NON-PARAMETRIC MACHINE LEARNING METHODS FOR CLUSTERING AND VARIABLE SELECTION

Qian Liu

A dissertation submitted to the faculty at the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Biostatistics.

Chapel Hill
2014

Approved by:

Eric Bair

Michael Kosorok

Andrew Nobel

Gary Slade

Donglin Zeng

ABSTRACT

Qian Liu: Non-parametric machine learning methods for clustering and variable selection
(Under the direction of Eric Bair)

Non-parametric machine learning methods have been popular and widely used in many scientific research areas, especially when dealing with high-dimension low sample size (HDLSS) data. In particular, clustering and biclustering approaches can serve as exploratory analysis tools to uncover informative data structures, and random forest models have their advantage in coping with complex variable interactions.

In many situations it is desirable to identify clusters that differ with respect to only a subset of features. Such clusters may represent homogeneous subgroups of patients with a disease. In this dissertation, we first propose a general framework for biclustering based on the sparse clustering method. Specifically, we develop an algorithm for identifying features that belong to biclusters. This framework can be used to identify biclusters that differ with respect to the means of the features, the variances of the features, or more general differences. We apply these methods to several simulated and real-world data sets, and the results of our methods compare favourably with previous published methods, with respect to both predictive accuracy and computing time.

As a follow up to the biclustering study, we further look into the sparse clustering algorithm, and point out a few limitations of their proposed method for tuning parameter selection. We propose an alternative approach to select the tuning parameter, and to better identify features with positive weights. We compare our algorithm with the existing sparse clustering method on both simulated and real world data sets, and

the results suggest that our method out-performs the existing method, especially in presence of weak clustering signal.

For the last project, we consider random forest variable importance (VIMP) scores. We propose an alternative algorithm to calculate the conditional VIMP scores. We test our proposed algorithm on both simulated and real-world data sets, and the results suggested that our conditional VIMP scores could better reveal the association between predictor variables and the modelling outcome, despite the correlation among predictor variables.

I would like to dedicate this Doctoral dissertation to my one year old daughter, Emmalyn Pu, who came like an angel and brightened up my life. I wish all the best for her.

And with this opportunity, I would like to address a special thank you to my dear husband, Dongqiuye Pu. We met in college when we were both young and full of dreams. Now we have been holding hands and sharing 10 years of wonderful life together. With laughter and tears, we grow as a family. I hope we can continue this sweet journey for another 10, 20, 30 years.

Acknowledgments

I would like to first acknowledge the inspirational instruction of Dr. Eric Bair, who is the best advisor I can ever wish for, and also a good supporting friend outside the scope of biostatistics.

I would also like to acknowledge my fantastic committee members, Drs. Michael Kosorok, Andrew Nobel, Gary Slade, and Donglin Zeng. They have been super nice and helpful, both on my research and on my career development.

Last but not the least, I would like to acknowledge my friends and family. I can not go this far without your generous support and love. I am blessed to have you in my life.

Table of Contents

List of Tables

List of Figures

Chapter 1

Literature Review

1.1 Clustering and biclustering

Unsupervised exploratory methods play an important role in the analysis of high-dimension low sample size (HDLSS) data, such as microarray gene expression data. Such data sets can be expressed in the form of a $n \times p$ matrix X, where each row corresponds to one observation and each column corresponds to one predictor variable. In clustering and biclustering studies, we refer to observations as objects, and predictor variables as features. Both terms will be used interchangeably in the following sections.

Unsupervised learning is a powerful tool for discovering interpretable structures within HDLSS data without reference to external information. In particular, clustering methods partition observations into subgroups based on their overall feature patterns. In general, the observations are assigned into different sub-categories in such a way that the objects within the same sub-category (called a cluster) are more similar to each other than to those outside. And there is no single standard in determining the similarity among observations – different clustering algorithms have different definitions, depending on the study purpose and the data structure encountered. But clustering methods in general will consider all the features when making the decision.

In many situations, these clusters may differ with respect to only a subset of the features. Such data structure could be overlooked if one clusters using all the features.

Biclustering methods may be useful in situations where clusters are formed by only a subset of the features. Biclustering aims to identify sub-matrices U within the original data matrix X. The results may be visualized as two-dimensional signal blocks (after reordering the rows and columns) containing only a subset of the observations and a subset of the features. For example, in a gene expression data set collected from cancer patients, there may exist a subset of genes whose expression levels differ among patients with a more aggressive form of cancer. Identifying such a bicluster may aid in the treatment of the cancer patients.

In general, biclustering results in rectangular signal blocks (called biclusters) where the entries within are more similar to each other than to those outside. Different biclustering algorithms have different definitions for this similarity measure, thus leading to different biclusters of interest. Here we define biclusters as sub-matrices U of the original data matrix X such that the observations within U are different from the observations not contained in U with respect to the features in U. In other words, the choice of features influences which observations form the biclusters.

We can view clustering as a one-step partitioning method that partitions only the set of observations. Biclustering, on the other hand, is a two-step partitioning method that identifies partitions with respect to both features and observations. However, given a set of features, the problem of biclustering reduces to the problem of partitioning the observations with respect to this set of features, a problem which can be solved using conventional clustering methods. Thus, one may identify biclusters by identifying the features that define the biclusters and then clustering with respect to these features. In recent years several methods have been proposed for identifying features that define such clusters. In the first part of the dissertation, we will show how the "sparse clustering" method (29) may be used to identify biclusters under this framework. The proposed method can be used to detect biclusters with heterogeneous means and/or

variances as well as more complex differences.

1.1.1 An overview of competing biclustering approaches

As the association between objects and features are defined variously, different biclustering algorithms can lead to different results. One intuitive and commonly used method is to independently cluster rows and columns through a multivariate clustering method (9). This method can be further improved through simultaneous clustering, which is also known as co-clustering (12). Or, one can rearrange the rows and columns and search for rectangular blocks whose entries are on average large and positive (red) or large and negative (green) (28). All these criteria make applicable sense, depending on the specific data structures encountered. In this section, we briefly describe the biclustering methods to which we will compare our algorithms in the first part of the dissertation.

The Plaid method

Bicluster was first proposed in a way that the elements of a bicluster U should be well fitted by a two-way ANOVA model (6). Based on that, the Plaid algorithm was developed as an iterative procedure to approximate the data matrix X by a sum of sub-matrices whose entries follow two-way ANOVA models (15). For a given n $\times$ p data matrix, the Plaid model wants to achieve a small value of

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{p}(Y_{ij}-\theta_{ij0}-\sum_{k=1}^{K}\theta_{ijk}\rho_{jk}\kappa_{ik})^2, \tag{1.1}$$

Here Y_{ij} denotes the ijth entry within the data matrix; θ_{ijk} denotes the response shared by all features in the kth layer, with θ_{ij0} being the background layer; ρ_{jk} is 1 if feature

j is in the kth feature block and 0 otherwise; and κ_{ik} is 1 if object i is in the kth object block and 0 otherwise.

Built on two-way ANOVA models, the Plaid algorithm treats objects and features symmetrically, which is convenient and suitable for data structures where rows and columns are interchangeable. Biclusters identified by the Plaid algorithm have entries more similar to each other than to those outside, with respect to both objects and features under the two-way ANOVA model setting. Multiple biclusters can be detected by this method.

For the algorithm comparison in this biclustering study, the default setting of the Plaid algorithm was used, and datasets were feature/column scaled before running through the algorithm.

The Large Average Submatrix (LAS) method

As the data dimension increases, it is more difficult to analyse the data through simple two-way ANOVA models; and treating objects and features symmetrically by ignoring their natural relationship can be a waste of information in some situations. The LAS algorithm was proposed by defining biclusters U as sub-matrices with the averages of entries large and positive (red) or large and negative (green) (20) . The algorithm is based on an additive model where the data matrix X can be expressed as a sum of K constant sub-matrices with noise, as follows:

$$x_{ij} = \sum_{k=1}^{K} \alpha_k I(i \in A_k, j \in B_k) + \varepsilon_{ij}, \quad i \in [m],\, j \in [n], \tag{1.2}$$

where $A_k \subseteq [m]$ and $B_k \subseteq [n]$ are the row and column sets of the kth sub-matrix ([s] denotes the set of integers from 1 to s), $\alpha_k \in \mathbb{R}$ is the level of the kth sub-matrix, and ε_{ij} are independent $N(0,1)$ random variables. When $K = 0$, the model reduces to the simple null model where X is an $m \times n$ Gaussian random matrix. This null model leads

to a significance based score function for sub-matrices/biclusters. Specifically, the score assigned to a $k \times l$ sub-matrix U of X with average $\text{Avg}(U) = \tau > 0$ is defined by

$$S(U) = -\log\left[\binom{m}{k}\binom{n}{l}\Phi(-\tau)\sqrt{kl}\right]. \tag{1.3}$$

The score function (1.3) can be viewed as a significance measure of departure from the null model, which accounts for the size of the sub-matrix U and also the mean of the entries within U.

LAS algorithm also treats objects and features symmetrically, but instead of fitting two-way ANOVA models, the algorithm looks for signal blocks where the average entries within are more close to each other than to those outside, with respect to the numeric mean of the entries. The LAS algorithm has built in transformation functions recommended for certain data structures, which is convenient and useful. LAS algorithm is able to identify multiple biclusters within a given data set, and to output the numeric mean and associated LAS score as a reference of the statistical significance for each bicluster detected.

For the algorithm comparison in this biclustering study, default setting of the LAS method was applied, including the default data transformation if recommended by the method.

The Sparse singular value decomposition (SSVD) method

The SSVD method was developed as another alternative for biclustering, which searched for a low-rank, checkerboard structured matrix approximation to the original data matrix X (16). For example, we can obtain the rank-K approximation through

the first $K \leq \tau$ rank-one matrices of X, in the form of

$$\mathbf{X} = \mathbf{UDV}^T = \sum_{k=1}^{\tau} s_k \mathbf{u}_k \mathbf{v}_k^T \approx \mathbf{X}^{(k)} \equiv \sum_{k=1}^{K} s_k \mathbf{u}_k \mathbf{v}_k^T, \tag{1.4}$$

where τ is the rank of $\mathbf{X}$, $\mathbf{U} = (u_1, \ldots, u_\tau)$ is a matrix of orthonormal left singular vectors, $\mathbf{V} = (v_1, \ldots, v_\tau)$ is a matrix of orthonormal right singular vectors, $\mathbf{D} = \text{diag}(s_1, \ldots, s_\tau)$ is a diagonal matrix with positive singular values $s_1 \geq \ldots \geq s_\tau$ on its diagonal.

With the adaptive lasso penalty, the minimizing objective for SSVD can be expressed as

$$||\mathbf{X} - s\mathbf{uv}^T||_F^2 + s\lambda_u \sum_{i=1}^{n} w_{1,i}|u_i| + s\lambda_v \sum_{j=1}^{d} w_{2,j}|v_j|, \tag{1.5}$$

where s is a positive scalar, $\mathbf{u}$ is a unit n-vector, $\mathbf{v}$ is a unit d-vector, and $w_{1,i}$'s and $w_{2,i}$'s are data-driven weights, as explained in (30).

The sparsity-inducing penalties force the singular vectors to contain many zero entries, with the non-zero entries corresponding to objects and features that form the bicluster of interest. The SSVD method also treats objects and features symmetrically, and searches for a checkerboard structured matrix to approximate the original data matrix. The resulting approximation matrix has real value entries, with non-zero entries correspond to the biclusters of interest.

In the algorithm comparisons in this biclustering study, default setting for the SSVD method was applied. For result visualization on the figures, we transformed the resulting singular vectors to be $0/\pm 1$ based on the signs of the entries. In prediction accuracy and reproducibility comparisons, we further dichotomized the results into $0/1$ as we only cared about whether an object or feature was inside the sub-matrix U.

Heterogeneous SSVD (HSSVD) method

Based on the framework of SSVD, another group developed the HSSVD method to detect both mean and variance biclusters in the presence of unknown heterogeneous residual variance (5). HSSVD defines biclusters as subsets of the data matrix with the same mean and variance. It assumes a background layer where all the elements share a common mean and variance, and all the biclusters have rectangular structures with distinct mean or variance compared to the background layer. The HSSVD method assumption can be expressed in the form of a random effect model as follows

$$\mathbf{X} = \mathbf{\Xi} + \rho^2 \mathbf{\Sigma} \times \mathbf{\Phi} + b\mathbf{J}, \tag{1.6}$$

where $\mathbf{X}$ is the observed data, $\mathbf{\Xi} = (\xi_{ij})$ is an $n \times p$ matrix representing the signal, and $\mathbf{\Phi} = (\phi_{ij})$ is an $n \times p$ matrix with i.i.d. random components with mean 0 and variance 1. The heterogeneous variance signal is represented by this $n \times p$ matrix $\mathbf{\Sigma} = (\sigma_{ij})$, with ρ, a finite positive number serving as a common scale factor. $\mathbf{J}_{n \times p}$ is an $n \times p$ matrix with all values equal to 1, and this finite number b serves as a common location factor.

The HSSVD model makes the sparsity assumption that the majority of ξ_{ij} values are 0 and the majority of σ_{ij} values are 1, and the mean structure $\mathbf{\Xi}$ and the variance structure $\mathbf{\Phi}$ are both low rank. Beyond the scope of the entry mean differences, the HSSVD algorithm also studies the behavior of entry variances. It introduces a new concept of biclusters with heterogeneous variances, which are commonly encountered in some scientific research areas such as DNA methylation, where the methylation level influences the variance of the measurements.

For algorithm comparisons in this biclustering study, default setting for the HSSVD method was applied. Similar to the SSVD method, the resulting singular matrices $\mathbf{u}$

and $\mathbf{v}$ were transformed into $0/\pm 1$ for easier visualization in the figures, and further dichotomized into $0/1$ for prediction accuracy and reproducibility comparisons.

1.1.2 Sparse clustering and GAP statistic

Sparse clustering

The standard k-means clustering algorithm partitions a data set into k subcategories by maximizing the between cluster sum of squares (BCSS). The BCSS is calculated by taking the sum of the BCSS's for each individual feature. This implies that all the features are equally important. However, as we have discussed previously, in many situations the clusters differ with respect to only a fraction of the features, and these truly associated features do not necessarily contribute equally. In such situations, giving equal weights to all features when clustering may produce inaccurate results. This is especially true for HDLSS problems, where the number of the features is much bigger than the number of objects. To overcome this problem, the Tibshirani group proposed a novel clustering method which they called "sparse clustering" (29). Under sparse clustering where the original data matrix is with dimension $n \times p$, each feature is given a non-negative weight $w_j, j = 1, 2, \ldots, p$, and the following weighted version of the BCSS is maximized:

$$\begin{aligned} &\text{maximize}_{C_1,\ldots,C_K,\mathbf{w}} \left\{ \sum_{j=1}^{p} w_j \left(\frac{1}{n} \sum_{i=1}^{n} \sum_{i'=1}^{n} d_{i,i',j} - \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i,i' \in C_k} d_{i,i',j} \right) \right\} \\ &\qquad \text{subject to } ||\mathbf{w}||^2 \leq 1, ||\mathbf{w}||_1 \leq s, w_j \geq 0 \, \forall \, j. \end{aligned} \tag{1.7}$$

Here X_{ij} represents observation i for feature j of the data matrix X and $i \in C_k$ if and only if observation i belongs to cluster k. $d_{i,i',j}$ is a distance metric between any pair of observations in X with respect to feature j. For k-means clustering, we take

$d_{i,i',j} = (X_{ij} - X_{i'j})^2$. The L_1 bound on $\mathbf{w}$, s, is a tuning parameter, and can be either pre-specified by the customer or selected through certain procedure, which we will discuss in detail later.

They also describe an iterative procedure for maximizing (1.7) (29):

1. Initially let $w_1 = w_2 = \dots w_p$.

2. Maximize (1.7) with respect to $C_1, C_2, \dots, C_K$ by applying the standard k-means algorithm with the appropriate weights. In other words, apply the k-means algorithm where the dissimilarity between observations i and i' is defined to be $\sum_{j=1}^{p} w_j d_{i,i',j}$.

3. Maximize (1.7) with respect to the w_j's by letting

$$w_j = \frac{S(b_j, \Delta)}{\|S(b_j, \Delta)\|_2} \tag{1.8}$$

   Here b_j is the (unweighted) between cluster sum of squares for feature j and $S(x, y) = \text{sign}(x)(|x| - y)_+$ is a soft-threshold operator. Δ is chosen so that $\sum_j |w_j| = s$ ($\Delta = 0$ if $\sum_j |w_j| \leq s$). See (29) for the justification for (1.8).

4. Iterate steps 2 and 3 until the algorithm converges.

Note that (1.8) implies that as s increases, the number of nonzero w_j's decreases. Thus, for sufficiently small values of s, only a subset of the features contribute to the cluster assignments, and the magnitude of w_j represents the contribution of feature j to the clustering result. So, this method is useful in situations where the clusters differ with respect to only a subset of the features. We can further imagine that in extreme cases, when the tuning parameter s is selected properly and $K = 2$, we might be able to identify the contributing features and to group the observations into two sub-categories

based on these features, which leads to a bicluster with only a subset of observations and a subset of features.

A variant of this procedure can be used to perform sparse hierarchical clustering as well. In sparse hierarchical clustering, each feature is once again given a non-negative weight and the cluster hierarchy is constructed using these weighted features. The value of the weights again depends on a tuning parameter s, and some weights are forced to 0 when the tuning parameter is sufficiently small (29).

GAP statistic and tuning parameter selection

The GAP statistic was first proposed as a reference tool to select the optimal number of clusters (26). They defined W_k as the pooled within-cluster sum of squares around the cluster mean in the following manner:

$$W_k = \sum_{r=1}^{k} \frac{1}{2n_r} D_r, \tag{1.9}$$

where k is the number of clusters, n_r is the number of observations within the rth cluster, and $D_r = \sum_{i,i' \in C_r} d_{ii'}$ is the sum of the pairwise distances for all points in cluster r. The GAP statistic is further defined as

$$GAP_n(k) = E_n^* \log(W_k) - \log(W_k), \tag{1.10}$$

where E_n^* denotes the expect ion under a sample of size n from the reference distribution.

The sparse clustering method suggested the following algorithm to select the tuning parameter s from a set of candidate values based on the GAP statistic through independent observation permutation (29):

1. Obtain permuted data sets $X_1, \ldots, X_B$ by independently permuting the observations within each feature.

2. For each s, compute $O(s) = \sum_j w_j(\frac{1}{n}\sum_{i=1}^{n}\sum_{i'=1}^{n} d_{i,i',j} - \sum_{k=1}^{K}\frac{1}{n_k}\sum_{i,i'\in C_k} d_{i,i',j})$ as the objective obtained by performing sparse K-means clustering within tuning parameter value s on the original data set X, and $O_b(s)$ as the corresponding objective for the permuted data set X_b. The GAP statistic for a given value of s is then calculated as $GAP(s) = \log(O(s)) - \frac{1}{B}\sum_{b=1}^{B}\log(O_b(s))$.

3. The optimal tuning parameter s^* is chosen such that the corresponding $GAP(s^*)$ is the largest.

Through this algorithm, they want the GAP statistic to measure the strength of the clustering obtained on the real data relative to the clustering obtained on null data. And their argument for the independent observation permutation is that even though there may be strong correlation between the features in the original data set X, the feature in the permuted data sets $X_1, \ldots, X_B$ are uncorrelated with each other. And this uncorrelated structure is desired as in their reference null distribution, all the features should be uncorrelated and there should be no subgroups (e.g. clusters) on the object dimension (29).

1.2 Random forests and VIMP

1.2.1 Trees and random forests

A decision tree is a tree-shape structure for modelling decisions and their possible consequences. Each internal node on the tree represents a test or a decision rule of an attribute, each branch represents a possible outcome of the test, and each leaf node represents a class label, e.g. the decision taken after considering all attributes. Decision trees share the advantage of easy interpretation, and they are usually preferred

compared to parametric models when dealing with non-linear effects, arbitrary interactions, and missing values. However, it also has one embedded drawback as the lack of accuracy.

Random forests are a data mining method for classification that are based on simple decision trees (3), and can be used to evaluate the association between a response variable and a large number of predictors. By constructing multiple decision trees on randomly selected training sets and taking the average of the outputting class labels from individual trees, random forests are more accurate and reliable than simple decision trees (3). In order to build a decision tree in random forests, a bootstrap sample is first selected from the data, with the observations excluded from the bootstrap sample known as the "out of bag" (OOB) observations. A decision tree is then fit based on this bootstrap sample, using only a subset of features on each node. Repeat the process multiple times and take the average of the results, we will end up with a random forest model. Since the randomly drawn bootstrap samples are diverse in nature, the resulting decision trees are unstable, but their average result, e.g. the random forest model, will produce a more stable and accurate prediction of the modelling outcome. Specifically, the prediction accuracy was approved by smoothing the hard cut decision boundaries due to the splitting in single decision trees, which also reduced the variance of the prediction (4).

In high dimensional data setting, the VIMP scores have been suggested for random forest models in selecting associated predictor variables. In particular, the VIMP scores measure how much the predictive accuracy of the model is decreased when a given variable is measured with error (3). So, variables with higher VIMP scores are more likely to be associated with the modelling outcome. Although these VIMP scores are a useful tool, they have certain shortcomings. For example, our limited experience suggests that the predictors that are not associated with the outcome variable can have

high VIMP scores if they are strongly correlated with another predictor that is truly associated.

1.2.2 VIMP: reference for variable selection

As we have discussed earlier, there is no standard method for significance test in random forests, which means that we can not tell whether or not a random forest model is reliable, or which of the predictor variables are truly associated with the modelling outcome. In order to solve the problem, VIMP is proposed to serve as a reference for predictor significance.

When a single decision tree in random forests is calculated, the OOB samples can serve as the testing set and pass down the tree with the predictive accuracy recorded. Then the values for a given predictor variable are permuted in the OOB samples, and the predictive accuracy is recorded again. The variable importance of this predictor variable is the average decrease in accuracy over all trees (3). We can imagine that if a given predictor variable in the OOB sample is irrelevant to the modelling outcome, then the permutation will not change its predictive accuracy and its VIMP would be low. In this way, variables with high VIMP scores are more likely to be associated with the modelling outcome. The basic rationale of the conventional VIMP permutation is that by randomly permuting the predictor variable x_j, it is believed that its original association with the response variable y is broken. Then, when we use this permuted variable x_j together with the other non-permuted variables to modelling the response variable y, if the result does not change, it suggests that the permutation of x_j does not affect the model fitting, hence this variable x_j is not associated with y. In the later sections, we will refer to this conventional VIMP scores as the Breiman's VIMP scores.

However, these VIMP measures can only serve as references, not deterministic rules.

Even though we can rank the predictor variables based on their importance scores, due to the lack of standard testing procedure, it is difficult to decide how important is important, and we can not simply choose a cut off value for the VIMP scores and decide which predictor variables are associated with the response variable and should be included in the model. In practice, we still need standard statistical tests to evaluate the model fitting and the variables' statistical significance. On the other hand, VIMP scores may be inaccurate when the predictor variables are correlated. In particular, the inaccuracy may result from the preference of correlated predictor variables in early splits of decision trees, as well as the permutation scheme used in computing the permutation importance (23).

1.2.3 Conditional VIMP scores

As a reference measure, VIMP has an embedded bias towards correlated predictor variables, and we want to avoid this kind of false positive detection. In other words, we want to judge the importance of a certain predictor variable without the misleading influence from the other covariates. Note that the conventional VIMP score is calculated by sampling from the marginal distribution of x_j, and the correlation with other predictor variables will affect the result and possibly lead to false detection. So, in theory, if we can calculate VIMP by sampling from the conditional distribution of $x_j|X_{-j}$, we will be abel to avoid the influence due to variable correlation.

The concept of "conditional VIMP" is motivated by this idea of conditioning on all the other predictors while judging the variable importance (23). This new method did not fully solve the problem, but provided a possible alternative. In the later sections, we will refer to their conditional VIMP score as the Strobl's VIMP scores. Specifically, they propose a conditional permutation scheme as following:

1. Compute the before permutation oob-accuracy.

2. For all variables Z to be conditioned on, extract the cut-points that split this variable in the current tree and create a grid by bisecting the sample space in each cut-point.

3. Within the grid, permute the values of x_j and compute the after permutation oob-prediction accuracy.

4. Take the difference between the before and after permutation prediction accuracies as the importance score of variable x_j for one tree. And the conditional VIMP score of x_j for the forest is computed as the average over all trees.

In the random forests VIMP study, we will propose a different approach to obtain the conditional VIMP scores through the conditional distribution of a predictor variable, and compare the performance with the other VIMP scores.

1.2.4 Review on VIMP statistical test and variable selection

Along with the proposed Breiman's VIMP scores (3), they also suggested a simple significance test based on the normality of z-scores developed by scaling the permutation importance (3). In particular, the z-score for variable j is calculated as

$$z_j = \frac{VIMP_{Breiman}(x_{j0})}{\hat{\sigma}/\sqrt{ntree}}, \tag{1.11}$$

where $VIMP_{Breiman}(x_{j0})$ is the Breiman's VIMP score for variable j, $ntree$ is the number of trees in the forest, and $\hat{\sigma}$ is the observed standard deviation of the p VIMP scores. However, this test has some strange statistical properties (22) and might be questionable.

There are also other approaches for random forests variable selection. For example, backward elimination by throwing out least important variables until OOB prediction accuracy dropped (8); applying plots and significance test by randomly permuting the response values to mimic the overall null hypothesis that none of the predictor variable was relevant (7, 19). However, all of these approaches, together with the simple significance test as indicated by (1.11), were biased and had preference of correlated predictor variables (1).

In general, there were two basic strategies for random forest variable selection depending on different selection objectives, either to find important variables highly related to the response variable for interpretation purpose, or to find a small number of variables sufficient for parsimonious prediction of the response variables (11). Based on the two selection objectives, variables could be selected either by constructing nested random forest models and chose the one with the smallest OOB error, or constructing an ascending sequence of random forest models and chose the one by invoking and testing. However, they all require fitting of numerous random forest models and comparing certain model fit statistics, which are computationally intensive. Moreover, the concept of 'better interpretation' is somehow arbitrary and case specific.

Chapter 2

Biclustering via sparse clustering

2.1 Introduction

Unsupervised exploratory methods play an important role in the analysis of high-dimension low sample size (HDLSS) data, such as microarray gene expression data. Such data sets can be expressed in the form of a $n \times p$ matrix X, where each row corresponds to one observation each column corresponds to a feature. Unsupervised learning is a powerful tool for discovering interpretable structures within HDLSS data without reference to external information. In particular, clustering methods partition observations into subgroups based on their overall feature patterns. In many situations, these underlying subgroups may differ with respect to only a subset of the features. Such subgroups could be overlooked if one clusters using all the features.

Biclustering methods may be useful in situations where clusters are formed by only a subset of the features. Biclustering aims to identify sub-matrices U within the original data matrix X. The results may be visualized as two-dimensional signal blocks (after reordering the rows and columns) containing only a subset of the observations and features. For example, in a gene expression data set collected from cancer patients, there may exist a subset of genes whose expression levels differ among patients with a more aggressive form of cancer. Identifying such a bicluster may aid in the treatment of cancer patients.

We define biclusters as sub-matrices U of the original data matrix X such that the observations within U are different from the observations not contained in U with respect to the features in U. In other words, the choice of features influences which observations form the biclusters. In general, we can view clustering as a one-dimensional partitioning method that partitions only the set of observations. Biclustering, on the other hand, is a two-dimensional partitioning method that identifies partitions with respect to both features and observations. However, given a set of features, the problem of biclustering reduces to the problem of partitioning the observations with respect to this set of features, a problem which can be solved using conventional clustering methods. Thus, one may identify biclusters by identifying the features that define the biclusters and then clustering with respect to these features. In recent years several methods have been proposed for identifying features that define such clusters. We will show how the "sparse clustering" method (29) may be used to identify biclusters under this framework. The proposed method can be used to detect biclusters with heterogeneous means and/or variances as well as more complex differences. We compare our algorithms with some other existing biclustering approaches by applying the methods to a series of simulation studies and real data sets.

2.2 Methods

2.2.1 Sparse Clustering

The standard k-means clustering algorithm partitions a data set into k sub-categories by maximizing the between cluster sum of squares (BCSS). The BCSS is calculated by taking the sum of the BCSS's for each individual feature. This implies that all features are equally important. However, in many situations the clusters differ with respect to only a fraction of the features. In such situations, giving equal weight to all features

when clustering may produce inaccurate results. This is especially true for HDLSS problems. To overcome this problem, (29) proposed a novel clustering method which they called "sparse clustering." Under sparse clustering, each feature is given a non-negative weight w_j, and the following weighted version of the BCSS is maximized:

$$\text{maximize}_{C_1,\ldots,C_K,\mathbf{w}} \left\{ \sum_{j=1}^{p} w_j \left(\frac{1}{n} \sum_{i=1}^{n} \sum_{i'=1}^{n} d_{i,i',j} - \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i,i' \in C_k} d_{i,i',j} \right) \right\} \tag{2.1}$$
$$\text{subject to } ||\mathbf{w}||^2 \leq 1, ||\mathbf{w}||_1 \leq s, w_j \geq 0 \, \forall \, j.$$

Here X_{ij} represents observation i for feature j of the data matrix X and $i \in C_k$ if and only if observation i belongs to cluster k. $d_{i,i',j}$ is a distance metric between any pair of observations in X with respect to feature j. For k-means clustering, we take $d_{i,i',j} = (X_{ij} - X_{i'j})^2$.

They also describe an iterative procedure for maximizing (2.1) within the sparse clustering algorithm (29):

1. Initially let $w_1 = w_2 = \ldots w_p$.

2. Maximize (2.1) with respect to $C_1, C_2, \ldots, C_K$ by applying the standard k-means algorithm with the appropriate weights. In other words, apply the k-means algorithm where the dissimilarity between observations i and i' is defined to be $\sum_{j=1}^{p} w_j d_{i,i',j}$.

3. Maximize (2.1) with respect to the w_j's by letting

$$w_j = \frac{S(b_j, \Delta)}{\|S(b_j, \Delta)\|_2} \tag{2.2}$$

   Here b_j is the (unweighted) between cluster sum of squares for feature j and $S(x, y) = \text{sign}(x)(|x| - y)_+$ is a soft-threshold operator. Δ is chosen so that $\sum_j |w_j| = s$ ($\Delta = 0$ if $\sum_j |w_j| \leq s$). See (29) for the justification for (2.2).

4. Iterate steps 2 and 3 until the algorithm converges.

Note that (2.2) implies that as s increases, the number of non-zero w_j's decreases. Thus, for sufficiently small values of s, only a subset of the features contribute to the cluster assignments, so this method is useful in situations where the clusters differ with respect to only a subset of the features.

A variant of this procedure can be used to perform sparse hierarchical clustering. In sparse hierarchical clustering, each feature is once again given a weight and the cluster hierarchy is constructed using these weighted features. The value of the weights again depends on a tuning parameter, and some weights are forced to 0 when the tuning parameter is sufficiently small (29).

2.2.2 Biclustering Via Sparse Clustering

As described earlier, the objective of biclustering is to identify submatrices U of a data matrix X such that the observations containing in U differ from the observations not contained in U with respect to the features contained in U. One possible strategy to identify such biclusters is to apply 2-means sparse clustering. One could define the observations of U to be the observations in the smaller cluster identified by the procedure and the features in U to be the features with non-zero weights.

The list of features with non-zero weights depends on the tuning parameter s, so this approach to biclustering requires one to choose the correct value of this tuning parameter. One possible approach for choosing s is described in the sparse clustering algorithm (29), but in our experience it tends to give non-zero weights to too many features. Thus, we propose an alternative method for identifying the features that belong to the bicluster. First, note that if sparse clustering is applied with $s = \sqrt{p}$, then no soft thresholding will be performed on the weights and all weights be non-zero. The motivation for our method is the following: Suppose that sparse 2-means

clustering is applied with $s = \sqrt{p}$. Let $w_{(1)}, w_{(2)}, \ldots, w_{(p)}$ denote the weights produced by the sparse clustering procedure, and let $w_{(1)_0}, w_{(2)_0}, \ldots, w_{(p)_0}$ denoted the expected values of the weights under the null hypothesis that no bicluster exists. If this null hypothesis is true, then we would expect that $w_{(j)} \approx w_{(j)_0}$ for all j. However, if the first m features form a bicluster, then we would expect that $w_{(j)} > w_{(j)_0}$ for $j \leq m$ and $w_{(j)} < w(j)_0$ otherwise.

Thus, our proposed biclustering method is described below:

1. Apply the 2-means sparse clustering algorithm with $s = \sqrt{p}$ to obtain clusters C_1 and C_2 and weights $w_1, w_2, \ldots w_p$.

2. Perform a Kolmogorov-Smirnov test of the null hypothesis that the distribution of $w_1, w_2, \ldots, w_p$ is the same as the expected distribution of the weights under the null hypothesis of no clusters.

3. If the test in Step 2 fails to reject the null hypothesis, then terminate the procedure and report that no biclusters were identified.

4. If the test in Step 2 rejects the null hypothesis, then let

$$m = \arg\max_j \left(w_{(p-j+1)} - w_{(p-j+1)_0}\right) - \left(w_{(p-j)} - w_{(p-j)_0}\right) \tag{2.3}$$

5. Return a bicluster containing the m features with the largest weights and the observations belonging to either C_1 or C_2 (whichever is smaller).

We recommend that the data matrix be normalized such that all features have mean 0 and standard deviation 1 before applying the procedure.

Let b_j denote the between cluster sum of squares for feature j. Suppose that the mean of the observations in C_1 is $\mu_{1,j}$ and the mean of the observations in C_2 is $\mu_{2,j}$.

Then it is easy to verify that

$$E(b_j) = 1 + np(1-p)(\mu_{1,j} - \mu_{2,j})^2 \tag{2.4}$$

where p is the probability that a given observation belongs to C_1. This implies that $E(b_j) = 1$ if $\mu_{1,j} = \mu_{2,j}$, which would be the case of feature j does not belong to the bicluster. However, if $\mu_{1,j} \neq \mu_{2,j}$, then $E(b_j)$ will increase as n increases. Thus, assuming that $\mu_{1,j} \neq \mu_{2,j}$ for at least one j (which will always be the case when a bicluster exists), (2.2) and the law of large numbers implies that $w_j \to 0$ as n increases for all j such that $\mu_{1,j} = \mu_{2,j}$ (i.e., all j that do not belong to the bicluster). This indicates that the criteria (2.3) is consistent for selecting the features that belong to the bicluster, assuming that C_1 and C_2 are correctly identified and the conditions of the law of large numbers are satisfied.

One may wish to identify secondary biclusters in a data set after identifying a primary bicluster. One simple approach to identify such secondary biclusters is described below:

1. Identify a primary bicluster U_1 as described above.

2. Define a matrix X' as follows:

$$x'_{ij} = \begin{cases} x_{ij} & \text{if } x_{ij} \notin U_1 \\ x_{ij} - \bar{X}_{U_1,j} + \bar{X}_{U'_1,j} & \text{if } x_{ij} \in U_1 \end{cases} \tag{2.5}$$

   Here $\bar{X}_{U_1,j}$ denotes the sample mean of the jth feature of U_1 and $\bar{X}_{U'_1,j}$ denotes the sample mean of the jth feature of the elements of X that are not in U_1.

3. Apply the biclustering algorithm to the matrix X'.

The above procedure may be repeated as many times as desired to identify multiple

biclusters in the same data sets (although the procedure should be terminated if it fails to reject the null hypothesis that no biclusters exist in Step 2).

2.2.3 Estimating the Null Distribution of the Weights

This method requires one to know the expected order statistics of the weights under the null hypothesis that no clusters exist. If this distribution is unknown, it may be approximated as follows:

1. Apply the 2-means sparse clustering algorithm with $s = \sqrt{p}$ to obtain clusters C_1 and C_2, as before.

2. Fix C_1 and C_2 and permute the rows of X to calculate weights $w_1^*, w_2^*, \ldots, w_p^*$.

3. Repeat Step 2 B times.

4. Approximate $w_{(j)_0}$ as $w_{(j)_0} = \sum_k w_{(j)k}^*/B$, where $w_{(j)k}^*$ represents the jth order statistic of the weights from the kth iteration of Step 2.

This procedure will provide an estimate of the expected values of the order statistics of the weights, but it is very expensive computationally for large data sets. It would be desirable to develop a faster alternative. Fortunately, if the sparse clustering procedure is modified slightly, the exact distribution of the weights can be calculated under mild assumptions.

First, note that the criterion in (2.1) can be written as $\sum_j w_j b_j$, where b_j is the between cluster sum of squares for feature j. If we modify the procedure to minimize $\sum_j w_j \sqrt{b_j}$ rather than $\sum_j w_j b_j$, then (2.2) implies that the optimal w_j's are given by

$$w_j = \frac{\sqrt{b_j}}{\sqrt{\sum_k b_k}} \tag{2.6}$$

assuming $s = \sqrt{p}$ (implying that $\Delta = 0$ in (2.2)). Now under the null hypothesis that no clusters exist, there is no difference in the means of the observations in C_1 and C_2 for all features, implying that $b_j \sim \chi^2_1$ for all j. Thus, (2.6) implies that w_j^2 has a Beta$(1/2, (p-1)/2)$ distribution. Thus, if we use this criterion to select the clusters, we can test the null hypothesis that no bicluster exists by performing a Kolmogorov-Smirnov test of the null hypothesis that the w_j^2's have a Beta$(1/2, (p-1)/2)$ distribution. Similarly, in (2.3), $w_{(j)_0} = E(\sqrt{B_{(j)}})$, where $B \sim$ Beta$(1/2, (p-1)/2)$. Although there is no simple closed form expression for $E(\sqrt{B_{(j)}})$, it can be easily approximated numerically. We will use this method to approximate the null distribution of the weights in all subsequent examples unless otherwise noted.

2.2.4 Variance Biclustering and Other Variations

Note that sparse 2-means clustering is only used in the initial step of our biclustering procedure. In principle any clustering procedure that produces two clusters could be used in place of sparse 2-means clustering. Sparse 2-means clustering is an obvious choice to identify putative biclusters since it is designed to identify clusters that differ with respect to only a subset of the features. However, in some situations it may be desirable to use a different clustering procedure to identify the putative biclusters.

One important application where it may be useful to use an alternative clustering procedure is variance biclustering. The biclustering method described in Section 2.2.2 is designed to identify biclusters whose mean differs from the mean of the observations that do not belong to the bicluster. In some situations, however, one may wish to identify biclusters that have unusually high (or low) variance compared to observations that are not in the bicluster. For example, when analysing DNA methylation data, biclusters that exhibit high variance may reveal possible functional regions in the genome.

To identify variance biclusters, we propose the following simple modification of 2-means clustering in order to identify clusters whose variances differ from one another:

1. Initially assign each observation to each cluster 1 or cluster 2.

2. For $i = 1, 2, \ldots, n$, move observation i from cluster 1 to cluster 2 (or from cluster 2 to cluster 1) if
$$\sum_{j=1}^{p} \log(|s^2_{j,C_1} - s^2_{j,C_2}| + 1) \tag{2.7}$$
is increased after moving the observation to the other cluster. Here s_{j,C_k} represents the standard deviation of feature j for the observations in cluster k.

3. Repeat Step 2 until the procedure converges.

Note that we did not specify how the initial cluster assignments in step 1 were performed. The simplest approach is to simply assign each observation to a cluster randomly. An alternative approach is to calculate the variance of the data for each observation across the features. The observations are then partitioned based on their variances: half of the observations with the largest variances are initially assigned to cluster 1 and the other half of the observations (with the smallest variances) are initially assigned to cluster 2. Our preliminary work suggests that both approaches produce comparable results but the latter approach tends to be faster, so we will use this approach in all subsequent examples.

Also, note that this procedure can be easily modified to consider feature weights by replacing (2.7) with
$$\sum_{j=1}^{p} w_j \log(|s^2_{j,C_1} - s^2_{j,C_2}| + 1) \tag{2.8}$$
A sparse version of this algorithm (motivated by the sparse clustering algorithm) is also possible, as described below:

1. Initially let $w_1 = w_2 = \ldots w_p$.

2. Maximize (2.8) with respect to C_1 and C_2 by applying the above procedure with the appropriate weights.

3. Maximize (2.8) with respect to the w_j's by letting

$$w_j = \frac{S(b_j, \Delta)}{\|S(b_j, \Delta)\|_2} \tag{2.9}$$

   where $b_j = \log(|s^2_{j,C_1} - s^2_{j,C_2}| + 1)$.

4. Iterate steps 2 and 3 until the algorithm converges.

By replacing 2-means sparse clustering with the procedure described above, the biclustering algorithm described in Section 2.2.2 can be used to identify variance biclusters. If one wishes to identify secondary variance biclusters, one may define a matrix X' as follows:

$$x'_{ij} = \begin{cases} x_{ij} & \text{if } x_{ij} \notin U_1 \\ \frac{x_{ij}\sigma_{U'_1,j}}{\sigma_{U_1,j}} & \text{if } x_{ij} \in U_1 \end{cases} \tag{2.10}$$

where $\sigma_{U_1,j}$ denotes the standard deviation of the jth feature of U_1 and $\sigma_{U'_1,j}$ denotes the standard deviation of the jth feature of the elements of X that are not in U_1.

Note that this procedure requires an estimate of the null distribution of the w_j's. This null distribution may be estimated by permuting the rows of X as described in Section 2.2.3. Alternatively, one can take advantage of the fact that $n_1 s^2_{j,C_1} \sim \chi^2_{n_1}$ and $n_2 s^2_{j,C_2} \sim \chi^2_{n_2}$ for all j under the null hypothesis of no variance biclusters, where n_1 and n_2 are the number of observations in C_1 and C_2, respectively. The null distribution of the b_j's (and hence the w_j's) can be estimated by simulating chi-square random variables and calculating the w_j's for each set of simulated values. We will use this method to approximate the null distribution in all examples in this manuscript, since the permutation-based approach is much slower.

Other variations of this biclustering procedure are possible. For example, rather than using sparse 2-means clustering to identify the putative biclusters in the first step of the procedure, one could use some form of hierarchical clustering and then partition the cluster hierarchy into two clusters. We will provide a simulated example below where applying hierarchical clustering with single linkage to identify the biclusters produces better results than sparse 2-means clustering.

2.2.5 Existing Biclustering Methods

A variety of biclustering methods have been proposed. One simple and commonly used approach is to independently apply hierarchical clustering to both the rows and columns of a data set (9). Several improvements of this simple approach have been proposed (12, 28). Other biclustering methods directly search for submatrices U such that the mean of the observations in U is higher than the mean of the observations not in U. The "Plaid" method approximates a data matrix X as a sum of submatrices whose entries follow two-way ANOVA models (15). At each step of the procedure, the algorithm searches for a submatrix that maximizes the reduction in the overall sum of squares. Similarly, the "Large Average Submatrix" (LAS) method assumes that the data matrix can be expressed as a sum of constant submatrices plus Gaussian noise (20). These submatrices are identified using an iterative search procedure. Also, the "sparse biclustering" method assumes that the n observations belong to K unknown and non-overlapping classes, and the p features belong to R unknown and non-overlapping classes (25). The mean value of all the features in each class is assumed to be the same. Class labels are obtained by maximizing the log likelihood, and sparsity is obtained by imposing an ℓ_1 penalty on the log likelihood.

Other methods for identifying biclusters utilize the singular value decomposition

(SVD) of the data matrix. The SSVD method searches for a low-rank "checkerboard-structured" approximation for a data matrix by calculating a weighted form of the SVD (16). An adaptive lasso penalty (30) is applied to the weights, forcing both the left and right singular vectors to be sparse. The non-zero entries in the resulting (sparse) singular vectors correspond to the observations and features forming the bicluster. One generalization of this method is called "Heterogeneous Sparse Singular Value Decomposition" (HSSVD) (5). HSSVD approximates the data as the sum of a "mean layer" and a "variance layer" (plus random noise) and identifies biclusters in these two layers. The inclusion of a "variance layer" allows one to identify variance biclusters as well as mean biclusters.

While these methods have been useful for many problems, they have certain shortcomings. As we will demonstrate below, they may fail to identify biclusters in simple simulations. Also, with the exception of the HSSVD method, these existing methods can only identify biclusters whose means differ from the observations not in the bicluster. (HSSVD can also identify biclusters whose variances differ.) However, biclustering methods based on the SVD have other shortcomings. These methods can identify the presence of biclusters but cannot determine which observations and features belong to the bicluster without using arbitrary cut-offs.

2.2.6 Evaluating the Reproducibility of Biclusters

We propose an intuitive method to evaluate the reproducibility of the biclusters identified by each method. We randomly partition the original data matrix X into two submatrices X_1 and X_2, each of which contains half of the observations. Denote the primary bicluster identified within X as U, and let U_1 and U_2 be the primary biclusters within X_1 and X_2, respectively. We treat U as the reference or the "correct" bicluster, and record four rates: **1)** The percentage of observations that are misclassified (i.e.

the percentage of observations that are either in U_1/U_2 but not U or in U but not in U_1/U_2); **2)** The percentage of false negatives (i.e. the average percentage of features in U that are not in U_1/U_2); **3)** The percentage of false positives (i.e. the average number of features in U_1/U_2 that are not in U); and **4)** The percentage of features that are misclassified (i.e. features that are identified as significant on sub-matrix U_1 but not U_2, or vice versa). We repeated the procedure 10 times on each simulated dataset and averaged over the 10 iterations.

2.2.7 Computational Details

In the later sections, we will compare our proposed biclustering algorithm with several existing methods, specifically Plaid, LAS, SSVD, and HSSVD. The Plaid algorithm was implemented in the R package "biclust." The default setting was used, and the data sets were feature/column scaled before running through the algorithm. The LAS algorithm is available at https://genome.unc.edu/las/. The default settings were used, including the data transformation step if recommended by the method. The SSVD functions are available at http://www.unc.edu/∼haipeng/, and the HSSVD functions can be found at http://impact.unc.edu/impact7/HSSVD. Again, the default settings were used for both methods. For easy visualization of these two SVD based methods, we transformed the resulting singular vectors/matrices to be $0/\pm 1$ based on the signs of the entries when making the plots. When comparing the prediction accuracy and reproducibility of these methods, we further dichotomized the results as 0/1, since we only care about whether an object or feature is inside the sub-matrix U. The sparse biclustering algorithm was implemented using the "sparseBC" R package with the default settings. We used $K = 2$ and $R = 2$ to force the sparse biclustering algorithm to identify only a single bicluster so that its results can be compared with other methods that identify one bicluster at a time. Our proposed method was implemented using a

modified version of the "sparcl" R package. All calculations were performed using a single core of a 2.66 GHz Intel Core 2 Quad processor on a Linux-based system.

2.3 Results

2.3.1 Simulation Studies

We first evaluated the performance of our method on a variety of simulated data sets and compared its performance with the biclustering methods described in Section 2.2.5. The methods were compared with respect to computing time, prediction accuracy, and reproducibility (defined in Section 2.2.6). To evaluate the prediction accuracy when identifying a single bicluster, we compared three rates: observation misclassification rate, feature false positive rate (FPR), and feature false negative rate (FNR).

For the sequential biclustering simulations (simulations 3 and 5), the identification of the current bicluster depends on all the biclusters identified previously and there is no "correct" sequence for identification. Thus, we recorded the prediction accuracy in a different manner. Specifically, when there existed two biclusters, the reasonable result would be the identification of either bicluster 1 or 2, or a larger bicluster that covers both the signal blocks, which will be referred to as "bicluster 1+2." For each simulation, we determined which of the three biclusters was identified by each method. For each method, we recorded the percentage of simulations when each of the three possible biclusters was identified. Also, instead of comparing the mismatch rates for observations and features separately, we recorded the FPR and FNR of the entries. The reproducibility analysis described in Section 2.2.6 was only performed on simulation studies 1, 2, and 4 for computational reasons.

We also compare the performance of the stopping rule in these methods on simulation 1 and 3, by recording the total number of biclusters identified by each method.

Primary Bicluster Identification

In this study, each simulated data set contained four non-overlapping bicluster signals generated from normal distributions, and the comparison was focused on identifying the primary bicluster. Each simulated data set comprised a 100×200 matrix with independent entries where each column represents a feature and each row represents an observation. The background entries followed a standard normal distribution with mean 0 and standard deviation 1. We denote the distribution as $N(0, 1)$, where $N(a, b)$ represents a normal random variable with mean a and standard deviation b. The four non-overlapping rectangular shaped biclusters were constructed in the following manner: bicluster 1, consisting of observations 1-20 and features 1-20 (denoted as [1-20, 1-20]) added a $N(2, 1)$ layer to the background, bicluster 2 [16-30, 51-80] added a $N(3, 1)$ layer to the background, bicluster 3 [51-90, 61-130] added a $N(3, 1)$ layer to the background, and bicluster 4 [66-100, 151-200] added a $N(2, 1)$ layer to the background. Bicluster 3 was the primary bicluster, since it was the largest bicluster and had the largest mean difference from the background, so we expected the algorithms to detect this bicluster as the first layer. Figure 2.4 shows the biclustering results from one of the simulations. Under the given data structure, the Plaid algorithm failed to identify any biclusters for all the simulations. Each simulated data set was partitioned as described in Section 2.2.6 to evaluate the reproducibility of the biclusters.

Departure from Normality

In this study, we simulated data sets with four non-overlapping bicluster signals similar to the data sets that were simulated in Section 2.3.1. The main difference is that the data were generated from Cauchy distributions with infinite moments. Each simulated data set comprised a 100×200 matrix with independent entries. The background entries followed a Cauchy distribution with location shift 0 and scale 1. We denote

the distribution as Cauchy$(0, 1)$, where Cauchy(a, b) represents a Cauchy random variable with location shift a and scale b. The four non-overlapping rectangular shaped biclusters were constructed in the following manner: bicluster 1 [1-20, 1-20] added a Cauchy$(75, 1)$ layer to the background, bicluster 2 [16-30, 51-80] added a Cauchy$(50, 1)$ layer to the background, bicluster 3 [51-90, 71-110] added a Cauchy$(200, 1)$ layer to the background, and bicluster 4 [71-100, 156-200] added a Cauchy$(75, 1)$ layer to the background. Bicluster 3 was the primary bicluster, and we expected the algorithms to detect this bicluster as the first layer. Figure 2.4 compares the biclustering results of each method for one of the simulations. SSVD had valid identification for 37 out of the 100 simulations, Plaid method had 62, and sparse biclustering had 13. For these three methods, the performance was averaged only on the simulations where they had valid results. For a 'valid result', we mean that the bicluster contains at least 2 observations and 2 features. Each simulated data set was partitioned as described in Section 2.2.6 to evaluate the reproducibility of the biclusters.

Sequential Biclusters with Overlap

In this study, we simulated datasets with overlap between two biclusters. Each simulated data set comprised of two layers, each of which was a 100×200 matrix with independent entries. The background data (i.e., observations that do not belong to the bicluster) were $N(0, 0.5)$. The first layer contained a bicluster [1-40, 1-40] generated from $N(7, 2)$, and the second layer contained a bicluster [21-60, 21-60] generated from $N(-5, 3)$. The final data set was the sum of the two layers. Note that observations 21-40 and features 21-40 are contained in both biclusters. Plaid method had 98 valid identification results for 98 out of the 100 simulations, and its performance was averaged over the 98 simulations. Figure 2.4 shows the biclustering results from one of the simulations. Reproducibility of the biclusters was not evaluated for this simulation

scenario.

Non-Spherical Biclusters

Most existing biclustering methods seek to maximize the Euclidean distance between the center of the putative bicluster and the remaining data values. This assumes that the biclusters are approximately "spherical" (in the appropriate number of dimensions). Although this assumption is reasonable in many situations, it can cause these methods to fail if the assumption is violated. See Figure 2.4 for an example of non-spherical clusters in the case of two dimensions. Hierarchical clustering (with single linkage) will do a better job of identifying clusters similar to the clusters in Figure 2.4 than k-means clustering (which also assumes that the clusters are spherical). One strength of SC-Biclust is the fact that it can use clustering methods other than 2-means clustering to identify biclusters (see Section 2.2.4). Thus, it is reasonable to expect that SC-Biclust (with single linkage hierarchical clustering) will outperform competing biclustering methods when the biclusters are non-spherical.

The purpose of this study was to provide an example where SC-Biclust using hierarchical clustering can identify biclusters that existing biclustering methods would fail to identify. Each 1200×75 data set was simulated as follows. For $1 \leq j \leq 25$:

$$X_{i,2j} = -2I(i \leq 500) + 5\sin(\theta_i + \pi I(i > 500)) + \epsilon_i$$

$$X_{i,2j-1} = 5I(i \leq 500) + 5\cos(\theta_i + \pi I(i > 500)) + \epsilon_i$$

Here the ϵ_i's are iid $N(0, 0.2)$ and the θ_i's are iid Uniform$(0, \pi)$. For all $j > 50$, the X_{ij}'s are $N(0, 1)$. Figure 2.4 shows the biclustering results from one of the simulations. Each simulated data set was partitioned as described in Section 2.2.6 to evaluate the reproducibility of the biclusters.

Variance Biclustering

Another limitation of most existing biclustering methods is that they are only capable of detecting biclusters whose mean values differ from the data points not in the bicluster. In some situations, however, one may wish to identify biclusters with higher (or lower) variance than the data points not contained in the bicluster. As described in Section 2.2.4, SC-Biclust can be modified to identify biclusters with heterogeneous variance. The goal of this simulation is to evaluate the ability of SC-Biclust to identify such biclusters. We simulated data sets with two non-overlapping biclusters with heterogeneous variances. Each simulated data set consisted of a 150×500 matrix with independent entries. The background entries were all $N(1, 2)$. The first bicluster [1-30, 1-200] was generated as $N(1, 15)$, and the second bicluster [31-50, 201-400] was generated as $N(1, 5)$. Figure 2.4 shows the biclustering results from one of the simulations. The prediction accuracy of the methods were evaluated in the same way as the third simulation scenario, and no reproducibility was assessed.

Simulation Results

We simulated 100 data sets with the same structure for each simulation scenario. Table 2.1 shows the average computing time for each method for each simulation scenario. Tables 2.2 and 2.3 show the prediction accuracy and the number of valid predictions out of 100, Table 2.4 shows the reproducibility results for simulations 1, 2, and 4, and Table 2.5 shows the stopping rule comparison for simulation 1 and 3.

SC-Biclust performed very well in the first simulation scenario. No observations were misclassified across all 100 simulations and the proportion of features that were misclassified was also very low. The reproducibility of the biclusters identified by SC-Biclust was also very good. The sparse biclustering method also produced good results, except for the relatively high feature misclassification rate in the reproducibility

analysis. SSVD, HSSVD, and LAS tended to include spurious features in the bicluster (as evidenced by their higher FPR), and Plaid did not select any features for this simulation scenario. The results of the second simulation scenario were similar. SC-Biclust assigned both the correct observations and features to the bicluster with nearly perfect accuracy and produced a noticeably lower error rate than competing methods. It is noteworthy that the performance of SC-Biclust decreased only slightly when the data was non-normal whereas other methods (particularly SSVD, HSSVD, and sparse biclustering) performed much worse. It is interesting to note that sparse biclustering performed very poorly in this simulation despite nearly perfect performance in the first simulation, indicating that it is not robust to departures from normality in the data.

In the third simulation scenario, SC-Biclust identified both biclusters with perfect accuracy in all the simulations. LAS also identified the first bicluster with high accuracy but it tended to include many spurious entries when identifying the second bicluster. SSVD and HSSVD tended to identify bicluster 1+2 (combining the two biclusters into one), and the performance of Plaid was poor. The sparse biclustering method identified single biclusters, but with very high false negative rates.

SC-Biclust had a much lower proportion of misclassified observations in the fourth simulation scenario and excellent reproducibility. This is not surprising, since the other biclustering methods assume that the biclusters are spherical, and this assumption is violated for this simulation. However, these results illustrate that SC-Biclust can be used to identify biclusters in situations where existing methods will fail.

In the fifth simulation scenario, SC-Biclust identified the first variance bicluster with high accuracy. It usually detected the second variance bicluster as well, although many of the entries were false negatives. HSSVD tended to identify bicluster 1+2, with higher FNR and FPR than SC-Biclust. The other methods performed poorly, which is not surprising, since they are not designed to identify variance biclusters.

In terms of computing time, SC-Biclust was generally faster than HSSVD and LAS but slower than SSVD, Plain, and sparse biclustering. This was true across almost all five simulation scenarios.

The performance of the stopping rules were compared on simulation 1 and 3 only. For the SC-Biclust method, the maximum number of biclusters to be identified for simulation 1 was set to be 7, and it was set to be 5 for simulation 3. Recall that the Plaid method failed to detect any bicluster on simulation 1, so it was excluded on simulation 1 comparison. And it failed 2 times on simulation 3, meaning that the bicluster identified contained a single observation or/and a single feature, so it was included on simulation 3 comparison and the 'invalid' biclusters also count. Since the SSVD method only identified a single layer for all simulation scenario regardless of the data structure, it was excluded for the stopping rule comparison.

2.3.2 Analysis of OPPERA data

OPPERA is a prospective cohort study on Temporomandibular Disorders (TMD), which are a set of painful conditions that affect the jaw muscles, the jaw joint, or both. Both TMD-free participants and chronic TMD patients were enrolled in the study. Each study participant completed a quarterly questionnaire, and participants who showed signs of first-onset TMD returned to the clinic for a formal examination. The median follow up period was 2.8 years. The data set contained 185 chronic TMD patients and 3258 initially TMD-free individuals, 260 of whom developed TMD before the end of the study. Among the TMD-free individuals, 521 did not complete any follow up questionnaires and were excluded from the analysis. The remaining 2737 were used for survival analysis in the later sections, where development of first-onset TMD is the event of interest. For a more detailed description of the OPPERA study, see (21) or (2).

Three sets of possible risk factors for TMD were measured in OPPERA, including autonomic measurements like blood pressure and heart rate (44 total variables), psychosocial measurements like depression and anxiety (39 total variables), and quantitative sensory testing (QST) measurements (33 total variables) that evaluate participants' sensitivity to experimental pain. See (10), (13), and (17) for more detailed descriptions of these variables.

The SC-Biclust algorithm identified 3 significant biclusters within the OPPERA data set. The first bicluster contained 30 measures of autonomic function, the second bicluster contained 29 measures of psychological distress, and the third bicluster contained 6 measures of pain sensitivity. There were no overlap in the features selected in the three biclusters. Thus, the biclusters identified by SC-Biclust were consistent with the known structure of the data set. The biclusters identified by the other methods did not correspond to the three different types of measurements known to exist in this data set. See Table 2.6 for a summary of the results.

Membership in the biclusters identified by each method of interest was evaluated as a possible risk factor for both chronic TMD and first-onset TMD. (Subjects with chronic TMD were excluded from the analysis for first-onset TMD.) The association between each bicluster and chronic TMD is shown in Table 2.7, and the association between each bicluster and first-onset TMD is shown in Table 2.4. Kaplan-Meier plots for first-onset TMD for selected biclusters are shown in Figure 2.4. All three biclusters identified by SC-Biclust were associated with chronic TMD. The second bicluster was also associated with first-onset TMD. The second and third biclusters identified by LAS were associated with first-onset TMD, and the second bicluster was also associated with first-onset TMD. The remaining biclusters were associated with neither chronic TMD nor first-onset TMD. SC-Biclust was faster than HSSVD and LAS but slower than SSVD and Plaid. The sparse biclustering algorithm failed to detect any biclusters.

2.3.3 Analysis of a breast cancer gene expression dataset

The data set used in this section contains gene expression measurements on 4751 genes from a total number of 78 breast cancer subjects. The survival time of each subject is also available. See (27) for a more detailed description of this data set.

The primary bicluster identified by the SC-Biclust algorithm contains 16 subjects and 8 features. The primary bicluster identified by the LAS algorithm contains 16 observations and 1421 features. Interestingly, the 16 observations identified by SC-Biclust and LAS are exactly the same. The primary bicluster identified by the sparse biclustering method contains 60 observations and 553 features. HSSVD method identified 8 mean bicluster layers and 3 variance bicluster layers, for which we will only study the primary mean layer. The Plaid method failed to identify any bicluster within the dataset; and the SSVD method and the HSSVD variance identification resulted in no observation cluster. Detailed biclustering results are provided in Table 2.9.

We tested the null hypothesis of no association between each putative bicluster and survival using log rank tests. Table 2.9 and Figure 2.4 show the associations between survival and the biclusters identified by SC-Biclust, HSSVD (mean layer only), LAS, and sparse biclustering. The putative biclusters identified by SC-Biclust, LAS, and sparse biclustering were associated with survival, but the putative bicluster identified by HSSVD was not. The running time for SC-Biclust was also significantly lower than the running time of the other methods.

2.3.4 Analysis of methylation data

We applied SC-Biclust (and existing biclustering methods) to a methylation data set comparing cancer patients with normal patients. Methylation data were evaluated at 384 different cancer-specific differentially methylated regions (cDMRs) for 138 normal samples and 152 cancer samples. Details of the data set are described in (14), who

reported that the cancer samples had hyper variability in certain cDMRs compared to controls.

We first applied the SC-Biclust algorithm to identify two mean biclusters and then used the residual matrix for variance bicluster identification, as described in Section 2.2.4. We chose the top two variance biclusters for comparison with the other methods. The HSSVD method identified two layers of mean biclusters and six layers of variance biclusters. The Plaid method identified two biclusters. The sparse biclustering method failed to detect any biclusters. For the LAS method, we report the top three biclusters. Comparison of the biclustering results are summarized in Table 2.10. The first mean bicluster identified by SC-Biclust was strongly associated with cancer, as were all the first three variance biclusters. Indeed, we can see that the three variance biclusters identified by the SC-Biclust algorithm contained cancer samples exclusively. The other biclustering methods also identified biclusters that were associated with cancer. It is interesting to compare the variance biclusters identified by SC-Biclust to the variance biclusters identified by HSSVD. SC-Biclust tends to identify small variance biclusters that contain only cancer patients whereas HSSVD tends to identify larger biclusters that are more heterogeneous. Note that under the 0/1 transformation, SSVD and the mean layers of HSSVD identified biclusters containing all of the observations. The running time for SC-Biclust was greater than the running time of other methods, except for the HSSVD method.

2.4 Discussion

Biclustering is an unsupervised learning algorithm that is a powerful tool for studying HDLSS data. In this paper, we have proposed a general framework for biclustering based on sparse clustering. We have developed algorithms for heterogeneous mean and

variance biclusters as well as more complex structures that can be identified using hierarchical clustering. The algorithms we described in this paper are special cases of this framework, and similar methods can be developed for other bicluster structures of interest.

The biclusters identified by SC-Biclust compared favourably with the biclusters identified by competing methods for both the simulated and real data sets. We believe that SC-Biclust has several other advantages compared to existing biclustering methods. First, unlike some other biclustering methods (15, 25), SC-Biclust does not assume that all features in a bicluster have the same mean. This is a strong assumption that is likely to be violated for many data sets. Indeed, SC-Biclust does not even necessarily assume that the bicluster has different means than the observations not in the bicluster. In general, SC-Biclust can be applied given an arbitrary function whose value increases as the "difference" between the bicluster and the remaining observations increases and a method for maximizing this function with respect to the observations. For example, as noted earlier, SC-Biclust can be used to identify biclusters with heterogeneous variance.

Second, SC-Biclust is noticeably faster than other biclustering methods, particularly HSSVD and LAS. This was particularly true when these methods were applied to high dimensional data. Thus, SC-Biclust may be useful for Big Data problems where other methods are too expensive computationally.

It is interesting to compare the results of SC-Biclust and HSSVD for variance biclustering. In the examples considered in this manuscript, SC-Biclust tended to identify smaller, more homogeneous biclusters whereas HSSVD tended to identify biclusters that were larger and more heterogeneous. It is unclear if this result is true in general or if it is merely an artifact of these particular data sets. Also, it is possible that the method used by SC-Biclust to identify variance biclusters could be improved. The

identification of variance biclusters is a relatively new topic and an important area for future research.

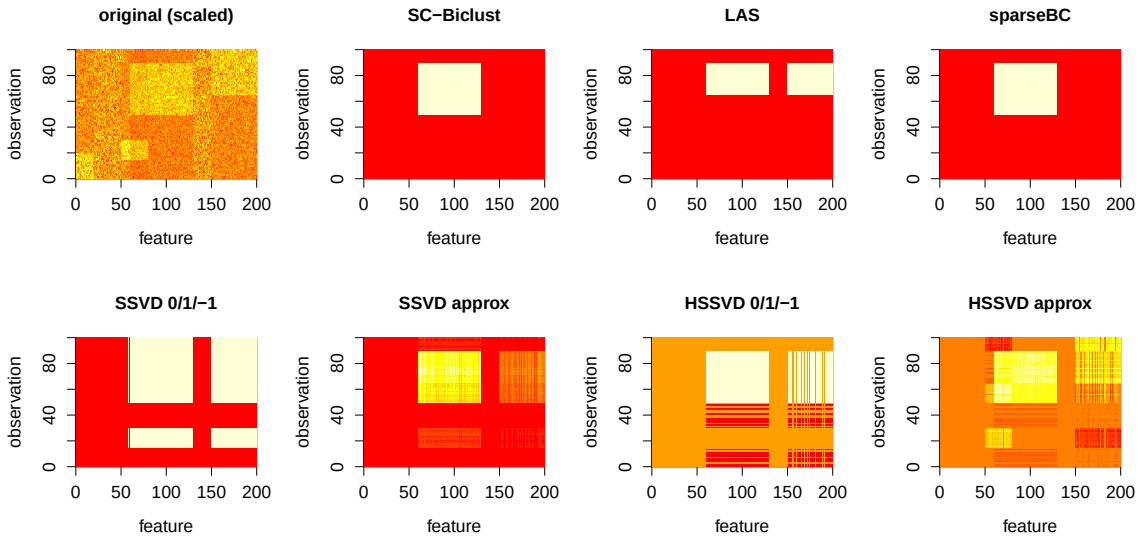

Figure 2.1: Simulation example: primary bicluster identification.

This is an illustration of a single simulation from the first simulation scenario. The first panel shows a heat map of the (scaled) data. The primary bicluster is the rectangular yellow block in the middle. The remaining panels show the biclusters identified by SC-Biclust, LAS, sparse biclustering, SSVD, and HSSVD, with the white regions corresponding to the biclusters. For SSVD and HSSVD, both the 0/1/-1 indicator matrix and the approximation matrix are plotted.

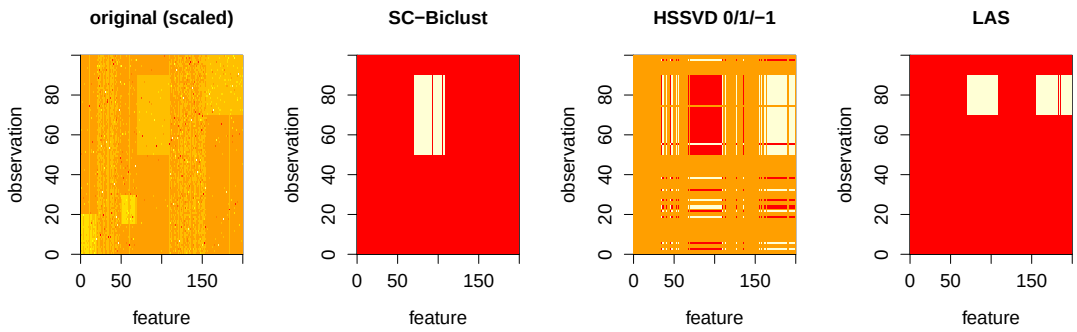

Figure 2.2: Simulation example: departure from normality.

This is an illustration of a single simulation from the second simulation scenario. The first panel shows a heat map of the (scaled) data. The primary bicluster is the rectangular yellow block in the middle. The remaining panels show the biclusters identified by SC-Biclust, HSSVD, and LAS, with the white regions corresponding to the biclusters.

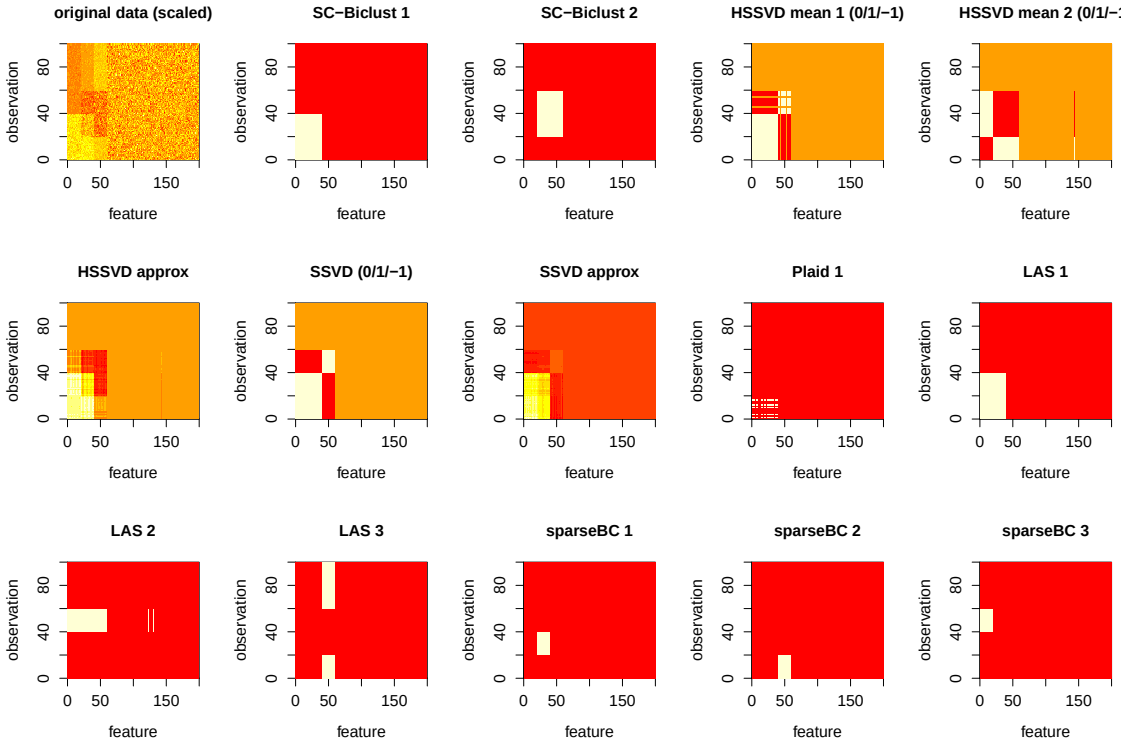

Figure 2.3: Simulation example: sequential biclusters with overlap.

This is an illustration of a single simulation from the third simulation scenario. The first panel shows a heat map of the (scaled) data. The two overlapping biclusters are in the bottom left cover of the data matrix; one is in red and the other is in yellow. The remaining panels show the first two biclusters identified by SC-Biclust and HSSVD, the first bicluster identified by SSVD and Plaid, and the first three biclusters identified by LAS and sparse biclustering. The white regions correspond to the biclusters. For SSVD and HSSVD, both the 0/1/-1 indicator matrix layers and the overall approximation matrices are plotted.

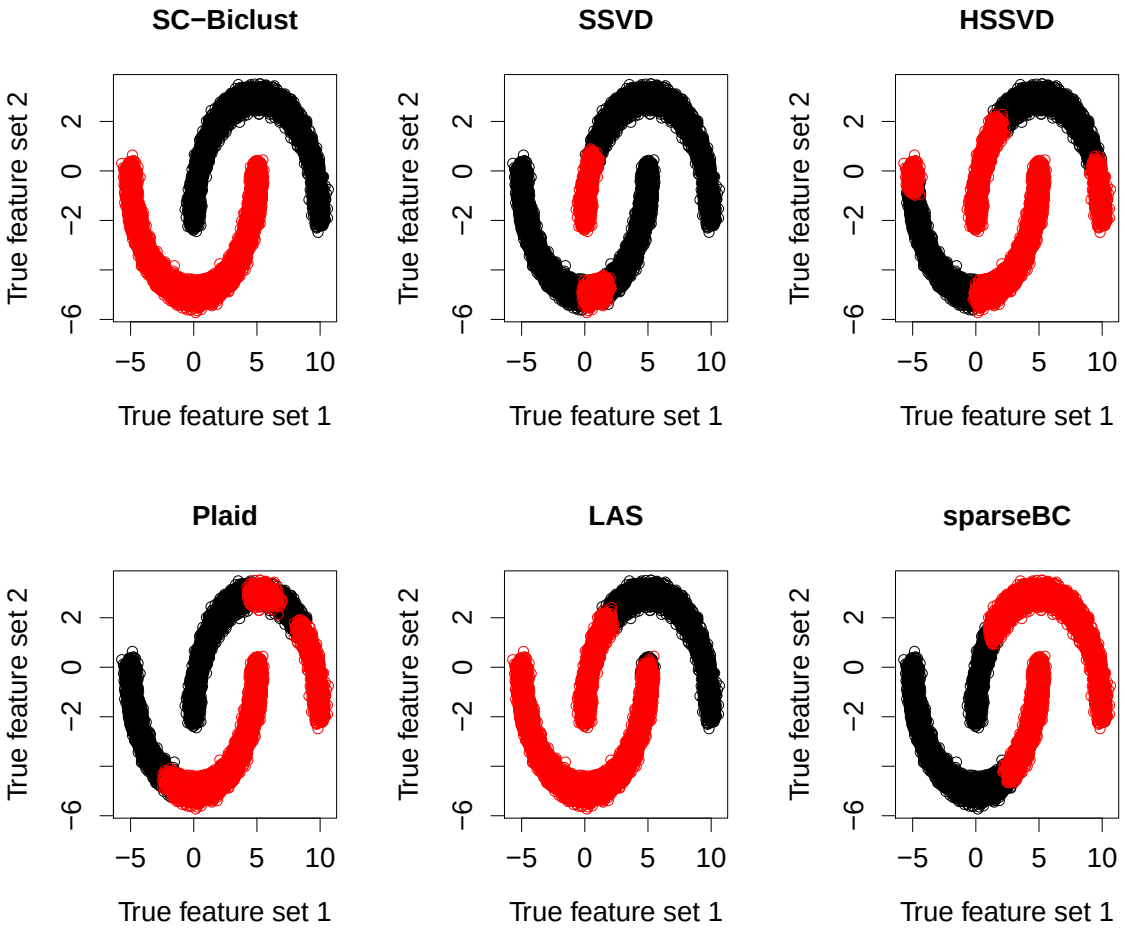

Figure 2.4: Simulation example: non-spherical biclusters.

Each panel shows a plot of the second feature versus the first feature for a single simulation from the fourth simulation scenario. Note that the data forms two non-spherical clusters. Each panel shows the result of applying a biclustering method (specifically SC-Biclust, SSVD, HSSVD, Plaid, LAS, and sparse biclustering) to this data set. Observations that belong to the putative bicluster are labelled in red.

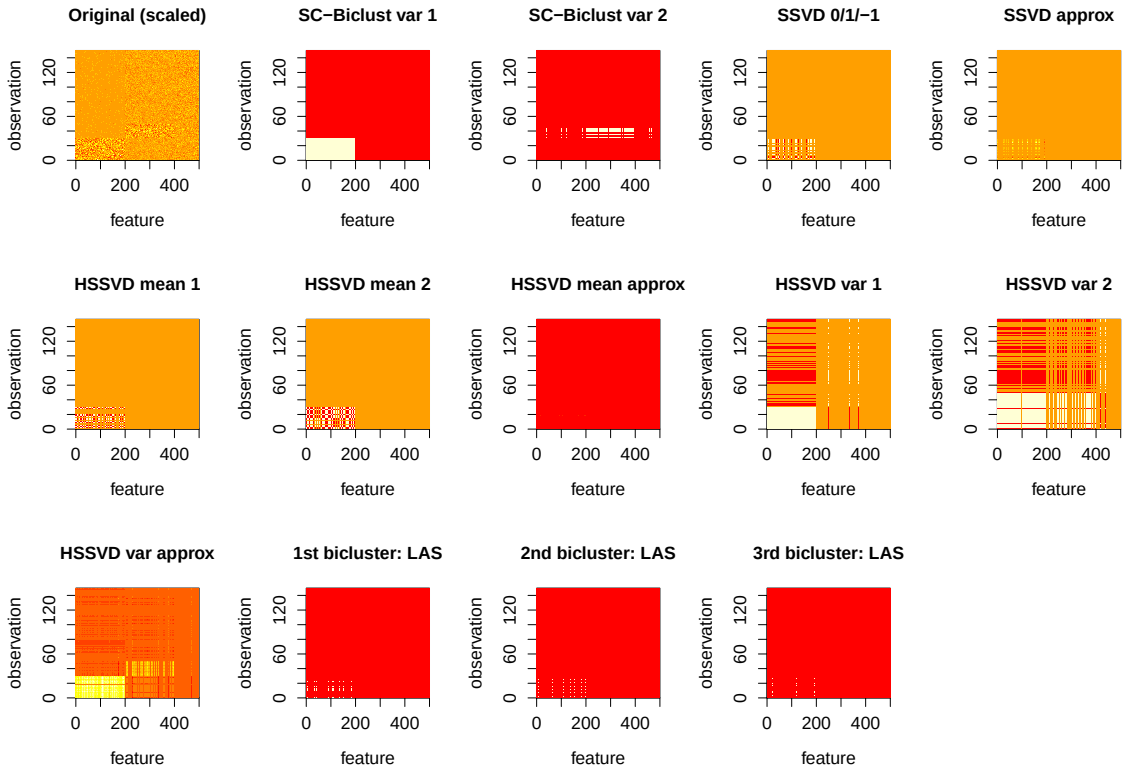

Figure 2.5: Simulation example: variance biclustering.

This is an illustration of a single simulation from the fifth simulation scenario. The first panel shows a heat map of the (scaled) data. The two non-overlapping variance biclusters are on the bottom left corner. The remaining panels show the first two variance biclusters identified by SC-Biclust, result from SSVD and HSSVD, and the first three bicluster identified by LAS. The white regions correspond to the biclusters. For SSVD and HSSVD, both the 0/1/-1 indicator matrix layers and the overall approximation matrices are plotted.

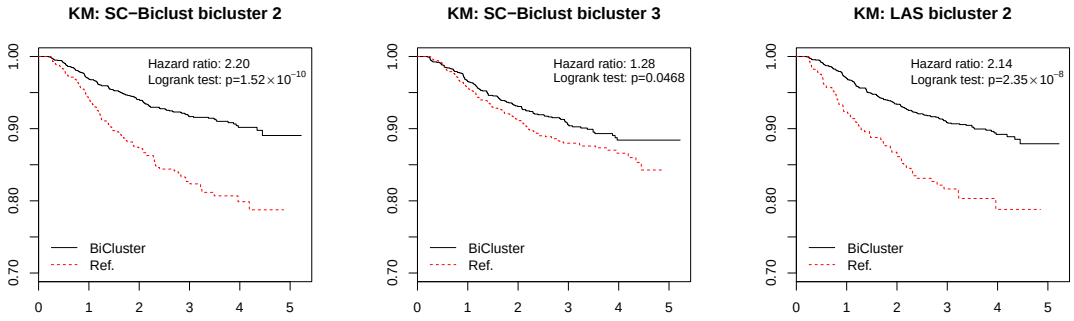

Figure 2.6: OPPERA Kaplan-Meier plots.

The Kaplan-Meier plots showing the association between first-onset TMD and the biclusters identified by SC-Biclust (layer 2 and 3) and LAS (layer 2).

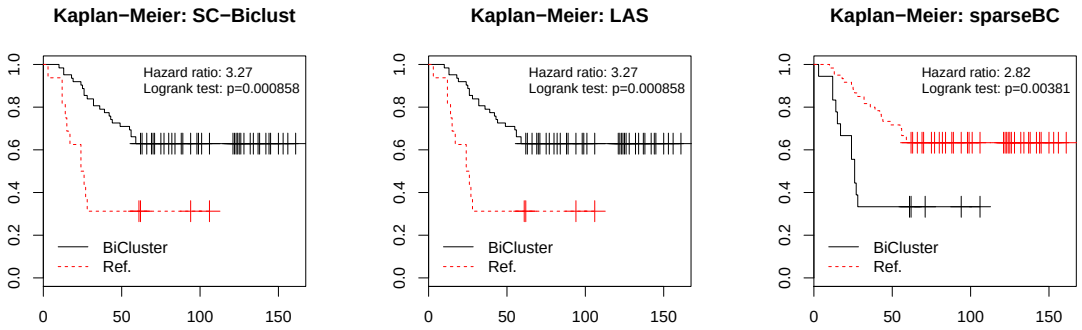

Figure 2.7: Breast cancer gene expression Kaplan-Meier plot.

The Kaplan-Meier plots showing the association between survival and the biclusters identified by SC-Biclust, LAS, and sparse biclustering.

Table 2.1: Comparison of computing times (average of 100 simulations)

Algorithm	Simulation 1	Simulation 2	Simulation 3	Simulation 4	Simulation 5
SC-Biclust	0.416 sec	0.42 sec	0.79 sec	4.93 sec	1.91 min
SSVD	0.28 sec	0.62 sec	0.39 sec	37.34 sec	51.41 sec
HSSVD	1.25 min	1.27 min	1.28 min	2.36 min	5.05 min
Plaid	NA	0.081 sec	0.21 sec	0.58 sec	NA
LAS	12.50 sec	1.27 min	9.54 sec	41.91 sec	3.44 min
Sparse Biclustering	0.85 sec	0.99 sec	23.95 sec	4.62 sec	NA

Table 2.2: Comparison of prediction accuracy: simulations 1, 2, and 4 (average of 100 simulations)

Simulation 1: primary bicluster identification				
Algorithm	Obs. misclassification rate	Feature FNR	Feature FPR	valid identifications
SC-Biclust	0	0.15	0.0024	100
SSVD	0.25	0	0.39	100
HSSVD	0.18	0	0.32	100
Plaid	NA	NA	NA	0
LAS	0.14	0.0021	0.38	100
Sparse Biclustering	0	0	0.0035	100
Simulation 2: departure from normality				
Algorithm	Obs. misclassification rate	Feature FNR	Feature FPR	valid identifications
SC-Biclust	0.18	0.085	0.050	100
SSVD	0.18	0.43	0.072	37
HSSVD	0.40	0.070	0.53	100
Plaid	0.28	0.33	0.13	62
LAS	0.20	0.017	0.27	100
Sparse Biclustering	0.00077	0.0058	0.0038	13
Simulation 4: non-spherical biclusters				
Algorithm	Obs. misclassification rate	Feature FNR	Feature FPR	valid identifications
SC-Biclust	0.058	0	0	100
SSVD	0.41	0	0	100
HSSVD	0.45	0	0	100
Plaid	0.29	0.5	0	100
LAS	0.12	0	0	100
Sparse Biclustering	0.47	0.5	0	100

Table 2.3: Comparison of prediction accuracy: simulations 3 and 5 (average of 100 simulations)

Simulation 3: sequential biclusters with overlap				
Algorithm	Identification	Entry FNR	Entry FPR	valid identifications
SC-Biclust layer 1	Bicluster 1 100%	0	0	100
SC-Biclust layer 2	Bicluster 2 100%	0	0	
SSVD	Bicluster 1+2 100%	0.0048	0	100
HSSVD mean layer 1	Bicluster 1 26%, Bicluster 1+2 74%	0.088	0.013	100
HSSVD mean layer 2	Bicluster 1+2 100%	0.00017	0.00033	
Plaid	Bicluster 1 98%	0.82	0.000073	98
LAS layer 1	Bicluster 1 100%	0.022	0	100
LAS layer 2	Bicluster 2 100%	0.50	0.022	
LAS layer 3	Bicluster 1 100%	1	0.064	
Sparse Biclustering layer 1	Bicluster 1 85%, Bicluster 2 15%	0.92	0.043	100
Sparse Biclustering layer 2	Bicluster 1 67%, Bicluster 2 33%	0.87	0.026	
Sparse Biclustering layer 3	Bicluster 1 75%, Bicluster 2 25%	0.91	0.071	
Simulation 5: variance biclustering				
Algorithm	Identification	Entry FNR	Entry FPR	valid identifications
SC-Biclust layer 1	Bicluster 1 99%, Bicluster 2 1%	0.052	0.0000023	100
SC-Biclust layer 2	Bicluster 1 5%, Bicluster 2 95%	0.76	0.0099	
SSVD	Bicluster 1 91%, Bicluster 2 9%	0.78	0.0011	100
HSSVD mean layer 1	Bicluster 1 100%	0.59	0.00013	100
HSSVD mean layer 2	Bicluster 1 100%	0.20	0.00032	
HSSVD variance layer 1	Bicluster 1 100%	0.0016	0.16	
HSSVD variance layer 2	Bicluster 2 2%, Bicluster 1+2 98%	0.23	0.35	
Plaid	NA	NA	NA	0
LAS layer 1	Bicluster 2 100%	1	0.0036	100
LAS layer 2	Bicluster 2 100%	1	0.0033	
LAS layer 3	Bicluster 2 100%	1	0.0024	
Sparse Biclustering	NA	NA	NA	0

Table 2.4: Comparison of reproducibility (average of 100 simulations × 10 partitions)

	Simulation 1: primary bicluster identification			
Algorithm	Obs. misclassification rate	Feature FNR	Feature FPR	Feature misclassification rate
SC-Biclust	0.11	0.18	0.041	0.14
SSVD	0.015	0.012	0.012	0.024
HSSVD	0.11	0.32	0.0075	0.13
LAS	0.061	0.15	0.023	0.19
Sparse Biclustering	0.05	0.096	0.088	0.18
	Simulation 2: departure from normality			
Algorithm	Obs. misclassification rate	Feature FNR	Feature FPR	Feature misclassification rate
SC-Biclust	0.29	0.12	0.093	0.19
SSVD	0.08	0.37	0.041	0.14
HSSVD	0.16	0.21	0.24	0.30
Plaid	0.37	0.29	0.15	0.19
LAS	0.048	0.21	0.010	0.19
Sparse Biclustering	0.20	0.42	0.0030	0.093
	Simulation 4: non-spherical biclusters			
Algorithm	Obs. misclassification rate	Feature FNR	Feature FPR	Feature misclassification rate
SC-Biclust	0.073	0	0	0
SSVD	0.011	0	0	0
HSSVD	0.27	0.001	0	0.0005
Plaid	0.77	0.34	0.17	0.32
LAS	0.0047	0	0	0
Sparse Biclustering	0.25	0	0	0

Table 2.5: Comparison of stopping rule: simulations 1 and 3 (average of 100 simulations)

	Simulation 1: primary bicluster identification
Algorithm	number of biclusters identified (%)
SC-Biclust	4 (44%) 5 (54%) 6 (2%)
HSSVD mean	2 (5%) 3 (39%) 4 (54%) 5 (2%)
HSSVD var	2 (100%)
LAS	8 (2%) 9 (98%)
Sparse Biclustering	5 (1%) 6 (99%)
	Simulation 3: sequential biclusters with overlap
Algorithm	number of biclusters identified (%)
SC-Biclust	2 (99%) 3 (1%)
HSSVD mean	2 (63%) 3 (37%)
HSSVD var	2 (100%)
Plaid	1 (40%) 2 (30%) 3 (22%) 4 (7%) 5(1%)
LAS	7 (100%)
Sparse Biclustering	4 (100%)

Table 2.6: OPPERA: comparison of different biclustering algorithms

Algorithm (computing time)	Layer	Bicluster composition	
		# obs. (case; non-case)	# features (Auto; Psy; QST)
SC-Biclust (2.59 min)	Layer 1	1561 (110; 1451)	30 (30; 0; 0)
	Layer 2	998 (89; 909)	29 (0; 29; 0)
	Layer 3	1619 (118; 1501)	6 (0; 0; 6)
SSVD (21.28 sec)	Layer 1	3443 (185; 3258)	98 (44; 22; 32)
HSSVD (12.47 min)	Mean 1	3443 (185; 3258)	115 (44; 39; 32)
	Mean 2	3443 (185; 3258)	116 (44; 39; 33)
	Var 1	3378 (184; 3194)	109 (44; 36; 29)
	Var 2	3408 (185; 3223)	111 (44; 36; 31)
Plaid (14.08 sec)	Layer 1	68 (6; 62)	23 (23; 0; 0)
	Layer 2	6 (1; 5)	21 (21; 0; 0)
	Layer 3	23 (2; 21)	21 (7; 14; 0)
LAS (14.66 min)	Layer 1	817 (33; 784)	24 (24; 0; 0)
	Layer 2	638 (73; 565)	43 (0; 23; 20)
	Layer 3	945 (78; 867)	24 (24; 0; 0)

Table 2.7: OPPERA: association between biclusters and chronic TMD

Algorithm	Bicluster 1		Bicluster 2		Bicluster 3	
	χ^2 (df=1)	p-value	χ^2 (df=1)	p-value	χ^2 (df=1)	p-value
SC-Biclust	15.13	1.00×10^{-4}	33.75	6.26×10^{-9}	21.34	3.84×10^{-6}
HSSVD var	1.23	0.27	1.08	0.30	NA	NA
Plaid	1.01	0.32	0.10	0.75	0.06	0.81
LAS	3.41	0.065	55.27	1.05×10^{-13}	20.49	6.01×10^{-6}

Table 2.8: OPPERA: association between biclusters and first-onset TMD (logrank test)

Algorithm	Bicluster 1		Bicluster 2		Bicluster 3	
	Statistic (df)	p value	Statistic (df)	p value	Statistic (df)	p value
SC-Biclust	2.72 (df=1)	0.099	41.01 (df=1)	1.52×10^{-10}	3.95 (df=1)	0.047
HSSVD var	0.4 (df=1)	0.53	0.26 (df=1)	0.61	NA	NA
Plaid	2.87 (df=1)	0.090	0.42 (df=1)	0.52	0.07 (df=1)	0.80
LAS	0.5 (df=1)	0.48	31.18 (df=1)	2.35×10^{-8}	1.71 (df=1)	0.19

Table 2.9: Gene expression: Comparison of biclustering and survival analysis results.

Algorithm	Computing time	Obs.	Feature	Score (logrank) test	
				Statistic (df)	p value
SC-Biclust	8.72 sec	16	8	11.11 (df=1)	8.58×10^{-4}
HSSVD mean	8.30 min*	75	1046	0.42 (df=1)	0.515
LAS	22.87 min	16	1421	11.11 (df=1)	8.58×10^{-4}
Sparse Biclustering	30.80 min	60	553	10.2 (df=1)	0.0014

Table 2.10: Methylation: association between biclusters and cancer

Algorithm (Computing time)	Layers	Fisher's exact test p-value	Bicluster composition # obs. (cancer; normal)	# features
SC-Biclust	Mean 1	2.00×10^{-11}	115 (88; 27)	274
(5.74 min)	Mean 2	0.81	115 (59; 56)	221
	Var 1	0.00011	14 (14; 0)	299
	Var 2	0.00011	14 (14; 0)	345
HSSVD	Mean 1	NA	290 (152; 138)	243
(9.02 min)	Mean 2	NA	290 (152; 138)	261
	Var 1	1.07×10^{-14}	190 (69; 121)	235
	Var 2	0.097	247 (124; 123)	369
	Var 3	0.13	249 (126; 123)	373
	Var 4	0.00021	262 (128; 134)	376
	Var 5	0.047	273 (139; 134)	378
	Var 6	0.047	273 (139; 134)	379
Plaid	Layer 1	4.71×10^{-5}	13 (0; 13)	104
(1.22 sec)	Layer 2	0.031	6 (6; 0)	80
LAS	Layer 1	0.45	53 (25; 28)	232
(3.98 min)	Layer 2	$< 2.2 \times 10^{-16}$	60 (60; 0)	171
	Layer 3	$< 2.2 \times 10^{-16}$	58 (58; 0)	100

Chapter 3

Soft-thresholding sparse clustering

3.1 Introduction

Machine learning methods are popular and widely used in many scientific research areas, especially when analysing HDLSS data, such as gene expression data. In particular, clustering methods can be used to uncover informative data structure for further analysis on the association between observations and predictor variables. Clustering methods seek to partition a data set into homogeneous observation subgroups. And the conventional clustering methods (such as k-means clustering or hierarchical clustering) use all the features to identify clusters. However, in many situations it is desirable to identify clusters that differ with respect to only a subset of the features, and clustering based on all the features could lead to misleading result. Such clusters may represent homogeneous subgroups of patients with a disease, such as cancer or chronic pain. In a gene expression data set collected from cancer patients, for example, it is likely that the subgroups are defined by only a small proportion of the genes. In such situations, we need a clustering method that can not only correctly cluster the observations into homogeneous subgroups, but also identify the features that truly contribute to the observation clustering.

3.1.1 Sparse clustering

The standard k-means clustering algorithm partitions a data set into k subcategories by maximizing the between cluster sum of squares (BCSS). The BCSS is calculated by taking the sum of the BCSS's for each individual feature. This implies that all the features are equally important. However, as we have discussed previously, in many situations the clusters differ with respect to only a fraction of the features, and these truly associated features do not necessarily contribute equally. In such situations, giving equal weights to all features when clustering may produce inaccurate results. This is especially true for HDLSS problems, where the number of the features is much bigger than the number of objects. To overcome this problem, (29) proposed a novel clustering method which they called "sparse clustering." Under sparse clustering where the original data matrix is with dimension $n \times p$, each feature is given a non-negative weight $w_j, j = 1, 2, \ldots, p$, and the following weighted version of the BCSS is maximized:

$$\begin{aligned} &\text{maximize}_{C_1,\ldots,C_K,\mathbf{w}} \left\{ \sum_{j=1}^{p} w_j \left(\frac{1}{n} \sum_{i=1}^{n} \sum_{i'=1}^{n} d_{i,i',j} - \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i,i' \in C_k} d_{i,i',j} \right) \right\} \\ &\qquad \text{subject to } ||\mathbf{w}||^2 \leq 1, ||\mathbf{w}||_1 \leq s, w_j \geq 0 \, \forall \, j. \end{aligned} \tag{3.1}$$

Here X_{ij} represents observation i for feature j of the data matrix X and $i \in C_k$ if and only if observation i belongs to cluster k. $d_{i,i',j}$ is a distance metric between any pair of observations in X with respect to feature j. For k-means clustering, we take $d_{i,i',j} = (X_{ij} - X_{i'j})^2$. The L_1 bound on $\mathbf{w}$, s, is a tuning parameter, and can be either pre-specified by the customer or selected through certain procedure, which we will discuss in detail later.

They also describe an iterative procedure for maximizing (3.1) in the sparse clustering paper (29):

1. Initially let $w_1 = w_2 = \ldots w_p$.

2. Maximize (3.1) with respect to $C_1, C_2, \ldots, C_K$ by applying the standard k-means algorithm with the appropriate weights. In other words, apply the k-means algorithm where the dissimilarity between observations i and i' is defined to be $\sum_{j=1}^{p} w_j d_{i,i',j}$.

3. Maximize (3.1) with respect to the w_j's by letting

$$w_j = \frac{S(b_j, \Delta)}{\|S(b_j, \Delta)\|_2} \tag{3.2}$$

   Here b_j is the (unweighted) between cluster sum of squares for feature j and $S(x, y) = \text{sign}(x)(|x| - y)_+$ is a soft-threshold operator. Δ is chosen so that $\sum_j |w_j| = s$ ($\Delta = 0$ if $\sum_j |w_j| \leq s$). See (29) for the justification for (3.2).

4. Iterate steps 2 and 3 until the algorithm converges.

Note that (3.2) implies that as s increases, the number of non-zero w_j's decreases. Ideally, for sufficiently small values of s, only a subset of the features contribute to the cluster assignments, and the magnitude of w_j represents the contribution of feature j to the clustering result. So, this method is useful in situations where the clusters differ with respect to only a subset of the features. Also note that in this algorithm, each value of s corresponds to a particular Δ, and one can obtain w_j's without s if Δ is defined instead.

Under their general framework, variants of this procedure can be used to perform sparse clustering in various form, such as sparse hierarchical clustering (29).

3.1.2 GAP statistic and tuning parameter selection

The GAP statistic was first proposed as a reference tool to select the optimal number of clusters (26). They defined W_k as the pooled within-cluster sum of squares (WCSS) around the cluster mean in the following manner:

$$W_k = \sum_{r=1}^{k} \frac{1}{2n_r} D_r, \tag{3.3}$$

where k is the number of clusters, n_r is the number of observations within the rth cluster, and $D_r = \sum_{i,i' \in C_r} d_{ii'}$ is the sum of the pairwise distances for all points in cluster r. The GAP statistic is then defined as

$$GAP_n(k) = E_n^* \log(W_k) - \log(W_k), \tag{3.4}$$

where E_n^* denotes the expectation under a sample of size n from the reference distribution. In this way, the statistic $GAP_n(k)$ measures the difference of the clustering obtained on the real data relative to the clustering obtained on the null data, and the calculation of this $E_n^* \log(W_k)$ term depends on the definition of the null data.

In the sparse clustering method, they suggested the following algorithm to select the tuning parameter s from a set of candidate values based on the GAP statistic through independent observation permutation (29):

1. Obtain permuted data sets $X_1, \ldots, X_B$ by independently permuting the observations within each feature. The permuted data sets are referred to as the null data sets.

2. For each tuning parameter s, compute
   $O(s) = \sum_j w_j (\frac{1}{n} \sum_{i=1}^{n} \sum_{i'=1}^{n} d_{i,i',j} - \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i,i' \in C_k} d_{i,i',j})$
   as the objective obtained by performing sparse K-means clustering on the original

data set X with tuning parameter s, and $O_b(s)$ as the corresponding objective for the permuted data set X_b. The GAP statistic for a given value of s is then calculated as $GAP(s) = \log(O(s)) - \frac{1}{B}\sum_{b=1}^{B} \log(O_b(s))$.

3. The optimal tuning parameter s^* is chosen such that the corresponding $GAP(s^*)$ is the largest.

Through this algorithm, the GAP statistic is desired to measure the strength of the clustering obtained on the real data relative to the clustering obtained on null data, and this null data is approximated by the permuted data sets in the first step. Their argument for the independent observation permutation is that all the features in the null data set should be uncorrelated and there should be no subgroups (e.g. clusters) on the observation dimension (29).

3.1.3 Simulation example of sparse clustering null distribution

In this simulation example, we artificially created a 100×200 data matrix with independent entries from a standard normal distribution with mean 0 and standard deviation 1. We denote the distribution as $N(0,1)$, where $N(a,b)$ represents a normal random variable with mean a and standard deviation b. All the features are uncorrelated in this simulated data set and there is no designed cluster on the object dimension. According to the sparse clustering method, this could be viewed as their reference null distribution. And if we run the sparse K-means clustering algorithm on this data set, we expect to see that all of the features should have roughly equal weights as none of them is significantly associated with the object clusters that do not exist.

So, we ran sparse 2-means clustering on this simulated null distribution data matrix, and the algorithm clustered the 100 observations into two subgroups, each contained 49 and 51 observations specifically. Figure 3.1 plots the associated tuning parameter

s, feature weights, and the GAP statistics from the sparse clustering method, together with the feature weights that were associated with the optimal tuning parameter selected by the algorithm. According to the plot, we can see that based on the optimal tuning parameter, one of the features had significantly bigger weight compared to the others. According to (29), this result suggested that this feature was associated with the object clusters obtained, which we know was not true based on the way we simulated the data.

To further reveal the limitation of this null distribution, we then permuted this simulated null data set 15 times. During each permutation, the observations from the simulated data set were permuted independently within each feature. According to (29), all these 15 permuted data sets could also serve as the reference null data. And once again, for each permuted data set, all the features should have roughly equal weights as none of them was significantly associated with the observation clusters that did not exist. Figure 3.2 plots the resulting sparse clustering feature weights from each of the 15 permuted data sets. The results were similar to what we have seen in Figure 3.1, where one of the features had significantly bigger weight compared to the others. For each permuted data set, the result suggested that this feature was highly associated with the object clusters obtained by the sparse clustering algorithm, which we know is not true. It was also interesting to note that as we kept the feature indices unchanged across the 16 null data sets (1 from Figure 3.1 and 15 from Figure 3.2), the features being picked by the algorithm were not consistent.

3.2 ST-Spcl: soft-thresholding sparse clustering

According to the simulation example we presented above, the null distribution proposed by the sparse clustering method does not lead to equal feature weights. Instead, one of the features was assigned significantly larger weight compared to the others,

which violated the sparse clustering assumption for weight assignment. The results suggest that selecting the smallest tuning parameter s that corresponds to the largest GAP statistic is not the appropriate and optimal strategy, and the algorithm could be improved if we can predefine a subset of features that we believe are truly associated with the observation clustering.

3.2.1 Iterative procedure of the ST-Spcl algorithm

In order to improve the sparse clustering algorithm in terms of tuning parameter selection, here we propose a new method named ST-Spcl (soft-thresholding sparse clustering). This new algorithm is based on the idea of biclustering for variable selection (see Chapter 2 for details), and then iterating to obtain the optimal observation clusters and feature weights. And instead of using permutations to select the smallest tuning parameter s that corresponds to the largest GAP statistic, we work directly with the soft-threshold operator that was introduced in (3.2) by defining the Δ term without doing numeric search based on s. In this way, we can significantly speed up the process and bypass the selection of s based on the GAP statistics, whose null distribution assumption was not appropriate according to our simulation example. The proposed iterative procedure is as follows:

1. Start with sparse K-means clustering with tuning parameter fixed as $s = \sqrt{p}$. Obtain the initial clustering index CI_0 and the reference feature weights $w = \{w_1, w_2, \ldots, w_p\}$.

2. Perform a Kolmogorov-Smirnov test of the null hypothesis that the distribution of $w_1, w_2, \ldots, w_p$ is the same as the expected distribution of the weights under the null hypothesis of no clusters.

3. If the test in Step 2 fails to reject the null hypothesis, then terminate the procedure

and report that no clusters was identified.

4. If the test in Step 2 rejects the null hypothesis, then let

$$m = \arg\max_j \left(w_{(p-j+1)} - w_{(p-j+1)_0}\right) - \left(w_{(p-j)} - w_{(p-j)_0}\right) \tag{3.5}$$

   Define the non-zero weight indicator ws_{ind} as $ws_{ind,j} = 1$ if feature j belonged to the top m features with the highest reference feature weights, and $ws_{ind,j} = 0$ if feature j had a small reference feature weight.

5. Define $\Delta = \max\{w_j, \text{where } ws_{ind,j} = 0, j \in \{1, 2, \ldots, p\}\}$ and apply the soft-threshold operator to maximize (3.1) with respect to $\mathbf{w}$, the same as suggested by (3.2).

6. Maximize equation (3.1) with respect to CI.

7. Iterate step 5 and 6 until the algorithm converges.

We recommend that the data matrix be normalized such that all features have mean 0 and standard deviation 1 before applying the procedure.

3.2.2 Estimating the Null Distribution of the Weights

This method requires one to know the expected order statistics of the weights under the null hypothesis that no clusters exist. If this distribution is unknown, it may be approximated as follows:

1. Apply the K-means sparse clustering algorithm with $s = \sqrt{p}$ to obtain clustering index CI_0, as before.

2. Fix the clustering index CI_0 and permute the rows of X to calculate weights $w_1^*, w_2^*, \ldots, w_p^*$.

3. Repeat Step 2 B times.

4. Approximate $w_{(j)_0}$ as $w_{(j)_0} = \sum_k w^*_{(j)k}/B$, where $w^*_{(j)k}$ represents the jth order statistic of the weights from the kth iteration of Step 2.

This procedure will provide an estimate of the expected values of the order statistics of the weights, but it is very expensive computationally for large data sets. It would be desirable to develop a faster alternative. Fortunately, if the sparse clustering procedure is modified slightly, the exact distribution of the weights can be calculated under mild assumptions for K=2 cases.

First, note that the criterion in (3.1) can be written as $\sum_j w_j b_j$, where b_j is the between cluster sum of squares for feature j. If we modify the procedure to minimize $\sum_j w_j \sqrt{b_j}$ rather than $\sum_j w_j b_j$, then (3.2) implies that the optimal w_j's are given by

$$w_j = \frac{\sqrt{b_j}}{\sqrt{\sum_k b_k}} \tag{3.6}$$

assuming $s = \sqrt{p}$ (implying that $\Delta = 0$ in (3.2)). Now under the null hypothesis that no clusters exist, there is no difference in the means of the observations in C_1 and C_2 for all features, implying that $b_j \sim \chi^2_1$ for all j. Thus, (3.6) implies that w_j^2 has a Beta$(1/2, (p-1)/2)$ distribution. Thus, if we use this criterion to select the clusters, we can test the null hypothesis that no bicluster exists by performing a Kolmogorov-Smirnov test of the null hypothesis that the w_j^2's have a Beta$(1/2, (p-1)/2)$ distribution. Similarly, in (3.5), $w_{(j)_0} = E(\sqrt{B_{(j)}})$, where $B \sim$ Beta$(1/2, (p-1)/2)$. Although there is no simple closed form expression for $E(\sqrt{B_{(j)}})$, it can be easily approximated numerically. We will use this method to approximate the null distribution of the weights in all subsequent 2-mean clustering examples unless otherwise noted.

3.3 Simulation example to test the performance of ST-Spcl

We first evaluated the performance of our method on a few simulation scenarios, and compared its performance with the sparse clustering method. For each simulation scenario, we simulated 100 data sets with the same data structure, and the numeric comparison results were summarized in Table 3.1.

The first simulation scenario is the same as the simulation example presented in Section 3.1.3, where each simulated data set comprised a 100×200 matrix with independent entries from $N(0, 1)$ distribution, and each column represents a feature and each row represents an observation. All the features are uncorrelated, and no observation cluster exists. Figure 3 shows the clustering result from one of the simulations. According to the plot, we can see that the sparse clustering method clustered observations into two groups; and the algorithm assigned non-zero weights to a lot of features, which were all false positive due to the way we simulated the data set. On the other hand, the ST-Spcl method resulted in no observation cluster, as desired; and the method only assigned non-zero weights to a few features, which indicated low feature false positive rate. The numeric comparison on computing time and feature false positive rate across 100 simulations were summarized in Table 3.1.

In the second simulation example, we simulated 100 data sets with size 100×200. The background entries followed $N(0, 1)$ distribution. Observations 1-40 and features 1-70 had elevated mean, with a signal layer added to the existing data matrix where the entries in the signal layer follow distribution $N(3, 1)$. Figure 4 shows the clustering result from one of the simulations. According to the plot, we can see that both the sparse clustering method and the ST-Spcl method successfully identified the true observation clusters, but the sparse clustering method assigned non-zero weights to all the features while the ST-Spcl method only assign non-zero weights to the truly associated features. The result suggested that the sparse clustering method had high feature false

positive rate, which was not desirable as we want to pick out only the features that were associated with the observation clusters. The numeric comparison on computing time and feature false positive/negative rates across 100 simulations were summarized in Table 3.1.

3.4 Real data application on breast cancer gene expression.

The data set used in this section contains gene expression measurements on 4751 genes from a total number of 78 breast cancer subjects. The survival time of each subject is also available. See (27) for a more detailed description of this data set.

We ran 2-means clustering analysis on this gene expression data set with both the sparse clustering method and the ST-Spcl method. The sparse clustering algorithm took 11.07 minutes to run, identified two clusters with 17 and 61 observations respectively. In comparison, the ST-Spcl algorithm took 15.97 seconds to run, and identified two clusters with 16 and 62 observations respectively. The observation clustering results from these two methods were comparable to each other as there was only one observation categorized into different clusters. In terms of feature selection, the sparse clustering method assigned non-zero weights too all of the 4751 features, while the ST-Spcl method only picked out 8 features to be significantly associated with the observation clusters identified. Detailed comparison results are summarized in Table 3.2.

We also tested the null hypothesis of no association between each putative observation clustering result and survival using log rank tests. Table 3.2 and Figure 3.5 show the associations between survival and the observation clusters identified by sparse clustering and ST-Spcl. As we can see from Table 3.2, the putative observation clusters identified by both methods were associated with survival, while the running time for ST-Spcl was significantly lower than the running time for sparse clustering.

3.5 Discussion

In this chapter, we point out a few limitations of the sparse clustering method, especially in the null distribution assumption and the tuning parameter selection algorithm. Specifically, we use simulation examples to show that the null distribution assumption is questionable, hence their proposed method for tuning parameter selection based on the GAP statistic can be further improved. For instance, they expect to see that all the features have roughly equal weights when there is no observation cluster, and such data sets were the null reference data sets they used for method assumption (29). However, the results we observed violated the assumption, with one of the features assigned significantly larger weight at the end of the iterative procedure.

To solve this problem and to further improve the sparse clustering method, we proposed an alternative approach to assign feature weights. Specifically, we proposed a sparse clustering algorithm called ST-Spcl, that incorporated both the biclustering method and the sparse clustering method, in identifying observation clusters and also revealing the sparsity property of feature contribution to these clusters. By working directly with the soft threshoding operator, we greatly reduced the computing time. And by introducing the feature selection step from the biclustering method, we were able to focus more on the features that truly contribute to the observation clusters, and to force the weights of the none associated features go to zero.

We compared the performance of these two algorithms both on simulated data sets and a real data set from a breast cancer gene expression study. The results suggested that the ST-Spcl method was much faster than the existing sparse clustering method, especially with the special cases where K equals 2 and we were able to parametrically specify the null distribution of the features weights with minor modification to the general algorithm. On top of that, the ST-Spcl method was more accurate in terms of lower false negative/positive discovery rates. More importantly, ST-Spcl was more

successful in uncovering the sparsity property of the features, where the existing sparse clustering methods tended to assign none zero weights to too many features, thus the result was less informative if one really wanted to select only a subset of the features that were truly associated with the observation clusters identified.

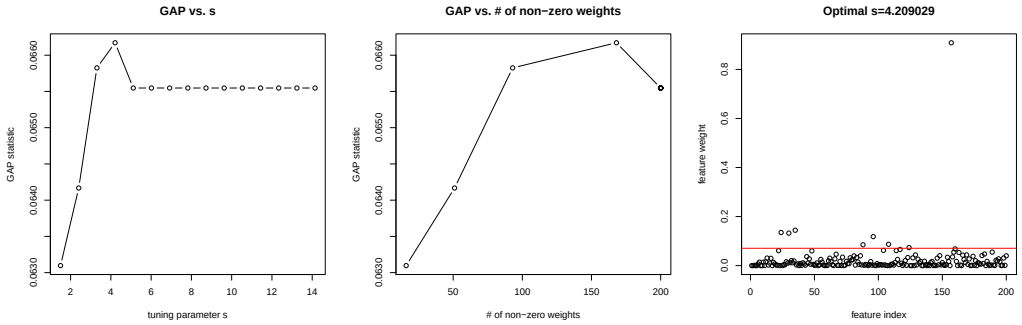

Figure 3.1: Simulation example of sparse clustering null distribution.

The first two figures plot the association between the GAP statistic and the tuning parameter s, and the association between the GAP statistic and the number of null zero weights. The last figure plots the feature weights resulted from the optimal tuning parameter selected by the sparse clustering algorithm, where the horizontal line corresponds to a reference weight when all the feature have equal weights.

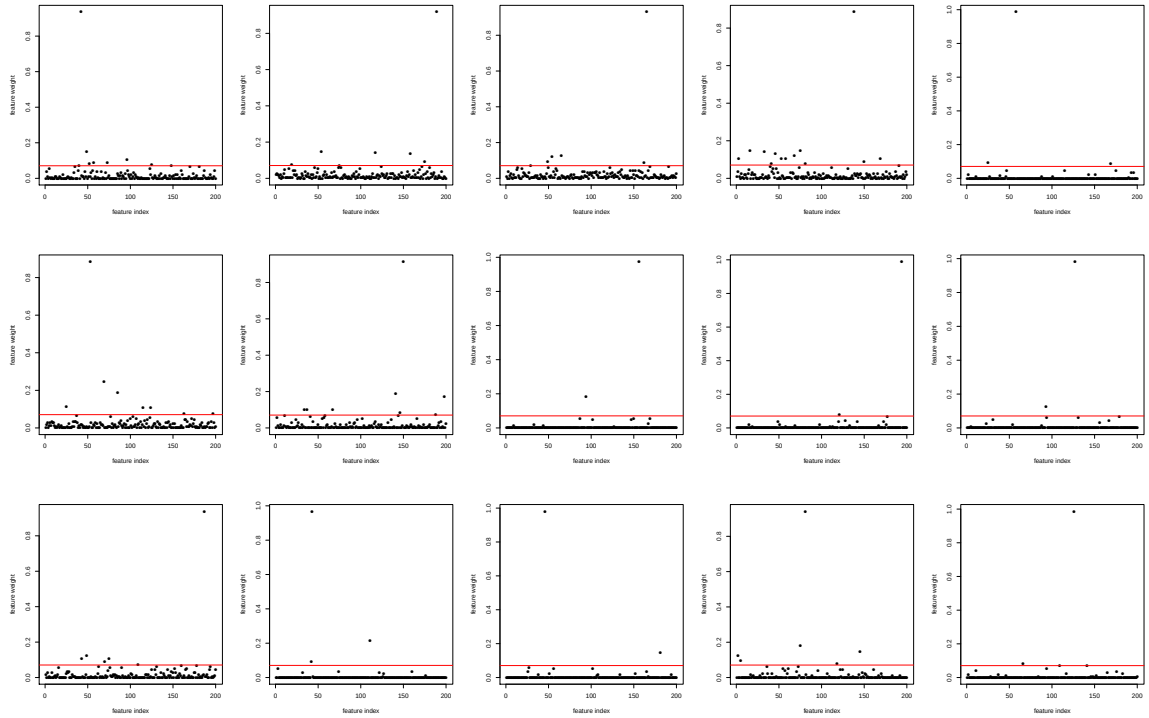

Figure 3.2: Simulation example of feature weights from permuted sparse clustering null data.

Each plot in this figure corresponds to a random permutation of the null data described previously, which can also be viewed as the null data. On each permuted data set, the sparse clustering method results in one feature with significant larger weight compared to the others, and the horizontal line corresponds to a reference weight when all the feature have equal weights. For each permuted data set, the result suggests that this feature is highly associated with the object clusters obtained, which we know is not true.

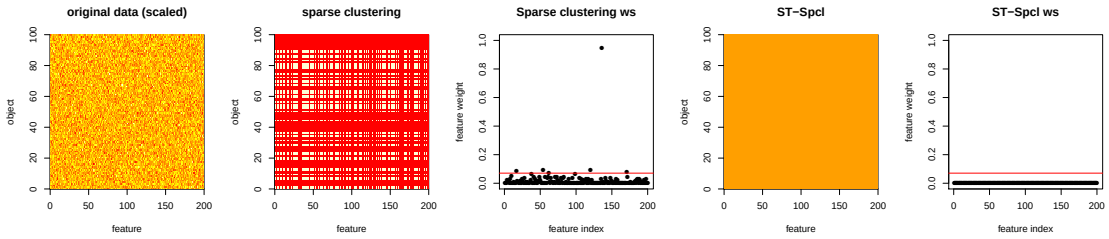

Figure 3.3: Simulation example of sparse clustering null data: ST-Spcl vs. sparse clustering.

This figure corresponds to one of the 100 simulations for the first simulation scenario. Sparse clustering assigns large weight to one of the features. In comparison, ST-Spcl assigns weight 1 to one feature and 0 to the others, and returns an error message indicating the null data situation.

Figure 3.4: Simulation example of bicluster with homogeneous mean: ST-Spcl vs. sparse clustering.

This figure corresponds to one of the 100 simulations for the second simulation scenario. Sparse clustering assigns positive weights to all the featurest. In comparison, ST-Spcl assigns positive weights only to the features that truly contribute to the observation clustering.

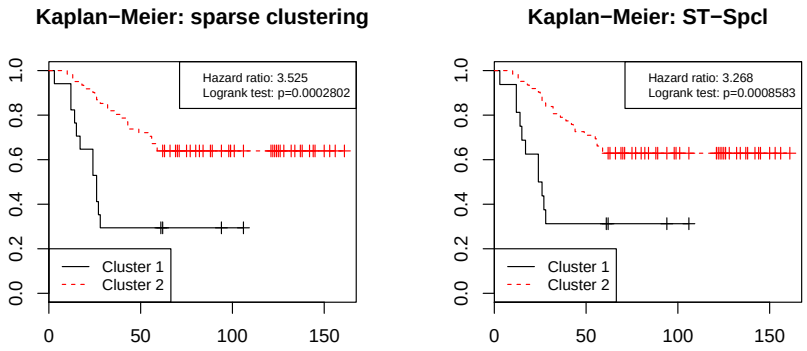

Figure 3.5: Breast cancer gene expression Kaplan-Meier plot.

The Kaplan-Meier plots showing the association between survival and the observation clusters identified by sparse clustering and ST-Spcl.

Table 3.1: Sparse clustering comparison (average of 100 simulations)

Simulation 1: sparse clustering null data			
Algorithm	Computing time	Feature false negative rate	Feature false positive rate
Sparse clustering	0.597 mins	NA	0.339
ST-Spcl	0.505 sec	NA	0.00567
Simulation 2: bicluster with homogeneous mean			
Algorithm	Computing time	Feature false negative rate	Feature false positive rate
Sparse clustering	0.433 min	0	1
ST-Spcl	0.383 sec	0	0

Table 3.2: Gene expression: Comparison of clustering and survival analysis results.

Algorithm	Computing time	Obs.	Feature	Score (logrank) test	
				Statistic (df)	p value
Sparse clustering	11.07 mins	61	4751	13.2(df=1)	0.00028
ST-Spcl	15.97 sec	62	8	11.11(df=1)	0.00086

Chapter 4

Random forests variable importance

4.1 Introduction

Random forests are a data mining method based on simple decision trees, and can be used to evaluate the association between a response variable and a large number of predictors (3). Random forests are usually used for prediction, and assessment for variable importance. The response variable can be either numeric or categorical, and so are the predictor variables. As an alternatives to traditional parametric regression models, random forests have became popular and widely used in some scientific research areas, such as genetics and neurosciences (24, 8). They are able to cope with high dimension low sample size (HDLSS), namely "small n large p" problems, and to deal with high order interactions and complex correlations among predictor variables.

Random forest is built on multiple decision trees, where each tree takes a bootstrap sample that is randomly selected from the original observations. These trees are diverse, both because of the randomly selected bootstrap samples and the randomly preselected splitting variables in each tree. Particularly, in building each decision tree, the observations excluded from each bootstrap sample are known as the "out of bag" (OOB) observations, and this step is also known as "bagging"; and at each node, only a subset of the features are used to build the decision rule, which are known as the splitting variables and the size can be pre-specified. By taking the average of the outputs

from multiple tress, random forest model is more accurate and reliable than simple decision trees. In particular, it was showed that the prediction accuracy was approved by smoothing the hard cut decision boundaries due to the splitting in single trees, which also reduced the variance of the prediction (4).

In high dimensional data setting, random forest VIMP scores have been suggested to assess variable importance and thus be used for variable selection (3). In the later sections, we will refer to their proposed VIMP scores as the Breiman's VIMP scores. In particular, the Breiman's VIMP scores measure how much the predictive accuracy of the model is decreased when a given variable is measured with error, meaning that the observations of this variable are replaced by a random sample of itself with replacement. In general, variables with higher Breiman's VIMP scores are more likely to be associated with the modelling outcome. However, in practice, predictors that are not associated with the modelling outcome can have high VIMP scores if they are strongly correlated with some truly associated variables (1). In this paper, we propose an alternative approach to calculate the conditional VIMP sores and to avoid the bias due to correlation.

4.1.1 Breiman's VIMP: reference for variable selection

Recall that a random forest model is built on multiple decision trees, where a single tree is calculated using a bootstrap sample randomly selected from the original observations, and the OOB samples can serve as the testing set and pass down the tree with the predictive accuracy recorded. In calculating the Breiman's VIMP scores, the values for a given predictor variable are permuted in the OOB samples, and the variable importance of this variable is the average decrease in accuracy over all trees. The Breiman's VIMP is obtained based on variable permutation, thus it is also referred to as the permutation importance, as a contrast to the Gini importance which is the

mean Gini gain produced by this predictor variable over all trees (22). Not like the Gini importance which has a bias towards continuous variables and categorical variables with more categories, the permutation importance is unbiased with respect to both continuous and categorical variables (22).

In general, variables with high Breiman's VIMP scores are more likely to be associated with the modelling outcome (3). For notation purpose, let's denote x_{j0} as the observed value of predictor variable j; X_{-j} as the data matrix excluding variable j; and $L(x_j, X_{-j})$ as the associated loss function of choice, which could be MSE for continuous output, for example. The basic rationale of the Breiman's VIMP score calculation is that by randomly permuting the predictor variable x_{j0}, its original association with the response variable y is broken. Then, when we use this permuted values of x_{j0}, denoted by x'_{j0}, together with the other non-permuted variables X_{-j} to model the response variable y, if the result does not change significantly, it suggests that the permutation of x_{j0} does not affect the model fitting, hence this variable j is not significantly associated with y.

However, the Breiman's VIMP scores can only serve as a reference measure rather than a deterministic decision rule for variable selection. For example, even though we can rank the predictor variables based on their Breiman's VIMP scores, due to the lack of standard testing procedure, we can not simply choose a cut off value for the VIMP scores and decide which predictor variables are associated and which are not. On the other hand, Breiman's VIMP scores may be inaccurate when the predictor variables are correlated. In particular, this inaccuracy may be resulted from the preference of correlated predictor variables in early splits of decision trees, as well as the permutation scheme used in computing the Breiman's VIMP scores (23).

4.1.2 Strobl's VIMP: motivation and one possible solution

As a reference measure, Breiman's VIMP score is known to have an embedded bias towards correlated predictor variables, which means that irrelevant predictor variables can have high VIMP scores if they have strong correlation with some other truly associated predictor variables. In practice, we want to avoid this kind of false positive detection and to judge variable importance without the misleading influence from the other covariates. Recall the Breiman's VIMP score for predictor variable j is calculated by independently permuting x_{j0}, which is equivalent to sampling from the marginal distribution of x_{j0}. In this way, the correlation of variable j with other predictor variables will affect the result and possibly lead to false detection. So, if we can calculate the VIMP scores by sampling from the conditional distribution of $x_{j0}|X_{-j}$, we will be able to avoid the influence due to variable correlation.

The concept of "conditional VIMP" is motivated by the idea of calculation variable importance while conditional on the other covariates (23). In the following sections, we will refer to their VIMP scores as the "Strobl's VIMP". Specifically, they propose a conditional permutation scheme as following:

1. Compute the before permutation OOB-accuracy based on x_{j0}.

2. For all variables to be conditioning on, extract the cut-points that split this variable in the current tree and create a grid by bisecting the sample space in each cut-point.

3. Within the grid, permute the values of x_{j0} and compute the after permutation OOB-prediction accuracy.

4. Take the difference between the before and after permutation prediction accuracies as the importance score of variable x_{j0} for one tree. And the Strobl's VIMP

score of variable j for the forest is computed as the average difference over all trees.

Comparing to the Breiman's VIMP, the Strobl's VIMP scores serve as better measurements because they can adjust for variable correlation when evaluating variable importance. In the later sections we will present a simulation example where we compare these two VIMP scores on a random forest model with high variable correlation among a subset of the predictor variables, and we will see that the Strobl's VIMP scores are able to adjust for variable correlations, even though they do not solve the problem completely and are still biased.

4.1.3 Review on VIMP statistical test and variable selection

Along with the proposed Breiman's VIMP scores, they also suggested a simple significance test based on the normality of z-scores developed by scaling the permutation importance (3). In particular, the z-score for variable j is calculated as

$$z_j = \frac{VIMP_{Breiman}(x_{j0})}{\hat{\sigma}/\sqrt{ntree}}, \tag{4.1}$$

where $VIMP_{Breiman}(x_{j0})$ is the Breiman's VIMP score for variable j, $ntree$ is the number of trees in the forest, and $\hat{\sigma}$ is the observed standard deviation of the p VIMP scores. However, this test has some strange statistical properties and might be questionable (22).

There are also other approaches for random forests variable selection. For example, backward elimination by throwing out least important variables until OOB prediction accuracy dropped (8); applying plots and significance test by randomly permuting the response values to mimic the overall null hypothesis that none of the predictor variable was relevant (7, 19) . However, all of these approaches, together with the

simple significance test as indicated in 4.1, were biased and had preference of correlated predictor variables (1).

In general, there were two basic strategies for random forest variable selection depending on different selection objectives, either to find important variables highly related to the response variable for interpretation purpose, or to find a small number of variables sufficient to a good parsimonious prediction of the response variables (11). Based on the two selection objectives, variables could be selected either by constructing nested random forest models and chose the one with the smallest OOB error, or constructing an ascending sequence of random forest models and chose the one by invoking and testing. However, they all required fitting of numerous random forest models and comparing certain model fit statistics, which were computationally intensive. Moreover, the concept of 'better interpretation' was somehow arbitrary and case specific.

4.2 Methods

4.2.1 An alternative approach to calculate conditional VIMP

In order to calculate the variable importance conditioning on all the other predictor variables, we first propose an approach to estimate the conditional distribution of $x_{j0}|X_{-j}$:

1. For a single decision tree, fit a model to predict x_{j0} based on X_{-j}. This could be a simple linear regression model or a random forest model, depending on the data structure encountered.

2. Let $\hat{x}_{i,j}$ denote the predicted value of observation i based on the model from step 1, and $\hat{\epsilon}_i$ denote the corresponding residual. Define $x^*_{i,j} = \hat{x}_{i,j} + \hat{\epsilon}^*_i$, where $\hat{\epsilon}^*_i$'s are a random permutation of the $\hat{\epsilon}_i$'s.

3. Let $x_j^* = \{x_{i,j}^*\}$, which can be viewed as a random sample drawn from the conditional distribution of $x_{j0}|X_{-j}$.

By using x_j^* instead of an independent permutation of x_{j0}, which was denoted as x'_{j0} in section 1.2.2, now we are sampling from the conditional distribution of $x_{j0}|X_{-j}$ instead of the marginal distribution of x_{j0}. In this way, we adjust for variable correlation and are able to obtain the conditional VIMP scores by following the same manner of calculating the Breiman's VIMP, as briefly described in section 1.2.2. In the following sections, we will refer to our proposed VIMP scores and the "conditional VIMP".

4.3 Simulation example of various VIMPs comparison

As we have discussed before, the Breiman's VIMP scores may be inaccurate and have bias towards correlated predictor variables. Specifically, irrelevant variables may have high VIMP scores if they are strongly correlated with some truly associated variables. The Strobl's VIMP scores tried to adjust for the variable correlation and did reduce the bias, but not completely solve the problem. In this section, we use a simulation example to compare the performance of the three VIMP scores (23).

In this simulation example, we have 12 predictor variables labelled as $X_1, X_2, \ldots, X_{12}$, following normal distribution with mean 0 and covariance matrix $\Sigma = \{\sigma_{i,j}\}$, denoted by $N(0, \Sigma)$. $\sigma_{i,i} = 1$ for $i = 1, 2, \ldots, 12$; $\sigma_{i,i'} = 0.9$ for $i, i' \leq 4, i \neq i'$; and $\sigma_{i,j} = 0$ otherwise. The outcome variable y is simulated in the following manner:

$$y_i = 5x_{i,1} + 5x_{i,2} + 2x_{i,3} - 5x_{i,5} - 5x_{i,6} - 2x_{i,7} + \epsilon_i, \tag{4.2}$$

where $\epsilon_i \sim N(0, 0.5)$.

According to the set up, the true model should only contain predictor variable X_1, X_2, X_3, X_5, X_6, and X_7, where the other 6 variables are irrelevant and should be

excluded. Note that variable X_4 is irrelevant, but it is strong correlated with variable X_1, X_2, and X_3. We simulated 100 data sets, and fit a random forest model to each simulated data set, and calculate three types of VIMP scores, including the Breiman's VIMP, the Strobl's VIMP, and the conditional VIMP proposed in Section 4.2.1. Table 4.5 summarized the median VIMP scores from 100 simulations. The comparison results were also provided in Figure 4.5, with the median VIMP scores plotted together with the 25th and 75th percentiles..

According to the results, the Breiman's VIMP scores had a bias toward correlation, and the predictor variable X_4 had high VIMP scores because of its strong correlation with variable X_1, X_2, and X_3, despite the fact that it is not associated with Y at all. The Strobl's VIMP scores were able to adjust for variable correlation, and the VIMP scores for variable X_4 went lower, but they were still above zero and even higher than the VIMP score for variable X_7, which was actually associated with Y. Among the three VIMP scores, the conditional VIMP proposed in our method successfully selected the truly associated predictor variables, and the VIMP scores for variable X_4 were zero, just as desired. The results suggested that the conditional VIMP scores proposed in our method could better adjust for variable correlation and reveal the true statistical association between the predictor variables and the modelling outcome, compared to the Breiman's and the Strobl's VIMP scores.

In conclusion, our conditional VIMP scores were the only method that was able to identify X_4 as a non-significant covariate, despite its high correlation with with variable X_1, X_2, and X_3; while still identify X_7 as truly associated. The Strobl's VIMP scores could adjust for variable correlation and avoid bias to a certain level, compared to the Breiman's VIMP scores; but our conditional VIMP scores had the best performance among the three.

4.4 Real data application on OPPERA

OPPERA is a prospective cohort study on Temporomandibular Disorders (TMD), which are a set of painful conditions that affect the jaw muscles, the jaw joint, or both. Both TMD-free participants and chronic TMD patients were enrolled in the study. Each study participant completed a quarterly questionnaire, and participants who showed signs of first-onset TMD returned to the clinic for a formal examination. The median follow up period was 2.8 years. The data set contained 185 chronic TMD patients and 3258 initially TMD-free individuals, 260 of whom developed TMD before the end of the study. For a more detailed description of the OPPERA study, see (21) or (2).

Possible risk factors for TMD were measured in OPPERA at baseline, including socio-demographic characteristics like age and gender (6 total variables), clinical characteristics like jaw injury and parafunction (78 total variables), autonomic measurements like blood pressure and heart rate (44 total variables), psychosocial measurements like depression and anxiety (39 total variables), and quantitative sensory testing (QST) measurements (33 total variables) that evaluate participants' sensitivity to experimental pain. See (18, 10, 13, 17) for more detailed descriptions of these variables.

In this study, we applied random forest models to this OPPERA data set, and the aim was to identify potential risk factors that were associated with elevated risk of chronic TMD and the first-onset of TMD. We calculated two types of VIMP scores based on the random forest model, including the Breiman's VIMP and the conditional VIMP proposed in Section 4.2.1. Table 4.5 summarized the comparison results on the top 10 predictor variables with the highest Breiman's VIMP scores.

We can see that the results from the two types of VIMP scores were quite different. For example, in the study of chronic TMD, the variable of Pressure Pain Threshold on Temporalis had the 5th highest Breiman's VIMP score, but its conditional VIMP

score was below zero. As a matter of fact, the Pressure Pain Threshold was a set of 5 measurements, which measured pain sensitivity on 5 different spots on human body and were naturally highly correlated. The results suggested that its Breiman's VIMP was biased because of its strong correlation with Pressure Pain Threshold on Masseter and Pressure Pain Threshold on TMJ. Once we conditioning on the other predictor variables, the measurement of Pressure Pain Threshold on Temporalis itself did not bring extra information on chronic TMD, thus the conditional VIMP was low. Similarly, in the study of first-onset TMD, the variable Count of 20 Comorbid Pain Conditions (CPSQ) was a summary measure of a few other predictor variables. It had the highest Breiman's VIMP score as 100, but once conditioning on the other predictor variables, its conditional VIMP decreased to 12.2, which suggested that it was "important" mainly due to its correlation with the other variables, and its Breiman's VIMP score was biased.

4.5 Discussion

In this chapter, we focus on random forests variable importance and variable selection. As a reference measure, the original Breiman's VIMP scores have embedded bias towards correlation. The Strobl's VIMP score tried to adjust for variable correlation and did perform better than the Breiman's VIMP scores, especially in presence of variable correlation; however, the conditioning approach they proposed didn't not fully remove the misleading influence from the correlated variables, and their results were still biased, even though the level of bias were largely reduced.

In this chapter, we mainly discussed these two existing VIMP scores, and compared their performance with our proposed conditional VIMP scores in both simulated and real-world data sets. The results suggested that our conditional VIMP scores were the only method that was able to fully adjust for variable correlation, and to evaluate

variable importance while conditioning on the other covariates. In this way, we are able to judge variable importance without the misleading bias due to complex variable correlation.

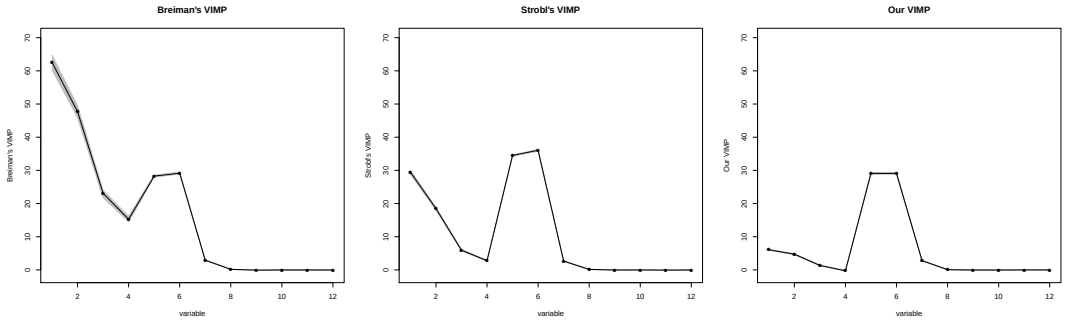

Figure 4.1: Simulation example of VIMP scores.

Among the 12 predictor variables, X_1, X_2, X_3, X_5, X_6, and X_7 are truly associated with the outcome variable Y, while the others are irrelevant. Three types of random forest VIMP scores are calculated, including the Breiman's VIMP, the Strobl's VIMP, and the conditional VIMP proposed in Section 4.2.1. The median of the VIMP scores from 100 simulations are plotted, with the 25% and the 75% percentiles.

Table 4.1: Simulation example: median VIMP across 100 simulations

Predictor variable	Breiman's	Strobl's	Conditional
X_1	62.61	29.49	6.07
X_2	47.73	18.51	4.74
X_3	23.11	5.95	1.36
X_4	15.25	2.78	-0.27
X_5	28.28	34.50	29.09
X_6	29.14	36.07	29.07
X_7	2.93	2.66	2.80
X_8	0.17	0.15	0.08
X_9	-0.07	-0.04	-0.04
X_{10}	-0.02	-0.02	-0.04
X_{11}	-0.02	-0.05	-0.01
X_{12}	-0.05	-0.03	0.00

Table 4.2: VIMP comparison for chronic/first-onset TMD

VIMP scores for chronic TMD		
Predictor variable	Breiman's	Conditional
Somatic Awareness (SCL 90R)	100	100
Pressure Pain Threshold on Masseter	93.7	56.5
Pressure Pain Threshold on TMJ	70.6	57.8
Somatic Awareness (PILL Global Score)	64.3	42.2
Pressure Pain Threshold on Temporalis	44.1	-7.8
Race	30.1	26.5
Global Psychological Distress (SCL 90R)	28.9	-3.1
Anxiety (SCL 90R)	24.8	-4.2
Sleep Quality (PSQI)	21.3	4.6
Catastrophization (PCS Helplessness Scale)	20.1	15.6

VIMP scores for first-onset TMD		
Predictor variable	Breiman's	Conditional
Count of 20 Comorbid Pain Conditions (CPSQ)	100	12.2
Bodily Pain (SF-12)	67.0	3.5
Somatic Awareness (SCL 90R)	51.2	24.8
Current Lower Back Pain (CPSQ)	48.1	4.0
OPPERA Study Site	37.6	50.0
Count of 6 Nonspecific Orofacial Symptoms (CPSQ)	30.7	100
Somatic Awareness (PILL Global Score)	30.2	28.9
Age	29.0	52.3
Sleep Quality (PSQI)	28.0	-3.1
Parafunctional Habits (OBC)	26.7	73.5

Chapter 5

Conclusion

Non-parametric machine learning methods are popular and widely used in many scientific research areas, especially in dealing with HDLSS data, or in presence of complex data structure and variable correlation. In this dissertation, we include three projects in studying non-parametric machine learning methods in clustering and variable selection.

We first look into the problem of biclustering, and develop a general framework for biclustering based on the sparse clustering method, named SC-Biclust. We also propose specific algorithms to identify biclusters with heterogeneous means and variances, or more general differences. We test the performance of the SC-Biclust algorithm in several simulated and real-world data sets, and compare with some existing biclustering algorithms. Our methods compare favourably with existing biclustering algorithms, in terms of prediction accuracy, reproducibility, and computing time.

As a follow up study, we then focus on the sparse clustering method, and point out a few limitations with the existing method in tuning parameter selection. We develop a new sparse clustering algorithm called ST-Spcl, which is able to identify observation clusters, and on top of that, to assign feature weights based on their individual contribution to the observation clusters. Our algorithm out-performs the existing sparse clustering algorithm in both simulated and real-world data sets, especially in presence of week cluster signals.

For the last project, we study on random forest variable importance and variable selection. Specifically, we propose an alternative approach to calculate the VIMP scores while conditioning on the other covariates. We test our method in both simulated and real-world data sets, and the results suggest that our conditional VIMP scores are the only method that was able to fully adjust for variable correlation, and to avoid bias due to complex variable correlation structure.

Bibliography

[1] Archer, Kellie J and Ryan V Kimes. 2008. "Empirical characterization of random forest variable importance measures." *Computational Statistics & Data Analysis* 52(4):2249–2260.

[2] Bair, Eric, Naomi C Brownstein, Richard Ohrbach, Joel D Greenspan, Ronald Dubner, Roger B Fillingim, William Maixner, Shad B Smith, Luda Diatchenko, Yoly Gonzalez et al. 2013. "Study protocol, sample characteristics, and loss to follow-up: The OPPERA prospective cohort study." *The Journal of Pain* 14(12):T2–T19.

[3] Breiman, Leo. 2001. "Random forests." *Machine learning* 45(1):5–32.

[4] Bühlmann, Peter and Bin Yu. 2002. "Analyzing bagging." *The Annals of Statistics* 30(4):927–961.

[5] Chen, Guanhua, Patrick F Sullivan and Michael R Kosorok. 2013. "Biclustering with heterogeneous variance." *Proceedings of the National Academy of Sciences* 110(30):12253–12258.

[6] Cheng, Yizong and George M Church. 2000. Biclustering of expression data. In *Ismb*. Vol. 8 pp. 93–103.

[7] Diaz-Uriarte, Ramón. 2007. "GeneSrF and varSelRF: a web-based tool and R package for gene selection and classification using random forest." *BMC bioinformatics* 8(1):328.

[8] Díaz-Uriarte, Ramón and Sara Alvarez De Andres. 2006. "Gene selection and classification of microarray data using random forest." *BMC bioinformatics* 7(1):3.

[9] Eisen, Michael B, Paul T Spellman, Patrick O Brown and David Botstein. 1998. "Cluster analysis and display of genome-wide expression patterns." *Proceedings of the National Academy of Sciences* 95(25):14863–14868.

[10] Fillingim, Roger B, Richard Ohrbach, Joel D Greenspan, Charles Knott, Ronald Dubner, Eric Bair, Cristina Baraian, Gary D Slade and William Maixner. 2011. "Potential psychosocial risk factors for chronic TMD: descriptive data and empirically identified domains from the OPPERA case-control study." *The Journal of Pain* 12(11):T46–T60.

[11] Genuer, Robin, Jean-Michel Poggi and Christine Tuleau-Malot. 2010. "Variable selection using random forests." *Pattern Recognition Letters* 31(14):2225–2236.

[12] Getz, Gad, Erel Levine and Eytan Domany. 2000. "Coupled two-way clustering analysis of gene microarray data." *Proceedings of the National Academy of Sciences* 97(22):12079–12084.

[13] Greenspan, Joel D, Gary D Slade, Eric Bair, Ronald Dubner, Roger B Fillingim, Richard Ohrbach, Charlie Knott, Flora Mulkey, Rebecca Rothwell and William Maixner. 2011. "Pain sensitivity risk factors for chronic TMD: descriptive data and empirically identified domains from the OPPERA case control study." *The Journal of Pain* 12(11):T61–T74.

[14] Hansen, Kasper Daniel, Winston Timp, Héctor Corrada Bravo, Sarven Sabunciyan, Benjamin Langmead, Oliver G McDonald, Bo Wen, Hao Wu, Yun Liu, Dinh Diep et al. 2011. "Increased methylation variation in epigenetic domains across cancer types." *Nature genetics* 43(8):768–775.

[15] Lazzeroni, Laura and Art Owen. 2002. "Plaid models for gene expression data." *Statistica sinica* 12(1):61–86.

[16] Lee, Mihee, Haipeng Shen, Jianhua Z Huang and JS Marron. 2010. "Biclustering via sparse singular value decomposition." *Biometrics* 66(4):1087–1095.

[17] Maixner, William, Joel D Greenspan, Ronald Dubner, Eric Bair, Flora Mulkey, Vanessa Miller, Charles Knott, Gary D Slade, Richard Ohrbach, Luda Diatchenko et al. 2011. "Potential autonomic risk factors for chronic TMD: descriptive data and empirically identified domains from the OPPERA case-control study." *The Journal of Pain* 12(11):T75–T91.

[18] Ohrbach, Richard, Eric Bair, Roger B Fillingim, Yoly Gonzalez, Sharon M Gordon, Pei-Feng Lim, Margarete Ribeiro-Dasilva, Luda Diatchenko, Ronald Dubner, Joel D Greenspan et al. 2013. "Clinical orofacial characteristics associated with risk of first-onset TMD: The OPPERA prospective cohort study." *The Journal of Pain* 14(12):T33–T50.

[19] Rodenburg, Wendy, A Geert Heidema, JM Boer, IM Bovee-Oudenhoven, EJ Feskens, EC Mariman, Jaap Keijer et al. 2008. "A framework to identify physiological responses in microarray-based gene expression studies: selection and interpretation of biologically relevant genes." *Physiological Genomics* 33(1):78–90.

[20] Shabalin, Andrey A, Victor J Weigman, Charles M Perou and Andrew B Nobel. 2009. "Finding large average submatrices in high dimensional data." *The Annals of Applied Statistics* pp. 985–1012.

[21] Slade, Gary D, Eric Bair, Kunthel By, Flora Mulkey, Cristina Baraian, Rebecca Rothwell, Maria Reynolds, Vanessa Miller, Yoly Gonzalez, Sharon Gordon et al. 2011. "Study methods, recruitment, sociodemographic findings, and demographic representativeness in the OPPERA study." *The Journal of Pain* 12(11):T12–T26.

[22] Strobl, Carolin and Achim Zeileis. 2008. "Danger: High power!–exploring the statistical properties of a test for random forest variable importance.".

[23] Strobl, Carolin, Anne-Laure Boulesteix, Thomas Kneib, Thomas Augustin and Achim Zeileis. 2008. "Conditional variable importance for random forests." *BMC bioinformatics* 9(1):307.

[24] Svetnik, Vladimir, Andy Liaw, Christopher Tong, J Christopher Culberson, Robert P Sheridan and Bradley P Feuston. 2003. "Random forest: a classification and regression tool for compound classification and QSAR modeling." *Journal of chemical information and computer sciences* 43(6):1947–1958.

[25] Tan, Kean Ming and Daniela M Witten. 2013. "Sparse biclustering of transposable data." *Journal of Computational and Graphical Statistics* . In press.

[26] Tibshirani, Robert, Guenther Walther and Trevor Hastie. 2001. "Estimating the number of clusters in a data set via the gap statistic." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63(2):411–423.

[27] van't Veer, Laura J, Hongyue Dai, Marc J Van De Vijver, Yudong D He, Augustinus AM Hart, Mao Mao, Hans L Peterse, Karin van der Kooy, Matthew J Marton, Anke T Witteveen, George J Shreiber, Ron M Kerkhoven, Chris Roberts, Peter S Linsley, Ren'e Bernards and Stephen H Friend. 2002. "Gene expression profiling predicts clinical outcome of breast cancer." *Nature* 415(6871):530–536.

[28] Weigelt, Britta, Zhiyuan Hu, Xiaping He, Chad Livasy, Lisa A Carey, Matthew G Ewend, Annuska M Glas, Charles M Perou and Laura J van't Veer. 2005. "Molecular portraits and 70-gene prognosis signature are preserved throughout the metastatic process of breast cancer." *Cancer research* 65(20):9155–9158.

[29] Witten, Daniela M and Robert Tibshirani. 2010. "A framework for feature selection in clustering." *Journal of the American Statistical Association* 105(490).

[30] Zou, Hui. 2006. "The adaptive lasso and its oracle properties." *Journal of the American statistical association* 101(476):1418–1429.