CS4761 Project: Visualizing Clusters in Microarray Data

John Alfieri
jalfieribioe@aol.com

Hart Lambur
hal2001@columbia.edu

Blake Shaw
bs2018@columbia.edu

Tao Wang
tw2004@columbia.edu

May 8, 2005

Introduction

There is a pressing need for efficient, and effective algorithms to reduce, cluster and visualize the high dimensionality gene-expression data gathered from microarray analysis. Analyzing microarray data is essential for gaining insight into subtle differences in biological processes. For example, successfully treating cancer patients heavily depends on targeting the treatment to distinct types of tumors. By analyzing the expression levels of genes in a variety of cancer patients we can hope to aggregate the multitude of epigenetic changes caused by the disease into distinct classes, allowing for better diagnosis, and therefore better treatment.

Throughout the course of the term, the class has examined the increasingly high-throughput "systems" integration of experimental and computational technologies[1]. Our project group has sought to further understand the nature of leading-edge, cross-disciplinary, and integrated problem structures through a shared interest in large-scale experimental data, functional genomics, and microarray based cancer research.

[1]See, for example, Leroy Hood, The Institute for Systems Biology, 4225 Roosevelt Way, Seattle WA 98105.

Efficiently clustering and visualizing data as distinct classes is an interesting computational problem, under investigation in various academic fields including machine learning, image analysis, data mining and biology. In essence, we want to extract a small set of information which captures common traits allowing for a compact and lucid representations—through visual illustrations and clustering—of a large amount of data. Depending on the metric chosen, clustering can group genes that serve similar biological functions, or experiment samples that affect genes in similar ways. Groupings discovered in this way can be compared to existing knowledge of gene functions to provide a deeper understanding of the relationships between groups of genes which may have been previously unknown or considered to be unrelated, or can discover/confirm new classes of samples which were previously unknown.

A general goal of our group has been to develop an understanding of large-scale, integrated technologies working from a strongly computational perspective. Golub and colleagues genomic and computational approaches to cancer biology and cancer medicine represent seminal efforts in cancer microarray study design and analysis. Early microarray efforts, such

as Eisen's 1998 [1] study of genome-wide expression patterns, were more descriptive than analytical in nature, and have focused on cell cultures rather than primary patient material. Golubs 1999 leukemia data set [4] is a first generation [7] benchmark [5] methodology for the molecular classification of leukemia using more analytical and systematic techniques. Although this data set is regarded as easy to classify [6], the study elucidated Principal Component Analysis (PCA) and more recently Non-negative Matrix Factorization (NMF) [3] for the analysis of microarray data.

Visualization of gene expression data is extremely important to biologists. Clear illustrations of the relationships between genes or samples reveal new and often unanticipated relationships. For this reason, many biologists have long favored simple clustering techniques such as hierarchical clustering which force microarray data into a 'clustered' tree structure. This 'forced' structure allows biologists to visually identify the clusters or groups that interest them, and to further examine those groups. Algorithms such as PCA and NFA are not hierarchical, and require the biologist to know the *rank* or number of clusters they are looking for *a priori*. This presents a major problem for most biologists, since many researchers do not know the rank of the data they are analyzing, or even if their data is clustered at all.

For this reason, new visualization techniques are needed to allow biologists to 'see' clusters without imposing a forced hierarchical structure. One promising algorithm to fit this purpose is Locally Linear Embedding (LLE), an algorithm which allows for a non-linear dimensionality reduction of high-dimensional data sets. LLE permits a biologist to collapse very high-dimensional space of clustered points into 2 or 3 dimensions. Combined with clustering algorithms like k-means, PCA and NMF, LLE gives biologists a good guess for the number of clusters that may be interesting, as well as providing an alternative view of gene expression or sample relationships.

In this paper we combine the visualization abilities of LLE with modern clustering algorithms like NMF. We take two of Golub's original data sets, his human acute leukemias data and his childhood brain tumors medulloblastomas data, and run LLE to produce visual illustrations of the sample clustering. By visually inspecting the data, we identify and enumerate interesting clusters. This *rank* is then used to cluster the data using either the k-means or NMF techniques. The visualization/cluster technique is shown to produce accurate results in line with Golub's work.

As a final, more interesting experiment, we apply our novel technique to the yeast "stress" data presented by Gasch and colleagues (Gasch 2000) to further investigate our multi-model analytical approach in terms of a more complex multi-class classification problem. Gasch applied ten different stress shocks, such as temperature shocks, hydrogen peroxide, amino acid starvation, etc. to 173 yeast samples. Our visualization/cluster technique is used to group the shock sample into 'cliques' of shocks that affect the yeast in similar ways. Below we outline the different data sets in greater detail, describe the algorithms, and perform our analysis.

Background

Data Sets

Todd Golub's genomic and computational approaches to cancer biology and cancer medicine represent seminal efforts in cancer microarray study design and analysis. Golub's 1999 analysis of his leukemia and medulloblastomas data sets using hierarchical clustering (HC) and self-organizing map (SOM) techniques is a first generation benchmark

methodology for molecular classification. His continuing work in the problem led Golub to nonnegative matrix factorization (NMF) decomposition algorithms that can reduce the dimensionality of microarray data from hundreds of thousands of genes to a handful of 'metagenes'.

Luekemia Data

It is well know that acute leukemias may be classified as acute lymphoblastic leukemia (ALL) originating from lymphoid precursors or acute myeloid leukemia (AML) originating from myeloid precursors. ALL types can be further classified into T-lineage and B-lineage subtypes. Ramaswamy and Golub noted that "distinct cellular precursors likely account for the robust expression signatures that distinguish these two cancers." [4] This distinction is critical for treatment planning but it is clinically difficult to determine. Golub applied a variety of clustering algorithms to systematically determine cell class without subjective analysis.

Golubs initial (training) data set consisted of 38 bone marrow samples (27 ALL, 11 AML) obtained from patients before treatment. Biotinylated RNA prepared from patients bone marrow cells was hybridized to the Affymetrix (high-density oligonucleotide array) HU 6800 gene chip containing probes for 6,817 genes. A test data set consisted of 6,817 genes across 34 patients (20 ALL, 14 AML) derived from bone marrow as well as from peripheral blood samples provided biological (vs. experimental) heterogeneity to the study.

Medulloblastomas (Brain Tumor) Data

Golub's second data set analyses gene expression data from childhood brain tumors known as medulloblastomas. While the pathogenesis of these brain tumors is not well understood, it is widely accepted that two know hisotogical subclasses exist: classic and desmoplastic, which are easily differentiated under the microscope. The dimensionality of the medulloblastomas data set is 6,817 genes across 34 samples containing 25 classic medulloblastomas and 9 desmoplastic clinical histological classes.

Yeast Shock Data

Gasch et al. describe yeast stress experiments based upon whole-genome microarrays containing 6152 genes sampled over 173 experiments. These stresses include a variety of temperature shocks, nutrient depletions, chemical and radiation effects, osmotic changes, and oxidizing and reducing treatments. Several key results were obtained, including a description of global expression programs which respond to a diverse set of stresses [2]. This "environmental stress response" as it has been termed, define two groups of genes which were consistently over- and under- expressed across a majority of the stress tests.

Current Clustering Techniques

Basic Clustering Techniques

Hierarchical Clustering (HC): HC is a frequently used and commonly accepted approach in microarray analysis. This method, however, imposes a strict tree structure on the problem and is very sensitive to the metric used to determine similarity. Furthermore this technique often requires subjective human analysis to identify clusters, which is problematic for systematic analysis (although many biologists prefer to visual 'inspect' their data).

Self-organizing Maps (SOM): SOMs have been successfully used in many applications, but can be unstable and are heavily dependent on initial

conditions. For biologists, the difficulty of finding appropriate initial conditions makes SOM an unattractive technique.

Non-negative Matrix Factorization (NMF)

NMF techniques decompose gene data into a series of 'metagenes'. NMF techniques have been used to reduce images of human faces into parts reminiscent of features such as eyes, nose, etc.; comparable PCA analysis yields eigenfaces with no obvious interpretation. NMF is more difficult algorithmically due to its nonnegativtity requirement, but yields gene components with easy biological interpretations and more intuitive decompositions.

To describe NMF mathematically, consider N genes in M samples in a matrix A. NMF's goal is to find a small number of metagenes, each defined as a positive linear combination of the N genes. This represents a factorization of $A \sim WH$, where W is a small $N \times k$ matrix describing k metagenes, and H is a small $k \times M$ matrix describing k metasamples.

LLE/NMF Method

Our new technique combines the visualization power of LLE and the robust clustering ability of NMF to easily inspect clusters in a data set. LLE allows us to first visually inspect the data, condensing the high dimensional data into a 2-dimensional plane, in order to estimate how many clusters are distinguishable. We can then apply NMF to more rigorously analyze how the data is clustered.

Locally Linear Embedding attempts to represent significant relationships between N items in a D dimensional space using only two dimensions, allowing the data to be plotted, and easily visualized.

LLE is comprised of 3 steps. First distances from each point to its K nearest neighbors are computed. Then matrix of reconstruction weights W minimizing the cost specified by this distance matrix is constructed. Lastly, W is used to reduce the problem of finding a low dimensional embedding to a sparse eigenvalue problem.

There are two tunable parameters for our algorithm. LLE involves a parameter K, which controls how many nearest neighbors the algorithm uses when computing the reconstruction weights. K is generally a measure of the non-linearity of the manifold mapping the high-dimensional space to a low-dimensional one. NMF takes as an argument a value for the number of clusters it is trying to find.

In our analysis we visualize the Leukemia, Medulloblastomas, and Yeast data sets. The placement of points on the image are calculated by LLE. The colors represent distinct clusters found by NMF. Furthermore lines are drawn between samples found to be in the same cluster by NMF.

Analysis

Analysis of Leukemia Data

The results of our LLE/NMF combination technique on the leukemia data set are very promising (see figure 1). As described above, human acute leukemia can be classified into ALL and AML types, and ALL can be further classified into T and B cell subtypes. In the LLE embedding, we clearly see 3 distinct regions in the 2-dimensional space, implying that the embedding calculated by LLE is strong and the clusters are well-formed. Presented with this information, we ran the NMF algorithm with rank$=3$ to cluster the data into three unique metagene 'types'. The results from that clustering were then fed back into LLE to produce the colored picture seen in figure 1.

From the coloring and the LLE embedding we can clearly see how LLE and NMF agreed on the labels

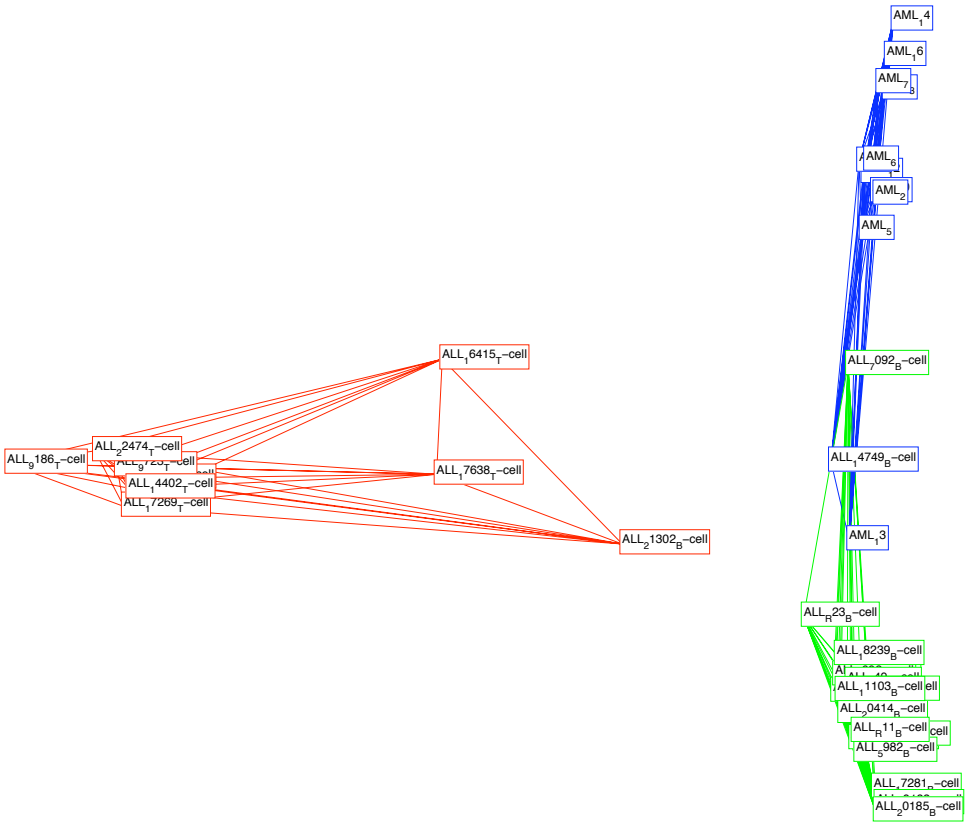

Figure 1: Analysis of using LLE/NMF on Golub's leukemia data set. k=6, and the number of clusters = 3. Our algorithm correctly visualizes the 3 distinct types of leukemia: ALL type T, ALL type B, and AML.

for most the sample points. The ALL type-T subtypes are tightly clustered on the left side of the image and are clearly labeled red (corresponding to the first cluster identified by NMF). The AML and ALL type-B subypes are displayed along the right side of the image, with most AML samples gathered near the top and most ALL type-T samples clustered on the bottom. There is some overlap in the clustered position in the middle of the group, where ALL sample 13 is clustered by the NMF algorithm with the blue AML group. This, however, is completely consistent with Golub's results which note that a couple samples appear to have been mislabeled/erroneous, and are not correctly classified by any current technique.

Based on this dataset and analysis, the LLE/NMF technique appears to be very effective. The nested structure of the image clusters and the agreement between the LLE embedding and the NMF labeling is quite promising. However, it should be noted that this dataset has become a benchmark in the cancer classification community [3] and is not consider par-

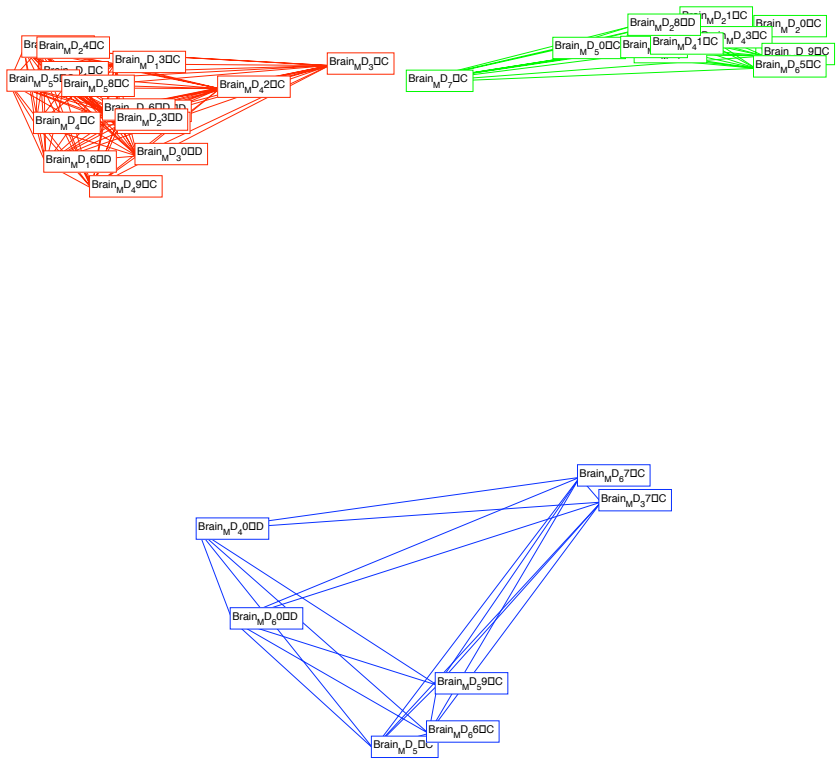

Figure 2: Analysis of using LLE/NMF on Golub's medulloblastomas data set. k=5, and the number of clusters = 3. Our algorithm visualizes 3 distinct regions which are not neccesarily verified by known biological differences of the samples. However the results of the algorithm strongly suggest 3 distinct classes of medulloblastomas.

ticularly difficult to analyze.

Analysis of Medulloblastomas Data

As compared with the clear differentiation offered by the leukemia data, our results for the medulloblastomas data set are not as clear. Although NMF and LLE strongly agree on the existence of three distinct clusters (see figure 2), this result is not verified by the known biological differences of the samples. There are two known subclasses: classic (C) and desmoplastic (D), which are clearly distinguished using a microscope [3].

Golub mentions the difficulty of clustering this data. Hierarchical clustering techniques are unable to provide any clear class distinctions, and are unable to distinguish the D cell types. NMF is able to find a distinct demoplastic cluster when the rank $= 5$ (see the blue region in figure 3), where one of the discovered classes is almost exclusively comprised of D cell types. However, we believe that due to the strong cross-validation of the LLE and NMF techniques that a more significant difference in gene expression data is being represented by these three

clusters. It is possible that there are three fundamentally different processes happening at a genetic level which only produce two distinct detectable features upon examination with a microscope. Without a strong understanding of the biological processes at work, it is difficult to further explore this hypothesis, but the data presented here remains very compelling.

Analysis of Yeast Shock Data

As a final, more interesting test of our clustering/visualization techniques, we attempted to discover the presence of sample clusters in the yeast shock data of Gasch et al [2]. As described earlier, the Gasch group explored genomic expression patterns in yeast responding to a wide range of environmental stress tests. The data set includes 6152 genes and 173 samples.

Initially, we ran the LLE algorithm over a wide range of k values and visually inspected the results, looking for pronounced clusters and groupings. While it remained difficult to see distinct, tight clusters for all 173 samples, we could see groupings of similar tests in different regions of the embedding. After a number of trials, we choose a rank $= 7$ and ran the NMF algorithm. The results can be seen is figure 4.

We can see from this result how certain groups of stress tests appear to gather together. For example, the nitrogen depletion stress tests from a long spike (colored black) down the lower left side of the image. The longer tests protrude further than the shorter ones (for example, the 5 day test is father below the 3 day test, which is below the 1 day test, etc). This provides very strong evidence for a biologically significant clustering of these samples.

Other interesting features include the blue cluster of YPD stationary phases. The blue cluster almost exclusively captures all the stationary phase samples, and within the cluster, samples are positioned according to the length of the experiment (like with the nitrogen tests).

Heat shock treatments of different lengths, temperatures and types appear to cluster significantly at near the top of the LLE embedding. The NMF algorithm with rank $= 7$ does not appear to uniquely classify them as well an the nitrogen and stationary phase tests, and instead groups the samples with a variety of other tests. This is likely a function of the large number of sample points used in the LLE embedding: with so many sample points available, it is difficult to visualize and choose an accurate rank for the NMF clustering.

Biologically, our results offer interesting observations of the Gasch environmental stress data set. It is quite possible to infer that nitrogen depletion, stationary phase and heat shock stresses affect the yeast cells in a way fundamentally different from the majority of the other stress tests. This observation is partially supported by the original Gasch research [2] which qualitatively comments on the gene expression level similarlity of these tests compared with (and unique from) the other stresses. By this hypothesis, the large green and yellow grouping of many different tests in the lower right spike of the image represents a group of tests which respond in similar ways to most of the stresses experienced.

Discussion and Future Directions

LLE/NMF is a valuable tool for biologists to visually inspect clusters in gene-expression data because it combines the simplicity of a rank-independent visualization technique (LLE) with a robust, mathematically rigorous clustering algorithm (NMF). Our implementation has been proven accurate analyzing simple data such as the leukemia data set, and has offered an interesting perspective on more difficult clustering problems as well, such as the medulloblas-

Figure 3: Analysis of using LLE/NMF on Golub's medulloblastomas data set. k=5, and the number of clusters = 5. Our algorithm visualizes 5 distinct regions. The desmoplastic class is properly identified by NMF however LLE still suggests 3 distinct classes of medulloblastomas.

tomas and yeast data sets.

There are two directions in which we would like to take this project in the future. The first is an automatic estimation of k for LLE. Although this parameter does not have a large effect on output (compared to NMF for example), picking a k value which is either much bigger or smaller then it should be can yield inaccurate results. We have started doing some simple analysis of the strength of the LLE embedding by looking at the eigenvalues found by the eigensolver in the last step of LLE. However, this process could be automated and optimized, to remove the initial step of tuning k to fit the non-linearity of the data. Secondly, we believe that this technique could be largely improved by adding a level of interactivity in the visualization/clustering process. When analyzing a large number of samples, it is important to have an interface that can scale the data representation appropriately. The ability to exclude certain samples from the visualization, focus on specific clusters, etc. would be a valuable feature, which would allow biologists to more easily

explore the data. Furthermore, NMF could be run interactively: a user could select an interesting region of the LLE embedding and run NMF to cluster the points found exclusively in that region. This level of interactivity would greatly improve user's ability to gain insight into inherent relationships in the data by further increasing the cross-validation of NMF and LLE, and simplifying the task of creating meaningful, informative visualizations.

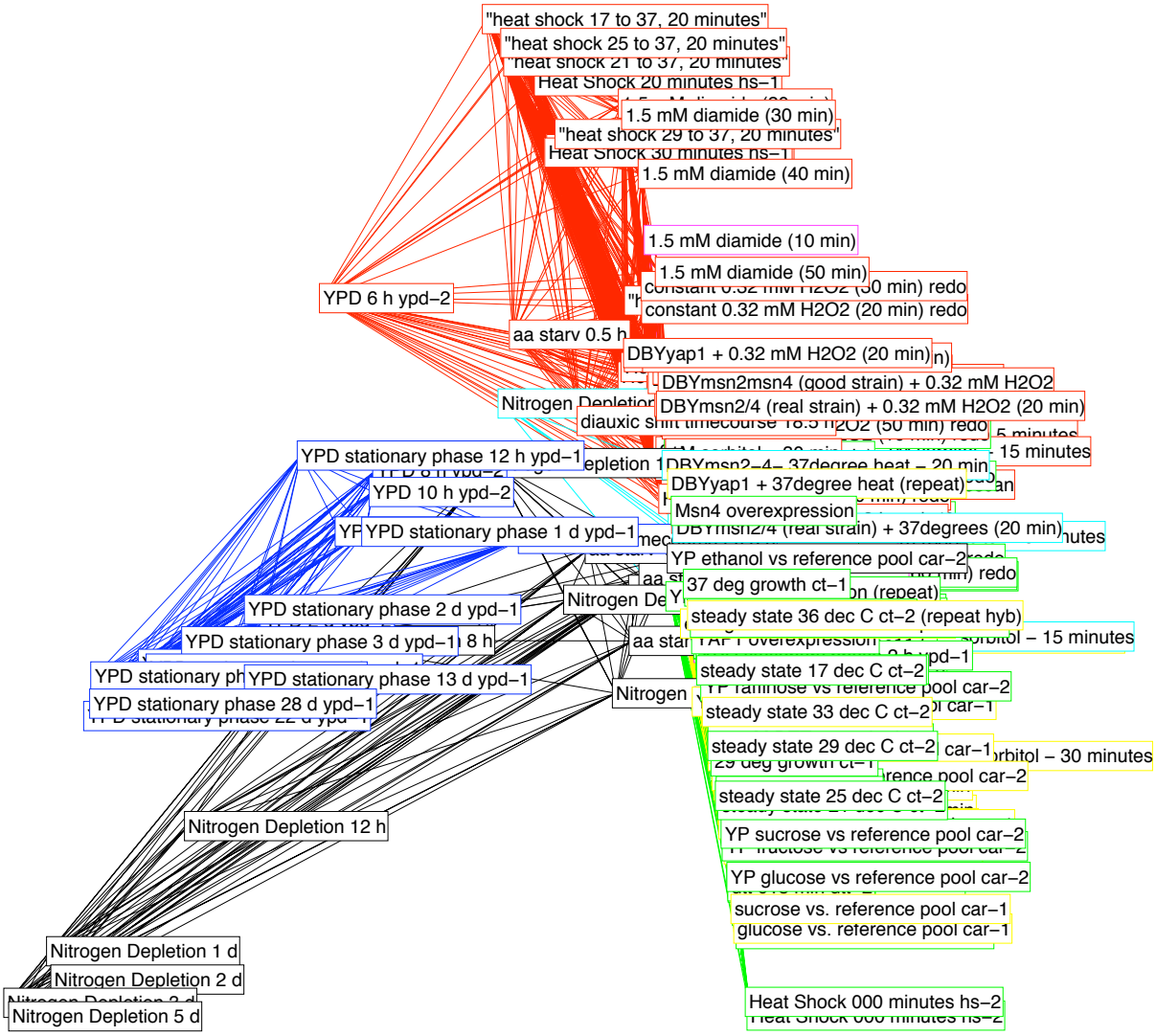

Figure 4: Analysis of using LLE/NMF on the yeast shock data set. k=30, and the number of clusters = 7. Our algorithm clearly shows certain shocks affecting the gene expression levels in similar ways.

References

[1] M. B. Eisen and P. O. Brown. Dna arrays for analysis of gene expression. *Methods Enzymol*, 303:179–205, 1999.

[2] Audrey P. Gasch et al. Genomic expression programs in the response of yeast cells to environmental changes. *Molecular Biology of the Cell*, 11:4241–4257, 2000.

[3] T.R. Golub and D.K. Slonim et al. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

[4] T. Golub J. P. Brunet, P. Tamayo and Jill Mesirov. Metagenes and molecular pattern discovery using matrix factorization. *PNAS*, 101:4164–4169, 2004.

[5] S. M. Lin and K. F. Johnson, editors. *Methods of Microarray Data Analysis: Papers from CAMDA 2000*. Kluwer Academic, Boston, 2002.

[6] D. V. Nguyen and D. M. Rocke. *Classification of Acute Leukemia Based on DNA Microarray Gene Expressions Using Partial Least Squares*. Kluwer Academic, Boston, 2002.

[7] P. Tamayo and S. Ramaswamy. *Expression Profiling of Human Tumors: Diagnostic and Research Applications*. June 2002.

A Code Listing

A.1 clusterAndPlot.m

```
% Cluster and Plot
% Blake Shaw & Hart Lambur
% Columbia University
%
% This script runs the nmf/lle combination algorithm on
% a variety of different datasets



%set this variable to the path which contains the data folders
%leukemia_data, medulloblastomas_data, shock_data
dataFolder = '/Users/blake/Desktop/metagenes/';

%1 -- leukemia data set
%2 -- brain tumor data
%2 -- yeast data
demo = 2;


if demo==1
    % params
    numClusters = 3;
    K = 6;

    % get data
    disp(sprintf('Loading Data'));
    load([dataFolder 'leukemia_data/ALL_AML_data.txt']);
    AMLLabels = readtextfile([dataFolder...
 'leukemia_data/ALL_AML_samples.txt']);

    data = ALL_AML_data;
    labels = AMLLabels;

elseif demo==2
    % params
    numClusters = 3;
    K = 5;

    % get data
    disp(sprintf('Loading Data'));
    load([dataFolder 'medulloblastomas_data/Medulloblastoma_data.txt']);
```

```
    med_labels = readtextfile([dataFolder...
 'medulloblastomas_data/Medulloblastomas_samples.txt']);

    data = Medulloblastoma_data;
    labels = med_labels;

elseif demo==3
    % params
    numClusters = 7;
    K = 30;

    % get data
    disp(sprintf('Loading Data'));
    load([dataFolder 'shock_data/gasch_data.txt']);
    gaschLabels = readtextfile([dataFolder...
 'shock_data/gasch_samples.txt']);

    data = gasch_data + 10;
    data = data(:, :);
    labels = gaschLabels;

end

% print info
[D,N] = size(data);
disp(sprintf('Dimensions: %d, Number of Items: %d', D, N));

% nmf
disp(sprintf('Running NMF -- Finding %d clusters', numClusters));
[w, h] = nmf(data, numClusters, 0);
[num, indices] = max(h);
clusters = indices';

% lle
disp(sprintf('Running LLE -- K = %d', K));
[Y, nearestNeighbors] = lle(data, labels, K, 2, 0, 1);

% nmf
disp(sprintf('Plotting'));

figure(1);
clf;

for i=1:N
   for j=1:N
```

```
        if i<j
            if clusters(i) == clusters (j)
                line( [Y(1, i), Y(1, j)], [ Y(2, i), Y(2, j)],...
 'Color', getColor(clusters(i)));
            end
        end
   end
end

for i=1:N
    text(Y(1, i), Y(2, i), labels(i, :), 'EdgeColor',...
 getColor(clusters(i)), 'BackgroundColor', 'white' );
end

axis([-1, 1, -1, 1]);
```

A.2 lle.m

```
% Locally Linear Embedding
% Blake Shaw & Hart Lambur
% Columbia University
%
% [Y, nearestNeighbors, eigStrength] = lle(X, labels, K,dOutput, plotType,...
% distanceType)
%
% X = D x N matrix where N is the number of points in a D dimensional space
% K = number of neighbors
% distanceType = 1 -- euclidean distance, 2 -- KL Divergence
% labels = labels for points
% dOutput = number of dimensions for output Y
% Y = points in a dOutput dimensional space
% nearestNeighbors = matrix of nearest neighbors for each item
% dimensions

function [Y, nearestNeighbors, eigStrength] = lle(X, labels, K,dOutput,...
plotType, distanceType)

[D,N] = size(X);
disp(sprintf('LLE for %d points in %d dimensions',N,D));

disp(sprintf('Calculating Distance Matrix'));
distance = ones(N, N);


if distanceType == 1
```

```
    disp(sprintf('Using Euclidean Distance'));
    for i=1:N
        %disp(sprintf('%d out of %d', i, N));
        for j=1:N
            dist = 0;
            for k=1:D
                if(i ~= j)
                    %euclidean distance
                    A = X(k, i);
                    B = X(k, j);
                    dist = dist + (A - B)^2;
                end
            end
            distance(i, j) = dist^0.5;
        end
    end
elseif distanceType == 2
    disp(sprintf('Using KL Divergence'));
    for i=1:N
        for j=1:N
            dist = 0;
            for k=1:D
                if(i ~= j)
                    %kl divergence
                    P = X(k, i);
                    Q = X(k, j);
                    if (P/Q)==0
                        disp(sprintf('Error for KL divergence: P=%f, Q=%f', P, Q));
                    end
                    dist = dist + P * log(P / Q);
                end
            end
            distance(i, j) = dist;
        end
    end
end

disp(sprintf('Finding %d nearest neighbors', K));
[sortedDistances,indexes] = sort(distance);


% disp(sprintf('Printing Neighbors to file...'));
% fid = fopen('UserDistances.txt', 'w');
% for printUser=1:N
%     fprintf(fid, 'Neighbors for %s\n', labels(printUser, :));
```

```
%     for(i=1:N)
%         fprintf(fid,'     --->%s -- %6f\n', labels(indexes(i, printUser), :),...
% distance(printUser, indexes(i, printUser)));
%     end
% end
%fclose(fid);

nearestNeighbors = indexes(2:(1+K),:); %index 1 should be the point itself


disp(sprintf('Solving for weights'));

z = zeros(D, K);
W = zeros(K,N);
for i=1:N

   %subtract Xi from every column of Z
   for j=1:K
       z(:, j) = X(:,nearestNeighbors(j,i)) - X(:,i);
   end

   C = z'*z;
   C = C + eye(K,K); %help numerical instabilities
   W(:,i) = inv(C) * ones(K, 1);
   W(:,i) = W(:,i)/sum(W(:,i));
end

disp(sprintf('Computing embedding coordinates using weights'));


M = sparse(1:N,1:N,ones(1,N),N,N,4*K*N); %sparse matrix
for i=1:N
   w = W(:,i);
   j = nearestNeighbors(:,i);
   M(i,j) = M(i,j) - w';
   M(j,i) = M(j,i) - w;
   M(j,j) = M(j,j) + w*w';
end

%M2 = ((eye(K, N) - W)' * (eye(K, N) - W)) % why doesnt this work?

%options.disp = 0;
%[Y,eigenvals] = eigs(M,N-1);
%eigenvals(N-2-dOutput:N-2, :);
numEigens = dOutput + 4;
```

```
[Y, eigenvals] = jdqr(M, numEigens, 0);
eigValues = diag(eigenvals);

disp(sprintf('# of eigenvalues: %d', length(eigValues)));
eigStrength = eigValues(dOutput+1+1) - eigValues(dOutput+1);
%eigStrength = 0;
Y = Y(:,2:dOutput+1)';

if plotType == 1
    disp(sprintf('Plotting'));
    figure(1);
    clf;
    % plot(Y(1, :), Y(2, :), 'r+');
    % axis([-1, 1, -1, 1]);

    if dOutput == 3
        for i=1:N
           for j=1:min(5, K)
              line( [Y(1, i), Y(1, nearestNeighbors(j,  i))],...
  [ Y(2, i), Y(2, nearestNeighbors(j, i))], [ Y(3, i), Y(3, nearestNeighbors(j, i))] );
           end
        end

        for i=1:N
           text(Y(1, i), Y(2, i), Y(3, i), labels(i, :),...
    'EdgeColor', 'blue', 'BackgroundColor', 'white' );
        end
        axis([-1, 1, -1, 1, -1, 1]);
    else
       for i=1:N
           for j=1:min(5, K)
              line( [Y(1, i), Y(1, nearestNeighbors(j,  i))],...
   [ Y(2, i), Y(2, nearestNeighbors(j, i))]);
           end
        end

        for i=1:N
           text(Y(1, i), Y(2, i), labels(i, :), 'EdgeColor',...
    'blue', 'BackgroundColor', 'white' );
        end
        axis([-1, 1, -1, 1]);
    end
    title('Map');

     figure(2);
```

```
     clf;
     bar(eigValues(2:dOutput+1+2));
     axis([0, length(eigValues) + 1, 0, max(eigValues) + 0.1]);
     title('Eigenvalues');
elseif plotType == 2
    disp(sprintf('Plotting'));
    figure(1);
    clf;
    plot(Y(1, :), Y(2, :), 'r+');
    axis([-1, 1, -1, 1]);
end
```

A.3 nmf.m

```
function [w,h]=nmf(v,r,verbose)
%
% Jean-Philippe Brunet
% Cancer Genomics
% The Broad Institute
% brunet@broad.mit.edu
%
% Edited by:
% Blake Shaw & Hart Lambur
% Columbia University
%
% This software and its documentation are copyright 2004 by the
% Broad Institute/Massachusetts Institute of Technology. All rights are reserved.
% This software is supplied without any warranty or guaranteed support whatsoever.
% Neither the Broad Institute nor MIT can not be responsible for its use, misuse,
% or functionality.
%
% NMF divergence update equations :
% Lee, D..D., and Seung, H.S., (2001), 'Algorithms for Non-negative Matrix
% Factorization', Adv. Neural Info. Proc. Syst. 13, 556-562.
%
% v (n,m) : N (genes) x M (samples) original matrix
%           Numerical data only.
%           Must be non negative.
%           Not all entries in a row can be 0. If so, add a small constant to the
%           matrix, eg.v+0.01*min(min(v)),and restart.
%
% r       : number of desired factors (rank of the factorization)
%
% verbose : prints iteration count and changes in connectivity matrix elements
%           unless verbose is 0
```

```
%
% Note : NMF iterations stop when connectivity matrix has not changed
%        for 10*stopconv interations. This is experimental and can be
%        adjusted.
%
% w    : N x r NMF factor
% h    : r x M NMF factor


% test for negative values in v
if min(min(v)) < 0
    error('matrix entries can not be negative');
    return
end
if min(sum(v,2)) == 0
    error('not all entries in a row can be zero');
    return
end


[n,m]=size(v);
stopconv=40;      % stopping criterion (can be adjusted)
niter = 2000;     % maximum number of iterations (can be adjusted)

cons=zeros(m,m);
consold=cons;
inc=0;
j=0;

%
% initialize random w and h
%
w=rand(n,r);
h=rand(r,m);


for i=1:niter

    % divergence-reducing NMF iterations

    x1=repmat(sum(w,1)',1,m);
    h=h.*(w'*(v./(w*h)))./x1;
    x2=repmat(sum(h,2)',n,1);
    w=w.*((v./(w*h))*h')./x2;
```

```
    % test convergence every 10 iterations

    if(mod(i,10)==0)
        j=j+1;

    % adjust small values to avoid undeflow
        h=max(h,eps);w=max(w,eps);


    % construct connectivity matrix
        [y,index]=max(h,[],1);   %find largest factor
        mat1=repmat(index,m,1);  % spread index down
        mat2=repmat(index',1,m); % spread index right
        cons=mat1==mat2;

        if(sum(sum(cons~=consold))==0) % connectivity matrix has not changed
            inc=inc+1;                    %accumulate count
        else
            inc=0;                        % else restart count
        end

        if verbose                     % prints number of changing elements
            fprintf('\t%d\t%d\t%d\n',i,inc,sum(sum(cons~=consold))),
        end

        if(inc>stopconv)
            break,                % assume convergence is connectivity stops changing
        end

        consold=cons;

    end
end
```

A.4 readtextfile.m

```
function tab=readtextfile(filename)
```

```
% Read a text file into a matrix with one row per input line
% and with a fixed number of columns, set by the longest line.
% Each string is padded with NUL (ASCII 0) characters
%
% open the file for reading
ip = fopen(filename,'rt');   % 'rt' means read text
if (ip < 0)
    error('could not open file');% just abort if error
end;
% find length of longest line
max=0;          % record length of longest string
cnt=0;          % record number of strings
s = fgetl(ip);              % get a line
while (ischar(s))     % while not end of file
   cnt = cnt+1;
   if (length(s) > max)    % keep record of longest
max = length(s);
   end;
    s = fgetl(ip);      % get next line
end;
% rewind the file to the beginning
frewind(ip);
% create an empty matrix of appropriate size
tab=char(zeros(cnt,max));% fill with ASCII zeros
% load the strings for real
cnt=0;
s = fgetl(ip);
while (ischar(s))
   cnt = cnt+1;
   tab(cnt,1:length(s)) = s;% slot into table
    s = fgetl(ip);
end;
% close the file and return
fclose(ip);
return;
```

A.5 getcolor.m

```
function [pColor] = getColor(i)
if i==1
    pColor = [1, 0, 0];
elseif i==2
    pColor = [0, 1, 0];
elseif i==3
    pColor = [0, 0, 1];
```

```
elseif i==4
    pColor = [0, 0, 0];
elseif i==5
    pColor = [0, 1, 1];
elseif i==6
    pColor = [1, 1, 0];
elseif i==7
    pColor = [1, 0, 1];
elseif i==8
    pColor = [0.5, 0, 0];
elseif i==9
    pColor = [0, 0.5, 0];
elseif i==10
    pColor = [0, 0, 0.5];
else
    pColor = [1, 1, 1];
end
```