Bagged Gene Shaving for the Robust Clustering of High-Throughput Data

Bradley M. Broom*, Erik P. Sulman†, Kim-Anh Do‡, Mary E. Edgerton§, and Kenneth D. Aldape§

*Department of Bioinformatics and Computational Biology, UT MD Anderson Cancer Center, Houston, Texas 77030
†Department of Radiation Oncology, UT MD Anderson Cancer Center, Houston, Texas 77030
‡Department of Biostatistics, UT MD Anderson Cancer Center, Houston, Texas 77030
§Department of Pathology, UT MD Anderson Cancer Center, Houston, Texas 77030

Abstract—The analysis of high-throughput data sets, such as microarray data, often requires that individual variables (genes, for example) be grouped into small clusters such that all the variables in any given cluster have highly correlated values across all samples. Gene shaving is an established method for generating such clusters, but is overly sensitive to the input data: changing one sample out of hundreds can determine whether or not an entire cluster is found, even if the search is continued for many hundreds of clusters. This paper describes a clustering method based on the bootstrap aggregation (bagging) of gene shaving clusters, which overcomes this and other problems. The paper also describes the application of the new method to a large meta-data set of gene expression microarray data from brain tumor samples. Index Terms—bootstrap aggregation, clustering, gene shaving, glioblastoma.

I. INTRODUCTION

The analysis of high-throughput data sets, such as microarray data, often requires that individual variables (genes, for example) be grouped into small clusters such that all the variables in any given cluster have highly correlated values across all samples. Unlike hierarchical clustering in which all variables are recursively aggregated, no matter how different, into ever larger clusters to form a tree-like dendrogram, we are interested only in clusters containing variables with very similar measurements in all samples. Such clusters are sometimes called meta-genes or meta-variables, which we use to denote the average measurement for each sample of all the variables in the cluster. Figure 2 below shows an example of a gene cluster. One use of these clusters is to infer the relatedness of individual variables from their membership in a common cluster. A second use is to suggest possible functions for individual variables of interest, based on the functions of other variables in the cluster, and to suggest additional related variables that might also be of interest. A third use is to calculate a cluster meta-variable for each sample by averaging the individual variables in the cluster. The cluster meta-variables might yield more robust measurements and tests of sample characteristics than the individual variables. Converting the individual variables into meta-variables also reduces the number of variables, and makes searching high dimensional spaces for interaction effects more tractable.

A. Gene Shaving

Gene shaving [6] is an established method for generating such clusters. Gene shaving is so named because it was first developed in the context of analyzing gene expression microarrays, which measure tens of thousands of probe sets for each sample. Each probe set measures the amount of messenger RNA in a sample for a specific gene, but any particular microarray design may contain zero, one, two, or more probe sets for any specific gene. Despite its name, gene shaving can be applied to any high-throughput data set containing continuous data.

The gene shaving method begins by computing the first principal component of the data and ranking all variables (genes) by their correlation with that principal component. It then shaves off, or removes, a small number of the variables that are the least correlated with that principal component, recomputes the principal component of the reduced data set, and ranks the remaining variables according to the new principal component. The shaving process is continued until only two variables remain. We call the set of variables remaining at each step in this process a shaving.

The cluster itself is chosen from the set of shavings such that it maximizes the gap statistic between the shavings based on the original data and the shavings based on randomly permuted data. Multiple, independently permuted copies of the data are used to estimate the gap statistic.

Unlike other clustering methods, gene shaving allows both positively and negatively correlated variables to belong to the same cluster. (This is not shown in any of the examples in this paper.) When computing the cluster’s meta-variable the sign of each variable’s correlation must also be considered, so we use the terms signed meta-variable or signed meta-gene.

Once the cluster is chosen, the data set is orthogonalized with respect to the cluster’s meta-variable. Additional clusters can then be generated by repeating the entire process until a fixed number of clusters have been generated, or the “optimal” cluster would contain all remaining variables. The idea behind orthogonalization is that it removes the effect of the generated cluster from the data, but the variables stay in the data set so that, potentially at least, individual variables may belong to multiple clusters.

We developed a fast implementation of gene shaving called GeneClust [4], consisting of a user-friendly Java based user interface, a fast C implementation of gene shaving, and an R library interface to the C function. We have used GeneClust to analyze numerous high-throughput data sets.
B. Limitations of Gene Shaving

In our experience, gene shaving is overly sensitive to the input data. For example, we applied gene shaving to a large data set containing hundreds of glioma samples. In an early attempt to estimate the sensitivity of gene shaving to the data, we performed leave-one-out cross-validation and noticed that in many cases entire clusters would appear in some data sets but not others, even if we generated hundreds of clusters from each data set. In this example, a known cluster containing genes of interest did not appear in the clusters generated from the entire data set, but did appear as one of the first clusters generated in many of the data sets from which one sample had been left out.

We believe this occurs because the first principal component that is selected is indicative of a general trend across a large number of genes, but the shaving and cluster selection process narrows this down to a much smaller, very specific set of genes. The orthogonalization of the data set after the cluster is selected not only removes the effect of the selected cluster’s meta-variable, but also greatly reduces the influence of similar variables, even though these variables have not been assigned to any cluster, such that they will never be selected as a cluster. Thus, we believe that when the data is changed slightly, the first principal component is also changed only slightly, but the shaving process narrows down to a similar but ultimately different cluster. The orthogonalization of the data set then allows only whichever cluster was chosen first to be detected, with all other similar clusters orthogonalized away. Since the initial principal component determined for the first cluster may reflect a major trend over most of the variables in the data set, we view this exclusion of all related variables as a major problem.

Although the stated reason for orthogonalizing the data set, instead of removing the variables concerned from further consideration, is to allow individual variables to appear in more than one cluster, we have rarely observed this in practice. Thus, orthogonalization has a significant detrimental impact for little benefit.

Another limitation of gene shaving is that individual variables either belong to a cluster or they do not. It would be helpful to have some measure of how sensitive a variable’s membership in a specific cluster is to the data. In many cases, the gap curve for determining the size of a cluster shows two distinct peaks, the relative heights of which will determine whether the generated cluster is small or large. We believe these two peaks correspond to a group of core variables and another group of very similar but slightly distinct variables. Standard gene shaving will make this decision without considering the distribution of the input data, and if the smaller cluster is chosen will not output the additional variables in any cluster due to orthogonalization.

C. Outline

The following section describes a clustering method based on the bootstrap aggregation (bagging) [2] of gene shaving clusters, which overcomes the limitations of gene shaving described above.

Section III describes the application of the new method to a large meta-data set of gene expression microarray data from brain tumor samples.

Section IV concludes the paper and discusses topics for further research.

II. BAGGED GENE SHAVING

The basic idea of bagged gene shaving is very simple:
1) generate multiple bootstrap resamples of the data set,
2) run gene shaving independently on each resample,
3) cluster variables that frequently co-occur in the output of the different resamples.

The specifics of the first step will depend on the data set concerned, so we will defer any discussion of resampling to discussion of our brain cancer example in section III.

The second step is also straight-forward, although we note that even without orthogonalization of the data set after each cluster is generated, individual variables can sometimes occur in one cluster and sometimes in another. In our brain cancer example, we chose to eliminate the variables in a cluster from the data set instead of orthogonalizing against the cluster meta-variable. In the resulting output, variables frequently occur in different clusters.

The final step is the most challenging, in large part because of the fact that variables frequently occur in different clusters. The first challenge, however, is that there is not necessarily any correspondence between similarly numbered clusters generated from different resampled data sets. To determine the degree to which two variables, \( i \) and \( j \), co-occur we therefore compute a variable adjacency matrix, \( A_{i,j} \). Matrix elements are non-negative real values, with zero indicating the two variables never occur in the same cluster. Larger values mean the two variables co-occur more frequently and/or do so in better clusters, and are thus more adjacent.

To reduce the influence of poor clusters, we incorporate two measures of cluster quality into the adjacency matrix. The first is the percent variance, \( R_{K} \), explained by a cluster \( K \). The total variance, \( V_{T,K} \), of the cluster can be decomposed into the variance within the samples, \( V_{W,K} \), and the variance between the samples, \( V_{B,K} \). The percent variance explained by the cluster is then \( R_{K} = 100 \times V_{B,K}/V_{T,K} \). The second quality measure is the gap statistic, \( G_{K} \), which measures the difference between the percent variance explained by generated cluster, \( R_{K}^{2} \), and the percent variance explained by similarly sized clusters obtained from randomly permuted data.

Thus, the adjacency matrix is defined by

\[
A_{i,j} = \sum_{K|K \in \mathcal{K}, j \in K} G_{K}R_{K}^{2}.
\]

That is, the adjacency between two variables is the sum of \( G_{K}R_{K}^{2} \) over all clusters \( K \) that contain both variables.

Having obtained the adjacency matrix, we must now extract clusters from it. This raises the question of what we mean by a cluster, especially when some variables associate to a
greater or lesser degree with a wide range of other variables. We define a cluster to be a set of variables that co-occur with similarly frequency and which show similar patterns of co-occurrence with other variables. Variables will belong to at most one cluster. (This restriction could potentially be lifted. See section IV for further discussion.)

To extract a cluster, we find the two variables with the largest entry in the adjacency matrix. (In the case of a tie, an arbitrary pair of variables is chosen.) We then expand the cluster by considering additional variables whose adjacency to any current member of the cluster exceeds a fixed fraction of the new variable’s highest adjacency. That is, we exclude variables that are weakly adjacent to the current cluster, but which are much more strongly adjacent to other variables. We also require the new variable to be adjacent to all existing members. The cluster is expanded in this way until no other suitable variables are available.

The potential cluster may contain variables that are strongly adjacent to all other members of the potential cluster, but which show a different pattern of adjacency strengths. This occurs when there are multiple distinct but strongly adjacent groups of variables. We chose to separate these into distinct clusters. If the potential cluster is large enough (at least 8 variables), we compute the Pearson correlation of the adjacency strengths within the potential cluster between the first variable in the cluster and with every other variable, and eliminate any variable for which the p value of the correlation exceeds 0.05. (This sometimes results in a cluster containing just the first variable. Such clusters are discarded in a subsequent step.)

The remaining variables are output as a cluster, eliminated from further consideration, and the process repeated until all variables have been used or a sufficiently large number of clusters have been generated.

The above method generates cohesive clusters, but the order in which they are generated depends only on the maximum adjacency and not at all on cluster size. We define the quality of a cluster $K$, $Q_K$, to be the mean adjacency within the variables belonging to the cluster multiplied by the number of variables in the cluster:

$$ Q_K = 1/|K| \sum_{i,j \in K} A_{i,j}. $$

The generated clusters are sorted in order of decreasing $Q_K$.

Adjacencies between clusters are summarized by computing a cluster adjacency matrix, $B_{K_1 \times K_2}$. Each entry in the cluster adjacency matrix is the mean of the adjacencies of the variables in the two clusters concerned:

$$ B_{K_1 \times K_2} = \frac{\sum_{i \in K_1} \sum_{j \in K_2} A_{i,j}}{|K_1| \times |K_2|}. $$

Our earlier implementation of gene shaving, GeneClust, could extract a few tens of clusters from data sets containing up to about 100 samples in a reasonable time, but had a computational time complexity of $O(N^3)$ in the number of samples. Performing gene shaving on a large number of bootstrap resamples of our much larger data set was computationally infeasible.

To enable all of the computations required to be completed in a reasonable time, we have developed a much more efficient command line based implementation. This implementation uses multiple threads to take advantage of the multiple processors available on current workstations and cluster compute nodes. It is also highly optimized to make better use of cache and has improved locality of memory references. Although still $O(N^3)$, it is fast enough on current quad-processor machines to use on data sets containing about 1000 samples.

III. Application To Glioma Dataset

We applied bagged gene shaving to a large collection of gene expression microarray data obtained from infiltrating gliomas (WHO grade II-IV).

A. Dataset

The data consists of Affymetrix gene expression microarray results from a total of 739 infiltrating gliomas (WHO grade II-IV). The samples came from 10 different institutions and were measured on three different platforms (U133A, U133Plus2, and HT_U133A). Since the samples from some institutions were collected under different conditions, including different platforms, for the purpose of detecting and mitigating batch effects [1], we divided the samples into 15 distinct batches, which are summarized in table I. We split the data into a training set of 442 samples and a validation set of 297 samples, such that a similar proportion of training and test cases occur in each batch and in each disease grade.

B. Data preprocessing and bootstrap resample generation

Due to advances in annotation of the genome and transcriptome, the original Affymetrix probe set definitions are inaccurate [3]. Consequently, all of the original CEL files were quantified using updated probe set definitions based on

<table>
<thead>
<tr>
<th>Batch</th>
<th>Training Samples</th>
<th>Test Samples</th>
<th>Platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>AO-LBL</td>
<td>18</td>
<td>14</td>
<td>HT_U133A</td>
</tr>
<tr>
<td>AO-MDA</td>
<td>11</td>
<td>13</td>
<td>U133A</td>
</tr>
<tr>
<td>Belgium</td>
<td>12</td>
<td>9</td>
<td>U133Plus2</td>
</tr>
<tr>
<td>Collins</td>
<td>39</td>
<td>22</td>
<td>U133A</td>
</tr>
<tr>
<td>Duke</td>
<td>21</td>
<td>10</td>
<td>U133A</td>
</tr>
<tr>
<td>EORTC</td>
<td>40</td>
<td>25</td>
<td>U133Plus2</td>
</tr>
<tr>
<td>HF</td>
<td>68</td>
<td>37</td>
<td>U133Plus2</td>
</tr>
<tr>
<td>MDA</td>
<td>31</td>
<td>20</td>
<td>U133A</td>
</tr>
<tr>
<td>MDA-new</td>
<td>25</td>
<td>13</td>
<td>U133Plus2</td>
</tr>
<tr>
<td>TCGA</td>
<td>60</td>
<td>50</td>
<td>HT_U133A</td>
</tr>
<tr>
<td>UCLA</td>
<td>51</td>
<td>31</td>
<td>U133A</td>
</tr>
<tr>
<td>UCLA3</td>
<td>22</td>
<td>20</td>
<td>U133A</td>
</tr>
<tr>
<td>UCLA4</td>
<td>6</td>
<td>3</td>
<td>U133Plus2</td>
</tr>
<tr>
<td>UCLA5</td>
<td>21</td>
<td>16</td>
<td>U133A</td>
</tr>
<tr>
<td>UCSF</td>
<td>17</td>
<td>14</td>
<td>U133A</td>
</tr>
</tbody>
</table>
version 11 of the Entrez Gene definition. Using these new probe set definitions, there are 11911 probe sets common to the three different platforms used. Unlike the original Affymetrix probe set definitions, each gene is associated with at most one probe set. We will therefore use the term ‘gene’ to include the meaning ‘probe set’ in the remainder of this paper.

To reduce batch effects, the quantified data was combined as follows. The gene measurements for each sample were ranked from 1 to 11911. For the training data, the ranks for each gene were then ranked across the samples in its batch, and scaled to the range 0 to 1 by dividing by the number of samples in the batch. For the test data, each test sample was added individually to the training data for its batch, and the gene ranks were ranked across samples and scaled as described for the training data, and the scaled ranks for the test sample were recorded. (The scaled ranks for the training data obtained when the test sample was included were discarded.)

For resample generation, the final ranking across samples was weighted according to a draw from the Bayesian bootstrap distribution [8] instead of uniformly. We used Bayesian bootstrap resampling because it was easy to combine with our batch effect mitigation strategy and because it has better theoretical properties for small sample sizes, such as for some of the batches.

C. Bagged Gene Shaving

Bagged gene shaving was performed on the training data as described above using our optimized gene shaving software. We used MD Anderson’s large computational cluster to perform gene shaving on multiple resamples at once. The gene shaving for each resample was executed on a single four-processor compute node and required approximately ten hours of elapsed time. We used a total of 256 resamples.

For each resample, we generated a predetermined number of 600 clusters, removing the genes in each discovered cluster instead of orthogonalizing against the cluster mean gene. On average, the generated clusters contained a total of 10740 genes per resample. We then combined the cluster information to generate the raw gene adjacency matrix, from which we extracted a predetermined number of 1000 clusters. Of these, 193 contained only a single gene, leaving 807 clusters, which were then ranked across the samples in its batch, and the scaled ranks for the test sample were recorded. (The scaled ranks for the training data obtained when the test sample was included were discarded.)

For resample generation, the final ranking across samples was weighted according to a draw from the Bayesian bootstrap distribution [8] instead of uniformly. We used Bayesian bootstrap resampling because it was easy to combine with our batch effect mitigation strategy and because it has better theoretical properties for small sample sizes, such as for some of the batches.

D. Comparison to Unbagged Shaving

To compare the cluster quality obtained by the two methods, we clustered the 297 test samples using the clusters found by each method when applied to the training data, and calculated the percent variance explained, $R^2_k$, for each cluster. Figure 1 shows a histogram comparing the results. The results for the bagged gene clusters are significantly higher than those for unbagged gene shaving (a mean of 59 versus 38).

E. Results

For reference, figure 2 shows the first cluster. The samples (columns) are sorted so that the signed mean gene increases from left to right. The top two rows show color encoded representations of the batch the sample came from and the chip type used. Neither shows any obvious evidence of batch effects.
genes belong to a different cluster than the one to which they usually belong. Another possibility is that in some resamples the genes from both clusters are combined into a single, large cluster. Whether clusters 1 and 7 should remain separate or be combined into a single large cluster is a matter of degree. It is clear that the other clusters in figure 4 are distinct, even though there is a little adjacency between them. We believe that the separation of clusters 1 and 7 is also appropriate, since the adjacency within each is significantly higher and more consistent than the adjacency between them.

The cluster adjacency matrix records the average of the gene adjacency either within a cluster, or between any two clusters. Figure 5 shows the cluster adjacency matrix for the first 100 clusters. Cluster 1 is on the bottom left and cluster 100 on the top right. Adjacency between clusters 1 and 7 is clearly visible (on a suitably magnified view), as is adjacency between clusters 7 and 16.

The cluster adjacency matrix can be easily queried to determine the clusters that are most adjacent to any cluster of interest. Figure 6 shows the adjacency between the genes in the 10 clusters that have the highest average adjacency with cluster 1. Note that in this example, the clusters are sorted by their average adjacency with cluster 1, not by any measure of their own size or strength. From bottom left to top right, the clusters concerned are numbers 1, 7, 16, 26, 134, 81, 464, 201, 20, and 36. The fifth through eighth clusters in this example are relatively small and hard to discern.

To evaluate the correctness of the clusters, especially those with relatively low adjacency scores that are detected towards the end of the process, we consider separately false positives, in which a gene is assigned to a cluster to which it doesn’t actually belong, and false negatives, in which a gene is not assigned to a cluster to which it does belong. Answering this question is hard, because a cluster is (in general) an artificial construct, invented for our convenience, that means the variables in the cluster are highly correlated. It is not
the case that variables in different clusters are uncorrelated. Indeed, the adjacency between clusters 1 and 7 in figure 4 is indicative of a reasonable degree of correlation between the variables concerned.

When applied to random noise, many other clustering methods, such as hierarchical clustering, will happily produce many alleged clusters, but gene shaving is very unlikely to generate more than one or two such clusters and any that it does produce will have a very small gap statistic, \( G_k \), since that already includes a comparison to random permutations of the data. The bagged version of gene shaving is likely to spread any such weak, random adjacency across the adjacency matrix.

If the original Affymetrix probe set definitions are used, a useful consistency check is available, since many genes are measured by multiple probe sets. In earlier analyses based on these probe sets, we detected many clusters containing multiple probe sets for the same gene, and detected many clusters consisting solely of different probe sets for a single gene. Unfortunately, the revised probe set definitions used in this example contain at most one probe set for each gene, so such a check is not possible.

Nevertheless, there are several clusters containing only very similar genes. Cluster 207, for example, contains only the genes HOXD9, HOXD10, HOXD11, and HOXD13. Cluster 208 contains only the genes HIST1H2BF, HIST1H2BG, HIST1H2AE, and HIST1H3D. Cluster 685 contains only the genes FAM128A and FAM128B. Cluster 695 contains only the genes MT2A and MT4. These results suggest that many hundreds of bagged clusters are potentially indicative of true relationships.

In addition to clusters containing only very similar genes (or multiple probe sets for the same gene if using the original Affymetrix probe sets), we found two other major types of cluster. The first type consists of multiple genes related primarily by function. For example, cluster 1 is associated with the cell cycle, and contains many genes associated with nuclear replication, including the centromere proteins CENPA and CENPN, the kinesins KIF2C, KIF4A, KIF14, KIF20A, and KIF23, and the cell cycle genes CCNA2, CCNB1, and CCNB2.

The second type consists of multiple genes related by genomic location and not by function. For example, cluster 24 contains 17 genes all located on chromosome 10, while cluster 50 contains 15 genes all from cytogenetic band 19p13. These suggest the expression levels of the genes concerned are directly determined by a genome level alteration, such as a copy number change or epigenetic modifications to a large region of DNA. Chromosome 10, for example, is well known to be partially or completely lost in many glioblastomas [5]. (We do not regard clusters containing only very similar genes from the same genomic location, such as clusters 207 and 208 described above, as indicative of genome related expression levels.)

In addition to cluster 24, there are several additional clusters containing only genes from chromosome 10. Figure 7 shows the gene adjacency matrix for these clusters. The first cluster (number 24) is much stronger than the others. It is adjacent to the genes in the fifth cluster shown, although these adjacencies are much weaker than those within cluster 24. The remaining clusters are all fairly small, many containing only two genes, and adjacent among themselves. There is little adjacency between these clusters and the first and fifth clusters.

In this data set, we found clusters containing only genes from a single chromosome for 9 different chromosomes. Figure 8 shows the gene adjacency between the first cluster specifically associated with each of these chromosomes. Starting at the bottom left, these clusters are associated with chromosomes 10, 19, Y, 7, 17, 22, 14, 9, and 20. There is strong adjacency within each cluster, but virtually none between the clusters.

We compared the clusters from bagged gene shaving to the clusters obtained by applying gene shaving with orthogonalization to the original data. Bagged cluster 1 contains mostly genes from unbagged cluster 2, with ten additional genes not present in any of the unbagged clusters. These
additional genes are highly aligned with the functional aspects of the genes in bagged cluster 1. They include kinesian genes KIF14 and KIF23, cell cycle control genes CDC2, CDC6, and CDC20, a gene associated with the initiation of DNA replication (GINS1), a gene thought to be involved in the inhibition of spliceosome assembly during mitosis (MELK), a gene thought to be cell cycle regulator (DLGAP5), as well as two other genes (KIAA0101, TRIP13) whose function in DNA replication or cell cycle control is not known.

Unbagged cluster 2 contains mostly genes (46) from bagged cluster 1, but it also contains all of bagged clusters 7 (15 genes), 16 (8 genes) and 26 (8 genes). The genes in bagged cluster 7 are also associated with cell division and includes the centromere genes CENPE and CENPM, kinesin KIF11, and kinesin-8 related gene ZWINT. Bagged cluster 16 includes cell division cycle genes CDC43, CDC25C, and CDC45L, centromere gene CENPF. Bagged cluster 26 contains one kinesin gene (KIF15), a kinesin-8 component (SPC25), a component of the minichromosome maintenance complex (MCM10), a gene that regulates centriole duplication (PLK4), and genes associated with DNA repair (NEIL3), neuronal proliferation (RACGAP1), and chromatin condensation (NCAPH). All of these clusters are clearly related with nuclear replication. The question remains whether their further division into 4 subclusters is warranted.

In contrast, bagged cluster 2 contains 42 genes, of which 6 are present in unbagged cluster 12 and 1 is from unbagged cluster 1. The 6 genes in unbagged cluster 12 are LAPT5M (lysosomal-associated multispanning membrane protein), TYROBP (transmembrane signaling polypeptide), RNASE6 (ribonuclease), PTPRC (protein tyrosine phosphatase), ITGB2 (integrin chain component), and HCL1A (substrate of antigen receptor-coupled tyrosine kinase).

The remaining 35 genes, listed in table II are not output in any unbagged cluster. Figure 9 shows a heatmap of bagged cluster 2, including these 35 genes. The high percent variance explained by the cluster, the high adjacency between the genes in the cluster (figure 4), and the common theme of known immune system related function among many of the genes concerned, strongly suggests that it is a valid cluster.

We speculate they are not discovered by unbagged gene shaving because the genes in this cluster are somehow correlated with those in the first unbagged cluster. Indeed one gene, ARPC1B, from bagged cluster 1 occurs in unbagged cluster 1. Orthogonalization against cluster 1 makes it hard to detect that these genes are clustered, and further orthogonalization against those few genes found in cluster 12 makes it impossible.

IV. Conclusions

In this paper we described several issues that arise when applying the gene shaving clustering method to large high-throughput data sets. These include the sensitivity of the generated clusters to the input data, in particular the fact that large clusters are simply not found by gene shaving, as well as the lack of information concerning the sensitivity of membership in those clusters that are found to the input data.
We described the bagged gene shaving method for overcoming these issues by using bootstrap aggregation of gene shaving results from multiple bootstrap resamplings of the original data, and briefly described a high-performance implementation of gene shaving that makes such computations feasible.

We applied bagged gene shaving to a large, multi-institutional data set of infiltrating glioma samples and showed that bagged gene shaving finds large gene clusters not found by unbagged gene shaving. We also showed that genes that unbagged gene shaving simply lumps together into a single cluster have a definite and pronounced substructure that is resolved into distinct clusters by bagged gene shaving.

The gene clusters found by bagged gene shaving include small clusters of similar genes, large clusters of functionally related genes, and clusters of genes related only by genomic location.

The generated gene clusters can be used in many ways not described in this paper to gain insight into the biology of infiltrating gliomas. The signed meta-gene score computed for each cluster may be more reliable than using individual variables for detecting significant correlations between high-throughput data measurements and sample covariates of interest, such as patient survival. The smaller number of meta-genes also makes searching for high-dimensional interaction effects more tractable.

Methods such as Bayesian network analysis, for example, can be used to obtain a more global view of the significant interactions between variables, but do poorly when applied to data sets containing many highly-correlated variables, since the choice of one specific variable from a set of highly correlated variables will be driven principally by noise. Reducing several highly-correlated variables to a single meta-variable not only eliminates the correlation related problems, but also enables data sets containing more variables to be analyzed.

When generating clusters from the variable adjacency matrix, we required that variables belong to at most one cluster. The final clusters we generated also contained only a fraction of the variables in the entire data set, even though the gene shaving of each individual bootstrap resampling assigned nearly all variables to clusters. The current clusters could be expanded to include all variables that are more weakly adjacent to the existing variables in the cluster than the current, core members. We are not convinced by the benefits of such an approach. It would destroy the distinction between clusters 1, 7, 16, and 26, which were lumped together by the unbagged method. The current definition produces tight, consistent clusters, and the information about genes that co-occur in multiple clusters is available from the cluster adjacency matrix. Since variables that do not co-occur strongly enough to frequently join the same cluster are much more likely to be false positives, we believe that they should not be included.

Further tools are required to simplify understanding of the hundreds of generated clusters and their interactions. We have fed the genes in individual clusters into various gene ontology and pathway analysis tools but have not found them particularly enlightening compared to quickly scanning the biological roles of the genes concerned.

We would like to develop a database of clusters found across multiple diseases (at least cancers) so that we can more easily identify gene groupings that are common to many diseases and those that are specific to a particular disease. For example, finding a cluster of genes, such as cluster 1, related to proliferation in a cancer data set is not interesting, it is expected. More interesting would be to find genes in that cluster that do not belong to proliferation clusters from any other diseases.

ACKNOWLEDGMENT

This work was supported in part by the NIH/NCI SPORE in Brain Cancer (PP-3A) 1P50CA127001 01A1.

REFERENCES