UNSUPERVISED LEARNING FROM COMPLEX DATA: THE MATRIX INCISION TREE ALGORITHM

Ju Han Kim, M.D., Ph.D.
Children's Hospital Informatics Program, Children's Hospital, Harvard Medical School
300 Longwood Avenue, Boston, MA 02115, USA, juhan_kim@harvard.edu

Lucila Ohno-Machado, M.D., Ph.D.
Decision Systems Group, Brigham and Women's Hospital, Harvard Medical School
75 St. Francis Street, Boston, MA 02115, USA, machado@dsg.harvard.edu

Isaac S. Kohane, M.D., Ph.D.
Children's Hospital Informatics Program, Children's Hospital, Harvard Medical School
300 Longwood Avenue, Boston, MA 02115, USA, issac_kohane@harvard.edu

Analysis of large-scale gene expression data requires novel methods for knowledge discovery and predictive model building as well as clustering. Organizing data into meaningful structures is one of the most fundamental modes of learning. DNA microarray data set can be viewed as a set of mutually associated genes in a high-dimensional space. This paper describes a novel method to organize a complex high-dimensional space into successive lower-dimensional spaces based on the geometric properties of the data structure in the absence of a priori knowledge. The matrix incision tree algorithm reveals the hierarchical structural organization of observed data by determining the successive hyperplanes that 'optimally' separate the data hyperspace. The algorithm was tested against published data sets yielding promising results.

1. Introduction

A general question in many research areas is how to organize observed data into meaningful structures. DNA microarray technology constitutes a challenge to biomedical researchers, who wants to extract important biological information from the mRNA expression data set of very high dimensionality. Increasing number of methodologies for functional genomic clustering are being introduced to explore the extensive (and largely unlabeled) genomic data sets.

Cluster analysis involves the utilization of a collection of different algorithms to create hypothesized clusters. In exploratory phases of research, cluster analysis can be used to explore the underlying structure of the data and generate hypotheses. Because cluster analysis is essentially an exploration of the internal structural organization of observed data, there may be no single 'best' set of clusters or 'gold standard' and clustering algorithm is categorized as unsupervised learning.

In functional genomics, a typical first step in genomic data exploration is to examine the expression fold-differences before and after certain interventions. Genes with fold differences greater than a given threshold are considered clustered with the intervention[1,2].

Another way to approach the problem is to utilize algorithms that comprehensively compare all objects against each other to build either phylogenetic-type hierarchical trees[3,4] or other graphical representation of clusters, such as relevance networks using a variety of similarity or distance metrics: Euclidean distance, correlation coefficients, or mutual information[5]. Hierarchical tree clustering joins similar objects together into successively larger clusters in a bottom-up manner (i.e., from the leaves to the root of the tree), by successively relaxing the threshold of joining objects or sets. The relevance networks take the opposite strategy. It starts with a completely connected graph with the vertices representing each object and the edges representing a measure of association and then links are increasingly deleted to reveal 'naturally emerging' clusters at a certain threshold.

Another category of algorithms to explore functional genomic data can be classified as *partitional* clustering algorithms, such as K-means analysis and nearest neighbor clustering, which minimize within-cluster scatter or maximize between-cluster scatter[5]. Self-organizing maps (SOM), an artificial neural network learning algorithm[6], was demonstrated to be capable of finding meaningful clusters from functional genomic data[7,8].

This paper describes a novel unsupervised learning method to organize complex high-dimensional data space into successive lower-dimensional spaces based on a straightforward geometric property of the data structure in the absence of a priori knowledge. It uses a partitional clustering approach, as it determines the successive hyperplanes that 'optimally' separate observed data and reveal its internal hierarchical organization.

2. Method

In theory, the partitioning problem can be viewed as the simple process of selecting a criterion, evaluating it for all possible partitions, and selecting the partition that optimizes the criterion. However, even after we define a mathematically sound and intuitively appealing criterion, the number of the partitions is the astronomical Stirling number of the second kind[9,10]. For example, the number of possible combinations to create 4 clusters from 19 objects is 11,259,666,000 and bipartite clustering of N objects roughly explodes to 2^{N-1}. Exhaustive enumeration of the whole space is clearly not computationally feasible.

In the matrix incision tree algorithm, we first define a high-dimensional space of observations and its incisional hyperplanes that can separate the data space into sub-spaces. Then we try to find the 'optimal' incisional hyperplane based on a geometric criterion. To manage the combinatorial explosion of finding the 'optimal' incisional hyperplane, we have developed a representation to represent the partitioning problem as a much simpler matrix incision task, and devised an unsupervised learning algorithm, the matrix incision tree algorithm, to find the 'optimal' hyperplane.

2.1 Data Hyperspace and Incisional Hyperplanes

Figure 1 illustrates data hyperspace and incisional hyperplanes. In the completely connected graphs, the vertices represent objects and the edges represent links. In general, coding N objects in such a way that the distance between any two objects is equal requires a representation in (N-1)-dimensional space. For example, representing four vertices and all the six (4*3/2) edges of equal length is only possible in more than 3-D space but not on 2-D plane or 1-D line (Fig. 1c). The requirement of (N-1)-dimensional-space to code N objects also applies when the distances between objects are all different.

Infinite N-dimensional space is separable by infinite (N-1)-dimensional space. For example, plane (2-D) is separable by line (1-D) and 3-D space is separable by plane (2-D) (Fig. 1b and 1c).

In general, N objects and their geometric relationships can be fully represented in (N-1)-dimensional hyperspace and are separable into two lower-dimensional sub-spaces by a set of (N-2)-dimensional incisional hyperplanes. When a hyperplane separates N objects into two subgroups with m and n objects (N=m+n), the plane deletes m*n links among the total N(N-1)/2 links and there are 2^{N-1}-1 such incisional hyperplanes.

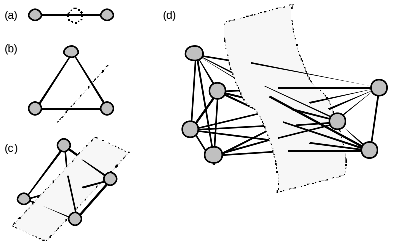

Figure 1. Data hyperspace and incisional hyperplanes. (a) Two objects can be arranged in 1-D 'line' space and are separable by 0-D 'point' space. (b) Three objects can be arranged in 2-D plane and are separable by 1-D line. (c) Four objects require 3-D space and are separable by 2-D plane. (d) Seven objects in 6-D hyperspace separated by 5-D hyperplane. Note that it is a severely distorted 3-D representation of the 6-D hyperspace, where all the 21 ((7 * 6)/2) links can have equal Euclidean length. The incisional hyperplane deletes 12 (4 x 3 = 12) links and only 9 (4x3/2 + 3x2/2) links will remain in the separated sub-spaces.

2.2 Object Link Strength Matrix

To manage complex observations like mRNA expression data, we can view each gene or array (i.e., cell line) as an object. The association between genes or arrays represents the *link strength* between two objects. In that way, we can create a comprehensive N-by-N object link strength matrix for N genes or N arrays. The average link strength for a group of objects is defined as the mean strength of all links,

$$\text{Within-group average link strength} = \sum_i L_i \,/\, \{N(N-1)/2\}$$

Similarly, average link strength between two groups is defined as the mean strength of all between-group links,

Between-group average link strength = $\sum_i L_i$ / (m * n).

N: number of objects within the group
L: link strength of the ith link
m, n: number of objects in each group (N=m+n)

2.3 Matrix Incision Index (MII)

Figure 2 is an equivalent but much more manageable representation of the hyperspace partitioning problem discussed in Figure 1. An (N-1)-dimensional space containing fully connected N objects can be represented as an N-by-N object link strength matrix. The rectangular area (a) in Figure 2 represents an incisional hyperplane that separates the N objects into two sub-spaces of m and n objects represented by the triangular areas (b) and (c), respectively.

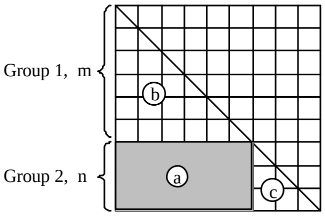

Figure 2. Matrix representation of data hyperspace and incisional hyperplane.
Matrix Incision Index (MII)
= {(m / (n+m)) * b + (n / (n+m)) * c} / a

Therefore, the 'optimal' partitioning problem of data hyperspace becomes a matrix incision problem of finding the rectangular area, (a), representing the 'optimal' incisional hyperplane with the minimum loss of link strength that produces the maximum link strengths within the resultant sub-spaces represented as the two triangular areas, (b) and (c). Matrix incision index (MII) is defined as the ratio of gain (i.e., the weighted mean of the within-group average link strengths of the two separated sub-spaces) to loss (i.e., the between-group average link strength for the incisional hyperplane) (see Fig. 2), as follows:

MII = {(m / (n+m)) * b + (n / (n+m)) * c} / a
m: number of objects in group 1
n: number of objects in group 2
a: between-group average link strength between groups 1 and 2
b: within-group average link strength of group 1
c: within-group average link strength of group 2

Although the matrix representation of hyperspace and hyperplanes makes the problem much simpler, it is worth noting that it is rather deceptively simple because we still have to manage the complicated 2-dimensional sorting of the rows and columns of the object link strength matrix as well as the N-1 possible rectangular areas under the diagonal line to correctly determine the 'optimal' incisional hyperplane. Hence, we need a computational algorithm to find the correct arrangement of objects in the matrix that returns the highest MII before we can obtain an analytic solution for this problem.

2.4 Geometric Aspect of Matrix Incision Index

Assume that the six objects in Figure 3 are completely connected with each other and have seven strong (r^2 = 0.8, solid lines) and eight weak (r^2 = 0.1, hidden lines in a completely connected graph) links (i.e., total 15 = 6*5/2 = 7 + 8). The broken lines represent four hyperplanes that separate the graph into two partitions. As shown in Figure 3, the magnitude of MII of incisional hyperplane (broken lines) captures the intuitive sense of 'optimality' of partitioning of the completely connected graph in hyperspace. The intuitively-most-appealing incisional hyperplane (a) corresponds to the highest MII (4.4) and the most unappealing hyperplane (d) corresponds to the lowest MII (0.64) among the four examples (out of 31 (2^{6-1} -1) possible planes). The hyperplanes (b) (MII = 1.8) and (c) (MII = 1.0) show intermediate relevance.

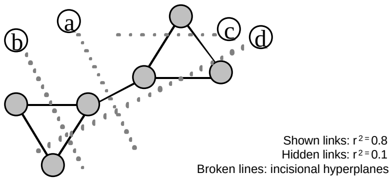
a
b
c
d
Shown links: r^2 = 0.8
Hidden links: r^2 = 0.1
Broken lines: incisional hyperplanes

Figure 3. Geometric property of matrix incision index (MII). The magnitude of MII of incisional hyperplane (broken lines) seems to capture the intuitive sense of 'optimal' partitioning of the completely connected graph in high dimensional space.

	Average loss	**Weighted average link strength**	**MII**
a.	(0.8 + 0.8)/9 = 0.18	0.5(0.8) + 0.5(0.8) = 0.8	4.4
b.	(1.6 + 0.6)/8 = 0.275	0.33(0.8) + 0.67(0.54) = 0.5	1.8
c.	(1.6 + 0.3)/5 = 0.38	0.2(1) + 0.8(0.45) = 0.38	1.0
d.	(3.2 + 0.4)/8 = 0.45	0.33(0.1) + 0.67(0.45) = 0.28	0.64

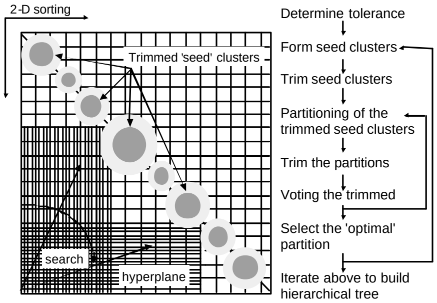

Figure 4. Matrix incision tree algorithm.

2.5 Matrix Incision Tree Algorithm

We have devised an unsupervised machine learning algorithm to find the 'optimal' matrix incision hyperplane with the highest MII (Fig. 4).

Basic Matrix Incision Tree Algorithm

Step 1: Determining computational tolerance level: If a system can comfortably compute 2^{N-1} MII's for a given object link strength matrix (see section 2.2), the computational tolerance level of the system for an exhaustive enumeration of all possible combinations of bipartite partitioning of N groups of objects (2^{N-1}) will be N.

Step 2: Developing 'seed' clusters: Most of the clustering algorithms tend to generate similar clusters. We created small representative 'seed' clusters up to the system's tolerance level (N) using available clustering algorithms such as K-means and SOM (see figure 4).

Step 3: Trimming outliers of 'seed' clusters: We applied a very simple strategy of trimming 20% of the objects with the lowest average link strength (i.e., between each object and all the others in the corresponding 'seed' cluster). It is because the central cores of the 'seed' clusters are likely to be relevant as the seeds of creating 'candidate' partitions in the following steps.

Step 4: For each enumeration of binary partitioning of the 'trimmed-seed' clusters of step 3 (i.e., for each of the 2^{N-1} binary partitioning of N 'trimmed-seed' clusters),

4.1: Trimming outliers of partitions: Using the same strategy with step 3, trim 20% of the most loosely linked objects for both partitions (created by fusion of the 'trimmed-seed' clusters).

4.2: Generating 'candidate' binary partitions by reassigning the 'trimmed' objects: For each object that was trimmed in steps 3 and 4.1 (36%, 1*0.2 + 0.8*0.2), calculate and compare the between-group average link strengths between the object and each of the two 'trimmed' partitions of step 4.1 and assign the object to the partition with the higher average link strength.

4.3: Measuring MII for the 'candidate' binary partitioning generated in step 4.2

Step 5: Choosing the 'optimal' partitioning: Select the 'candidate' binary partitioning (generated in step 4.2) with the highest MII (measured in step 4.3) among all the enumerations in step 4.

Step 6: Iterate step 2 to step 5 for the resultant subgroups (by the 'optimal' partitioning selected in step 5) successively down to individual leaves (i.e., objects), thereby creating the whole hierarchical tree.

The results described in the present study were all made with the basic matrix incision tree algorithm with 20 'seed' clusters and 20%-trimming strategy. The algorithm performed reasonably well with more than 9 'seed' clusters and 10-50% of trimming rate. A variety of sophistication of the algorithm such as cleverer creation of 'seed' clusters, better trimming strategy considering the distribution of link strengths, incremental trimming-and-reassignment for refined determination of 'optimal' partitioning, and backtracking to ensure the hierarchy of the incisional hyperplanes were tested and will be described elsewhere.

2.6 Tree Learning and Tree Traversal for Classification

The matrix incision tree algorithm is essentially an unsupervised learning algorithm that organizes observed data by determining the hierarchical nature of successive 'optimal' incisional hyperplanes purely based on the internal organization of the data structure. However, when we build a tree with a labeled data set without using the label information, the relationship between the labels and the resulting tree structure can be systematically applied to explore unseen observations.

Tree traversal is a successive process of assigning a traversing object with the closer branch at the level of each node starting from the root to the leaves of the tree. It is useful to consider it a type of nearest neighbor classification, applied at each branching point. The tree traversal index (TTI) is defined as the between-group average link strength involving two groups: (i) the traversing object, and (ii) the cluster represented by each branch. At each node, the traversing object moves to the branch with the higher TTI:

$$TTI = 1/N\ (\Sigma_i\ L_i)$$

N: number of objects in the corresponding branch.

L_i: link strength between traversing object and the *i*th object in the branch.

A measure of homogeneity of each branch can be used to determine whether to stop or go further down the tree.

2.7 Evaluation Data Sets

Although the matrix incision tree algorithm is primarily designed for unsupervised learning, well known, fully labeled, and publicized data sets can be used to evaluate its performance and characteristics. We have selected the Fisher's iris flower data set[11] as a simple and exhaustively utilized sample, and Golub's Leukemia mRNA expression data set[9], a real genomic sample to illustrate the performance of our algorithm.

The Fisher's iris data set consists of 150 observations of three species of Iris flowers (50 *Iris Setosa*, 50 *Iris Vesicolor*, and 50 *Iris Virginica*) and 4 discriminating measurements (petal and septal length and petal and septal width). The square of Pearson's correlation coefficient, r^2, was calculated with the four measurements and used as the link strength between objects. Many supervised and unsupervised learning algorithms have been tested against this data set. Golub's Leukemia data set has 6,817 human gene expression profiles of 74 cell lines (38 in training and 34 in test data sets, http://waldo.wi.mit.edu/MPR/data_set_ALL AML_train.txt, data_set_ALL_AML_independent.txt) of acute myeloid leukemia (AML) and acute lymphoblastic leukemia (ALL). Cancer cells were treated as objects and the square of correlation coefficients between gene expression profiles were used as object link strengths.

3. Result

3.1 Fisher's Iris Data

Figure 5 shows the matrix incision tree from the whole 150-case iris data. The first incisional hyperplane separated all *Setosa* perfectly from the other categories. The next plane separated *Vesicolor* and *Virginica* with five errors. Two *Virginica's* were clustered with 48 *Vesicolor's* and three *Vesicolor's* with 47 *Virgnica's*. Thus, the overall accuracy of relevant clustering with this three-group example was 97% (145/150). Half-split method was used to test the reliability of the method by splitting the 150 iris flowers into two subgroups with 50 and 100 cases, The overall accuracy of relevant clustering was 96% (48/50) and 98% (8/100), respectively (Fig. 6). All of the four errors (cases 66, 81, 40, 77) in the half-split study were also found among the five errors (cases 9, 40, 66, 77, 81) in the study with the whole data set (Fig. 5).

To test the learning performance we used the same split groups as training and test sets and also in reverse. After building a matrix incision tree with the training

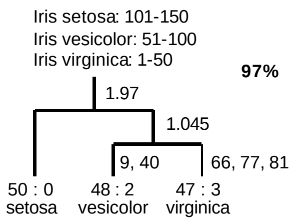

Figure 5. Matrix incision tree from Fisher's Iris data set. The matrix incision algorithm correctly clustered 97% (145/150) of the cases. The numbers next to intermediate branches are MII's and the lists of numbers next to terminal leaves are the numbers of the misclassified cases.

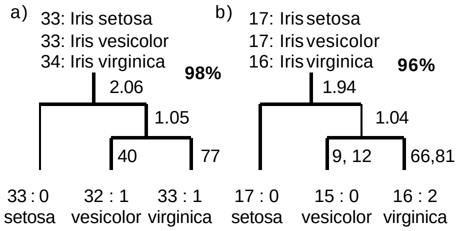

Figure 6. Matrix incision trees from half-split Fisher's Iris data set into 100-case and 50-case subgroups. Ninety-six percent (48/50) and 98% (98/100) were correctly clustered for each subgroup.

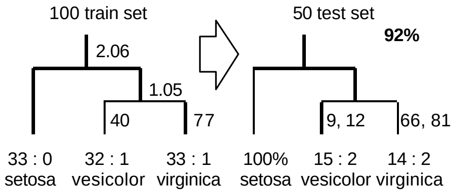

Figure 7. Learning performance of the matrix incision tree algorithm. Fisher's Iris data set was split into 100 training and 50 test sets. After building a matrix incision tree with the training set, the test set was put into the tree and classified by tree traversal method (see text). Overall classification accuracy was 92% (46/50).

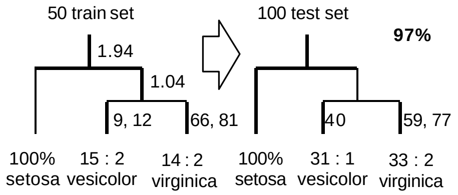

Figure 8. Learning performance of the matrix incision tree algorithm. Fisher's Iris data set was split into 50 training and 100 test sets. After building a matrix incision tree with the training set, the test set was put into the tree and classified by tree traversal method (see text). Overall classification accuracy was 97% (97/100).

set, the complementary test set was put into the tree and classified by tree traversal method (Fig.7 and 8). Accuracy of classification for the 100-training-50-testing sample was 92% (46/50) and for the 50-training-100-testing sample was 97% (97/100). Moreover, the six misclassified objects in the first study (Fig. 7) were all found among the seven misclassified ones in the opposite study (Fig. 8).

Self-tree traversal method (a tree traversal process for objects through its own tree) was also applied. The tree built with the whole 150 objects with five misclassified ones (9, 40, and 66, 77, 81) shown in Figure 5 returned one more misclassified ones (9, 12, 40, and 66, 77, 81) after self tree traversal. However, the original two misclassified ones (40 and 77, Fig. 6a) in the tree with 100 objects were reduced to one (77), and the original four (9, 12, 66, 81, Fig. 6b) in the tree with 50 objects were reduced to two (66, 81).

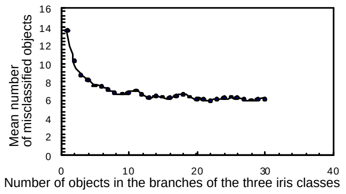

Figure 9. Mean number of misclassified objects with increasing number of objects in the branches of matrix incision tree. Randomly selected N objects equally from each of the three iris classes were assigned to their corresponding branches of the matrix incision tree (Fig. 5). Result from 100 repetitive experiments for each N demonstrates that the mean classification error decreases and forms a plateau with increasing number of objects in the branches of matrix incision tree.

One of the strengths of matrix incision tree that represents the structure of the successive 'optimal' incisional hyperplanes, when compared to the hierarchical tree clustering or relevance networks, is that the matrix incision tree algorithm always considers the group of observations as a whole (group effect) rather than one by one at each step with certain threshold (threshold effect). To evaluate this group effect, we measured the classification performance of matrix incision trees with varying number of objects in the branches of the tree. We have randomly selected N objects equally from each of the three iris classes and assigned them to the corresponding branches of the tree shown in Figure 5. Then we put all the 150 data into the tree to measure the number of misclassified cases. Figure 9 shows the result for 100 experiments for each N=1 to 30. The number of errors rapidly decreased from about 13 to about six as the number of objects in the branches of the matrix incision tree increased from 1 to 12 and then seemed to reach at a plateau with around six errors It seems to approach the 5 errors in the tree constructed from the whole data set (Fig. 5).

3.2 Leukemia data set

Figure 10 shows the matrix incision tree constructed from the 38-case training set with 6,817-gene expression profiles[9]. We put the 34-case test data set into the matrix incision tree with the tree traversal method (Fig. 10) to evaluate the performance for the 3-group distinction (B-cell ALL, T-cell ALL, and AML), using all 6,817 genes in the experiment. Because of the small sample size, all objects were assigned with a prediction label only when they arrived at a 100% homogenous branch. There were nine cases misclassified by the algorithm (74%, 9/34): five B-cell ALL's were labeled as AML's, 2 AML's as B-cell ALL's, and one B-cell ALL and one AML as T-cell ALL's.

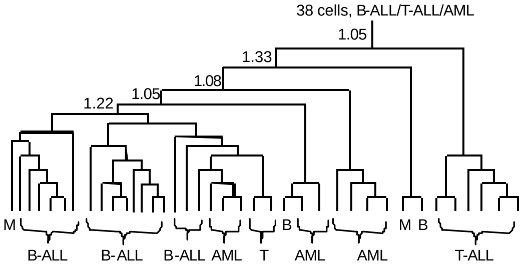

Figure 10. Matrix incision tree from the 38-case training set with three leukemia groups (B-ALL, T-ALL, AML, see text). Numbers next to the branches are matrix incision indices.

Self-tree traversal of the 38 training set into its own tree (Fig. 10) showed a very interesting result. Eleven objects arrived at a different homogeneous branch than their own. However, 10 out of the 11 were still tagged with the correct labels and only one AML (33) was wrongly labeled with B-cell ALL in the experiment.

In the same article, Golub et al. have selected 50 genes that were most highly correlated with the ALL/AML class distinction and provided the result of their own class discovery and

Training	ALL	AML
Cluster 1	25	0
Cluster 2	2	11

Test	ALL	AML
Cluster 1	19	1
Cluster 2	1	13

Figure 11. Matrix incision tree clustering of the Golub's ALL/AML leukemia data set (see text).

prediction analysis (http://waldo.wi.mit.edu/MPR/table_ALL_AML_predic.txt). The matrix incision tree algorithm was applied to the published 38-case training and 34-case test sets using only the 50 genes pre-selected by Golub et al. (Fig. 11). For the training set, one cluster had 25 ALL's with no AML and the other had 11 AML's with two ALL's (12, 25) with 95% (36/38, MII = 12.02) accuracy of relevant clustering. For the independent test set, one cluster had 19 ALL's with one AML (66) and the other had 13 AML's with one ALL (67) with 94% (32/34, MII = 10.41) accuracy of relevant clustering.

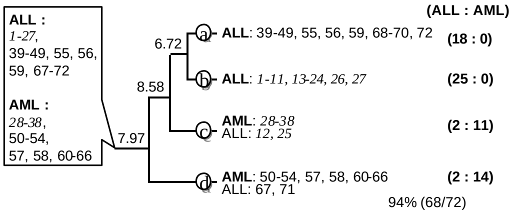

Figure 12. Matrix incision tree from the Leukemia data set (see text). The three successive incisional hyperplanes correctly clustered 64 of the overall 72 cell lines (94%). Note that the matrix incision tee successfully revealed not only the AML/ALL class distinction but also the distinction between the training (cases *1-38*, *italic*, branches (b) and (c)) and test (cases 39-72, branches (a) and (d)) sets (100%) which is suggested by the different sample collection conditions.

Figure 12 shows the matrix incision tree from the overall 72 cases using the selected 50 genes. The three successive incisional hyperplanes correctly clustered 94% (68/72) of the cases. The algorithm successfully captured not only the AML/ALL class distinction but also the difference between the training (cases *1-38*, *italic*, the second and third branches) and the test (cases 39-72, the first and fourth branches) sets perfectly (100%). The distinction may come from the potential difference of the independently collected two data sets as described by Golub et al.

Discussion

Increasing number of methodologies of mining valid information from large-scale gene expression data are available, and have evolved in parallel with the advance of gene expression profiling techniques[12, 13]. Organizing observed data into meaningful structures is one of the most fundamental modes of learning and understanding especially in the exploratory phase of research. Exploration can be best done in a systematic manner. Gene expression data can be viewed as matrices of genes and arrays (cell lines). The matrix incision tree method and algorithm provides a way of exploring the structure of complex data space in a systematic manner.

The matrix incision tree algorithm essentially works as a kind of partitional clustering algorithm generating a single partition of the observed data at each step in. However, it also has a property of hierarchical clustering by precisely defining the 'optimal' incisional hyperplane and iteratively exploring the hierarchical relationship of the successive 'optimal' incisional hyperplanes. Thus, what is basically represented in a matrix incision tree is the hierarchical organization of the successive 'optimal' incisional hyperplanes of observed data.

The matrix incision tree algorithm has no assumption about the distribution of the data nor that of its resultant clusters because it only uses an intuitive geometric property of each unique observation. The algorithm itself is independent of the similarity metrics, as are several other clustering algorithms. The matrix incision tree algorithm dose not require prior assumptions about the geometry or probable number of groups in observed data, as do SOM and K-means analysis. This simple nature of matrix incision tree algorithm seems to allow objective and systematic exploration of complex data, as our preliminary experiments point out. Moreover, it does not rely on any surrogate such as centroid or geometric grid to represent clusters as in K-means or SOM but takes all the observations into account.

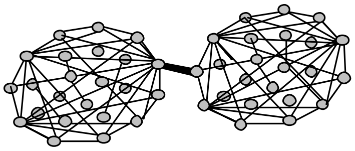

Figure 13. The 'Romeo and Juliet effect'. Line thickness represents object link strength and the omitted links of the fully connected objects represent weak links. Clustering algorithms that uses local threshold effect only will not directly reveal the distinct two-group structure because the thickest link in the middle that connects the two groups will not be separated, regardless of the global structure, until all other weaker links are completely separated

One weakness of certain clustering algorithms is that they operate entirely in a local fashion, using working criteria or thresholds that do not take into account a global perspective. For example, when a hierarchical-tree clustering amalgamates objects (or clusters of objects) to create a link, it takes only those objects into account, neglecting the others. The relevance networks approach also uses a single threshold at each comparison. Figure 13 demonstrates an example of this problem. Although there seem to be distinct two groups, if the link strength between the connecting two objects in the middle is stronger than other links, bottom-up hierarchical trees and relevance networks will not place these two objects in separate clusters. As an analogy, Romeo and Juliet will always be guaranteed to marry each other (i.e. be in the same cluster), as long as their love is stronger than any other, regardless of the hate (i.e., low link strength) between their families. This neglect of group effect (coming from the Capulets and the Montegues) in clustering algorithms is due to the use of local threshold effect. As exemplified in Figure 3, the matrix incision algorithm may discover the two groups if the group effect is larger than the link strength between the two objects.

In summary, our preliminary experiments suggest that the matrix incision tree algorithm has the potential to be a useful tool in functional genomics. Further testing using other data sets is underway.

Acknowledgments

LOM was funded under grant R29 LM06538-01 from the National Library of Medicine, NIH.

References

1. J. DeRisi, L. Penland, P.O. Brown, M.L. Bittner, P.S. Meltzer, M. Ray, Y. Chen, Y.A. Su, J.M. Trent, "Use of a cDNA microarray to analyse gene expression patterns in human cancer" Nat Genet 1996;14(4):457-60
2. R.A. Heller, M. Schena, A. Chai, D. Shalon, T. Bedilion, J. Gilmore, D.E. Woolley, R.W. Davis, "Discovery and analysis of inflammatory disease-related genes using cDNA microarrays" Proc Natl Acad Sci U S A 1997;94(6):2150-5
3. M.B. Eisen, P.T. Spellman, P.O. Brown, D. Botstein, "Cluster analysis and display of genome-wide expression patterns" Proc Natl Acad Sci U S A 998;95(25):14863-8
4. V.R. Iyer, M.B. Eisen, D.T. Ross, G. Schuler, T. Moore, J.C.F. Lee, J.M. Trent, L.M. Staudt, J. Hudson Jr., M.S. Boguski, D. Lashkari, D. Shalon, D. Botstein, P.O. Brown, "The transcriptional program in the response of human fibroblasts to serum" Science 1999;283(5398):83-7
5 . A.K. Jain, R.C. Dubes in *Algorithms for Clustering Data* "Partitional Clustering" (Prentice Hall, New Jersey, pp.89-133, 1988)
6. T. Kohonen, "Self-organized formation of topologically correct feature maps" Biological Cybernetics 1982;43:59-69
7. P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E.S. Lander, T.R. Golub, "Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation" Proc Natl Acad Sci U S A 1999;96(6):2907-12.
8. T.R. Golub, D.K. Slonim, P. Tamayo, C. Huard, M. Caasenbeek, J.P. Mesirov, H. Coller, M.L. Loh, J.R. Downing, M.A. Caligiuri, C.D. Bloomfield, E.S. Lander, "Molecular classification of cancer: class discovery and class prediction by gene expression monitoring" Science 1999;286:531-7.
9. J.J. Fortier, H. Solomon, in *Multivariate Analysis* "Clustering procedures" Ed. P. R. Krishnaiah (Academic Press, Inc., New York, pp.493-506, 1996)
10. R.E. Jensen, "A dynamic programming algorithm for cluster analysis" Operations Research 1969;17:1034-57
11. R.A. Fisher, "The Use of Multiple Measurements in Taxonomic Problems" Annals of Eugenics 1936;7, Part II:179-88.
12. M. Schena, D. Shalon, R.W. Davis, P.O. Brown, "Quantitative monitoring of gene expression patterns with a cDNA microarray" Science 1995;270:467-470 (1995).
13. G.S. Michaels, D.B. Carr, M. Askenazi, S. Fuhrman, X. Wen, R. Somogyi, "Cluster analysis and data visualization of large-scale gene expression data" Pac Symp Biocomput 1998;42-53