Autotuned Multilevel Clustering of Gene Expression Data

Hasin Afzal Ahmed¹, Priyakshi Mahanta¹, Dhruba Kumar Bhattacharyya¹*, Jugal Kumar Kalita²

¹Department of Comp. Sc. and Engg., Tezpur University, Napaam, 784028, India
²Department of Comp. Sc., University of Colorado, Colorado Springs, USA

Abstract DNA microarray technology has revolutionized biological and medical research by enabling biologists to measure expression levels of thousands of genes in a single experiment. Different computational techniques have been proposed to extract important biological information from the massive amount of gene expression data generated by DNA microarray technology. This paper presents a top-down hierarchical clustering algorithm that produces a tree of genes called GERC tree (GERC stands for Gene Expression Recursive Clustering) along with the generated clusters. GERC tree is an ample resource of biological information about the genes in an expression dataset. Unlike dendrogram, a GERC tree is not a binary tree. Genes in a leaf node of GERC tree represent a cluster. The clustering method was used with real-life datasets and the proposed method has been found satisfactory in terms of homogeneity, p value and z-score.

Keywords Hierarchical Clustering, Divisive Clustering, Mean Squared Residue, Gene Expression Data

1. Introduction

DNA microarray technology enables the biologists to monitor expression levels of thousands of genes in a single microarray experiment. There is a high demand of computational techniques to operate on the massive amount of expression data generated by DNA microarray technology to extract important biological information. Due to the large number of genes and complex gene regulation networks, clustering is a useful exploratory technique for analyzing such data. It groups data of interest into a number of relatively homogeneous groups or clusters where the intra-group object similarity is minimized and the inter-group object dissimilarity is maximized. Problems of automatically classifying data arise in many areas, and hierarchical clustering can be a very good approach in certain areas such as gene expression data analysis because it can present a hierarchical organization of the clusters.

Extracting important biological knowledge from biological data is a difficult task. One very useful approach for providing insight into the gene expression data is to organize the genes in a hierarchy of classes, where genes in a class are more similar compared to its ancestor classes in the hierarchy. In this paper, we present a polythetic divisive hierarchical clustering algorithm that operates in two distinct steps. The first step generates a number of initial clusters and in the second step, these initial clusters are further processed to form a set of finer clusters. The algorithm advances clustering of microarray data in following ways (a) Extraction of initial clusters is faster and it does not require any proximity measure. (b) During discovery of final clusters, a proximity measure referred here as MRD(Mean Residue Distance) is used to find mutual distance among genes within a particular initial cluster instead of operating on the entire set. (c) The algorithm is capable of tuning the threshold to be used to decompose a node to its child nodes itself. (d) The algorithm stores the tree structure which can be later used in different applications. (e) The algorithm allows overlapping of genes among child nodes of a particular node.

The rest of the paper is organized as follows. In section 2, we discuss related work. Section 3 presents the algorithm. Experimental results are reported in section 4. Finally, discussion and future work are reported in section 5 and section 6 respectively.

2. Related Work

Hierarchical clustering usually generates a hierarchy of nested clusters or, in other words, a tree of clusters, also known as a dendrogram. Hierarchical clustering methods are categorized into agglomerative (bottom-up) and divisive (top-down). A large number of clustering techniques have been reported for analyzing gene expression data, such as Unweighted Pair Group Method with Arithmetic Mean (UPGMA)[1], Self Organizing Tree Algorithm (SOTA)[2], Divisive Correlation Clustering Algorithm (DCCA)[3], Density-Based Hierarchical clustering method (DHC)[4] and Dynamically Growing Self-Organizing Tree (DGSOT)[5]. Unweighted Pair Group Method with Arithmetic Mean adopts an agglomerative method to graphically represent the...
clumped dataset. The method is much favored by many biologists and has become one of the most widely-used tools in gene expression data analysis. However, it suffers from lack of robustness, i.e., a small perturbation of the dataset may greatly change the structure of the hierarchical dendrogram. DHC is a popular density based clustering algorithm. DHC is developed based on ‘density’ and ‘attract’ of data objects. In the first level, an attraction tree is constructed to represent the relationship between the data objects in the dense area which is later summarized to a density tree. Another approach splits the genes through a divisive approach, called the Deterministic-Annealing Algorithm (DAA)[6]. Hierarchical clustering not only groups together genes with similar expression patterns but also provides a natural way to graphically represent the dataset allowing a thorough inspection. However, like UPGMA, a small change in the dataset may greatly affect the hierarchical dendrogram structure. Another drawback is its high computational complexity and vagueness of termination criteria.

### 3. The GERC Algorithm

GERC is a polythetic divisive hierarchical clustering algorithm that operates in two distinct steps. This is an extended version of the article[7] where the method was introduced. In the first step of the algorithm, an initial cluster is formed and this initial cluster is further processed in the second step to form finer clusters. The algorithm accepts four input parameters i.e., reference gene, step down ratio, preferred node volume and MRD threshold. However, the last three parameters can be statistically computed from the expression data. The technique can operate on any high dimensional numeric domain.

#### 3.1. Data Pre-processing

Often gene data available on the web are found to contain missing values. The quality of clusters largely depends on the handling of these missing values. Apart from missing value handling, pre-processing also involves normalization and discretization.

**Handling missing values**

We used the Local Least Squares Imputation method[8] to compute missing values in the datasets. There are two steps in the local least squares imputation method. The first step is to select $k$ genes by Pearson correlation coefficient. The second step is regression using the selected $k$ genes to estimate the missing values.

**Normalization**

The datasets are normalized using a common statistical method that converts each gene to a normal distribution with mean 0 and variance 1. This statistical method of normalization is often termed as Z score normalization[9] or Mean 0 Standard Deviation 1 normalization.

**Discretization**

The normalized matrix is discretized to a matrix by comparing a value in a column with the value in the next column of the same row. The normalized matrix consists of three discrete values 1 (if the next value is larger), -1 (if the next value is smaller) and 0 (if the next value is equal). The normalized matrix $G$ can be converted to the discretized matrix $G^d$ in the following manner.

$$
G^d(i,j) = \begin{cases} 
0, & \text{if } G(i,j) = G(i,j+1) \\
1, & \text{if } G(i,j) < G(i,j+1) \\
-1, & \text{if } G(i,j) > G(i,j+1) 
\end{cases}
$$

#### 3.2. Proximity Measure

In this paper, we introduce a simplified form of mean squared residue measure, i.e., Mean Residue Distance(MRD) to find mutual distance of two genes that aids in extracting the coherent patterns in the expression matrix. Like mean squared residue measure, MRD is a measure that works satisfactorily to detect coherence of constant valued genes, constant row genes, constant column genes and additive genes. The significance of these correlations in clustering of gene expression data is reported in[10]. Unlike MSR, MRD can operate in mutual mode i.e., it can compute correlation
between a pair of genes. Next we discuss the mean squared residue measure and then introduce MRD measure.

**Mean Squared Residue**

Mean squared residue is a measure that was used to find coherent objects in a data matrix by [11]. They tried to find a subset of genes along with a subset of conditions which has mean squared residue less than a threshold $\delta$. They termed such subspace clusters as $\delta$ biclusters. The measure is still considered a strong one to detect coherent objects if it is used carefully. Various subspace clustering algorithms use mean squared residue directly or with a bit of modification. Mean squared residue of an element $a_{ij}$ in gene expression matrix is given by,

\[
(a_{ij} - a_{\text{mean}} - a_{iJ} + a_{Ij})^2
\]

where,

- $a_{Ij} = \frac{1}{|J|} \sum_{j \in J} a_{ij}$, row mean of $i^{th}$ gene,
- $a_{ij} = \frac{1}{|I|} \sum_{i \in I} a_{ij}$, row mean of $j^{th}$ condition,
- $a_{IJ} = \frac{1}{|I||J|} \sum_{i \in I, j \in J} a_{ij}$, mean of the subspace cluster,

$I$ is the set of genes and $J$ is the set of conditions.

**Mean Residue Distance**

The mean residue distance of an element $a_{i}$ of gene $g_{i}= (a_{1}, a_{2}, ..., a_{n})$ with respect to another element $b_{j}$ of gene $g_{j}= (b_{1}, b_{2}, ..., b_{n})$ is defined as,

\[
MRD_{I}(g_{i}, g_{j}) = |a_{i} - a_{\text{mean}} - b_{j} + b_{\text{mean}}|
\]

Where $a_{\text{mean}}$ is the mean of all the elements of $g_{i}$ and $b_{\text{mean}}$ is the mean of all the elements of $g_{j}$. MRD of the gene pair $g_{1}$ and $g_{2}$ with respect to a subspace of conditions $\lambda$ can be computes as,

\[
MRD_{\lambda}(g_{1}, g_{2}) = \sum_{i \in \lambda} |a_{i} - a_{\text{mean}} - b_{i} + b_{\text{mean}}|
\]

Following definitions and theorems provide the theoretical basis and soundness of the proposed measure based on [12].

**Definition 1:** **Coherent genes:** Two genes are called coherent if similarity between the two genes is more than a given threshold in term of a particular proximity measure.

**Definition 2:** **Expression pattern:** The expression pattern of a gene is defined as the discretized form of the gene expression values. Two genes are said to have similar expression pattern if their discretized values are same. Mathematically, two genes $g_{i}$ and $g_{j}$ have similar expression pattern if

\[
G_{i}(i, k) = G_{j}(j, k), \text{ for } k = 1, 2, 3, ..., n
\]

Where $n$ is the total number of conditions.

**Definition 3:** **Constant valued genes:** For two genes $g_{i} = (a_{1}, a_{2}, ..., a_{n})$ and $g_{j} = (b_{1}, b_{2}, ..., b_{n})$ if $a_{1} = a_{2} = ... = a_{n} = b_{1} = b_{2} = ... = b_{n}$, then the genes are called constant valued genes.

**Definition 4:** **Constant row genes:** For two genes $g_{i} = (a_{1}, a_{2}, ..., a_{n})$ and $g_{j} = (b_{1}, b_{2}, ..., b_{n})$, if $a_{1} = a_{2} = ... = a_{n}$ and $b_{1} = b_{2} = ... = b_{n}$, then the genes are called constant row genes.

**Definition 5:** **Constant column genes:** For two genes $g_{i} = (a_{1}, a_{2}, ..., a_{n})$ and $g_{j} = (b_{1}, b_{2}, ..., b_{n})$, if $a_{1} = b_{1}, a_{2} = b_{2}, ..., a_{n} = b_{n}$, then the genes are called constant column genes.

**Definition 6:** **Additive genes:** For two genes $g_{i} = (a_{1}, a_{2}, ..., a_{n})$ and $g_{j} = (b_{1}, b_{2}, ..., b_{n})$, if $b_{1} = a_{1} + d, b_{2} = a_{2} + d, ..., b_{n} = a_{n} + d$, where $d$ is an additive constant, then the genes are called additive genes.

**Properties of MRD**

The MRD measure is capable of detecting four types of coherence (a) Coherence among constant valued genes (b) Coherence among constant row genes (c) Coherence among constant column genes (d) Coherence among additive genes. Next we present some of the properties of MRD.

**Theorem 1.** MRD of two constant valued genes is always zero.

**Proof:**

Let the two genes be $g_{i} = (a_{1}, a_{2}, ..., a_{n})$ and $g_{j} = (b_{1}, b_{2}, ..., b_{n})$.

Since the two genes are constant valued so $a_{1} = a_{2} = ... = a_{n} = b_{1} = b_{2} = ... = b_{n} = x$ (say).

Now mean of the two genes will be, $a_{\text{mean}} = b_{\text{mean}} = x$.
MRD\[1, 2, \ldots, n] (g_1, g_2) = \sum_{i=1}^{n} |a_i - a_{\text{mean}} - b_i + b_{\text{mean}} | \\
= \sum_{i=1}^{n} |a_i - a_{\text{mean}} - b_i + b_{\text{mean}} | \\
= \sum_{i=1}^{n} |x - x - x + x | \\
= 0

**Theorem 2.** MRD of two constant row genes is always zero.

**Proof:**

Let the two genes be \( g_1 = (a_1, a_2, \ldots, a_n) \) and \( g_2 = (b_1, b_2, \ldots, b_n) \). Since these are constant row genes, so \( a_1 = a_2 = a_3 = \ldots = a_n = x \) (say) and \( b_1 = b_2 = b_3 = \ldots = b_n = y \) (say).

Now mean of the first gene, \( a_{\text{mean}} = \frac{1}{n} \sum_{i=1}^{n} x \)
\[= \frac{1}{n} (n \times x) = x \]

Similarly mean of second gene \( b_{\text{mean}} = y \)

**Theorem 3.** MRD of two constant column genes is always zero.

**Proof:**

Let the two genes be \( g_1 = (a_1, a_2, \ldots, a_n) \) and \( g_2 = (b_1, b_2, \ldots, b_n) \). Since these are constant column genes, so \( a_1 = b_1, a_2 = b_2, \ldots, a_n = b_n \).

Now mean of two genes will be \( a_{\text{mean}} = b_{\text{mean}} = m \) (say).

**Theorem 4.** MRD of two additive genes is always zero.

**Proof:**

Let the two genes be \( g_1 = (a_1, a_2, \ldots, a_n) \) and \( g_2 = (b_1, b_2, \ldots, b_n) \). Since the genes are additive, so \( b_i = a_i + d \), for \( i = 1, 2, 3, \ldots, n \). Here \( d \) is an additive constant.

Now mean of the first gene \( a_{\text{mean}} = \frac{1}{n} \sum_{i=1}^{n} a_i = m \) (say).

Mean of the second gene \( b_{\text{mean}} = \frac{1}{n} \sum_{i=1}^{n} b_i \)
\[= \frac{1}{n} \sum_{i=1}^{n} (a_i + d) \]
\[= \frac{1}{n} \sum_{i=1}^{n} a_i + d \]
\[= m + d \]

**Figure 2.** Structure of GERC tree

### 3.3. GERC Tree

Our algorithm results a tree called GERC tree. The leaves of this tree present the generated clusters of the algorithm. This tree can be used to derive additional biological information from a gene expression dataset. The overall structure of the tree generated for more than one reference gene is presented in Fig. 2.

**Dendrogram versus GERC tree**

A dendrogram is binary tree that presents the hierarchical structure of the clusters generated by a hierarchical algorithm. In divisive hierarchical algorithm, dendrogram is obtained by recursively splitting a node containing a set of objects into two child nodes based on the similarity among the object pairs until all nodes have a single object. Conversely, in agglomerative approach, the nodes containing a single object are recursively merged until all the objects are in the root node. But in our algorithm, GERC tree is obtained by
recursively splitting a single node containing the set of all objects into multiple nodes (with possibly common objects) until number of objects in all the processing nodes is less than or equal to a user given threshold, i.e., *preferred node volume*. Unlike Dendrogram, GERC tree is not a binary tree and the structure of the tree is flexible depending on the values of the set of input parameter. The structural difference between dendrogram and GERC tree is presented in Fig. 3.

**Table 2.** Comparison of Dendrogram VS GERC Tree

| SYMBOL   | MEANING                        |
|----------|--------------------------------|
| N_c      | Total number of conditions     |
| R_g      | Reference gene                 |
| n        | No of genes                    |
| G        | The gene expression matrix     |
| G'       | Discretized gene expression matrix |
| g_i      | i^th gene                      |
| T        | Preferred Node Volume          |
| C        | Initial Cluster                |
| η(i)     | i^th node of the tree          |
| N_t      | Number of tree nodes           |
| dist(g_i,g_j) | MRD between i^th and j^th genes as in the distance matrix |
| α        | MRD threshold                  |
| δ        | Step down ratio                |
| Neighbour β(g_i) | Neighbours of i^th gene with respect to MRD threshold β |
| Exp(g_i) | Expression pattern of i^th gene presented by i^th row of the discretized matrix |
| father(η(i)) | father of i^th node           |
| degree β(g_i) | Degree of i^th gene with respect to MRD threshold β |
| η(i)threshold | MRD threshold of i tree node  |
| η(1)     | The root node of the GERC tree containing all the genes |

**Definition 7:** Neighbour: Two genes g_i and g_j are said to be neighbours of each other with respect to a threshold β if dist(g_i, g_j) ≤ β.

**Definition 8:** Degree: The degree of a gene g_i with respect to threshold β is defined as the number of genes which are within the β-neighbourhood of g_i.

**Definition 9:** Initial Cluster: An initial cluster is defined as a subset of genes S such that for any two genes g_i, g_j in S, Exp(g_i) = Exp(g_j) in terms of at least \((N_c/2)+1\) numbers of conditions.

**Definition 10:** Finer Cluster: A finer cluster is defined as a set of genes S such that for any two genes g_i, g_j in S, dist(g_i, g_j) <= δ, where δ is greater than or equal to the level of the node to which the finer cluster is associated.

The GERC algorithm consists of two steps. In the first step of GERC, all genes which have expression pattern similar to the reference gene in terms of \((N_c/2)+1\) number of conditions are put in an initial cluster. In the second step of the algorithm, the initial clusters are processed to produce finer clusters. While processing an initial cluster, all genes are put in the root node. Then iterative clustering is performed using a tuned MRD threshold on genes of each node to produce its child nodes. The process is repeated till all nodes in the tree contain a number of genes greater than user specified preferred node volume. The tuning of MRD threshold is governed by the user defined parameter *step down ratio*. The leaves of the tree represent the generated clusters. The algorithm can be used to produce a wide range of clusters by considering more than one reference gene. The detail of the algorithm is presented next.

### 3.4. Proposed Algorithm: GERC

Following definitions, symbols and notations are used to describe the GERC algorithm.
STEP 1:
Place $R_g$ in C.
Add all genes $g_k$ in C such that
\[ \text{Exp}(g_k) = \text{Exp}(R_g) \] in terms of $N_c$ conditions.
STEP 2:
Place all the genes $g_k$ such that $g_k \in C$ in $\eta(l)$.
Initialize $N_{\eta}$ to 1.
\[ \eta(1).\text{threshold} = \frac{\alpha}{\delta}. \]
\[ j = 1. \]
While $j \leq N_{\eta}$ and number of genes in $\eta(j) \geq T$ do
Compute distance matrix for all genes $g_k$ such that $g_k \in \eta(j)$ using MRD.
Compute degree $\eta(j).\text{threshold} \times \delta(g_k)$ and neighbor $\eta(j).\text{threshold} \times \delta(g_k)$ for each $g_k \in \eta(j)$.
Place all genes $g_k \in \eta(j)$ in $L$.
While $L$ is not empty do
Find $g_k \in L$ such that degree $\eta(j)(g_k)$ is highest.
Increment $N_{\eta}$.
Create a new node $\eta(N_{\eta})$.
Place neighbor $\eta(j).\text{threshold} \times \delta(g_m)$ and $g_m$ in $\eta(N_{\eta})$.
\[ \eta(N_{\eta}).\text{threshold} = \eta(j).\text{threshold} \times \delta. \]
father($\eta(N_{\eta})$) = $\eta(j)$.
Remove $g_m$ and neighbor $\eta(j).\text{threshold} \times \delta(g_m)$ from $L$.
end while
if $\eta(N_{\eta})$ is the only child of father($\eta(N_{\eta})$) then
Decrement $N_{\eta}$.
\[ \eta(j).\text{threshold} = \eta(j).\text{threshold} \times \delta. \]
Decrement $j$.
elserie
Increment $j$.
end if
end while
father($\eta(1)$) = $\eta(0)$.

In GERC, the reference gene is used as an input parameter. The algorithm tries to find the initial cluster to which this gene potentially belongs to. While finding this initial cluster it tries to locate all genes which have similar expression pattern with the reference gene in terms of at least $(N_c/2)+1$ number of conditions. If we want to explore the entire dataset we can use each of the available genes or a set of stochastically selected genes as reference genes. Once we discover the initial clusters we move to step 2 of the algorithm. In the second step, we create a single node first with all genes of the initial cluster in it and then iteratively cluster each node (having genes more than preferred volume number of genes) of the tree until all the processing nodes in the tree have less than or equal to preferred volume number of genes. The input parameter step down ratio is actually used to reduce the value of MRD threshold as towards leaf nodes of the tree the similarity among genes increases and require a smaller value of MRD threshold, e.g., if the MRD threshold provided by user is 1 and step down ratio is .5, then the first node will use 1 as its MRD threshold while clustering. On successful division of the node to its children, the successive level nodes will use thresholds $.5(1*.5), .25(.5*.5) \ldots$ and so on. Finally the sub-trees (one sub-tree in case of single reference gene) generated from the initial clusters are combined to form the GERC tree. The roots of these subtrees are made children of a single node that contains the set of entire genes and this node becomes the root of generated GERC tree.

3.5 Complexity Analysis

Since GERC involves two distinct steps, hence the total complexity is the sum of the complexities of these two steps. Let a dataset contains $n$ number of genes. In the first step of the algorithm, the total number of comparisons done to put all $n$ number of genes in the initial cluster is $n-1$. In the second step, if the number of genes in the initial cluster is $p$, the computation of distance matrix involves $px(p-1)/2$ operations. If the average number of genes in the non leaf nodes is $m$ and the total number of non leaf nodes is $l$, creating the child nodes and hence the entire tree requires $l*m$ operations. So complexity of the second step is $O(p(x(p-1)/2 + l*m))$.

4. Experimental Results

We implemented the GERC algorithm in MATLAB and tested it on four publicly available benchmark microarray datasets mentioned in Table 4. The test platform was a desktop PC with Intel Core 2 Duo 2.00 GHz processor and 512 MB memory running Windows XP operating system. Fig. 5, 6 and 7 present a part of the tree generated from an initial cluster for Dataset 2, Dataset 3 and Dataset 4 respectively.

4.1 Cluster Quality

The GERC algorithm was compared with various clustering algorithms and the results were validated using average homogeneity score[18], p value[19] and z score[20].

Cluster Homogeneity: Homogeneity measures the quality of clusters on the basis of the definition of a cluster i.e., Objects within a cluster are similar while objects in different clusters are dissimilar. It is calculated as follows.

(a) Compute average value of similarity between each gene $g_i$ and the centroid of the cluster to which it has been assigned.
(b) Calculate average homogeneity for the clustering C weighted according to the size of the clusters.
The homogeneity values for the GERC algorithm and some other algorithms are reported in Table 5. It can be observed that the homogeneity value for GERC is highest from which we can conclude that the coherence of the clusters produced by GERC are better than those produced by competing algorithms.

**p value:** The biological relevance of a cluster can be verified based on the gene ontology (GO) annotation database [http://db.yeastgenome.org/cgi-bin/GO/goTermFinder](http://db.yeastgenome.org/cgi-bin/GO/goTermFinder). It is used to test the functional enrichment of a group of genes in terms of three structured controlled ontologies, viz., associated biological processes, molecular functions and biological components. The functional enrichment of each GO category in each of the clusters obtained is calculated by its p-value.

The p-value is computed using a cumulative hyper geometric functional enrichment score based on Molecular Function and Biological Process annotations. The genes belonging to the enriched functional categories are biologically significant in the corresponding clusters. To compute p-value, we used the software FuncAssociate[19]. FuncAssociate computes hyper geometric functional enrichment score based on Molecular Function and Biological Process annotations. The enriched functional categories for some of the clusters obtained by GERC on Datasets 1, 2 and 4 are listed in Tables 6, 7 and 8 respectively. From the Tables, we can conclude that GERC shows a good enrichment of functional categories and therefore projects a good biological significance.

### Table 4. Datasets used for experimental results

| Dataset | Name | Source | Instances | Attributes |
|---------|------|--------|-----------|------------|
| 1       | Subset of Yeast Cell Cycle [13] | http://faculty.washington.edu/ | 384 | 17 |
| 2       | Yeast Diauxic Shift [14] | http://www.ncbi.nlm.nih.gov/geo/query | 6089 | 7 |
| 3       | Yeast Cell cycle [15] | Sample input files in expander [17] | 698 | 72 |
| 4       | Yeast Sporulation [16] | http://cmgm.stanford.edu/pbrown/sporulation | 474 | 7 |

### Table 5. Homogeneity values for GERC and other comparable algorithms

| Datasets | Method Applied | No. of clusters | Homogeneity |
|----------|----------------|-----------------|-------------|
| Dataset1 | K-Means        | 16              | 0.671       |
|          | SOM            | 16              | 0.710       |
|          | Click          | 3               | 0.549       |
|          | DCCA           | 15              | 0.818       |
|          | GERC           | 10              | 0.878       |
| Dataset3 | K-Means        | 5               | 0.577       |
|          | SOM            | 6               | 0.514       |
|          | Click          | 5               | 0.501       |
|          | DCCA           | 43              | 0.699       |
|          | GERC           | 11              | 0.805       |

### Table 6. p-value of some of the finer clusters for Dataset 1

| Cluster | p-value | GO number | GO category |
|---------|---------|-----------|-------------|
| C1      | 1.09E-05| GO:000731 | DNA synthesis involved in DNA repair |
|         | 3.07E-06| GO:0005657| replication fork |
|         | 4.57E-06| GO:0007064| mitochondrial maturation |
|         | 1.08E-05| GO:0007062| sister chromatid cohesion |
|         | 6.63E-05| GO:0006302| double-strand break repair |
|         | 1.67E-05| GO:0006260| DNA replication |
|         | 9.68E-06| GO:0006281| DNA repair |
|         | 2.24E-05| GO:0006974| response to DNA damage stimulus |
|         | 9.84E-06| GO:0006259| DNA metabolic process |
|         | 8.55E-05| GO:0033554| cellular response to stress |
| C2      | 5.67E-05| GO:0043596| nuclear replication fork |
|         | 2.79E-08| GO:0005657| replication fork |
|         | 6.49E-08| GO:0007064| mitochondrial maturation |
|         | 3.84E-07| GO:0007062| sister chromatid cohesion |
|         | 2.34E-12| GO:0006260| DNA replication |
|         | 5.23E-05| GO:0006281| DNA repair |
|         | 1.98E-12| GO:0004427| DNA metabolic process |
|         | 1.67E-11| GO:0006259| DNA repair |
|         | 3.68E-08| GO:0006259| DNA repair |
|         | 2.02E-07| GO:0006364| cell cycle process |
|         | 7.58E-06| GO:0048519| nuclear chromosome part |
|         | 1.35E-09| GO:0044454| negative regulation of nuclear chromosome part |
| C3      | 5.81E-05| GO:0005856| DNA synthesis involved in DNA repair |
|         | 4.75E-05| GO:0005130| replication fork |
|         | 4.39E-05| GO:0016458| nuclear division |
|         | 4.95E-05| GO:0007062| sister chromatid cohesion |

Hasin Afzal Ahmed et al.: Autotuned Multilevel Clustering of Gene Expression Data
leaf nodes in a few numbers of levels. The tree is driven by the input parameters. The input parameters T and δ controls the height/depth of the tree. If we set the value of T large the genes in the root node will split to the children nodes of the tree. The structure of the generated tree where the children of a node represent the desired clusters. We keep on reducing the threshold used in the clustering process as we move on to deeper levels. The structure of the generated tree is controlled by the input parameters. The input parameters T and δ controls the height/depth of the tree. If we set the value of T large the genes in the root node will split to the leaf nodes in a few numbers of levels.

\[ z\text{-score} \] The z-score is used to calculate the z-score. A higher value of z-score indicates that genes are better clustered by function, indicating a more biologically relevant clustering result. Table 10 presents generated average z-score of GERC along with some other clustering algorithm. From the results it can be observed that GERC performed satisfactorily compared to the other algorithms in terms of z-score.

### 5. Discussion

We have presented here a clustering algorithm that generates a tree where the children of a node represent the clusters that are formed from the genes in that node. The leaf nodes of the tree will represent the desired clusters. We keep on reducing the threshold used in the clustering process as we move on to deeper levels. The structure of the generated tree is driven by the input parameters. The input parameters T and δ controls the height/depth of the tree. If we set the value of T large the genes in the root node will split to the leaf nodes in a few numbers of levels.

Table 7. p-value of some of the finer clusters for Dataset 2

| Cluster | p-value | GO number | GO category |
|---------|---------|-----------|-------------|
| C1      | 9.29E-07 | GO:0006486 | protein amino acid synthesis process |
|         | 9.29E-07 | GO:0043413 | glycolysis |
|         | 9.29E-07 | GO:0070085 | biopolymer glycolysis process |
|         | 4.09E-09 | GO:0005975 | carbohydrate metabolic process |
|         | 1.35E-14 | GO:003824 | catalytic activity |
|         | 7.48E-08 | GO:0016740 | transferase activity |
|         | 1.19E-07 | GO:0009149 | autophagy |

Table 8. p-value of some of the finer clusters for Dataset 4

| Cluster | p-value | GO number | GO category |
|---------|---------|-----------|-------------|
| C1      | 1.00E-13 | GO:002154 | cell differentiation |
|         | 1.00E-13 | GO:0043934 | sporulation |
|         | 1.00E-13 | GO:0030435 | sporulation resulting in formation of a cellular spore |
|         | 1.42E-11 | GO:0032502 | developmental process |
|         | 6.63E-07 | GO:0007131 | reciprocal meiotic recombination |
|         | 4.12E-09 | GO:0035825 | reciprocal DNA recombination |
|         | 7.76E-08 | GO:0051445 | regulation of meiotic cell cycle |

| Cluster | p-value | GO number | GO category |
|---------|---------|-----------|-------------|
| C2      | 7.68E-08 | GO:0070192 | chromosome organization involved |
|         | 6.11E-08 | GO:0007131 | reciprocal eiotic recombination |
|         | 6.11E-08 | GO:0035825 | reciprocal DNA recombination |
|         | 1.54E-16 | GO:0007126 | meiosis |
|         | 1.21E-08 | GO:0020154 | cell differentiation |
|         | 1.21E-08 | GO:0030435 | sporulation resulting in formation of a cellular spore |
|         | 1.21E-08 | GO:0043934 | sporulation |
|         | 1.67E-16 | GO:0022420 | anatomic structure formation |
|         | 3.68E-08 | GO:0022420 | cell cycle process |
|         | 2.02E-07 | GO:0005634 | involved in morphogenesis |
|         | 4.35E-09 | GO:0048646 | negative regulation of biological process |
|         | 1.35E-09 | GO:0010564 | regulation of cell cycle process |

| Cluster | p-value | GO number | GO category |
|---------|---------|-----------|-------------|
| C3      | 3.43E-11 | GO:0003476 | ascospore wall assembly |
|         | 3.43E-11 | GO:0042244 | spore wall assembly |
|         | 3.43E-11 | GO:0071940 | fungal-type cell wall assembly |
|         | 4.24E-11 | GO:0070726 | cell wall assembly |
|         | 8.31E-12 | GO:0001092 | cellular component assembly involved in morphogenesis |
|         | 6.19E-09 | GO:0030154 | cell differentiation |
|         | 6.19E-09 | GO:0027045 | cell differentiation |
|         | 6.19E-09 | GO:0043934 | sporulation |
|         | 1.07E-07 | GO:0048689 | cellular developmental process |
|         | 6.41E-07 | GO:0003006 | developmental process |
|         | 3.73E-09 | GO:0032502 | developmental process |

Figure 4. Tuning the value of δ for Dataset 3
If we set the value of $\delta$ small, it will lead to fewer levels as the difference between the thresholds used in the subsequent levels will become larger. $\alpha$ (MRD threshold) should be carefully chosen. A small value of $\alpha$ may leave out some upper levels of the hierarchical tree. The effect of these parameters on the tree structure is presented in Table 9. To decide the value of $\delta$, we drew two graphs (Fig. 4), one for number of nodes and another one for depth of the sub tree generated from an initial cluster against different values of $\delta$. We used a trade off between these two graphs and decided to use 0.7 as the value of $\delta$ for yeast cell cycle dataset.

| $\delta$ | $\alpha$ | $T$ | No of nodes | Depth |
|----------|----------|-----|-------------|-------|
| 0.4      | 2        | 10  | 11          | 3     |
| 0.9      | 2        | 10  | 16          | 7     |
| 0.7      | 2        | 3   | 33          | 5     |
| 0.7      | 2        | 12  | 7           | 2     |

6. Conclusions and Future Work

In this paper, we present a top down hierarchical clustering algorithm that produces a tree of genes in the neighbourhood of a reference gene called GERC tree along with the generated clusters. The algorithm can be used to generate a wide range of clusters by considering multiple reference genes. Clustering performed in the nodes at different levels of GERC tree adaptively chooses the values of threshold parameters. The complexity of our approach can be improved by using an appropriate heuristic method for estimating an effective set of parameter values which will guarantee for quality cluster results. The value of $\alpha$ and $R$ can also be calculated statistically from the set of input genes. Work is underway for integrating prior biological information to the clustering process to improve the results.

Table 9. Effect of Input Parameters

Table 10. $z$-scores for AMC and some counterparts for Dataset 2

| Method Applied | $z$-score |
|----------------|-----------|
| K means[16]    | 5.57      |
| SOM[17]        | 5.78      |
| DCCA[3]        | -0.78     |
| GenClus[18]    | 7.39      |
| AMC            | 7.35      |

Figure 5. Tree generated from an initial cluster for yeast Dataset 2
Figure 6. Tree generated from an initial cluster for yeast Dataset 3

Figure 7. Tree generated from an initial cluster for yeast Dataset 4
ACKNOWLEDGEMENTS

This work is an outcome of the research project in collaboration with CSIR, ISI, Kolkata funded by DST, Govt. of India. The work is also supported by INSPIRE programme of DST, Govt. Of India.

REFERENCES

[1] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein, “Cluster analysis and display of genome-wide expression patterns,” Proceedings of Natl. Acad. Sci. U.S.A., vol. 95, pp. 14 863–14 868, 1998.

[2] J. Dopazo and J. Carazo, “Phylogenetic reconstruction using an unsupervised neural network that adopts the topology of an aphylogenetic tree,” J Mol Eval, vol. 44, pp. 226–233, 2002.

[3] A. Bhattacharya and R. K. De, “Divisive correlation clustering algorithm (dccca) for grouping of genes,” Bioinformatics, vol. 24, no. 11, pp. 1359–1366, June 2008.

[4] D. Jiang, J. Pei, and A. Zhang, “Dhca density-based hierarchical clustering method for time series gene expression data,” Proceedings of IEEE International Symposium on Bioinformatic and Bioengineering, pp. 393–400, 2003.

[5] F. Luo, L. Khan, F. Bastani, I. L. Yen and J. Zhou, “A dynamically growing self-organizing tree (dgsot) for hierarchical clustering gene expression profiles,” Bioinformatics, vol. 20, no. 16, pp. 2605–2617, 2004.

[6] K. Rose, “Deterministic annealing for clustering, compression, classification, regression, and related optimization problems,” Proceedings of the IEEE, vol. 86, no. 11, pp. 2210–2239, 2002.

[7] H. A. Ahmed, P. Mahanta, D. K. Bhattacharyya and J. Kalita, “GERC: Tree based clustering for gene expression data,” Proceedings of 2011 IEEE 11th International Conference on Bioinformatics and Bioengineering, pp. 299-302, 2011.

[8] H. Kim, G. H. Golub and H. Park, “Missing value estimation for DNA microarray gene expression data: local least squares imputation,” Bioinformatics, vol. 21, no. 2, pp. 187–198, 2005.

[9] R. Das, J. Kalita and D. K. Bhattacharyya, “A new approach for clustering gene expression time series data,” Int. J. Bioinformatics Res. Appl., vol. 5, pp. 310–328, 2009.

[10] S. C. Madeira and A. L. Oliveira, “Biclustering algorithm for biological data analysis: A survey,” IEEE/ACM Trans. Comput. Biol. Bioinformatics, vol. 1, pp. 24–45, 2004.

[11] Y. Cheng and G. M. Church, “Biclustering of expression data,” Proceedings of the eighth International Conference on Intelligent Systems for Molecular Biology, vol. 1, pp. 93–103, 2000.

[12] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, “A novel coherence measure for discovering scaling biclusters from gene expression data,” J. Bioinformatics and Computational Biology, vol. 7, no. 5, pp. 853-868, 2009.

[13] R. J. Cho, M. J. Campbell, E. A. Winzeler, L. Steinmetz, A. Conway, L. Wodicka, T. G. Wolfsberg, A. E. Gabrielian, D. Landsman, D. J. Lockhart, and R. W. Davis, “A genome-wide transcriptional analysis of the mitotic cell cycle,” Molecular cell, vol. 2, no. 1, pp. 65–73, 1998.

[14] J. L. DeRisi, V. R. Iyer and P. O. Brown, “Exploring the metabolic and genetic control of gene expression on a genomic scale,” Science, vol. 278, no. 5388, pp. 680–686, 1998.

[15] P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein and B. Futcher, “Comprehensive identification of cell cycle–regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization,” Molecular biology of the cell, vol. 9, no. 12, pp. 3273–3297, 1998.

[16] S. Chu, J. DeRisi, M. Eisen, J. Mulholland, D. Botstein, P. O. Brown and I. Herskowitz, “The transcriptional program of sporulation in budding yeast,” Science, vol. 282, no. 5389, pp. 699–705, 1998.

[17] I. Ulitsky, A. Maron-Katz, S. Shavit, D. Sagir, C. Linhart, R. Elkon, A. Tanay, R. Sharan, Y. Shiloh and R. Shamir, “Expander: from expression microarrays to networks and functions,” nature protocols, vol. 5, no. 2, pp. 303–322, 2010.

[18] R. Sharan and R. Shamir, “Click: A clustering algorithm with applications to gene expression analysis,” Proceedings of the eighth International Conference on Intelligent Systems for Molecular Biology, vol. 8, pp. 307–316, 2000.

[19] G. F. Berriz, O. D. King, B. Bryant, C. Sander, and F. P. Roth, “Characterizing gene sets with FuncAssociate,” Bioinformatics, vol. 19, no. 18, pp. 225–236, 2011.

[20] F. Gibbons, F. Roth, “Judging the quality of gene expression-based clustering methods using gene annotation,” Genome Research, vol. 12, pp. 1574-1581, 2002.

[21] J. A. Hartigan and M. A. Wong, “A k-means clustering algorithm,” JSTOR: Applied Statistics, vol. 28, pp. 100–108, 1979.

[22] T. Kohonen, “The self-organizing map,” Proceedings of the IEEE, vol. 78, no. 9, pp. 1464-1480, 1990.

[23] S. Sarmah, R. D. Sarmah, D. K. Bhattacharyya, “An effective density-based hierarchical clustering technique to identify coherent patterns from gene expression data,” Proceedings of the 15th Pacific Asia conference on Advances in knowledge discovery and data mining, pp. 225-236, 2011.