Robust Overlapping Co-clustering

Meghana Deodhar Hyuk Cho Gunjan Gupta Joydeep Ghosh Inderjit Dhillon

{deodhar/ghosh}@ece.utexas.edu
{hyukcho/inderjit}@cs.utexas.edu
{gunjan}@lans.ece.utexas.edu

IDEAL-2008-09*

Intelligent Data Exploration & Analysis Laboratory

(Web: http://www.ideal.ece.utexas.edu/)

Department of Electrical and Computer Engineering
The University of Texas at Austin
Austin, Texas 78712
U.S.A.

February 20, 2008

Abstract

Clustering problems often involve datasets where only a part of the data is relevant to the problem, e.g., in microarray data analysis only a subset of the genes show cohesive expressions within a subset of the conditions/features. On such datasets, in order to accurately identify meaningful clusters, both non-informative data points and non-discriminative features need to be discarded. Additionally, since clusters could exist in different subspaces of the feature space, a co-clustering algorithm that simultaneously clusters objects and features is often more suitable as compared to one that is restricted to traditional "one-sided" clustering. We propose Robust Overlapping Co-clustering (ROCC), a scalable and very versatile framework that addresses the problem of efficiently detecting dense, arbitrarily positioned, possibly overlapping co-clusters in a dataset. ROCC works with a large variety of distance measures and different co-cluster definitions, making it applicable to a wide range of real life datasets. Through extensive experimentation we show that our approach is significantly more accurate in identifying biologically meaningful co-clusters in microarray data as compared to several other prominent approaches proposed for this task. We also point out other interesting applications of the proposed framework in solving challenging clustering problems.

1 Motivation

When clustering certain real world datasets, it has been observed that only a part of the data forms cohesive clusters. For example, in the case of microarray data, typically only a small subset of the genes cluster well and the rest can be considered non-informative [GG06]. Problems addressed by eCommerce businesses, such as market basket analysis and fraud detection involve huge, noisy datasets with coherent patterns occurring only in small pockets of the data. Moreover, for such data, coherent clusters could be arbitrarily positioned in subspaces formed by different, possibly overlapping subsets of features, e.g., different subsets of genes may be correlated across different subsets of experiments in microarray data. Additionally, it is possible that some features may not be relevant to any cluster.

Traditional clustering algorithms like k-means do not address either issue since they assign every data point to a cluster based on a similarity measure computed across all the features. Feature selection or feature clustering [DMK03, DB04] improve clustering results on high dimensional and noisy datasets, but do not allow clusters existing in different subsets of the feature space to be detected easily. Co-clustering simultaneously clusters the data along multiple axes, e.g., in the case of microarray data it simultaneously clusters the genes as well as the experiments [CC00] and can hence detect clusters existing in different subspaces of the feature space. In this paper we focus on real life datasets, where co-clusters are arbitrarily positioned in the data matrix, could be overlapping and are obfuscated by the presence of a large number of irrelevant points. Our goal is to discover dense, arbitrarily positioned and overlapping co-clusters in the data, while simultaneously pruning away non-informative objects and features.

2 Related Work

Density based clustering algorithms have a motivation similar to our proposed approach and use the notion of local density to cluster only a relevant subset of the data into multiple dense clusters. DBSCAN [EKSX96] and its improved versions such as OPTICS rely on the notion of density to find arbitrarily shaped clusters in large spatial databases in the presence of noise. These approaches however, are not scalable to high dimensional datasets and are limited to Euclidean or related distance measures. The One Class Information Bottleneck algorithm [CC04] and the Batch Ball One Class Clustering algorithm [GG05] are efficient and scalable algorithms that can work with a large class of distance measures. However, both algorithms find only a single dense region. The Bregman Bubble Clustering (BBC) technique [GG06] addresses the problem of discovering multiple, dense regions in a dataset while discarding the relatively non-coherent parts of the data. BBC provides a robust, scalable framework for clustering only a relevant fraction of the data. However, all of these approaches are developed for one-sided clustering only, where the data points are clustered based on their similarity across the entire set of features.

In contrast, both co-clustering (biclustering) and subspace clustering approaches locate clusters in subspaces of the feature space. The literature is both areas is recent but explosive, so we refer to the surveys and comparative studies in [MO04, PHL04, PSBea06] as good starting points. As we shall see in Section 3, none of the existing methods provide the full set of capabilities that the proposed method provides.

Co-clustering was first applied to gene expression data by Cheng and Church [CC00], who used a greedy search heuristic to generate arbitrarily positioned, overlapping co-clusters, based on a homogeneity constraint. However, their iterative insertion and deletion based algorithm is expensive, since it identifies individual co-clusters sequentially rather than all at once. The algorithm also causes random perturbations to the data while masking discovered biclusters, which reduces the clustering quality. The plaid model approach [LO02] improves upon this by directly modeling overlapping clusters, but still cannot identify multiple co-clusters simultaneously. These algorithms are not very general as they assume additive Gaussian noise models. Neither can they effectively handle missing data. The xMotif algorithm [MK03] is an iterative search method that identifies submatrices with genes that have near constant expression levels across a subset of experiments. Therefore, most generated biclusters are not very interesting from the biological point of view.

In addition to the greedy, iterative algorithms discussed above, deterministic algorithms such as BiMax [PSBea06] and OPSM [BDCKY02] have also been proposed. The BiMax approach, proposed by Prelic et al. is based on a simple, binary data model that assumes only a response/no response level for a gene with respect to an experiment. Accordingly, the expression data has to be discretized before application of the BiMax algorithm, which could cause loss of useful information. The algorithm exactly finds all the inclusion maximal biclusters consisting of genes that jointly respond across a subset of experiments. The number of co-clusters identified by BiMax is exponential in the number of genes and experiments, making it infeasible to store and evaluate all the co-clusters in case of large datasets. The order preserving sub matrix algorithm (OPSM) looks for submatrices in which the expression levels of all the genes induce the same linear ordering of the experiments. This algorithm although very accurate, is designed to identify only a single co-cluster. A recent extension to OPSM [ZWL08] finds multiple, overlapping co-clusters in noisy datasets, but is very expensive in the number of features.

Bregman Co-clustering (BCC), proposed by Banerjee et al. [BDG$^+$07], is a highly efficient, generalized framework for partitional co-clustering [MO04] that works with any distance measure that is a Bregman divergence, or equivalently any noise distribution from the regular exponential family. BCC includes several previously developed co-clustering algorithms like information theoretic co-clustering [DMM03] and minimum sum squared co-clustering [CD07a] as special cases. The BCC framework is however restricted to grid-based, partitional co-clustering and assigns every point in the data matrix to exactly one co-cluster i.e., the co-clustering is exhaustive and exclusive.

Subspace clustering is another approach related to the problem described in Section 1 that combines feature selection with clustering and can find clusters in different, possibly overlapping subspaces, using a "search and evaluate" technique. Parsons et al. [PHL04] present a survey of subspace clustering algorithms, which includes bottom-up grid based methods like CLIQUE and iterative top-down algorithms like PROCLUS. However, most of them are computationally intensive, need extensive tuning to get meaningful results and identify uniform clusters with very similar values rather than clusters with coherent trends or patterns. The pCluster model proposed by Wang et al. [WWYY02] generalizes subspace clustering and aims at discovering scaling or shifting patterns in clusters. Yoon et al. [YNBM05] improve upon this model and propose a biclustering algorithm that uses specialized data structures such as zero-suppressed binary decision diagrams to improve efficiency. Their algorithm is however

very complex and is exponential in the number of features in the worst case. The more recent reg-cluster model [XLTW06] is designed to identify arbitrary scaling and shifting co-regulations patterns along with negative correlations. However, unlike our proposed approach, these pattern based, heuristic approaches do not use a principled cost function and do not scale well due to high complexity in the number of features.

Finally, we would like to point out that most co-clustering or subspace clustering techniques try to find *all* the co-clusters that match a predetermined criterion, but do not explicitly aim at identifying the most coherent co-clusters.

3 Our Contributions

We propose Robust Overlapping Co-clustering (ROCC), a novel approach for discovering dense, arbitrarily positioned co-clusters in large, possibly high dimensional datasets. Our approach is robust in the presence of noisy and irrelevant objects as well as features, which our algorithm automatically detects and prunes during the clustering process. ROCC is based on a systematically developed objective function, which is minimized by an iterative procedure that provably converges to a locally optimal solution. ROCC is also robust to the noise model of the data and can be tailored to use the most suitable distance measure for the data, selected from a large class of distance measures known as Bregman divergences.

The final objective of ROCC is achieved in two steps. In the first step, the Bregman Co-clustering algorithm is adapted to automatically prune away non-informative data points and perform feature selection by eliminating non-discriminative features and hence cluster only the relevant part of the dataset. This step finds co-clusters arranged in a grid structure, but only a predetermined number of rows and columns are assigned to the co-clusters. Note however that this result cannot be achieved by simply removing some rows/columns from the BCC result. An agglomeration step then appropriately merges similar co-clusters to discover dense, arbitrarily positioned and even overlapping co-clusters. Figure 1 contrasts the nature of the co-clusters identified by ROCC with those found by BCC and illustrates the way in which they are conceptually derived from the partitional model of BCC.

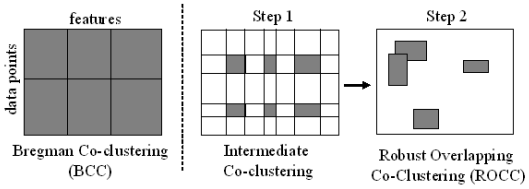


Figure 1: Nature of clusters identified by BCC and ROCC. Shaded areas represent clustered elements, rearranged according to cluster labels, while non-shaded areas denote discarded values.

The ROCC framework has the following key features that distinguish it from existing co-clustering algorithms.

1. The ability to mine the most coherent co-clusters from large and noisy datasets.

2. Detection of arbitrarily positioned and possibly overlapping co-clusters in a principled manner by iteratively minimizing a suitable cost function.

3. Generalization to all Bregman divergences, including squared Euclidean distance, commonly used for clustering microarray data and I-divergence, commonly used for text data clustering [DMM03].

4. The ability to naturally deal with missing data values, without introducing random perturbations or bias in the data.

5. Efficient detection all the co-clusters simultaneously rather than sequentially, enabling scalability to large and high-dimensional datasets.

As far as we know, no existing co-clustering algorithm [ZWL08, XLTW06, BDG+07, CC00] has all of the above features. Our contribution is significant, since as described in Section 1 there exist several applications where all these features are necessary for discovering meaningful patterns. Through extensive experimental evaluation on synthetic and real microarray data we illustrate that ROCC does significantly better than a variety of prominent, previously proposed co-clustering techniques. The application of ROCC to the simultaneous feature selection and clustering problem discussed in Section 9.1.1 highlights the ability of our approach to improve upon human curated feature selection in a completely unsupervised manner.

Notation: Lowercase letters (a,z) represent scalars, lowercase, bold faced letters ($\mathbf{b}, \mathbf{c}$) represent vectors, while uppercase letters (Z, W) represent matrices. An element of a matrix, e.g., Z is represented as z_{ij} where i and j are the row and column indices respectively.

4 Problem Definition

We begin with the formulation of the first step of the ROCC algorithm that prunes away irrelevant data points and features and clusters the rest into a grid of co-clusters (Refer to Figure 1). Let m be the total number of rows (data points) and n the total number of columns (features). The data can be represented as an $m \times n$ matrix Z of data points and features. Let s_r and s_c be the specified number of rows and columns respectively, to be retained after pruning. If the exact values are not known, it is sufficient to set s_r and s_c conservatively to large values since the algorithm (Section 5.3) does a second round of pruning as needed. Our aim is to simultaneously cluster s_r rows and s_c columns of Z, into a grid of k row clusters and l column clusters. The co-clusters will hence be comprised of $s_r \times s_c$ entries selected from the $m \times n$ entries of Z. Let K and L denote the sets consisting of the s_r clustered rows and the s_c clustered columns respectively. Let ρ be a mapping from the s_r rows $\in K$ to the k row clusters and γ be a mapping from the s_c columns $\in L$ to the l column clusters. Let squared Euclidean distance be the selected distance measure [1]. We want to find a co-clustering defined by (ρ, γ) and sets K and L for the specified s_r and s_c that minimize the following objective function

$$\sum_{g=1}^{k} \sum_{h=1}^{l} \sum_{u \in K:\rho(u)=g} \sum_{v \in L:\gamma(v)=h} w_{uv}(z_{uv} - \hat{z}_{uv})^2 \ , \tag{1}$$

where z_{uv} is the original value in row u, column v of the matrix, assigned to row cluster g and column cluster h and $\hat{z}_{uv}$ is the value approximated within co-cluster g-h. w_{uv} is the non-negative weight associated with matrix entry z_{uv}, which allows the algorithm to deal with

[1] A more general description, which allows any Bregman divergence as the loss function, is given in Section 5.3.

missing values and data uncertainties. For example, the weights for known values can be set to 1 and missing values can be effectively ignored by setting their weights to 0. The objective function is hence the element-wise squared error between the original and the approximated value, summed only over the clustered elements ($s_r \times s_c$) of the matrix Z. The value $\hat{z}_{uv}$ can be approximated in several ways, depending on the type of summary statistics that each co-cluster preserves. This is analogous to the notion of approximation schemes used by Bregman co-clustering [BDG$^+$07], where the data matrix is reconstructed to preserve a certain set of statistics within each co-cluster. Banerjee et al. [BDG$^+$07] identify six possible sets of summary statistics, of increasing complexity, that one might be interested in preserving in the reconstructed matrix $\hat{Z}$, which lead to 6 different co-clustering bases. Two of these approximation schemes for $\hat{z}_{uv}$ are described in Section 5.3.

In the next step of ROCC, the goal is to agglomerate similar co-clusters to recover the arbitrarily positioned co-clusters. In order to agglomerate co-clusters, we first define a distance measure between two candidate co-clusters ($cc1$ and $cc2$) as follows. Let cc denote the co-cluster formed by the union of the rows and columns in $cc1$ and $cc2$. The matrix entries $\hat{z}_{uv}$ in cc are approximated using the selected approximation scheme. The average element-wise error e for cc is computed as $e = \frac{1}{N}\sum_{z_{uv} \in cc}(z_{uv} - \hat{z}_{uv})^2$, where N is the number of elements in cc. The error e is defined to be the distance between $cc1$ and $cc2$.

5 ROCC Algorithm

5.1 Solving Step 1 of the ROCC Problem

A co-clustering (ρ, γ), that minimizes the objective function (1), can be obtained by an iterative algorithm. The objective function can be expressed as a sum of row or column errors, computed over the s_r rows and s_c columns assigned to co-clusters. If row u is assigned to row cluster g, i.e., $\rho(u) = g$, the row error is the error summed over the appropriate s_c elements in the row as

$$E_u(g) = \sum_{h=1}^{l} \sum_{v \in L:\gamma(v)=h} w_{uv}(z_{uv} - \hat{z}_{uv}(g))^2.$$

For a fixed γ, the best choice of the row cluster assignment for row u is the g that minimizes this error, i.e., $\rho^{new}(u) = \text{argmin}_g\, E_u(g)$. After computing the best row cluster assignment for all the m rows, the rows are sorted in increasing order of their row errors and the top s_r rows with minimum error are selected to participate in the current row clusters [2]. A similar approach is used to assign columns to column clusters. Note that the rows/columns that are not included in the current s_r/s_c rows/columns assigned to co-clusters are still retained since they could be included in the co-clusters in future iterations. The row cluster update step hence selects that fraction of rows that minimize the error among the entire set of rows. Also, the assignment of the selected rows to their corresponding row clusters is done in such a way that it directly minimizes the error. Hence, row cluster updates and similarly column cluster updates reduce the objective function. Updating column cluster assignments could cause the best row assignments to change and visa versa. Optionally, the row and column cluster reassignment steps can be repeated several times, in arbitrary order until row and column cluster memberships converge.

[2] This can be optimized to avoid unnecessary distance calculations by applying the triangle inequality [Elk03].

Given the current row and column cluster assignments (ρ, γ), the values $\hat{z}_{uv}$ within each co-cluster have to be updated by recomputing the required co-cluster statistics based on the approximation scheme. This problem is identical to the Minimum Bregman Information (MBI) problem presented in [BDG$^+$07] for updating the matrix reconstruction $\hat{Z}$. For example, for basis 2 and squared Euclidean distance as the Bregman divergence, the MBI solution is simply the mean of all the elements in the co-cluster. Solving the MBI problem for the update step is guaranteed to decrease the objective function.

This iterative procedure is described in Figure 2. Step 1(i) decreases the objective function due to the property of the MBI solution, while Steps 1(ii) and 1(iii) directly decrease the objective function. The objective function hence decreases at every iteration. Since this function is bounded from below by zero, the algorithm is guaranteed to converge. Note that the co-clustering problem is NP-complete, so convergence to only a local minimum is possible.

5.2 Solving Step 2 of the ROCC Problem

We now provide a heuristic to hierarchically agglomerate similar co-clusters. The detailed steps are as follows.

1. **Pruning co-clusters.** Since the desired number of co-clusters is expected to be significantly smaller than the number of co-clusters at this stage of the algorithm, co-clusters with the largest error values can be filtered out in this step. Filtering also reduces the computation effort required by the following merging step. If one has no idea of the final number of co-clusters, a simple and efficient filtering heuristic is to select the error cut-off value as the one at which the sorted co-cluster errors show the largest increase between consecutive values. The co-clusters with errors greater than the cut-off are filtered out. Alternatively, if the final number of co-clusters to be found is pre-specified, it can be used to prune away an appropriate number of co-clusters with the largest errors.

2. **Merging similar co-clusters.** This step involves hierarchical, pair-wise agglomeration of the co-clusters left at the end of the pruning step (Step 1) to recover the true co-clusters. Each agglomeration identifies the "closest" pair of co-clusters that can be well represented by a single co-cluster model and are thus probably part of the same original co-cluster, and merges them to form a new co-cluster [3]. "Closest" here is in terms of the smallest value of distance as defined in Section 4. The rows and columns of the new co-cluster consist of the union of the rows and columns of the two merged co-clusters. Merging co-clusters in this manner allows co-clusters to share rows and columns and hence allows partial overlap between co-clusters. If the number of co-clusters to be identified is pre-specified, one can stop merging when this number is reached. If not, merging is continued all the way until only a single co-cluster (or a reasonably small number of co-clusters) is left. The increase in the distance between successively merged co-clusters is then computed and the set of co-clusters just before the largest increase is selected as the final solution.

[3] A variant of this algorithm can be derived by adopting Ward's method [War63] to agglomerate co-clusters. Empirically we found little difference between the two approaches.

5.3 Overall ROCC Meta-Algorithm

In this Section we put together the procedures described in Sections 5.1 and 5.2 and present the complete ROCC algorithm. The key idea is to over-partition the data into small co-clusters arranged in a grid structure and then agglomerate similar, partitioned co-clusters to recover the desired co-clusters. The iterative procedure (Section 5.1) is run with large enough values for the number of row and column clusters (k and l). In particular, in our experiments on synthetic data, where the true number of co-clusters c^{true} is known, k and l were set to values $\geq 2 * c^{\text{true}}$. Similarly, the s_r and s_c input parameters are set to sufficiently large values. Since the pruning step (Step 2 in Section 5.2) takes care of discarding less coherent co-clusters, setting $s_r \geq s_r^{\text{true}}$ and $s_c \geq s_c^{\text{true}}$ is sufficient. The resulting $k \times l$ clusters are then merged as in hierarchical agglomerative clustering until a suitable stopping criterion is reached. The pseudo-code for the complete algorithm is illustrated in Figure 2.

Approximation Schemes. The ROCC algorithm can use each of the 6 schemes (co-clustering bases) listed by Banerjee et al. [BDG+07] for approximating the matrix entries $\hat{z}_{uv}$. For concreteness, we illustrate two specific approximation schemes with squared Euclidean distance, which give rise to block co-clusters and pattern-based co-clusters respectively [4]. The meta-algorithm in Figure 2 uses C to refer to the selected co-clustering basis.

Block co-clusters. Let the co-cluster row and column indices be denoted by sets U and V respectively. In this case, a matrix entry is approximated as $\hat{z}_{uv} = z_{UV}$, where $z_{UV} = \frac{1}{|U||V|}\sum_{u\in U, v\in V} z_{uv}$ is the mean of all the entries in the co-cluster. The total error of all the elements within a co-cluster is zero if and only if all the elements in the co-cluster are the same. This approximation method hence identifies fairly uniform blocks of the data matrix as co-clusters.

Pattern-based co-clusters. z_{uv} is approximated as $\hat{z}_{uv} = z_{uV} + z_{Uv} - z_{UV}$, where $z_{uV} = \frac{1}{|V|}\sum_{v\in V} z_{uv}$ is the mean of the entries in row u whose column indices are in V and $z_{Uv} = \frac{1}{|U|}\sum_{u\in U} z_{uv}$ is the mean of the entries in column v whose row indices are in U. The co-cluster error is zero if the submatrix described by U and V is of the form $xe^T + ey^T$ where $e = [11\cdots1]^T$ and both x and y are arbitrary vectors. This approximation can identify co-clusters that show a coherent trend or pattern in the data values, making it suitable for clustering gene expression data [CD07a].

Distance Measures. In Section 4 we developed the objective function (1) assuming squared Euclidean distance as the distance measure. The objective function and the iterative procedure to minimize it can be generalized to all Bregman divergences [BDG+07]. The selected Bregman divergence is denoted by d_ϕ in Figure 2.

Time Complexity. The iterative procedure (Step 1) for squared Euclidean distance and I-divergence is linear in the size of the input data with a time complexity of $O(mn + mkl + nkl)$ per iteration, for all six bases [BDG+07]. $O(mn)$ operations are due to the co-cluster model update step (Step (i)), the $O(mkl)$ operations are due to the computation of the distance of all the m rows from the k row clusters. Note that each such computation involves $O(l)$ operations rather than $O(n)$. Similarly, column reassignment involves $O(nkl)$ operations. The selection steps in the iterative procedure (Steps (iib) and (iiib)) can be performed in linear

[4] These co-cluster definitions correspond to basis 2 and basis 6 defined by the BCC framework [BDG+07] respectively.

Algorithm: ROCC
Input: $Z_{m\times n}$, s_r, s_c, k, l, basis C, d_ϕ
Output: Set of co-clusters

Step 1
Begin with a random co-clustering (ρ, γ)
Repeat
Step (i): Update co-cluster models, $\forall [g]_1^k, [h]_1^l$,
Update statistics for co-cluster (g, h) based on
basis C to compute new $\hat{z}$ values

Step (ii): Update ρ
(iia). $\forall [u]_1^m$,
$\rho(u) = \text{argmin}_g \sum_{h=1}^{l} \sum_{v \in L:\gamma(v)=h} w_{uv} d_\phi(z_{uv}, \hat{z}_{uv}(g))$
(iib). K = the set of s_r rows with least error
from among the m rows

Step (iii): Update γ
(iiia). $\forall [v]_1^n$,
$\gamma(v) = \text{argmin}_h \sum_{g=1}^{k} \sum_{u \in K:\rho(u)=g} w_{uv} d_\phi(z_{uv}, \hat{z}_{uv}(h))$
(iiib). L = the set of s_c columns with least error
from among the n columns

until convergence
Step 2: Post-process (see text for details)
(i) Prune co-clusters with large errors.
(ii) Merge similar co-clusters until stopping criterion is reached.
return identified co-clusters.

Figure 2: Pseudo-code for ROCC Meta-Algorithm

time. Rather than naively sorting all the row errors and selecting the top s_r rows with smallest error, a more efficient approach can be used. We can take advantage of the fact that the s_r rows with smallest error out of the m rows need not themselves be sorted. In this case, the problem reduces to finding the s_r^{th} smallest element out of a list of m elements, which can be done in linear time ($O(m)$) by the efficient order statistics based algorithm proposed by Blum et al. [BFP$^+$73]. The size of the data that Step 2 operates on is small as compared to the original dataset since only the most relevant matrix entries are retained at the end of Step 1. Although the hierarchical agglomerative clustering like merging procedure in Step 2 requires $O(M^2)$ operations for computation of all pair-wise co-cluster distances, where M is the number of co-clusters to be merged, M is typically small since several co-clusters are discarded in the pruning step (Step 2(i)).

Special Cases. The ROCC framework is very general with a lot of known algorithms as its special cases. Step 1 of ROCC with $s_r = m$ and $s_c = n$ is the same as BCC for all six bases and a selected Bregman divergence. With $l = n$, $s_c = n$ and basis 2 it is identical to Bregman Bubble Clustering [GG06]. With $l = n$, $s_c = n$, $s_r = m$ and basis 2 it becomes Bregman Clustering and additionally with squared Euclidean distance it is the same as k-means.

5.4 ROCC with Pressurization

The iterative minimization procedure in Step 1 of the ROCC algorithm begins with random initialization for ρ and γ, which could lead to poor local minima. The problem is particularly severe for small s_r and s_c since row and column clusters may not move much and the final clustering may be very heavily influenced by the initialization. This problem can be addressed by using the pressurization technique applied by BBC [GG06]. Pressurization begins by clustering all the data and iteratively shaving off data points and features till s_r rows and s_c columns are left. Let ${s_r^{press}}_{(j)}$ and ${s_c^{press}}_{(j)}$ denote the number of data points and features to be clustered using the Step 1 procedure (Figure 2) in the j^{th} iteration of pressurization. ${s_r^{press}}_{(1)}$ and ${s_c^{press}}_{(1)}$ are initialized to m and n respectively, after which these parameters are decayed exponentially till ${s_r^{press}}_{(j)} = s_r$ and ${s_c^{press}}_{(j)} = s_c$. The rate of decay is controlled by parameters β_{row} and β_{col}, which lie between 0 and 1. At iteration j, ${s_r^{press}}_{(j)} = s_r + \lfloor (m - s_r) * \beta_{row}^{j-1} \rfloor$ and ${s_c^{press}}_{(j)} = s_c + \lfloor (m - s_c) * \beta_{col}^{j-1} \rfloor$. The intuition behind pressurization is that by beginning with all the data being clustered and then slowly reducing the fraction of data clustered, co-clusters can move around considerably from their initial positions to enable the discovery of small, coherent patterns. For speed, each pressurization iteration need not be run to convergence and can be run for a small, fixed number of iterations. This gives competitive results in practice, without a large increase in run time.

5.5 Handling Very Sparse Datasets

Certain applications involve datasets that are extremely sparse, i.e., a very large fraction of the entries in the data matrix are missing. For instance, a dataset of user-movie ratings is typically very sparse, since it is unlikely for most users to have rated more than a negligible fraction of the entire set of movies. Text clustering and market basket analysis also involve large, sparse data matrices.

In such scenarios, the ROCC objective function given by Equation 1 is no longer suitable since the actual number of (non-missing) clustered elements could significantly differ for different choices of sets $\mathcal{K}$ and $\mathcal{L}$. If the objective function (1) was optimized as described in Section 5.1, the solution would be a very sparse subset of the data with a large number of missing entries, since missing entries have $w_{uv} = 0$ and do not contribute to the error. In the most extreme case, the solution would be an empty $s_r \times s_c$ block of the matrix Z with zero error.

To handle such datasets, the problem is reformulated to minimize the following objective function, instead of objective function (1):

$$\frac{1}{\sum_{u \in \mathcal{K}, v \in \mathcal{L}} w_{uv}} \sum_{g=1}^{k} \sum_{h=1}^{l} \sum_{u \in K: \rho(u) = g} \sum_{v \in L: \gamma(v) = h} w_{uv} (z_{uv} - \hat{z}_{uv})^2 \ , \tag{2}$$

Note that this modified objective function is based on the mean error over $s_r \times s_c$ elements of Z rather than the total error. For a sparse data matrix Z, the denominator in (2) will differ for different choices of sets $\mathcal{K}$ and $\mathcal{L}$ and hence needs to be explicitly considered in the cost function.

The algorithm to minimize the above objective function (ROCC Step 1) is very similar to that described in Section 5.1, with the only difference being in the update of sets $\mathcal{K}$ and $\mathcal{L}$

in the row and column cluster update steps respectively (Steps 1(iib) and 1(iiib) in Figure 2). Since the objective function (2) includes the normalization term, it cannot be decomposed over the rows. A naive greedy solution of forming set $\mathcal{K}$ by selecting the s_r rows with lowest mean row error is hence not optimal for the row selection problem. Instead, we select the s_r rows that minimize

$$\frac{\sum_{u \in \mathcal{K}} E_u(\rho^{new}(u))}{\sum_{u \in \mathcal{K}} \sum_{v \in \mathcal{L}} w_{uv}}.$$

An optimal selection of s_r rows can now be obtained by using a standard dynamic programming approach akin to the solution of the knapsack problem in [KT06]. A similar approach is used to select the columns that form set $\mathcal{L}$ in the column cluster update step.

Note that the complexity of these dynamic programming selection steps is $O(ms_r)$ and $O(ns_c)$, i.e., quadratic in the worst case. If the number of missing entries in the data matrix is comparatively small and the entries are missing at random, then the $\sum_{u \in \mathcal{K}, v \in \mathcal{L}} w_{uv}$ term in the objective function (2) does not vary too much over different $s_r \times s_c$ blocks of the matrix and can be ignored. In this case, the more efficient algorithm of Section 5.1, which performs a greedy selection of the s_r rows and s_c columns with the lowest row and column errors respectively can be applied. We verified empirically on the Lee microarray dataset (Section 8), which has around 15% of the data matrix entries missing, that there is no significant difference between the greedy strategy and dynamic programming in terms of co-cluster quality.

6 Generative Model for Soft ROCC

The formulation of Step 1 of the ROCC algorithm is based on a very intuitive generative model, which consists of a mixture of $k \times l$ exponential family distributions, corresponding to the $k \times l$ co-clusters, and a background uniform distribution, corresponding to the non-informative data matrix entries. Each element z_{ij} of the data matrix Z is assumed to be generated from this mixture model as follows

$$P(z_{ij}) = \sum_{I=1}^{k} \sum_{J=1}^{l} \alpha_{IJ} f_\psi(z_{ij}|\theta_{i,j,I,J}) + \alpha_0 p_0 \ , \ [i]_1^m, [j]_1^n, \tag{3}$$

where α_{IJ} denotes the co-cluster priors, f_ψ is an exponential family distribution with cumulant $\psi(.)$ and $\theta_{i,j,I,J}$ is the natural parameter. The form of $\theta_{i,j,I,J}$ depends on the co-clustering scheme. For scheme 2, $\theta_{i,j,I,J} = \theta_{I,J}$ but for other schemes $\theta_{i,j,I,J}$ also depends on i and j, e.g. for scheme 6 $\theta_{i,j,I,J} = \theta_i + \theta_j + \theta_{I,J}$ [AM07]. α_0 and p_0 denote the prior probability and the probability density of the uniform distribution. Note that the co-cluster assignments here are soft, i.e. each matrix element belongs to multiple co-clusters and to the background with different probabilities. By assuming that the matrix elements are generated i.i.d with weights w_{ij}, the incomplete data log-likelihood is given by

$$L(\Theta|Z) = \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} log P(z_{ij}), \tag{4}$$

where Θ denotes all the model parameters. It is however not possible to directly maximize this data log-likelihood and we associate latent variables $\rho(i)$ and $\gamma(j)$, denoting cluster membership,

with each element z_{ij}. $\rho(i)$ and $\gamma(j)$ take values from 0 to k and 0 to l respectively, where $\rho(i) = 0, \gamma(j) = 0$ are used to denote membership to the uniform background.

We use the standard EM technique to fit the mixture model described above. Similar to the analysis by Gupta and Ghosh [GG06], we assume that p_0 is set to an appropriately selected fixed value. Hence, the parameters we optimize over are the priors and the parameters of the exponential distributions. We first construct the free energy function [NH98] as a sum of the expected complete data log-likelihood and the entropy of the latent variables with respect to a distribution $\tilde{p}(\rho(i), \gamma(j))$

$$F(\tilde{p}, \Theta) = \sum_{ij} w_{ij} E_{\tilde{p}_{ij}} logP(z_{ij}, \rho(i), \gamma(j)) + \sum_{ij} w_{ij} H(\tilde{p}_{ij}), \text{ where} \tag{5}$$

$$E_{\tilde{p}_{ij}} logP(z_{ij}, \rho(i), \gamma(j)) = \sum_{IJ} \tilde{p}_{ij}(I, J) log(\alpha_{IJ} f_\psi(z_{ij}|\theta_{i,j,I,J})) + \tilde{p}_{ij}(0, 0) log\alpha_0 p_0,$$

$$H(\tilde{p}_{ij}) = \sum_{IJ} \tilde{p}_{ij}(I, J) log\tilde{p}_{ij}(I, J) + \tilde{p}_{ij}(0, 0) log\tilde{p}_{ij}(0, 0).$$

It can be shown that maximizing $F(\tilde{p}, \Theta)$ with respect to $\tilde{p}$ and Θ is equivalent to maximizing the data log-likelihood given by Equation 4 [NH98]. $F(\tilde{p}, \Theta)$ can be maximized by the EM procedure, which alternates between maximizing F w.r.t. $\tilde{p}$ for fixed Θ (E-step) and maximizing F w.r.t. Θ for fixed $\tilde{p}$ (M-step) and is guaranteed to converge to a local maximum of F. The details of the E and M step are as follows.

E-step: Here we have the constraint $\sum_{IJ} \tilde{p}_{ij}(I, J) + \tilde{p}_{ij}(0, 0) = 1 \ , \forall i, j$. By introducing Lagrange multipliers for the equality constraints and differentiating F w.r.t. $\tilde{p}$ one can arrive at the update step given below.

$$\tilde{p}_{ij}(I, J) = \frac{\alpha_{IJ} f_\psi(z_{ij}; \theta_{i,i,I,J})}{\sum_{IJ} \alpha_{IJ} f_\psi(z_{ij}; \theta_{i,i,I,J}) + \alpha_0 p_0} \ , [I]_1^k, [J]_1^l, \tag{6}$$

$$\tilde{p}_{ij}(0, 0) = \frac{\alpha_0 p_0}{\sum_{IJ} \alpha_{IJ} f_\psi(z_{ij}; \theta_{i,i,I,J}) + \alpha_0 p_0}.$$

M-step:

Updating priors: The constraint on the priors of the mixture components is $\sum_{IJ} \alpha_{IJ} + \alpha_0 = 1$. It can be shown that the update step is as follows

$$\alpha_{IJ} = \frac{\sum_{ij} w_{ij} \tilde{p}_{ij}(I, J)}{\sum_{ij} w_{ij}} \ , [I]_1^k, [J]_1^l,$$

$$\alpha_0 = \frac{\sum_{ij} w_{ij} \tilde{p}_{ij}(0, 0)}{\sum_{ij} w_{ij}}.$$

Updating parameters: The mixture component parameters can be estimated by differentiating F w.r.t. $\theta_{i,j,I,J}$ and equating to 0 as follows

$$\sum_{ij} w_{ij} \sum_{IJ} \tilde{p}_{ij}(I, J) \nabla_{\theta_{i,j,I,J}} log(\alpha_{IJ} f_\psi(z_{ij}; \theta_{i,i,I,J})) = 0.$$

We now consider the special case of a mixture of Gaussian components with fixed variance σ^2 and co-clustering basis 2. According to the bijection theorem between exponential families and Bregman divergences [BDG+07], the Gaussian components correspond to using squared Euclidean distance as the Bregman divergence. Since we are using basis 2, the natural parameter $\theta_{i,j,I,J} = \theta_{I,J}$ and $f_\psi(z_{ij}|\theta_{i,j,I,J}) = \frac{1}{\sqrt{(2\pi\sigma^2)}} e^{\frac{-(z_{ij}-\theta_{IJ})^2}{2\sigma^2}}$. The update step for θ_{IJ} can then be derived as

$$\theta_{IJ} = \frac{\sum_{ij} w_{ij}\tilde{p}_{ij}(I,J)z_{ij}}{\sum_{ij} w_{ij}\tilde{p}_{ij}(I,J)}.$$

One can observe that the E and M update steps illustrated above are generalizations of the update steps for the soft BBC algorithm that deals with the one-sided clustering formulation [GG06]. The soft ROCC model described above is very flexible since it does not impose any structure on the co-clusters. However, the standard EM algorithm used to estimate the model is not scalable and can be very slow in practice. To address this issue, one can impose structural constraints on the generative mixture model. We do so by restricting $\tilde{p}_{ij}$ to the form $\tilde{p}_{ij}(\rho(i),\gamma(j)) = \tilde{p}_i(\rho(i))\tilde{p}_j(\gamma(j))$, i.e. $\rho(i)$ and $\gamma(j)$ are independent. Hence, we assume that every row i of the data matrix Z belongs to row clusters $I = 1$ to k and the background row cluster $I = 0$ with probabilities $\tilde{p}_i(I)$ and similarly every column j belongs to column clusters $J = 1$ to l and the background column cluster $J = 0$ with probabilities $\tilde{p}_j(J)$. By decoupling the row and column cluster assignments, we can now estimate the model very efficiently by an EM based algorithm. The steps of the algorithm are illustrated in Figure 3. Each step monotonically decreases the free energy function, finally converging to a locally optimal solution. This algorithm is a generalization of step 1 of the *hard* ROCC algorithm described in Section 5, for soft cluster assignments.

Algorithm: EM for Soft ROCC
Input: $Z_{m\times n}$, k, l, exponential family with cumulant ψ
Output: Mixture component priors α, parameters θ and co-cluster assignments $\tilde{p}$

Begin with arbitrarily initialized assignments $\tilde{p}$
Repeat
E-step
Update Row Cluster Assignments:
$\tilde{p}_i(I) = c_i(\prod_{j,J=1 \text{ to } l}(\alpha_{IJ} f_\psi(z_{ij};\theta_{i,j,I,J}))^{w_{ij}\tilde{p}_j(J)}(\alpha_0 p_0)^{w_{ij}\tilde{p}_j(0)})^{\frac{1}{w_i}}\ \forall [i]_1^m, [I]_1^k,$
$\tilde{p}_i(0) = c_i(\prod_{j,J=0 \text{ to } l}(\alpha_0 p_0)^{w_{ij}\tilde{p}_j(J)})^{\frac{1}{w_i}}\ \forall [i]_1^m, I = 0,$
where c_i is a normalizing factor s.t. $\sum_{I=0}^{k}\tilde{p}_i(I) = 1$ and $w_i = \sum_j w_{ij}$.

Update Column Cluster Assignments:
$\tilde{p}_j(J) = c_j(\prod_{i,I=1 \text{ to } k}(\alpha_{IJ} f_\psi(z_{ij};\theta_{i,j,I,J}))^{w_{ij}\tilde{p}_i(I)}(\alpha_0 p_0)^{w_{ij}\tilde{p}_i(0)})^{\frac{1}{w_j}}\ \forall [j]_1^n, [J]_1^l,$
$\tilde{p}_j(0) = c_j(\prod_{i,I=0 \text{ to } k}(\alpha_0 p_0)^{w_{ij}\tilde{p}_i(I)})^{\frac{1}{w_j}}\ \forall [j]_1^n, J = 0,$
where c_j is a normalizing factor s.t. $\sum_{J=0}^{l}\tilde{p}_j(J) = 1$ and $w_j = \sum_i w_{ij}$.

M-step
Update Priors:
$\alpha_{IJ} = a(\sum_{ij} w_{ij}\tilde{p}_i(I)\tilde{p}_j(J))\ \forall [I]_1^k, [J]_1^l,$
$\alpha_0 = a(\sum_{ij} w_{ij}\sum_{J=1}^{l}\tilde{p}_i(0)\tilde{p}_j(J) + \sum_{I=1}^{k}\tilde{p}_i(I)\tilde{p}_j(0) + \tilde{p}_i(0)\tilde{p}_j(0))\ ,$
where a is a normalizing factor s.t. $\sum_{IJ}\alpha_{IJ} + \alpha_0 = 1$.

Update Mixture Component Parameters:
$\theta_{i,j,I,J} = argmax_\theta \sum_{ij} w_{ij}\sum_{IJ}\tilde{p}_i(I)\tilde{p}_j(J) log(\alpha_{IJ} f_\psi(z_{ij};\theta_{i,i,I,J}))\ \forall [I]_1^k, [J]_1^l.$
until convergence
return $(\tilde{p}, \alpha, \theta)$

Figure 3: EM algorithm for estimation of soft ROCC model

7 Experimental Evaluation on Synthetic Data

7.1 Synthetic Datasets

We first tested the ROCC approach in a controlled setting using synthetically generated data. An advantage of doing this is that synthetic data enables a good evaluation of the proposed approach since true co-cluster labels can be generated for the data matrix entries. For real world microarray datasets, cluster labels are commonly available for genes and/or experiments, but not for gene-experiment combinations. We also evaluate the ability of ROCC to find coherent, dense gene clusters on the Lee and Gasch microarray datasets, details of which are presented in Section 8. We compare the performance of ROCC with the biclustering algorithm proposed by Cheng and Church [CC00], referred to as Biclustering, which, similar to ROCC, aims at identifying arbitrary, overlapping co-clusters. The synthetic data was generated by the following procedure:

1. A matrix Z of values randomly selected from a uniform distribution (range 0-10) forms the background.

2. Rectangular blocks of coherent values representing co-clusters are arbitrarily placed in Z and are made to overlap in some datasets. The embedded co-clusters are either block based (basis 2) or pattern-based (basis 6). The specific data generation process for the two co-cluster definitions is described below.

   Block based. A block based co-cluster is generated by randomly selecting the co-cluster values from a Gaussian distribution with a mean in the range 0-10. Different co-clusters are generated from different Gaussian distributions. A co-cluster is hence a uniform block of constant values with additive Gaussian noise.

   Pattern based. To generate a pattern based co-cluster of size r by c, we first generate a vector (1 by c) of random values in the range 0-10. Each of the r rows of the co-cluster are generated by shifting this vector by a randomly selected constant factor. All the rows in the co-cluster hence have the same pattern across all the columns in the co-cluster. The values in the co-cluster are then perturbed by adding Gaussian noise.

3. Finally, the rows and columns of Z are randomly permuted.

Table 1 describes the synthetic datasets that were used for experimentation, available at `http://users.ece.utexas.edu/~deodhar/ROCCData/`.

Dataset	m,n	# co-clusters	basis
1	500, 500	3	2
2	500, 200	4	2
3	500, 500	3	6
4	500, 200	4	6

Table 1: Synthetic Datasets

7.2 Evaluation metrics

1. Relative non-intersection area (RNIA)

   The RNIA metric [PM06] compares co-clustering solutions when cluster labels for individual matrix entries are available. Given two co-clustering solutions S_1 and S_2, the RNIA of the two clusterings is defined as:

   $$RNIA(S_1, S_2) = \frac{|U| - |I|}{|U|} ,$$

   where $|U|$ and $|I|$ are the number of matrix elements in the union and intersection of the two clusterings respectively. RNIA can be applied to overlapping co-clusters by defining $|U|$ and $|I|$ as

   $$|U| = \sum_{i,j} \max(n_{ij}^{S_1}, n_{ij}^{S_2}) \ , \ |I| = \sum_{i,j} \min(n_{ij}^{S_1}, n_{ij}^{S_2}) \ ,$$

   where $n_{ij}^{S_1}$ and $n_{ij}^{S_2}$ are the number of co-clusters in S_1 and S_2 respectively that contain the matrix element z_{ij}. The RNIA is hence 0 if S_1 and S_2 are identical and 1 if they are completely disjoint. The RNIA of the co-clustering solution of an algorithm and the true co-clustering solution quantifies the ability of the algorithm to identify arbitrary co-clusters in the data matrix.

2. Unsupervised cost (UCOST)

   The unsupervised cost is a score that evaluates the quality of the identified co-clusters. It is defined as the average error between the original matrix entries and their approximation within the corresponding co-clusters, computed across only the entries assigned to co-clusters. Mathematically, the UCOST is computed as follows:

   $$UCOST = \frac{\sum_{i=1}^{k} \sum_{z_{uv} \in C_i} d_\phi(z_{uv}, \hat{z}_{uv})}{\sum_{i=1}^{k} |C_i|} ,$$

   where k is the number of identified co-clusters, C_i is the size of the i^{th} co-cluster and $\hat{z}_{uv}$ is the approximation of z_{uv} within the i^{th} co-cluster. The UCOST depends on the number of co-clusters k and decreases with a larger value of k for a given fraction of the data clustered.

7.3 Results

We now present a comprehensive evaluation of the ROCC algorithm on the datasets described above. On these synthetic datasets, ROCC is run with an additional local refinement step, which uses each identified co-cluster to seed the iterative procedure described in Section 5.1 with $k = 1$ and $l = 1$. This can help each co-cluster to move around locally, leading to a better solution. This step can be run in parallel for all the co-clusters, to refine all of them simultaneously. The true s_r and s_c values are the only inputs to the ROCC algorithm. The error cut-offs for the ROCC pruning and merging steps (Steps 2(i) and 2(ii)) are determined as described in Section 5.2 and the number of co-clusters is automatically discovered.

Matrix reconstruction. We begin our empirical analysis by visually comparing the quality of the data matrix reconstructed from the co-clustering results of ROCC and Biclustering.

The number of co-clusters is input to Biclustering. For each dataset, the α parameter for Biclustering is set to the average H-score [CC00] of the true co-clusters. Figures 4 and 5 display for datasets 1 and 4 the original data matrix and the matrix reconstructed by ROCC and Biclustering, after appropriate reordering. The entries within each identified co-cluster are approximated based on the co-cluster model, e.g. the suitable approximation scheme in case of ROCC. We can think of the background of the data matrix as being generated from a background model, which is assumed to be a uniform distribution. The matrix entries not assigned to any cluster by ROCC and Biclustering are used to estimate the parameters of this uniform distribution. The background of the data matrix is then reconstructed by randomly drawing points from the background model. The figures show that on both the datasets, the ROCC reconstruction is qualitatively better than that obtained by Biclustering and is reasonably close to the original dataset. Also, in case of both datasets ROCC is able to recover all the embedded co-clusters, in a completely unsupervised manner, without being given the number of co-clusters as an input.

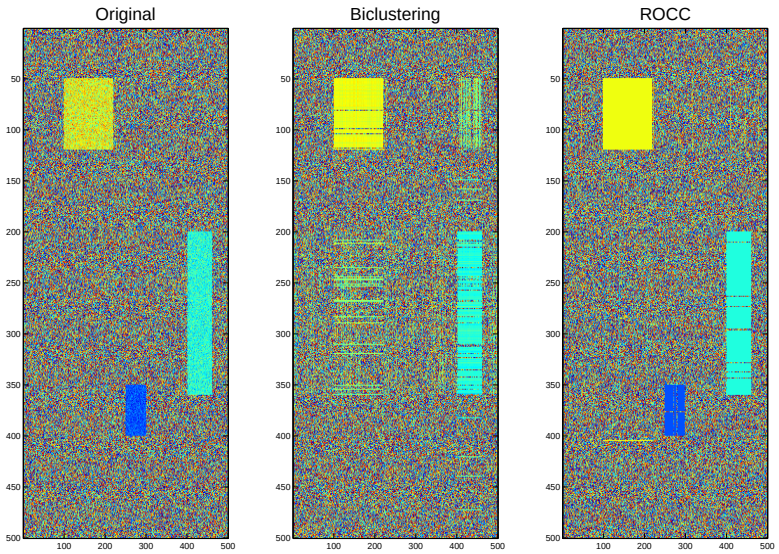


Figure 4: Dataset 1: matrix reconstruction by ROCC and Biclustering

RNIA comparison. Figure 6 compares the RNIA of the co-clusters identified by ROCC (with pressurization (Section 5.4)) and Cheng and Church's Biclustering algorithm [CC00] with the true co-clusters, across the 4 synthetic datasets. As before, the α parameter for Biclustering is set to the average H-score [CC00] of the true co-clusters. Note that the objective function optimized by the Biclustering algorithm is equivalent to the ROCC basis 6 objective with squared Euclidean distance, computed over a single bicluster, so datasets 3 and 4 actually match the underlying generative model of Biclustering. On datasets 1 and 2 which are relatively easy, ROCC does slightly better than Biclustering, while on datasets 3 and 4 ROCC performs significantly better.

Figure 7 displays the RNIA values for BCC and ROCC as s_r and s_c are varied from m and n to small values on synthetic datasets 3 and 4. The X-axis represents different fractions of

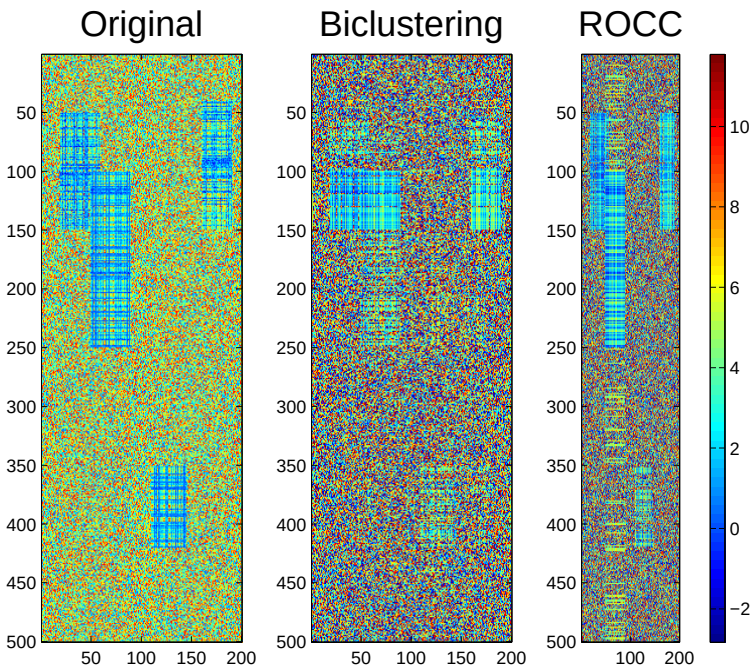


Figure 5: Dataset 4: matrix reconstruction by ROCC and Biclustering

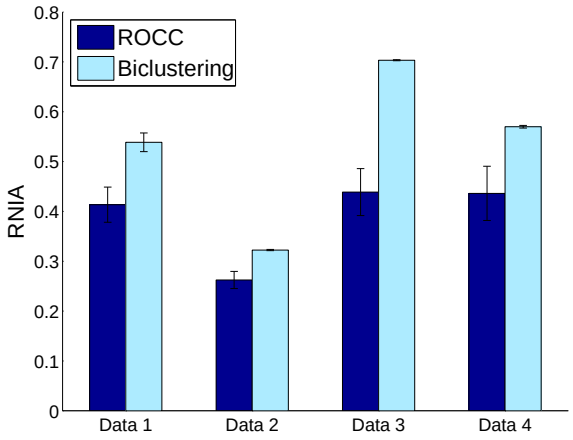
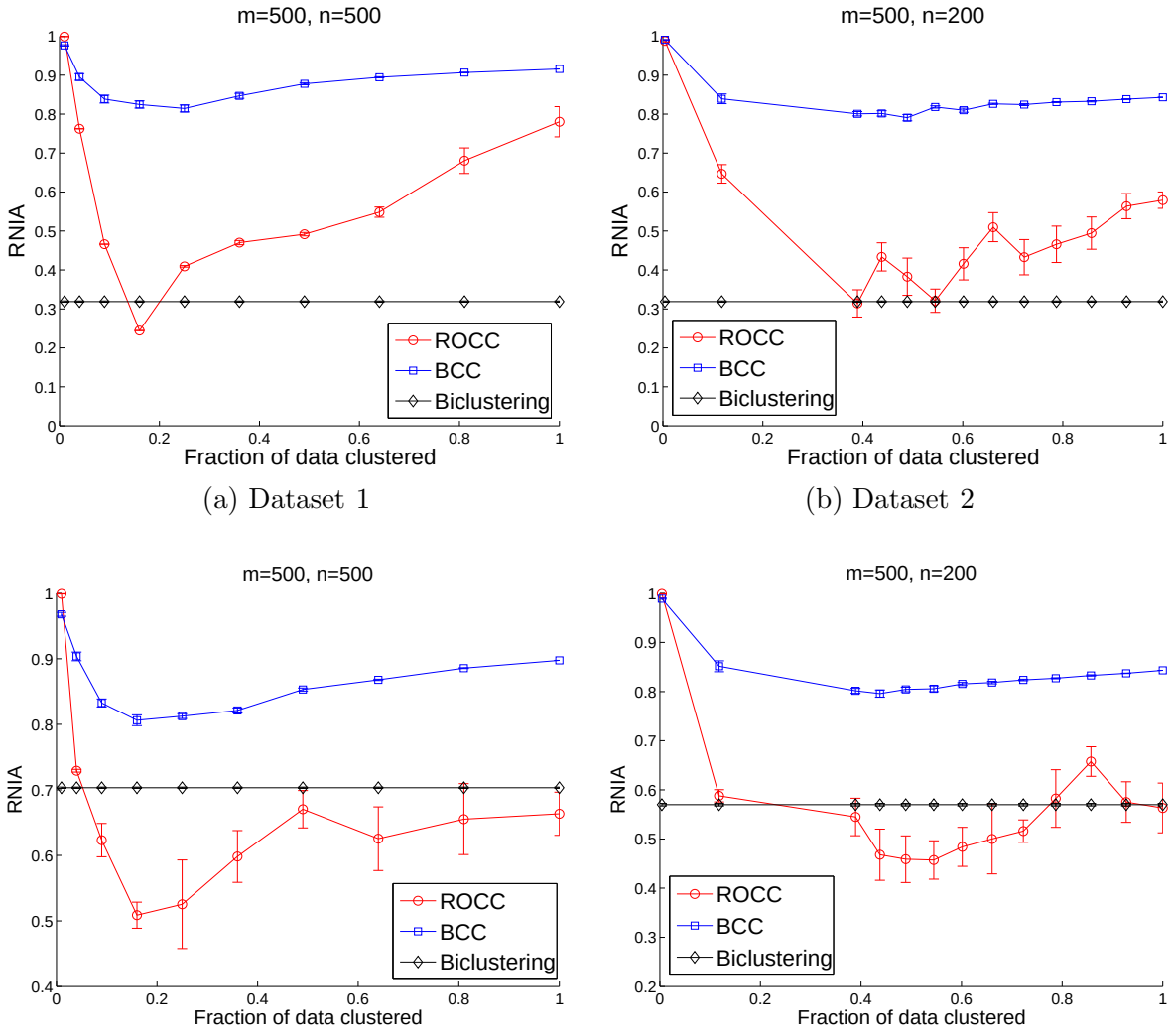


Figure 6: Comparison of ROCC and Biclustering with respect to RNIA on the 4 synthetic datasets.

the data being clustered, as rows and columns are pruned. Since BCC clusters all the data, pruning is carried out by a post-processing step. This step sorts the rows and columns by their distance to the corresponding cluster representatives and selects the s_r rows and s_c columns with smallest errors. Note that in case of ROCC, the actual fraction of the data clustered could be smaller than the corresponding X-axis value, due to the pruning of irrelevant co-clusters. The RNIA of the Biclustering solution, given the true number of clusters and a suitable α, is also displayed as a line on the same plot for comparison. Each plotted value is averaged over 10 randomly initialized runs. On these datasets the ROCC approach consistently achieves a significantly lower RNIA than BCC and does better than Biclustering, in the region near the true s_r and s_c values. We observed that on all the synthetic datasets, ROCC is able to identify the co-clusters and reconstruct the original data matrix very accurately. In almost all cases, ROCC also correctly recovers the true number of co-clusters.



(a) Dataset 1

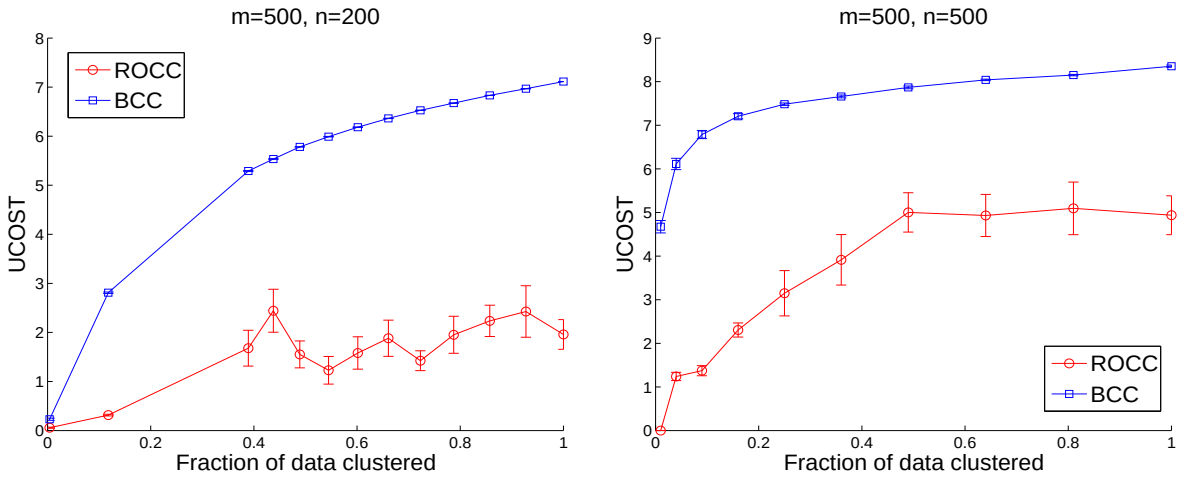
(b) Dataset 2

(c) Dataset 3

(d) Dataset 4

Figure 7: RNIA Plots comparing co-clustering approaches on synthetic data.

UCOST comparison. Figure 8 compares the average approximation error (UCOST) of the co-clusters discovered by BCC, and ROCC on datasets 2 and 3. Since ROCC discards less coherent co-clusters with large errors, whereas BCC retains all the identified co-clusters, ROCC has a significantly lower UCOST.



(a) Dataset 2 (b) Dataset 3

Figure 8: UCOST Plots comparing co-cluster quality on synthetic data.

Lesion studies. The ROCC algorithm has multiple steps, each of which aim at improving the accuracy of identifying the co-clusters embedded in the data. We performed lesion studies on the synthetic datasets as a systematic way of understanding the ROCC algorithm and the contributions of its different components. Table 2 displays the RNIA of the clusters obtained after each step of the ROCC algorithm on the four synthetic datasets and compares it to the RNIA of the clustering result of the BCC algorithm.

One can observe a dramatic improvement in the RNIA after the pruning step of ROCC (Step 2(i)). Since the merging step (Step 2(ii)) only restructures the co-clusters to try and identify the shapes and sizes of the original co-clusters, it does not affect the RNIA value much. The additional local refinement step improves the RNIA slightly on datasets 2,3 and 4. Hence, the maximum improvement to the RNIA seems to be attributed to Step 2(i). However since Step 1 of ROCC imposes a grid structure on the identified co-clusters, it is forced to include certain regions of the data matrix in its clustering solution, which results in a very high RNIA. To evaluate the improvement due to ROCC Step 1 over BCC, it is more suitable to use NMI (normalized mutual information [SG02]) to compare the assigned row and column cluster labels with the true labels. Note that since NMI does not consider overlapping clusters, for the purpose of this comparison rows/columns belonging to more than one cluster are assigned arbitrarily to only one of the clusters while assigning rows/columns their true cluster labels.

Tables 3 and 4 display respectively the NMI of the row and column cluster labels with the true labels for the partitional clustering obtained by BCC and ROCC Step 1. Note, that the least fitting rows and columns were discarded from the BCC solution in a post-processing step to obtain cluster assignments for s_r rows and s_c columns. The results illustrate that the simultaneous pruning of rows and columns while co-clustering, performed by ROCC Step 1

Algo./Step	Data 1	Data 2	Data 3	Data 4
BCC	0.809 (0.009)	0.765 (0.011)	0.794 (0.004)	0.799 (0.007)
ROCC Step 1	0.809 (0.009)	0.772 (0.014)	0.783 (0.006)	0.780 (0.007)
ROCC Step 2(i)	**0.105 (0.004)**	0.352 (0.053)	**0.463 (0.040)**	0.553 (0.056)
ROCC Step 2(ii)	0.324 (0.022)	0.366 (0.051)	0.509 (0.040)	0.555 (0.051)
Refining Step	0.415 (0.035)	**0.352 (0.052)**	0.472 (0.051)	**0.499 (0.058)**

Table 2: Lesion study results: RNIA after each step of ROCC on the synthetic datasets, averaged over 10 randomly initialized runs.

significantly improves the row and column clustering quality as compared to BCC. One can hence see that the ROCC Step 1 result cannot be obtained by simply post-processing the BCC result.

	Data 1	Data 2	Data 3	Data 4
BCC	0.633 (0.041)	0.787 (0.007)	0.591 (0.032)	0.548 (0.037)
ROCC Step 1	0.840 (0)	0.826 (0.003)	0.656 (0.042)	0.559 (0.037)

Table 3: Lesion study results: NMI of row cluster labels with true labels on the synthetic datasets.

	Data 1	Data 2	Data 3	Data 4
BCC	0.674 (0.033)	0.786 (0.013)	0.602 (0.039)	0.594 (0.035)
ROCC Step 1	0.894 (0)	0.903 (0.008)	0.691 (0.032)	0.663 (0.028)

Table 4: Lesion study results: NMI of column cluster labels with true labels on the synthetic datasets.

8 Experimental Results on Real Microarray Datasets

We now evaluate the performance of ROCC on two yeast microarray datasets, the Lee dataset [LDAM04] and the Gasch dataset [GSK+00]. The Lee dataset consists of gene expression values of 5612 yeast genes across 591 experiments and can be obtained from the Stanford Microarray Database (`http://genome-www5.stanford.edu/`). The Gasch dataset consists of the expression values of 6151 yeast genes under 173 environmental stress conditions and is available at `http://genome-www.stanford.edu/yeast_stress/`. Since the ground truth for both datasets is available only in the form of pair-wise linkages between the genes that are known to be functionally related, we compare the quality of the co-clusters identified by different co-clustering algorithms by computing the overlap lift [GG06] for the genes in each co-cluster. Overlap lift measures how many times more correct links are predicted as compared to random chance and is related to a normalized version of the proportion of disconnected genes measure used by [PSBea06]. On these datasets, the aim is to find the most coherent and biologically useful 150 to 200 co-clusters. We run ROCC (with pressurization) on the Lee dataset with the input parameters set to $s_r = 2000$, $s_c = 400$, $k = 50$ and $l = 10$ and on the Gasch dataset with $s_r = 500$, $s_c = 120$, $k = 80$, $l = 15$. These parameters were chosen based on a practical and intuitive understanding of the datasets, gained via exploratory analysis. Based on the final

number of clusters to be identified, Step 2 of ROCC prunes all but the best 200 co-clusters and then continues merging until 150 co-clusters are left. The set of co-clusters just before the largest increase in merge distance is returned as the solution.

Figure 9 compares the performance of ROCC with prominent co-clustering algorithms, i.e., Cheng and Church's Biclustering algorithm, the Order Preserving Submatrix algorithm (OPSM) [BDCKY02], the BiMax algorithm [PSBea06], and the BCC algorithm on the Lee and Gasch microarray datasets. Through extensive experimentation, Prelic et al. [PSBea06] show that the OPSM and the BiMax algorithms outperform other well known co-clustering algorithms like Samba [TSS02], ISA [BIB03] and xMotif [MK03] on real microarray data. The BiMax and OPSM results were generated using the BicAT software(`http://www.tik.ee.ethz.ch/sop/bicat/`) [BBP+06]. For application of the BiMax algorithm, we discretized the datasets using the discretization threshold suggested in [PSBea06]. Since it would be infeasible to evaluate the exponential number of co-clusters identified by BiMax, we selected the first 200 co-clusters for comparison. Though OPSM is designed to return only the best co-cluster, it is extended in BicAT to return up to 100 largest co-clusters among those that achieve the optimal score. The value of the l parameter for OPSM was set to 10. The Biclustering algorithm [5] is run with the number of clusters equal to 200. The value of the parameter α for the Lee dataset is set to the average H-score [CC00] of the clusters in the ROCC co-clustering solution with the highest overlap lift from Figure 15(a), i.e. $\alpha = 0.032$. Similarly α is set to 0.017 for the Gasch dataset [6]. We also attempted comparison with the state-of-art subspace clustering algorithm proposed by Yoon et al. [YNBM05], but due to its worst case exponential complexity in the number of features this technique did not scale to our datasets. In the Lee and Gasch datasets respectively, around 15% and 3% of the matrix entries are missing. As described in Section 5, ROCC and BCC can ignore missing entries by appropriately setting the weight matrix. The missing entries in the data matrix input to the other algorithms are replaced by random values in the same range as the known expression values. Both ROCC and BCC use squared Euclidean distance and find pattern-based co-clusters. BCC uses the same s_r and s_c values as ROCC for the post processing step as described in Section 7.3. The ROCC, BCC and Biclustering results are averaged over 10 trials, while OPSM and BiMax are deterministic.

Figure 9 shows that on both datasets, ROCC does much better than the other co-clustering approaches in terms of the overlap lift of the gene clusters. The figure also displays above each bar, the percentage of the data matrix entries clustered by the corresponding algorithm. On the Lee dataset, it is interesting that although ROCC clusters a much larger fraction of the data matrix entries than Biclustering, OPSM and BiMax, the co-clusters are of superior quality. The genes in the Lee dataset show better co-expression patterns as compared to Gasch, resulting in better performance. The Gasch dataset is more noisy, which explains why a larger fraction of the dataset has to be pruned as compared to Lee to get meaningful clusters. We found that while the co-clusters identified by OPSM showed a clear linear ordering of experiments, most of the co-clusters included very few experiments and almost all the genes, which is not very useful from the point of view of predicting gene linkages. ROCC does dramatically better than Biclustering, possibly due to its improved capabilities for identifying overlapping co-clusters and handling missing data. BiMax produced several tens of thousands of co-clusters, of which

[5] We used the implementation provided by Cheng and Church.

[6] We found the biclustering results to not be very sensitive to the choice of α (range of α values from 0.005 to 0.04 were tried).

we picked only the first 200, since it was infeasible to evaluate all of them.

The exact execution time values are not given since the algorithms were implemented in different languages and executed on different platforms, however it was evident that ROCC is faster than OPSM and BiMax, but slower than BCC and Biclustering.

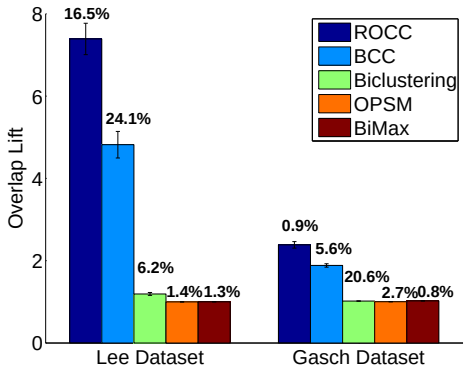


Figure 9: Comparison of ROCC with other co-clustering algorithms on the Lee and Gasch datasets. The number above each bar indicates the percentage of the data matrix entries clustered by each algorithm.

Most of the gene clusters identified by ROCC on the Lee dataset were biologically significant, with very low p-values [7]. Table 5 summarizes some of the identified high purity gene clusters. The coverage (x/y) indicates that x out of the y known members of a category were found. In contrast, the 10 best gene clusters identified by Biclustering had an *average p-value* of 5.50e-04.

# genes	Category(Coverage)	p-value
20	tRNA ligase (8/36)	6.63e-14
63	Endoplasmic reticulum membrane (14/84)	3.886e-14
108	Structural constituent of ribosome (104/206)	<1e-14
20	PF00270-DEAD (12/51)	<1e-14
12	Glycolysis (8/16)	<1e-14
37	Threonine endopeptidase (23/30)	<1e-14
24	PF00660-SRP1-TIP1 (22/30)	<1e-14
16	Translation factor, nucleic acid binding (7/47)	9.115e-12
32	Arp2/3 protein complex (5/7)	5.22e-11

Table 5: Examples of biologically significant clusters found by ROCC on the Lee dataset.

Lesion study results. With the aim of determining the importance of the different components of ROCC, as discussed earlier for synthetic data (Section 7.3), we now present lesion study results on the real microarray datasets. Table 6 displays the overlap lift, averaged over 10 randomly initialized runs, of the gene clusters identified by BCC and by each step of ROCC. Step 1 of ROCC gives a big improvement in overlap lift as compared to BCC. However, steps 2(i) and (ii) do not seem to be that useful. Firstly, since the co-clustering quality is measured in terms of the overlap lift of the predicted gene linkages, i.e., only gene clusters are considered, this method is not entirely suitable to evaluate the effect of the pruning (step 2(i)) and merging (step 2(ii)) steps on co-cluster coherence. Hence, Step 2(i), which prunes less coherent co-clusters from the grid structured co-clusters provided by step 1 will not affect the overlap lift

[7] The p-values were obtained using *Funspec* (`http://funspec.med.utoronto.ca/`)

unless an entire row cluster is pruned. This is possibly why the overlap lift is almost unchanged after the execution of step 2(i) on the Lee dataset. On Gasch this step results in a small improvement, possibly besause Gasch is more noisy and aggressive pruning helps eliminate a large number of noisy genes. Step 2(ii) actually reduces the overlap lift on both datasets. This is expected, since this step merges co-clusters, which reduces the number of models and hence the number of parameters representing the data.

Algo./Step	Lee	Gasch
BCC	4.819 (0.322)	1.884 (0.042)
ROCC Step 1	8.055 (0.350)	2.334 (0.045)
ROCC Step 2(i)	7.997 (0.347)	2.373 (0.085)
ROCC Step 2(ii)	7.980 (0.352)	2.269 (0.103)

Table 6: Lesion study results: Overlap lift after each step of ROCC on the Lee and Gasch datasets.

9 Other Applications of the ROCC Algorithm

In this section we discuss two interesting applications of the BBCC algorithm, which show that it can be very useful in diverse settings.

9.1 Simultaneous Feature Selection and Clustering

A particular instance of the ROCC algorithm can be used to perform feature selection along one axis, while simultaneously clustering along the other. ROCC interleaves feature selection with clustering and iteratively improves both, which is better than independently performing feature selection *a priori* and then clustering using the identified features. Additionally, ROCC also clusters related features, achieving simultaneous dimensionality reduction. We now consider an exemplary application of ROCC in the above context to four publicly available human cancer microarray datasets: Colon cancer [ABN$^+$99], Leukemia [GST$^+$99], Lung cancer [ASS$^+$02], and MLL [ASS$^+$02], all generated using Affymetrix technology. Table 7 summarizes information about the datasets, including the number of samples, genes and classes.

Table 7: Description of the human cancer microarray datasets.

Dataset	Colon	Leukemia	Lung	MLL
No. of genes	2000	7129	12533	12582
No. of samples	62	72	181	72
No. of sample classes	2	2	2	3
Sample class names	Normal(20) Tumor(42)	ALL(47) AML(25)	ADCA(150) MPM(31)	ALL(24) AML(25) MLL(23)

In this application, the aim is to cluster the samples, to recover the existing sample groups in an unsupervised manner, using the expression values of the genes as features. Of the thousands of genes present, many of them are known to be non-informative, redundant and have

noisy expression values, which makes feature selection an important issue. We expect that clustering and feature selection of the genes will bring about a substantial boost in the quality of the sample clusters. Note that feature selection interleaved with clustering in this manner is completely automatic and data driven as compared to the use of domain knowledge to select the best feature set as a preprocessing step. Additionally, the co-clusters of genes and samples provide explanatory power by describing the gene patterns that distinguish between the sample clusters, something that is not directly possible with regular one-sided clustering of the samples. For this application, ROCC (with pressurization) is set up to cluster all the samples and prune along the "gene" axis. Note, that the agglomeration procedure (Step 2) is not required for this application.

Data Transformation. Raw data values have a limitation that they do not disclose how they vary from the central tendency of the distribution. Therefore, the transformation of raw data is considered to be one of the most important data preprocessing steps since the variance of a variable will determine its importance in a given model [SLM94]. Cho et al. formally analyze the effect of various data transformations on their two co-clustering residue measures [CD07b], which correspond to our block and pattern-based co-cluster definitions respectively. They show that through column standardization (CS), the second residue is able to capture both scaling and shifting patterns. Therefore, in our experiments, we use CS as a data preprocessing step. Column standardization is defined as

$$a'_{ij} = \frac{a_{ij} - \mu_j}{\sigma_j}$$

for $i = 1, \cdots, m$ and $j = 1, \cdots, n$, where $\mu_j = \frac{1}{m}\sum_{i=1}^{m} a_{ij}$ and $\sigma_j^2 = \frac{1}{m}\sum_{i=1}^{m}(a_{ij} - \mu_j)^2$. Column standardization results in each column having zero mean and unit variance, and is also called "autoscaling". Through this standardization, the relative variation in intensity is emphasized. Since CS is a linear transformation, the relative positions of observations and the shape of the original distribution is retained.

Sample clustering evaluation measure. To evaluate the performance of the sample clustering solutions, we quantify cluster quality using the accuracy of the cluster labels with respect to the true class labels. Since we know the actual number of sample classes, we set the number of sample clusters equal to the number of classes in all our experiments. The cluster accuracy for the sample clusters is defined as follows.

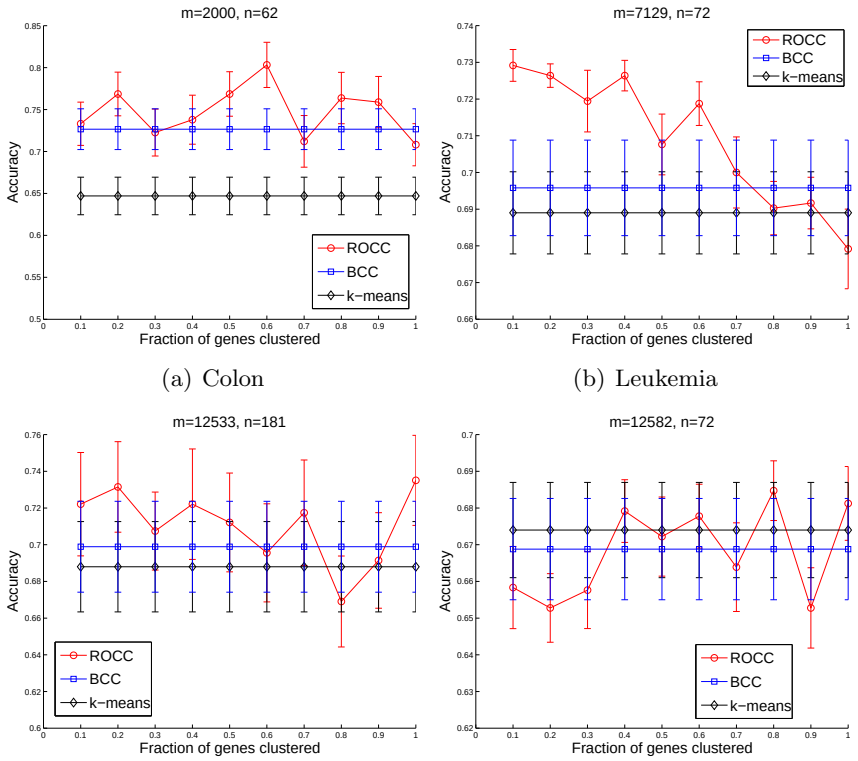
$$Accuracy(\%) = \frac{1}{n}\left(\sum_{i=1}^{l} t_i\right) \times 100,$$

where n is the total number of samples, l is the number of sample clusters, and t_i denotes the numbers of the samples correctly clustered into a sample class i. We first form a confusion matrix where $(i,j)^{\text{th}}$ entry is the number of samples in cluster i that belong to the true class j. Each t_i is a diagonal element of the corresponding confusion matrix whose cluster labels are permuted so that the sum of diagonal elements is maximized.

9.1.1 Results on Human Cancer Datasets

We compare the sample cluster accuracy of ROCC with BCC, which uses all the genes to obtain a co-clustering of genes and samples, as well as k-means, which uses all the genes as

features to cluster the samples. Along with an accurate clustering of the samples, ROCC also generates coherent gene clusters. A qualitative evaluation of the gene clusters is presented in Section 9.1.4. Figure 10 illustrates the sample cluster accuracy of ROCC at different fractions of the genes clustered, averaged over 20 trials, on the cancer datasets. For comparison, the sample cluster accuracy values of BCC and k-means are also plotted as straight lines in the same Figure. The displayed results are on the column standardized datasets. Both BCC and ROCC use basis 2 with squared Euclidean distance on the Leukemia, Lung and MLL datasets. Basis 2 is a better choice in this case as compared to basis 6 since these datasets contain a very large number of genes, many of which are known to be irrelevant and noisy and basis 6 might overfit the noise in the data. Our experiments support this hypothesis since we found basis 2 consistently outperforming basis 6 on these datasets. On the Colon dataset BCC and ROCC perform better with basis 6 since this dataset includes a relatively smaller, more carefully selected set of genes. k is set to 100 for BCC and ROCC. Overall ROCC performs better than k-means. On Leukemia and Colon, ROCC does a lot better than BCC as well (Figures 10(a), 10(b)). Since these datasets are known to show coherent patterns involving only a small fraction of all the genes, the automatic feature selection by ROCC contributes to its good performance.



(a) Colon (b) Leukemia

(c) Lung Cancer (d) MLL

Figure 10: Sample cluster accuracy at different fractions of the genes clustered.

9.1.2 Results on Reduced (Filtered) Cancer Datasets

The human cancer datasets described in Table 7 have been analyzed by several researchers, who have addressed the problem of noisy and redundant genes by simple preprocessing heuristics, based on domain knowledge commonly adopted in microarray experiments [BJ02, DB02, DF02]. These researchers used differential expression to filter redundant genes, and have made available reduced versions of these datasets containing only the most discriminative genes. We evaluated ROCC on these reduced datasets to determine whether the sets of reduced genes still contain non-informative genes, that can be pruned further, leading to a larger improvement in sample cluster accuracy. The reduced datasets were generated by filtering genes whose relative deviation ($|max/min|$) or absolute deviation ($|max - min|$) were less than predefined values, where max and min refer respectively to the maximum and minimum expression levels for a particular gene across all samples. The following describes the specific preprocessing steps taken to generate the reduced version of each human cancer dataset.

Colon Cancer. Genes with $|max/min| < 15$ and $|max - min| < 500$ were removed, leaving behind a reduced set of 1096 genes.

Leukemia. The reduced Leukemia dataset consists of 3571 genes, obtained by thresholding genes with a floor of 100 and ceiling of 16000 and then further screening out genes with $|max/min| < 5$ and $|max - min| < 500$.

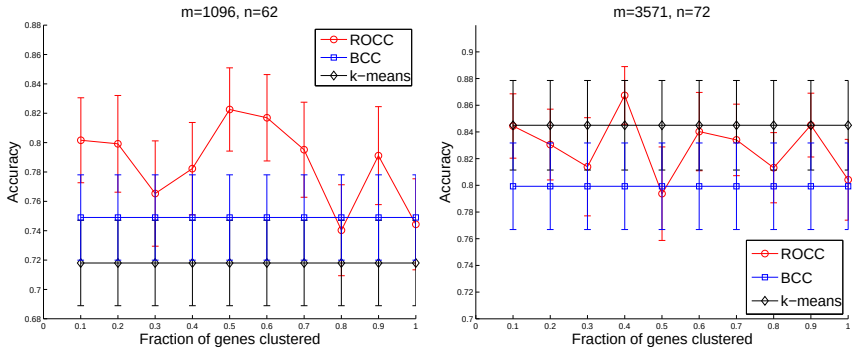
Lung cancer. The reduced Lung Cancer dataset has a total of 2401 genes with $|max - min| \geq 600$, obtained by screening out genes by fixing the relative deviation cut-off to 5 and varying the absolute deviation value so that the genes are pruned to 1/5-th.

MLL dataset. The reduced MLL dataset was generated using similar preprocessing steps as for the reduced Lung Cancer dataset and consists of 2474 genes with $|max - min| \geq 5500$.

Figure 11 compares the accuracy of the sample clusters obtained by ROCC with BCC and k-means on the reduced cancer datasets at different fractions of the genes clustered. Here BCC and ROCC use basis 6 with squared Euclidean distance and $k = 20$. The results are averaged over 20 runs. Overall the accuracy of all the algorithms improves since the datasets used here are preprocessed based on domain knowledge and include a more careful selection of genes. On all the datasets ROCC does better than BCC, indicating that the reduced set of genes still contains some less informative genes, which when pruned improves performance further. The results are most dramatic on the Lung Cancer dataset (Figure 11(c)), where ROCC is significantly better than BCC and k-means. On the other datasets, the performance of k-means is comparable to that of ROCC, which is not surprising since these datasets include genes carefully selected using domain knowledge.

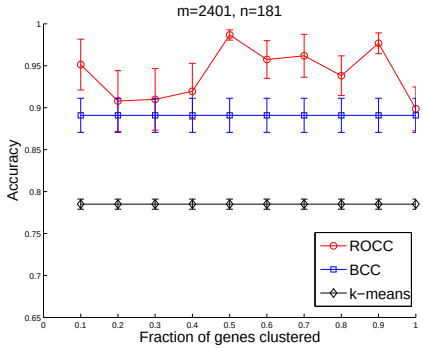
9.1.3 Visual Analysis of Co-clusters

In this Section we visually analyze the co-clusters discovered by BCC and ROCC by suitably plotting the matrix of data values to gain insights into the nature of the two co-clustering solutions. The genes are sorted in increasing order of their distance to their corresponding row cluster centroids and the top 100 genes are selected for visualization. Only the top 100 genes are selected, since a larger number of genes reduces the resolution of the plot, making it difficult to observe coherent co-cluster patterns. The plot includes the selected genes (rows) and all the samples (columns) of the data matrix, rearranged according to their row and column cluster labels. We now display a few such interesting plots.

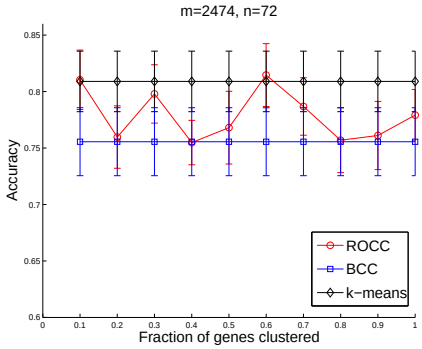


(a) Colon

(b) Leukemia



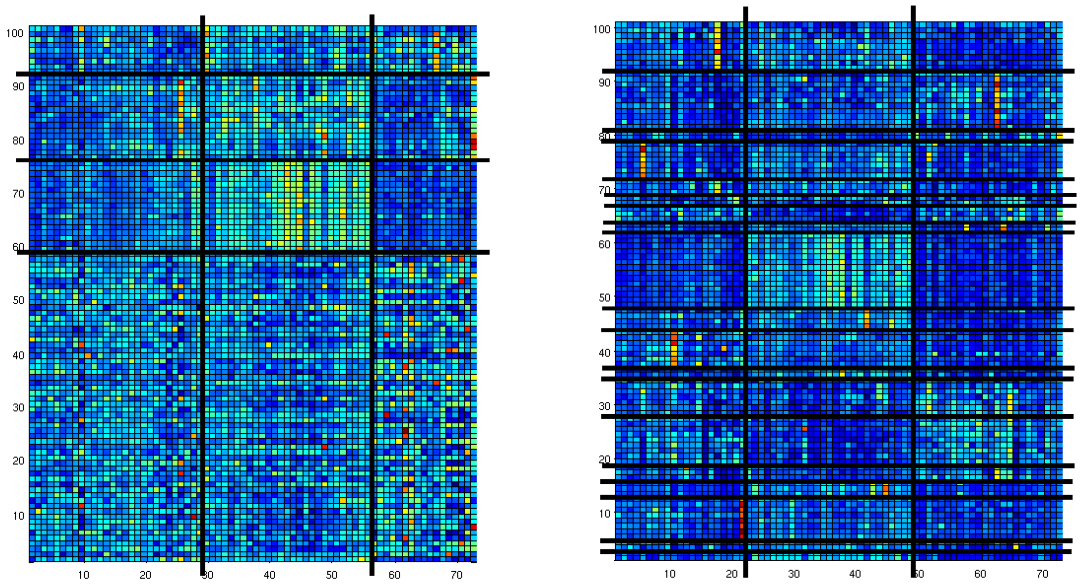
(c) Lung Cancer



(d) MLL

Figure 11: Sample cluster accuracy at different fractions of the genes clustered on the reduced cancer datasets.

The following figures display the plots of the rearranged data matrices obtained by BCC and ROCC after clustering 10% of the genes and all the samples of the column standardized, reduced human cancer datasets. On the MLL dataset, the plot produced by ROCC (Figure 12(b)) shows 3 distinct sample clusters, visually identifiable based on the expression patterns of only 100 genes. All the co-clusters appear to be very coherent and consist of highly correlated expression patterns. In contrast, in the BCC plot (Figure 12(a)) a very clear distinction of the 3 sample clusters is not visible and particularly, the first row cluster does not seem to be very coherent or discriminative with respect to the sample clusters. Lung cancer is a very imbalanced dataset with 31 samples belonging to one cluster and the remaining 150 belonging to the other. Figure 13 indicates that the expression patterns of the selected 100 genes identify the 2 sample clusters more accurately with ROCC than BCC. In the ROCC plot (Figure 13(b)) one can observe that the second sample cluster contains exactly 31 samples distinguished by a highly correlated gene expression pattern. On the Leukemia dataset (Figure 14), one can see the difference in the nature of the clusters identified by ROCC and BCC. While BCC identifies large co-clusters, ROCC identifies very dense, small co-clusters with a clear correlation between the included expression profiles.



(a) BCC (b) ROCC

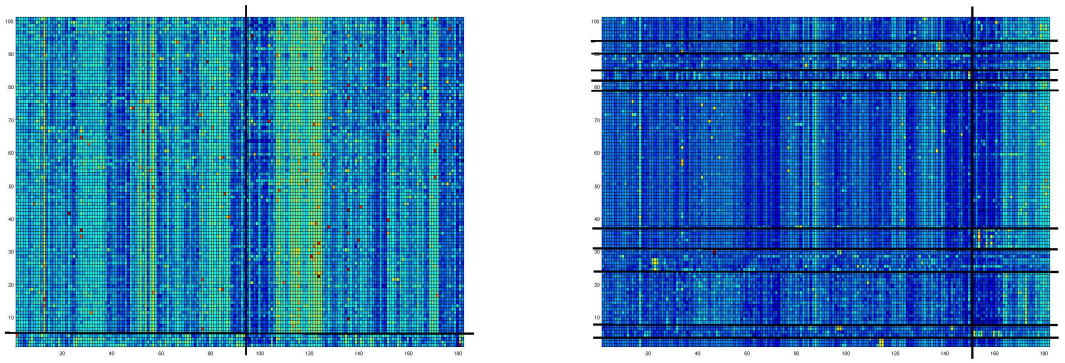
Figure 12: MLL Dataset

9.1.4 P-value Analysis of Co-clusters

Although our main objective in the experiments in Sections 9.1.1 and 9.1.2 was to cluster samples, identifying and investigating clusters of genes is also of interest to biologists. ROCC, by simultaneous pruning and co-clustering of genes and samples identifies gene clusters along with clustering the samples. Here we evaluate the quality of the gene clusters identified by ROCC on the reduced human cancer datasets.

We evaluate the quality of gene clusters by assessing their statistical significance in terms of the p-value. P-value, calculated using the hypergeometric distribution, represents the probability that the intersection of the genes assigned to a cluster with any given functional category occurs by chance. Here we focus on the Gene Ontology (GO) categories within "biological process (BP)" , "cellular component (CC)" and "molecular function (MF)". Among many publicly available functional profiling tools, we use the DAVID software (`http://david.abcc.ncifcrf.gov`) to compute gene cluster p-values. DAVID adopts the Fisher Exact Test to measure the gene enrichment in annotation terms. Categories with Fisher Exact p-value < 0.05 are considered statistically significant.

Table 8 illustrates for each gene cluster identified by ROCC on the reduced Leukemia dataset, the GO category with the lowest p-value that the cluster can be mapped to. These results are obtained by clustering 10% of the genes and all the samples of the column standardized dataset. Note that although only 10% of the genes were clustered, the set of background genes for p-value computation includes all the genes in the dataset. One can observe that a very large number of clusters have p-values < 0.05 and hence show statistically significant gene enrichment. ROCC is hence successful at identifying coherent and biologically significant gene



(a) BCC (b) ROCC

Figure 13: Lung Dataset

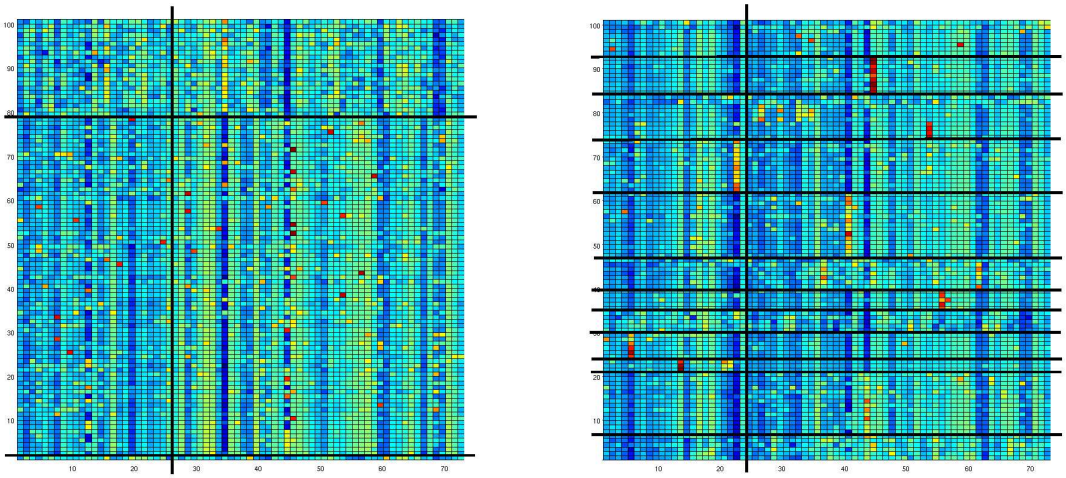
clusters along with improving the performance of sample clustering as seen in Sections 9.1.1 and 9.1.2. Due to space limitation, we list only the gene clusters of the Leukemia dataset. The results on the other human cancer datasets were very similar.

Cluster	GO Category	cluster size	# genes in category	p-value
1	CC-soluble fraction	42	4	$8.70E-02$
2	BP-primary metabolism	8	8	$2.20E-02$
3	CC-extracellular region	15	5	$2.10E-02$
4	MF-catalytic activity	40	22	$4.60E-03$
5	BP-regulation of transcription, DNA-dependent	4	3	$7.10E-02$
6	MF-G-protein coupled receptor activity	43	6	$1.50E-03$
7	MF-ubiquitin conjugating enzyme activity	15	2	$3.60E-02$
8	CC-mitochondrion	12	4	$1.30E-02$
9	MF-structural constituent of ribosome	28	4	$1.80E-03$
10	BP-reproduction	45	6	$7.30E-04$
11	BP-neurophysiological process	22	5	$1.30E-02$
12	BP-cellular lipid metabolism	15	4	$7.80E-03$
13	BP-DNA repair	5	2	$4.90E-02$
14	BP-locomotory behavior	12	3	$1.70E-02$
15	CC-intrinsic to plasma membrane	7	5	$1.60E-03$
16	CC-endoplasmic reticulum	8	3	$4.40E-02$
17	CC-intracellular organelle	13	8	$6.30E-02$

Table 8: Leukemia dataset: p-values of gene clusters identified by ROCC

9.2 Simultaneous Pruning of Irrelevant Data Points and Features

Identifying arbitrarily positioned, overlapping clusters in noisy datasets is a very challenging problem. However, in several datasets, simply identifying and pruning non-informative and noisy data points and features results in substantially better clusters than clustering all the data. We illustrate the ability of ROCC to do this on the Lee and Gasch microarray datasets. Figure 15 compares the overlap lift of the gene clusters identified by ROCC (with pressuriza-



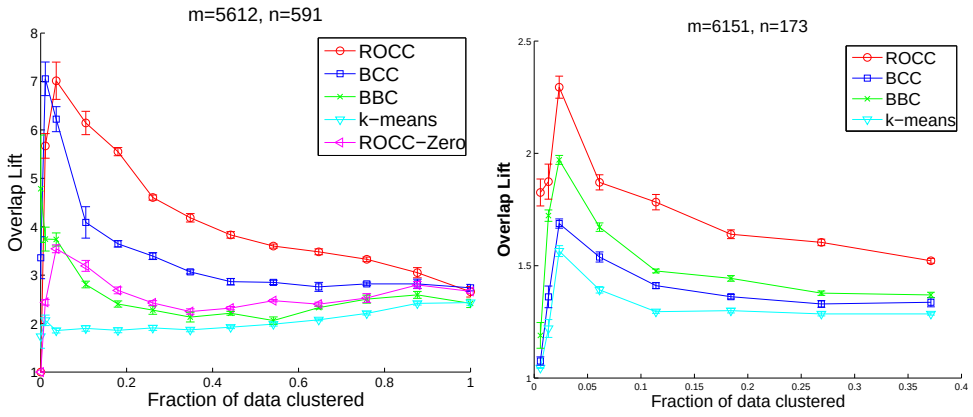
(a) BCC (b) ROCC

Figure 14: Leukemia Dataset

tion), BCC, BBC and k-means at varying fractions of the data matrix clustered. The number of genes and experiments clustered are gradually increased from very small values to the entire set of genes and experiments and the fraction of the data matrix entries actually assigned to clusters is represented along the X-axis. Since BCC, BBC and k-means assign every row to some row cluster, for a fair comparison of these algorithms with ROCC, pruning is carried out by a post processing step. This step ranks the rows in terms of their distance to the corresponding row cluster representatives and chooses the required fraction of rows as the top fraction from the ranked list. Note that in these experiments we only use step 1 of ROCC, thus needing much less computation.

The missing gene expression values in both datasets (around 15% in the Lee dataset and 3% in the Gasch dataset) are set to zero in the data matrix input to BBC and k-means. Note that setting a missing entry to zero is equivalent to assuming that there is no change in the gene expression value. On the other hand, as described in Section 5 the co-clustering algorithms ignore missing expression values by using an appropriately set weight matrix, rather than replacing missing entries with a fixed constant value. All the algorithms are run with squared Euclidean distance and ROCC and BCC are run with basis 6. On the Lee dataset, the values of k and l are set to 20 and 10, while on Gasch these values are set to 50 and 5.

ROCC shows a dramatic improvement in performance over all the other approaches. One can observe from Figure 15 that the overlap lift reduces as the fraction of genes and experiments clustered increases, highlighting the fact that the dense and coherent clusters existing in the data involve only a small fraction of the dataset. This improvement in performance achieved by ROCC is attributed to two factors, co-clustering along with pruning and being able to ignore missing values. In order to evaluate the improvement exclusively due to the pruning and co-clustering approach, we also display the performance of ROCC-Zero in Figure 15(a), where the input to the ROCC algorithm is the data matrix with missing entries set to 0. One can observe that pruning alone also brings about significant improvement over BBC and k-means.



(a) Lee Dataset (b) Gasch Dataset

Figure 15: Overlap lift at different fractions of the data clustered on the Lee and Gasch datasets.

We also compared the different algorithms on the basis of the biological significance of the gene clusters identified. This was done by computing the p-values of the gene clusters in the Lee dataset using *Funspec* (`http://funspec.med.utoronto.ca/`). Table 9 compares the algorithms with respect to the p-values of the gene clusters obtained with k set to 20, l set to 10, $s_r = 300$ and $s_c = 400$. Here in order to level the playing field, all the algorithms are run on the dataset with the missing entries set to 0. The objective is to qualitatively evaluate the improvement in the identified gene clusters brought about by the simultaneous co-clustering and pruning done by ROCC. One can observe that ROCC has the maximum percentage of clusters with low p-values and does very well as compared to BCC and k-means.

	% of clusters with p-values below		
Algorithm	10^{-4}	10^{-6}	10^{-10}
ROCC (20 clusters)	65	30	10
BBC (20 clusters)	60	30	10
BCC (13 clusters)	38.5	15.4	7.7
k-means (16 clusters)	37.5	18.75	6.25

Table 9: P-value comparison

10 Conclusions

In this paper, we have presented Robust Overlapping Co-clustering as a comprehensive framework capable of dealing with several challenges in clustering real life datasets. ROCC extends beyond the rigid structure of partitional co-clustering and detects arbitrarily positioned, overlapping co-clusters very accurately as illustrated in Sections 7.3 and 8. The ability of ROCC to operate with different distance measures and detect co-clusters with different structures, along with its scalability and robustness provided by pressurization makes it extremely versatile. ROCC is robust to the presence of irrelevant data points and features and discovers coherent co-clusters very accurately as illustrated in Section 7. Moreover, though ROCC requires several input parameters to be supplied, i.e., s_r, s_c, k and l, it is relatively very robust to the choice

of these parameters because of the post-processing steps as detailed in Section 5.3. While in this paper we focused on clustering microarray data, it would be worthwhile to investigate the applicability of suitable instances of the ROCC framework to clustering problems in different domains like text mining and market basket analysis.

11 Acknowledgments

This research was supported by NSF grants IIS 0325116, IIS 0307792, IIS 0713142 and CCF 0431257.

References

[ABN+99] U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. In *Proc. NAS*, volume 96, pages 6745–6750, 1999.

[AM07] D. Agarwal and S. Merugu. Predictive discrete latent factor models for large scale dyadic data. In *Proc. KDD '07*, pages 26–35, 2007.

[ASS+02] S. A. Armstrong, J. E. Staunton, L. B. Silverman, R. Pieters, M. L. den Boer, M. D. Minden, S. E. Sallan, E. S. Lander, T. R. Golub, and S. J. Korsmeyer. MLL translocations specify a distinct gene expression profile that distinguishes a unique leukemia. *Nature Genetics*, 30:41–47, 2002.

[BBP+06] S. Barkow, S. Bleuler, A. Prelic, P. Zimmermann, and E. Zitzler. Bicat: a biclustering analysis toolbox. *Bioinformatics*, 22(10):1282–1283, 2006.

[BDCKY02] A. Ben-Dor, B. Chor, R. Karp, and Z. Yakhini. Discovering local structure in gene expression data: the order-preserving submatrix problem. In *RECOMB '02: Proceedings of the sixth annual international conference on Computational biology*, pages 49–57, 2002.

[BDG+07] A. Banerjee, I. Dhillon, J. Ghosh, S. Merugu, and D. Modha. A generalized maximum entropy approach to bregman co-clustering and matrix approximation. *JMLR*, 8:1919–1986, 2007.

[BFP+73] M. Blum, R. Floyd, V. Pratt, R. Rivest, and R. Tarjan. Time bounds for selection. *J. Comput. Syst. Sci.*, 7(4):448–461, 1973.

[BIB03] S. Bergmann, J. Ihmels, and N. Barkai. Iterative signature algorithm for the analysis of large-scale gene expression data. *Phys Rev E Stat Nonlin Soft Matter Phys.*, 67, 2003.

[BJ02] T. H. Bø and I. Jonassen. New feature subset selection procedures for classification of expression profiles. *Genome Biology*, 3(4), 2002.

[CC00] Y. Cheng and G. M. Church. Biclustering of expression data. In *Proc. ICMB '00*, pages 93–103, 2000.

[CC04] K. Crammer and G. Chechik. A needle in a haystack: Local one-class optimization. In *Proc. ICML '04*, 2004.

[CD07a] H. Cho and I. Dhillon. Co-clustering of human cancer microarrays using minimum sum-squared residue co-clustering. *TCBB*, 2007.

[CD07b] H. Cho and I. Dhillon. Effect of data transformation on residue. *UTCS Technical Report TR-07-55*, pages Downloadable from `ftp://ftp.cs.utexas.edu/pub/techreports/tr07--55.pdf`, 2007.

[DB02] M. Dettling and P. Bühlmann. Supervised clustering of genes. *Genome Biology*, 3(12), 2002.

[DB04] J. Dy and C. Brodley. Feature selection for unsupervised learning. *JMLR*, 5:845–889, 2004.

[DF02] S. Dudoit and J. Fridlyand. A prediction-based resampling method for estimating the number of clusters in a dataset. *Genome Biology*, 3(7):research0036.1–0036.21, 2002.

[DMK03] I. Dhillon, S. Mallela, and R. Kumar. A divisive information-theoretic feature clustering algorithm for text classification. *JMLR*, 3:1265–1287, 2003.

[DMM03] I. Dhillon, S. Mallela, and D. Modha. Information-theoretic co-clustering. In *Proc. KDD '03*, pages 89–98, 2003.

[EKSX96] M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. KDD '96*, 1996.

[Elk03] C. Elkan. Using the triangle inequality to accelerate k-means. In *Proc. ICML '03*, pages 147–153, 2003.

[GG05] G. Gupta and J. Ghosh. Robust one-class clustering using hybrid global and local search. In *Proc. ICML '05*, pages 273–280, 2005.

[GG06] G. K. Gupta and J. Ghosh. Bregman bubble clustering: A robust, scalable framework for locating multiple, dense regions in data. In *Proc. ICDM'06*, pages 232–243, 2006.

[GSK+00] A.P. Gasch, P.T. Spellman, C.M. Kao, O. Carmel-Harel, M.B. Eisen, G. Stotz, D. Botstein, and P.O. Brown. Genomic expression program in the response of yeast cells to environmental changes. *Molecular Cell Biology*, 11:4241–4257, 2000.

[GST+99] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

[KT06] J. Kleinberg and E. Tardos. *Algorithm Design*. Pearson Education, 2006.

[LDAM04] I. Lee, S. Date, A. Adai, and E. Marcotte. A probabilistic functional network of yeast genes. *Science*, 306:1555–1558, 2004.

[LO02] L. Lazzeroni and A. B. Owen. Plaid models for gene expression data. *Statistica Sinica*, 12(1):61–86, 2002.

[MK03] T. Murali and S. Kasif. Extracting conserved gene expression motifs from gene expression data. *Pacific Symposium on Biocomputing*, 8:77–88, 2003.

[MO04] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE Trans. on Computational Biology and Bioinformatics*, 1(1):24–45, 2004.

[NH98] R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*, pages 355–368. MIT Press, 1998.

[PHL04] L. Parsons, E. Haque, and H. Liu. Subspace clustering for high dimensional data: a review. *SIGKDD Explor. Newsl.*, 6(1):90–105, 2004.

[PM06] A. Patrikainen and M. Meila. Comparing subspace clusterings. *IEEE Transactions on Knowledge and Data Engineering*, 18(7):902–916, 2006.

[PSBea06] A. Preli, P. Zimmermann S. Bleuler, and A. Wille et al. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, 22(9):1122–1129, 2006.

[SG02] A. Strehl and J. Ghosh. Cluster ensembles – a knowledge reuse framework for combining partitionings. In *Proceedings of AAAI 2002, Edmonton, Canada*, pages 93–98. AAAI, July 2002.

[SLM94] F. C. Sánchez, P. J. Lewi, and D. L. Massart. Effect of different preprocessing methods for principal component analysis applied to the composition of mixtures: detection of impurities in HPLC-DAD. *Chemometrics and Intelligent Laboratory Systems*, 25(2):157–177, 1994.

[TSS02] A. Tanay, R. Sharan, and R. Shamir. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18:136–144, 2002.

[War63] J. Ward. Hierarchical grouping to optimize an objective function. *Journal of American Statistical Association*, 58(301):236 – 244, 1963.

[WWYY02] H. Wang, W. Wang, J. Yang, and P.S. Yu. Clustering by pattern similarity in large data sets. In *Proc. KDD-2002*, pages 394–405, 2002.

[XLTW06] X. Xu, Y. Lu, A. Tung, and W. Wang. Mining shifting-and-scaling co-regulation patterns on gene expression profiles. In *Proc. ICDE '06*, page 89, 2006.

[YNBM05] S. Yoon, C. Nardini, L. Benini, and G. D. Micheli. Discovering coherent biclusters from gene expression data using zero-suppressed binary decision diagrams. *IEEE/ACM TCBB*, 2(4):339–354, 2005.

[ZWL08] M. Zhang, W. Wang, and J. Liu. Mining approximate order preserving clusters in the presence of noise. In *Proc. ICDE '08*, pages 160–168, 2008.