Error-Tolerant Clustering of Gene Microarray Data

Jay Cahill

Bachelor's Honors Thesis
April 16, 2002

Abstract

Gene microarray technology allows for unprecedented and massive production of biological data across multiple experimental conditions and in time series. Computer analysis of this data can help guide biological bench work toward the assignment of gene function, classification of cells and tissues and the ultimately assist in the diagnosis and treatment of disease. One approach to the analysis of microarray data is the identification of group of genes with common expression patterns or "clusters". The author implements an error-tolerant clustering algorithm due to Amir Ben-Dor, Ron Shamir and Zohar Yakhini. In their original paper, they defined a stochastic error model for microarray data, and, based on that model, prove that their algorithm recovers the underlying cluster structure of microarray data with high probability. In this paper, their results are replicated on artificial data. In addition, the author tests the stability of clusterings generated by the algorithm and compares the use of discretized and non-discretized similarity graphs.

Student:
Jay Cahill
Degree Candidate 2002, Bachelor of Arts in Computer Science
Boston College
Email: cahillwb@bc.edu
Tel: (617) 308-7218

Advisor:
Peter Clote, Ph.D., *Doctorat d'Etat*
Professor of Computer Science
Dept of Computer Science and Dept of Biology
Boston College
Email: clote@cs.bc.edu
Tel: (617) 552-1332

Note: This paper has been modified from its original form. Interest in the original should be directed to Peter Clote, clote@bc.edu or Jay Cahill, wjcahill@hotmail.com.

Part I: Motivation and Results

This paper is organized into three parts. The first part is intended to define the motivation and idea behind the project and present the results. The second part is intended as a user guide and documentation to enable the future maintenance and extension of the CAST software.
The third part offers conclusions and ideas for further study.

Section 1. Introduction

Proteins are the building blocks of life. Proteins make up our cells, tissues and organs and proteins make up the enzymes that control our cellular processes. By gathering information about the proteins in an organism, scientists can further understand the organism's makeup and biological processes.

The blueprints for the creation of new proteins are found in an organism's genetic code. The blueprint for a protein is transcribed from the cell's DNA to a mRNA molecule that transports this information to a ribosome to begin protein synthesis. Each mRNA molecule contains a transcript of a particular gene used in protein synthesis. By conducting an experiment that measures the amount of a particular mRNA molecule present in a cell at any given time, scientists can infer that a corresponding amount of the protein that the mRNA transcript codes for is being produced.

Gene microarray technology measures the level of expression of every gene in an organism at the same time by conducting many of these experiments in parallel. It should be noted that the level of expression is measured against a background frequency of mRNA molecules and as such is only accurate in measuring changes in gene expression levels, not absolute expression levels.

A time series of gene microarray experiments can show how gene expression changes throughout a cellular process. Spellman *et al.* use microarray data analysis to attempt to find all the genes in the yeast genome regulated by cell cycle.[1] Microarray experiments can also by conducted under different conditions such as heat shock or stress. Richmond *et al.* used multi-conditional microarray experiments including heat shock to identify genes in *E. coli*.[2] Analysis of microarray experiments in a

[1] Spellman *et al*., (1998). Comprehensive Identification of Cell Cycle-regulated Genes of the Yeast *Saccharomyces cerevisiae* by Microarray Hybridization. *Molecular Biology of the Cell* 9, 3273-3297.

[2] Craig S. Richmond, Jeremy D. Glasner, Robert Mau, Hongfan Jin, and Frederick R. Blattner (1999) Genome-wide expression profiling in *Escherichia coli* K-12. *Nucleic Acids Research* 27(19), 3821-3835.

time series or under varying conditions can improve the understanding of a gene's function in an organism.

Advances in gene microarray technology allow scientists to produce experimental data at a much faster rate than data analysis can be performed by humans; this field is often referred to as "high-throughput" biology. Large data sets lend themselves to computer analysis and visualization.

One method of computer analysis of microarray data is clustering. Clustering is a machine learning technique that defines clusters of things with common attributes. Clustering as an approach to the analysis of microarray data groups together genes with "similar" expression levels across conditions or during cellular processes. This paper will focus on the analysis of gene microarray data using a clustering technique due to Ben-Dor *et al.*[3]

Clustering is not a new idea; clustering algorithms have been used in the field of statistics for decades[4]. Clustering problem solving approaches are used in many fields including economics, medicine, biology, and many others, but the application of clustering algorithms to microarray data for the purpose of functional genomics, cell classification and disease diagnostics is fairly recent. Several different clustering algorithms have been applied to gene expression data. Eisen *et al.* used a hierarchical clustering approach, similar to those constructing evolutionary trees[5]. Tamayo *et al.* used self-organizing maps to differentiate between leukemia tumor types for the purpose of choosing the most appropriate chemotherapy treatment, a result of particular medical significance.[6] Other popular clustering techniques include *single-linkage*, *complete-linkage*, *group-average*, *centroid*, and *k-means*.[7]

As can be observed from results like Tamayo's, the application of computer analysis to sources of large quantities of biological data could contribute to major improvements in the diagnosis and treatment of disease. In the long term, advances in this field could have a significant economic, medical and humanitarian impact. The possibility of this type of advance motivates the creation of clustering algorithms specifically designed to analyze biological data.

One of the main challenges of microarray data analysis is the level of noise that exists in the data. Keller *et al.* [8] classify potential sources of noise as follows: the data contain 'technical' noise that can be introduced at a number of different stages, such as production of the DNA array, preparation of

[3] Amir Ben-Dor and Zohar Yakhini. Clustering gene expression patterns. In Proceedings of the Third Annual International Conference on Computational Molecular Biology (RECOMB 99), 1999. http://citeseer.nj.nec.com/305183.html.

[4] Hartigan, John A., 1975, *Clustering Algorithms*, New York, John Wiley and Sons.

[5] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. PNAS, 95:14863-14868, 1998.

[6] P Tamayo, D Slonim, J Mesirov, Q Zhu, S Kitareewan, E Dmitrovsky, ES Lander, TR Golub: Interpreting patterns of gene expression with self-organizing maps: methods and applications to hematopoietic differentiation. *Proc Natl Acad Sci USA* 1999, 96: 2907-2912

[7] B. Everitt. Cluster Analysis, Chapter 4. Edward Arnold, London, third edition, 1993.

[8] Keller, Andrew, et al. Bayesian Classification of DNA Array Expression Data. Technical Report UW-CSE-2000-08-01, August, 2000

the samples, hybridization between cDNA and the array, and signal analysis and extraction of the hybridization results. Additional 'biological' noise can come from non-uniform genetic backgrounds of the samples being compared, or from the impurity or misclassification of tissue samples. There are several papers evaluating the reliability of microarray data and attempting to reduce noise through statistical methods.[9,10]

While many different clustering algorithms have been suggested in the recent literature, the algorithm presented by Ben-Dor *et al.* is the only algorithm that the author is aware of that explicitly incorporates an error model. Due to high noise component of microarray data, it is important for a clustering algorithm to be error tolerant. Ben-Dor *et al.* demonstrates the error tolerance of the *Cluster Affinity Search Technique* (CAST) algorithm in their paper, and their results are replicated in Section 3.2. The error-tolerance of the CAST algorithm makes it particularly appropriate for biological applications. This clustering algorithm is also superior to others because it does not make assumptions about the number of clusters, their size, or structure; rather, the cluster structure is discovered from the data.

A clustering algorithm partitions a set of genes into clusters with similar features. Clustering can be thought of as an undirected graph where each node represents a gene and edges connect "similar" nodes. An optimal clustering would separate the graph into several disjoint subgraphs maximizing the number of edges within a "cluster" subgraph and minimizing number of edges outside of a "cluster" subgraph.

Determining an optimal clustering is NP-hard[11], so clustering algorithms are based heavily on heuristics and approximation. Ben-Dor *et al.* give a probabilistic proof of the optimality of their clustering algorithm and CAST is a practical heuristic based on that algorithm. Ben-Dor *et al.* refer to an implementation of CAST in MATLAB, but the implementation is not given in their paper. An implementation of their CAST algorithm in C++ is the main programming effort of this thesis.

For the purposes of improved matching, it is important that the clustering produced by CAST is *stable*, meaning that the same clustering is produced for multiple iterations of the algorithm. Deterministic clustering algorithms should be inherently stable, but due probabilistic nature of the CAST algorithm, stability is not guaranteed. Measurements of the stability of the CAST algorithm given data with increasing levels of noise are shown in Section 3.3.

[9] MK Kerr, GA Churchill. Bootstrapping cluster analysis: assessing the reliability of conclusions from microarray experiments. *Proc Natl Acad Sci USA* 2001, 98**:** 8961-8965

[10] M-LT Lee, FC Kuo, GA Whitmore, J Sklar. Importance of replication in microarray gene expression studies: statistical methods and evidence from repetitive cDNA hybridizations. *Proc Natl Acad Sci USA* 2000, 97**:** 9834-9839

[11] M. R. Garey and D. S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, 1979.

Section 2. The CAST Software

Software was created to support this paper and will be referred to as CAST. CAST combines a simple microarray data file parser and an implementation of the Ben-Dor *et al.*'s CAST algorithm in C++.

The output of the CAST's clustering module consists of a visual representation of the clustering in both real-valued and boolean similarity matrices and a listing of cluster membership by gene name and optionally gene description.

Section 3: Results

Testing the efficacy of the CAST software was difficult because its performance on experimental data is highly dependant on the characteristics of that data. Some experiments may have a strong underlying cluster structure; others may not. Also, the experiments sensitivity and tendency for error affects the algorithms performance.

The software was tested on both artificial data and experimental data. *Saccharomyces cerevisiae* (yeast) data from Michael Eisen were used as an example of a multi-conditional microarray experiment[12] and are referred to here as the AffyYeast data set. Data from the Stanford Microarray Database[13] were used as an example of a time-series microarray experiment and are referred to here as the yeast_cycle dataset. Pearson's correlation coefficient was used as the similarity measure for AffyYeast and L1 distance was used as the similarity measure for yeast_cycle.

3.1 Measurements

Ben-Dor *et al.* use two quantitative criteria for measuring the distance or "closeness" between two boolean matrices. Given two binary matrixes A and B of the same dimensions, let Nij be the number of entries on that A and B have values *i* and *j* respectively. The *matching coefficient* between the two matrices is the ratio of the total number of entries on that the two matrices agree, to the total number of entries: $(N_{00} + N_{11})/(N_{00} + N_{01} + N_{10} + N_{11})$. The *Jaccard coefficient* is the corresponding

[12] Supplement to P. Clote's BI370 class, http://www.cs.bc.edu/~clote/courses/BI370/hw/hw8/AffyYeast.txt, April 16, 2002

ratio when “negative matches” (N_{00}) are ignored: $N_{11}/(N_{01} + N_{10} + N_{11})$. Jaccard coefficient is used to evaluate CAST’s performance on artificial data.

To measure cluster stability, consider a *n x n* cluster membership matrix where *n* is the number of genes to be clustered and if two genes *i* and *j* are in the same cluster, then the *i*th/*j*th matrix entry is true. Otherwise, the entry is false. *Cluster stability* is defined as the Jaccard coefficient between two of these cluster membership matrices from different application of the CAST algorithm. When the algorithm is run twice on exactly the same data, you would expect cluster stability to be near 100% and this is verified in the results. For increasing levels of additional noise, cluster stability is reduced.

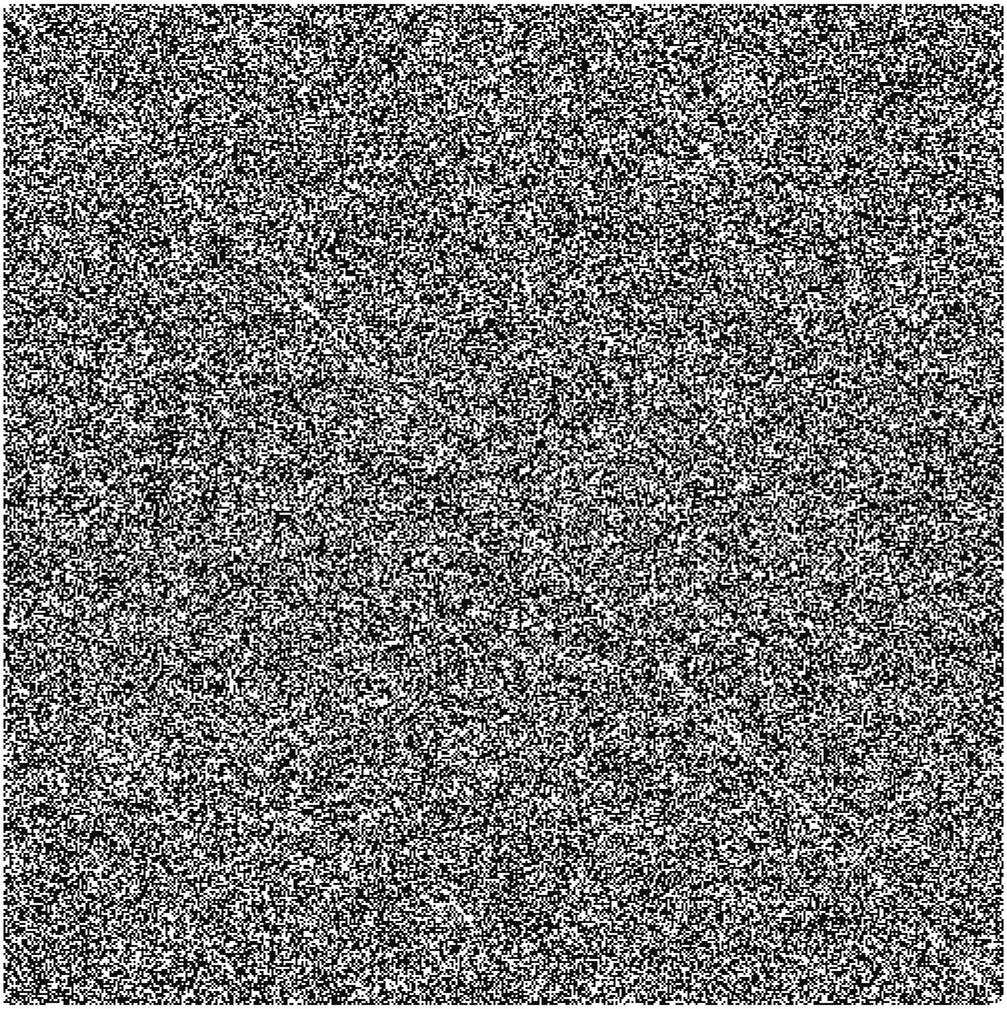
[13] Stanford Microarray Database, http://genome-www.stanford.edu/microarray/, April 16, 2002

3.2 Performance on Artificial Similarity Matrices

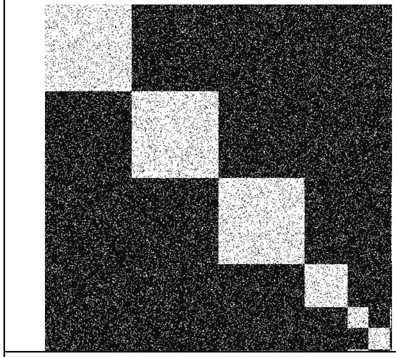
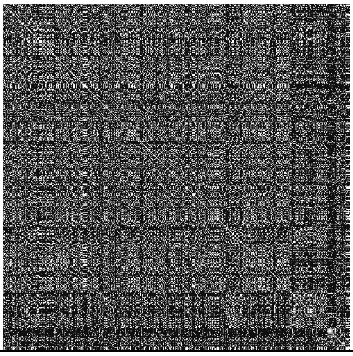
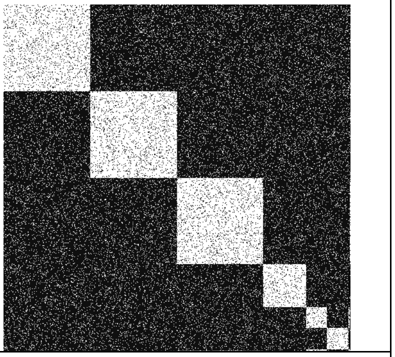
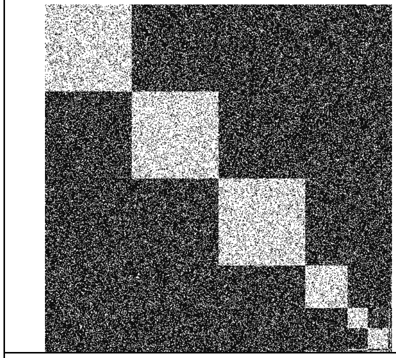
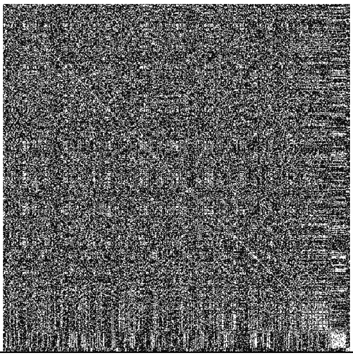
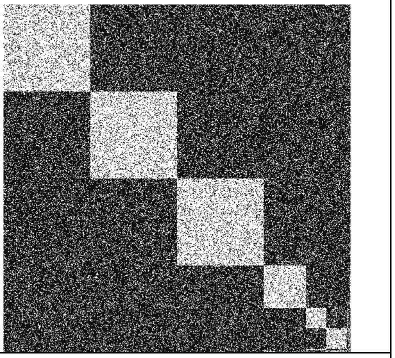
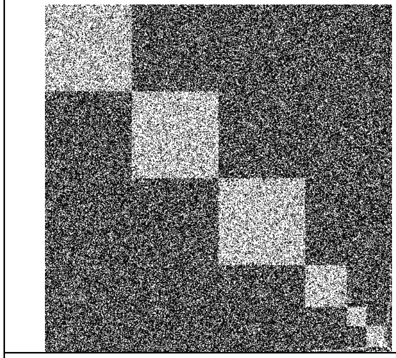
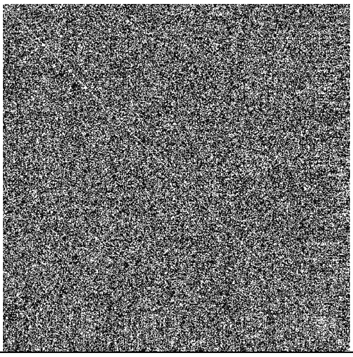
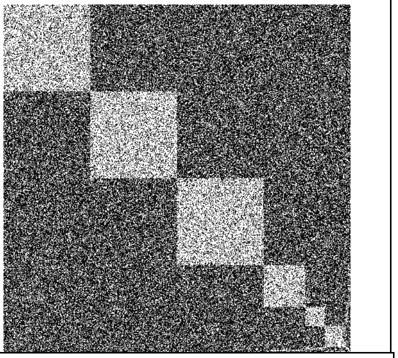
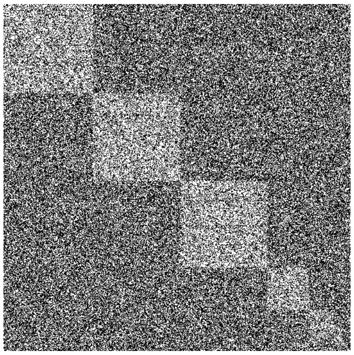
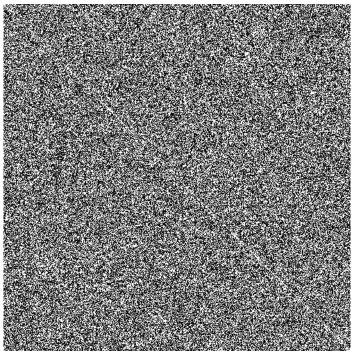
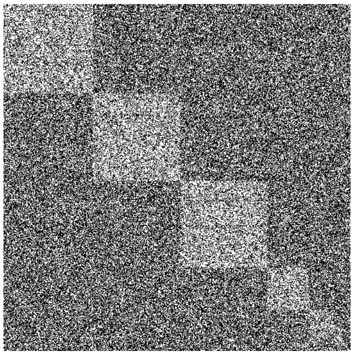
The algorithm's performance clustering artificial data is quite good both visually and quantitatively. Artificial similarity matrices meant to simulate a cluster structure that might be found in biological data were created with an arbitrary distribution of cluster membership where each gene has the following probability of being a member of the respective cluster: {1/4, 1/4, 1/4, 1/8, 1/16, 1/16}.

Increasing amounts of white noise were added to the similarity matrix to simulate false positives in the similarity measures on real data. Let alpha be the probability of error (a false positive or negative), then for each entry in the similarity matrix, that entry was flipped to be false with independent probability alpha.

The similarity matrices were then randomly permuted. These permuted matrices are the actual inputs to the algorithm. The permuted input to the human eye appears to be very close to white noise. The goal is to reconstruct the original cluster structure from the noisy, permuted data.

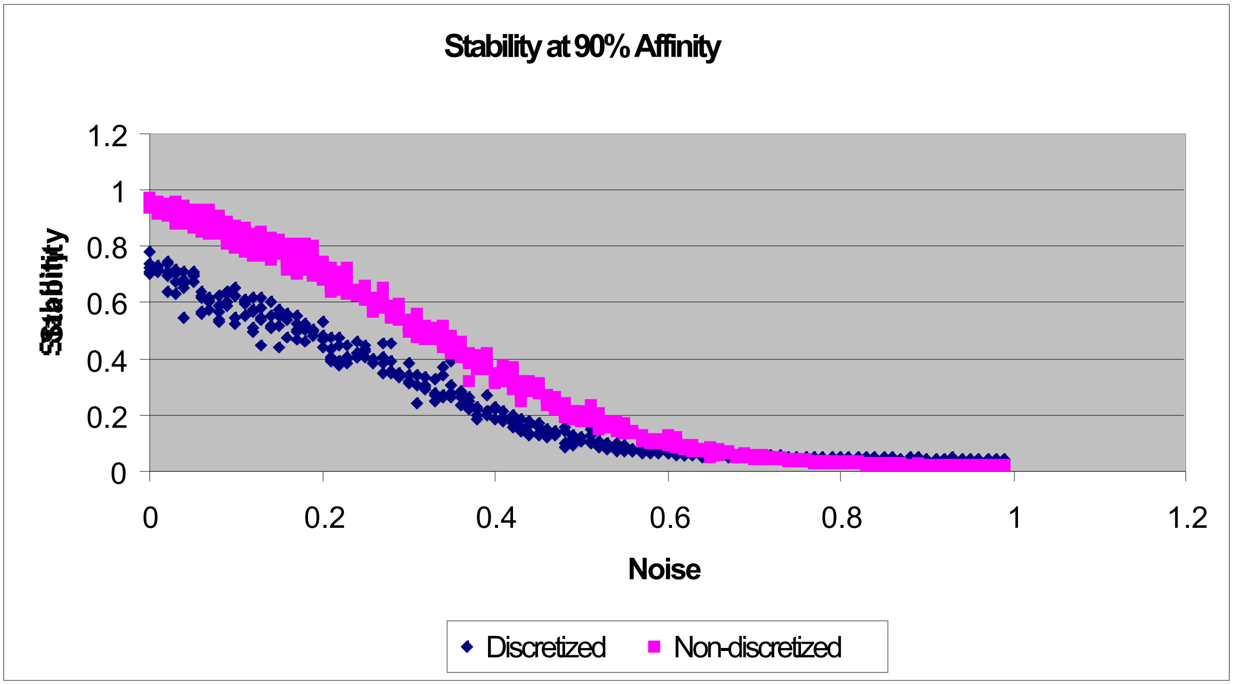


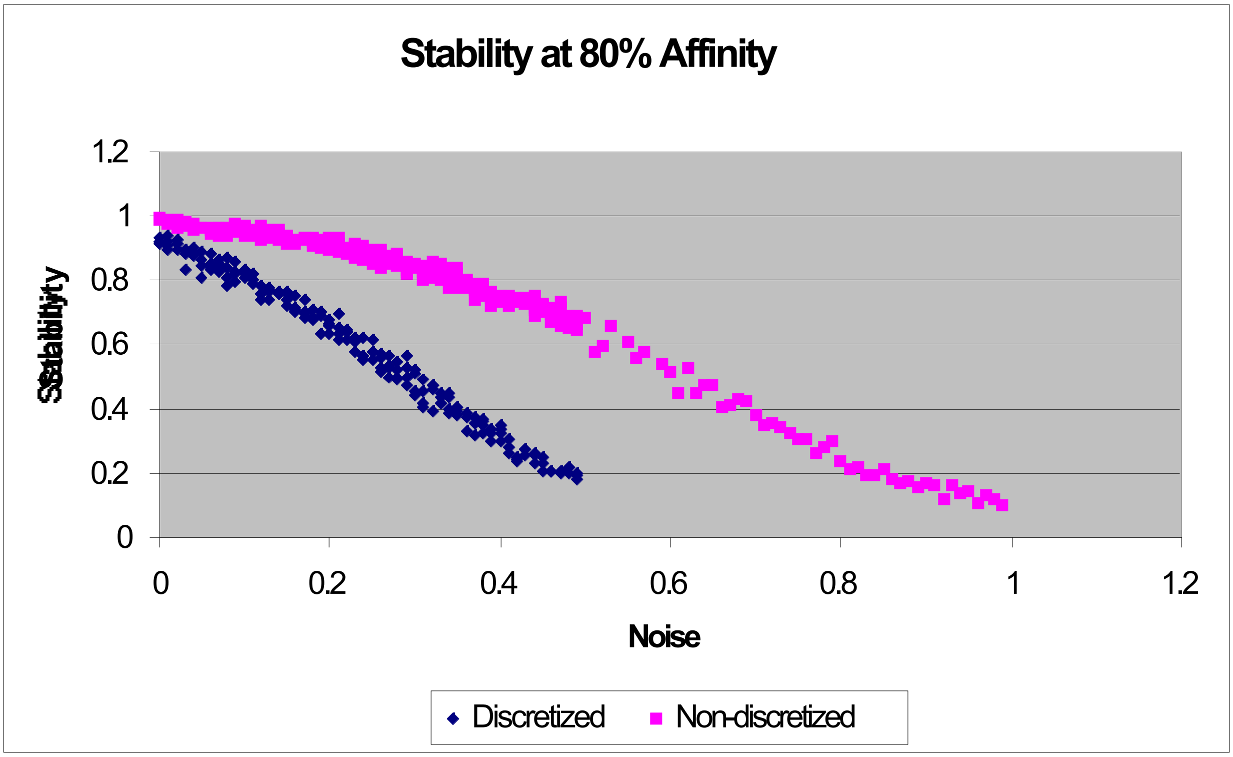
The clustering algorithm is successful in restoring the matrices to original form.

10% Error 		
20% Error 		
30% Error 		
40% Error 		

3.3 Cluster Stability

Stability was testing on the clustering produced with the AffyYeast.txt dataset. Both discretized and non-discretized clustering methods were tested at 80% and 90% affinity thresholds. The clusterings were more stable at the lower 80% cluster affinity threshold due to the creation of larger clusters. Non-discretized clusterings were more stable than clustering using discretized values.





Part II: Technical Discussion and User Guide

Section 1: Architecture and Design Decisions

The object model of CAST involves these objects:

- *GeneArray*: GeneArray is a collection of Genes. GeneArray parses microarray data files and stores them in memory.
- *Gene*: Represents gene expression data.
- *ClusterSet*: ClusterSet is a collection of Clusters. ClusterSet is the object that does most of the heavy lifting in CAST. The CAST algorithm is implemented in the ClusterSet object.
- *Clusters*: Clusters are collections of references to Genes.
- *Measure*: Based on the Functor design pattern, Measure serves an abstract base class/interface that generalizes the similarity measures used in CAST. This abstraction enables users to add similarity measures to the software by writing only one C++ function. The Measure interface is implemented by CorrelationCoefficient, L1Distance, and EuclideanDistance objects that are implementations of those similarity measures respectively. The user can add other measures as necessary. To standardize similarity thresholds, distance measures should be normalized to the format used Pearson's correlation coefficient where 1 represents the closest node or most similar gene and –1 represents the most distance node or most dissimilar gene. The Measure interface includes a boolean normalize field that should be set to true for distance measures.

In the implementation of the clustering algorithm, there was a choice between using a boolean similarity matrix discretized at a significance threshold or a real-valued similarity matrix. It was found that clustering based on a discretized similarity matrix did not guarantee that a gene would be in the cluster to which it had the highest affinity. This problem is due to inaccuracies caused by similarity values just above a significance threshold. Also, calculating cluster affinity based on integral boolean sums required frequent random tie-breaking in the choice of which gene to add or remove from a cluster. Random tie-breaking reduced the stability of clusterings, which is important for finding meaningful matchings. In the results for cluster stability, real-valued similarity matrices produced a slight improvement over discretized matrices.

Section 2: Clustering Implementation

Pseudocode of the CAST algorithm is as follows:

```
Let S(x,y) be the similarity measure between x any y.
Input:
        An n-by-n similarity matrix S, and an affinity threshold t.
Initialization:
        The collection of closed clusters: $C = \varnothing$
        Elements not yet assigned to any cluster: $U = \{1,\quad ,n\}$
Algorithm:
while ($U \neq \varnothing$) do
        Start a new cluster: $C_{open} = \varnothing$
        Reset affinity: a(.) = 0
        Repeat steps ADD and REMOVE as long as changes occur:
ADD: while $\max\{a(u) \mid u \in U\} >= t|C_{open}|$ do
        Pick an element $u \in U$ with maximum affinity.
        $C_{open} = C_{open} \cup \{u\}$
        $U = U \setminus \{u\}$
        For all $x \in U \cup C_{open}$ set $a(x) = a(x) + S(x,u)$
REMOVE:
        Pick an element $v \in C_{open}$ with minimum affinity.
        Remove v from $C_{open}$: $C_{open} = C_{open} \setminus \{v\}$
        Insert v into U: $U = U \cup \{v\}$
        Update the affinity: For all $x \in U \cup C_{open}$ set $a(x) = a(x) – S(x,v)$
Close the cluster: $C = C \cup \{C_{open}\}$
```

The implementation of the CAST algorithm takes as input a list of *n* real valued vectors each corresponding to the microarray data for one gene. A *n x n* real valued similarity matrix is computed where the ith/jth value is the similarity measure between the ith and jth gene. Pearson's correlation coefficient, Euclidean distance, and L1 distance are available as similarity measures. The similarity matrix can either be left with its real valued measurements or discretized with some threshold of similarity significance transforming the similarity matrix to a boolean matrix.

The deciding factor for cluster membership is the affinity between the cluster and its constituent genes. The affinity of one gene to a cluster is the sum of the similarity measure between all of the genes in that cluster to that gene divided by the size of the cluster; to put it another way, affinity is the average of the similarity measures between genes in the cluster and a unclustered gene. The algorithm is given a threshold value as the minimum average affinity to the cluster for any of its members. The algorithm adds genes to a cluster while affinity is greater than the threshold and removes genes from a cluster when affinity drops below the threshold. The algorithm terminates when all of the genes have been assigned to a cluster, even if that cluster is a singleton, meaning the gene is in a cluster by itself.

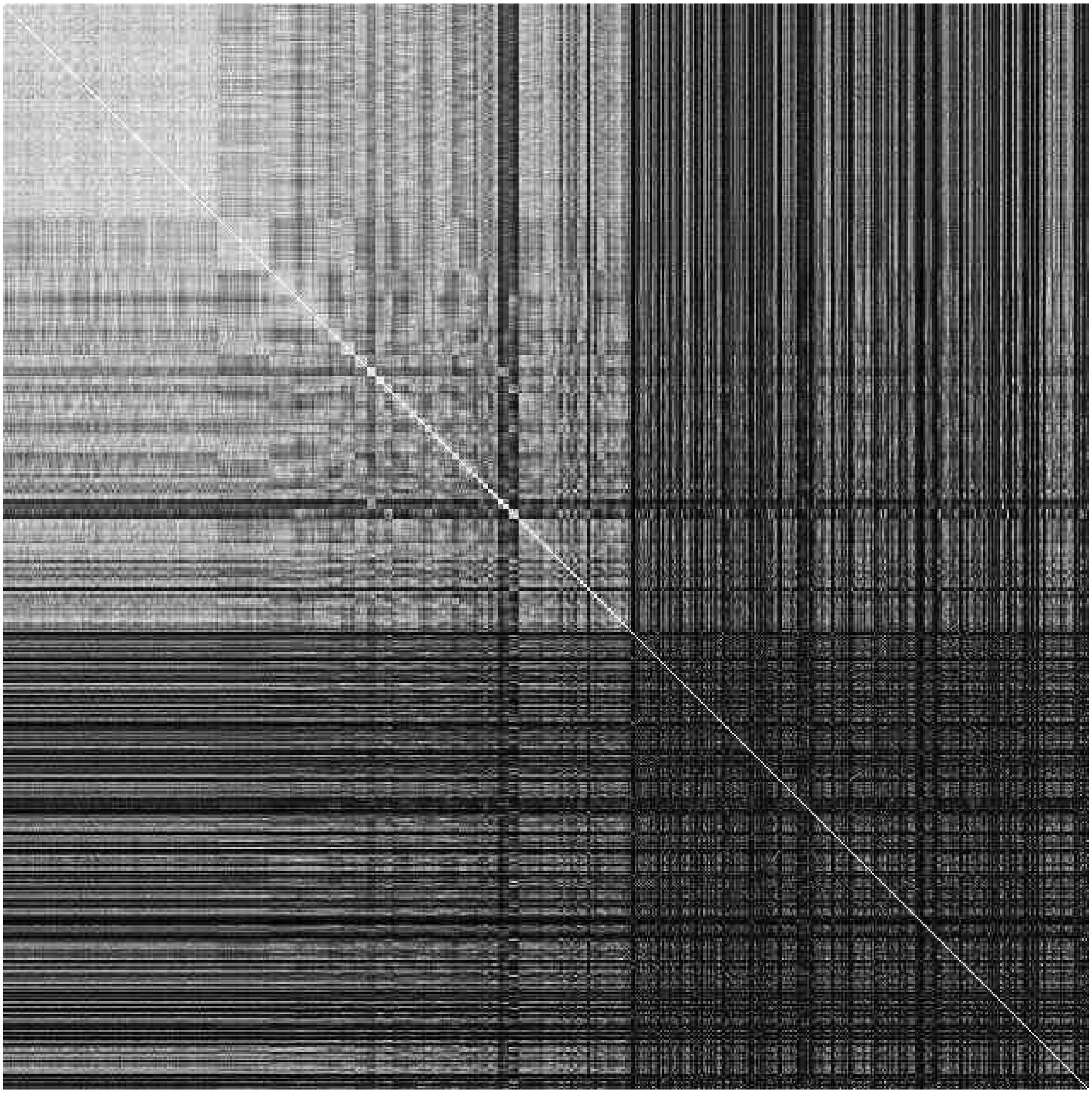
Having established cluster structure, the algorithm optionally performs a cleaning step that checks to make sure each gene is in the cluster to that it has maximum affinity; the algorithm guarantees that every gene will have affinity to it's cluster above the clustering threshold, but not that every gene is in the cluster to which it has maximum affinity. This cleaning step was implemented as a series of passes where, for each gene, the algorithm checks it's affinity to every cluster and moves the gene to the cluster of greatest affinity. As genes are swapped one at a time, cluster affinities change with each swap and the cleaning step can take many passes before a stable clustering is found.

Section 3: Cluster Visualization

To visualize the results of a clustering, the software displays the similarity matrix either as a monochromatic image for a discretized matrix or as a grayscale for a real-valued matrix. A white pixel represents similarity between two rows (genes) in the monochromatic and the white component of a grayscale pixel represents the magnitude of similarity between two rows (genes) in the grayscale image. The rows in the similarity matrix are reordered so that genes in the same cluster are next to each other and so that the clusters are sorted according to size, the largest first. This visualization produces an image with square clusters around the diagonal that are brighter than the surrounding image. This visualization is useful to get a rough idea of the quality of a clustering; a better clustering will have brighter squares around the diagonal and be darker in the regions outside the cluster.

Figure 3.1 is an example of the cluster visualization for the AffyYeast data set.

Figure 3.1:



Part III: Conclusion and Further Study

Section 1. Conclusion

The performance of the CAST algorithm on artificial data is very good and visually quite impressive; however, the addition of uniform white noise to the similarity matrices may not accurately simulate error in biological data where the definition of the underlying cluster structure is fuzzier and alpha values may be higher than 50%.

Given the non-deterministic nature of the clustering algorithm, the stability of its output in the presence of error is satisfactory. Overall, the CAST software functions well as a clustering tool.

Section 2. Further Study

1) Back-end Database Interface. The most needed and obvious improvement necessary in CAST is a back-end database interface. Using flat files to represent microarray data is woefully inadequate. An interface to existing web-based database of microarray data such as the Stanford Microarray Database would be ideal. An intermediate step toward this goal would be the abstraction of the GeneArray class into a interface/abstract base case in a manner similar to the design of the Measure interface.

2) A standardized file format for microarray data. The author is not aware of a standardized format for microarray data, but the establishment of such a standard would greatly aid non-biologists conducting research into microarray analysis techniques. The variety of microarray file formats severely limits the applicability of the CAST software. At present, CAST requires a parsing script hack for each different file format. An intermediate step would be more generalized parsing script or perhaps using Python as a scripting language to address the algorithm as a C++ object.

3) Better handling of missing data values. Currently, CAST interpolates missing data values that introduce error for multi-conditional experiments. The Measure interface must be modified to allow for similarity measures to consider missing data points.