A KNOWLEDGE AND DATA BASED HYBRID APPROACH TO GENE CLUSTERING

Abhishek K.1*, Karnick H.1, Mitra P.2
1 Indian Institute of Technology Kanpur, Kanpur, India; 2 Indian Institute of Technology Kharagpur, Kharagpur, India
*Corresponding author: e-mail: kumabhi@iitk.ac.in

Key words: gene ontology, gene expression, clustering, minimum spanning tree, cluster score, cluster quality

SUMMARY

Motivation: Traditional gene clustering algorithms focus only on the raw expression data for clustering genes whereas valuable information about genes is available in the form of GO trees. We aim to use this information along with expression data to produce better gene clusters.

Results: We propose an algorithm that produces good quality cohesive clusters and does not require the a priori specification of the number of clusters. The proposed algorithm comprehensively outperforms the k-means and random clustering algorithms on two yeast cell data sets.

Availability: Available on request.

INTRODUCTION

Microarray technology produces expression data for thousands of genes under different conditions. Interpretation of this huge volume of observed data requires the clustering of genes that have correlated expression profiles. Clustering can help identify genes that may have common functions or those that may be part of common regulatory networks. Traditional clustering algorithms have focused on the raw gene expression data for performing this task (Brazma, Vilo, 2000). However, valuable biological knowledge in the form of the Gene Ontology (GO) can provide useful inputs for producing meaningful clusters. The GO represents terms in a directed acyclic graph (DAG), covering three taxonomies namely molecular function, biological process, and cellular component. For example, the gene product cytochrome can be described as follows: molecular function terms: oxidoreductase activity; biological process terms: oxidative phosphorylation, induction of cell death; component terms: mitochondrial matrix, mitochondrial inner membrane. The DAG consists of terms represented as nodes connected by relationship edges. The ontology annotates gene products with different terms across the graph. In this work we adopt a hybrid approach towards gene clustering. We use knowledge available in the form of GO together with gene expression data to perform clustering.
ALGORITHM

Distance measure between GO nodes. We use the GO process ontology, to find the distance between GO nodes. Each GO node is annotated with a list of genes; some nodes can be un-annotated. Since GO is a DAG a GO node can have multiple parents and multiple children. As a particular node can have multiple parents the GO DAG is transformed to get a directed tree structure. The level of a node is defined as the number of nodes between the node and the root node. The procedure for converting the DAG to a directed tree is as follows:

\[
\text{Do}\{ \\
\text{Visit every node}\ x\ \text{in the DAG} \\
\text{If node}\ x\ \text{has} \ k > 1 \ \text{parent then:} \\
\quad \bullet \ \text{Create} \ k \ \text{nodes and make each newly created node a single child of a distinct parent of} \ x. \\
\quad \bullet \ \text{Replicate annotations of} \ x \ \text{across the} \ k \ \text{newly created nodes.} \\
\quad \bullet \ \text{Make each child of} \ x \ \text{a child of each newly created node.} \\
\quad \bullet \ \text{Remove node} \ x. \\
\} \ \text{Until} \ (\text{No change in DAG})
\]

We define a weight function \( f: \{1, 2, \ldots, N\} \rightarrow \mathbb{R} \) where \( N \) is the maximum level of any node in the obtained directed tree and \( \mathbb{R} \) is the real line such that \( f \) is a decreasing function i.e. \( f(i) > f(j) \) for all \( i < j \).

The distance between nodes \( i \) and \( j \) in the directed tree is defined to be the weight of the level of the Least Common Ancestor (LCA) of the nodes \( i \) and \( j \). The arguments in support of choosing this as a distance measure and the proof that it is in fact a distance metric can be found in Lee et al. 2004.

Calculation of gene distance matrix from the directed GO tree. We calculate the distance between all gene pairs which are annotated in the directed GO tree. We first prepare an \( n \times m \) table \( T \) where \( n \) is the total number of genes and \( m \) is the number of nodes in the GO tree. The rows represent the gene and the columns represent the attributes or GO nodes. We treat each GO node as an attribute. As a single gene may be annotated to multiple nodes we check for the list of annotations of a single gene and put a 1 in every attribute column with which the particular gene is annotated. So we get a binary table, which is used for calculating gene distance. We define \( \text{diff}(i, j) \) to be the number of differing entries in corresponding columns of rows \( i \) and \( j \). The gene distance between genes \( g_i \) and \( g_j \) \( \text{GOdist}(g_i, g_j) \) is defined to be:

\[
\text{GOdist}(g_i, g_j) = \frac{1}{\text{diff}(i, j)} \sum_{\alpha \in \alpha} \sum_{\beta \in \beta} (T_{\alpha \beta} - T_{\beta \alpha})^2 \cdot d(\alpha, \beta) ; \quad d(\alpha, \beta) \text{ is the GO distance between GO nodes } \alpha \text{ and } \beta .
\]

We get a gene distance matrix \( G \) of the order \( n \times n \) where \( n \) is the number of genes.

Definitions:

- Average GO distance \( \text{GOavg}(C) \) corresponding to a gene cluster \( C = g_i, 1 \leq i \leq k \),

\[
\text{GOavg}(C) = \sum_{1 \leq i, j \leq k} \frac{\text{GOdist}(g_i, g_j)}{\binom{k}{2}} ; \quad \text{GOdist}(g_i, g_j) \text{ is the GO distance measure between } g_i \text{ and } g_j \text{ based on the gene distance matrix } G.
\]

- Scatter of a cluster \( \text{Scatter}(C) \) \( C = g_i, 1 \leq i \leq k \), where \( g_i \)'s are the members of the cluster \( C \),

\[
\text{Scatter}(C) = \sum_{1 \leq i \leq k} (x_i - \mu)(x_i - \mu)^T ; \quad \text{where } x_i \text{ is the expression vector of gene } g_i \text{ and } \mu \text{ is the average expression vector for the cluster } C. \quad (x_i - \mu)^T \text{ is the transpose of the vector } x_i - \mu.
\]

- The objective function for partitioning the MST for a given number of clusters \( k \) \( F(k) \):

\[
F(k) = \frac{1}{k} \sum_{1 \leq c \leq k} \left( \text{GOavg}(C_i) + \text{Scatter}(C_i) \right)
\]
Scorei is the minimum value of the objective function F (i) obtained for a given number of clusters i and Clusteri is the optimal cluster set corresponding to it.

**Iterative Clustering Algorithm.** The sequence of steps followed by the algorithm is as follows:

2. Scale both gene distance matrix G and expression data matrix E to the same range and combine them to get net distance D.
3. Make a fully connected graph with genes as nodes with edge weight between node i and node j equal to the distance between gene i and gene j obtained from the net distance matrix D. Find the minimum spanning tree of this fully connected graph. To cluster, partition this Minimum Spanning Tree (MST) into k sub trees where k is the number of desired clusters (Xu et al., 2002).
4. The iterative algorithm is as follows:
   - INPUT: MST obtained from step 3.
   - a. Initialize k to 1.
   - b. while k < MAXCLUSTERS{
     - c. Perform a random k-partitioning of the Minimum Spanning Tree (MST) by removing k-1 edges. Then perform the following operation until the process converges. For each pair of adjacent clusters, go through all the edges in the merged cluster of the two to find the edge to cut, this globally optimizes the 2-partitioning of the merged cluster measured by objective function F(k).
     - d. Scorek = F(k), save the optimal cluster set obtained in above step as Clusterk
     - e. Increment k.
   - f. Search the list of scores to find the minimum element Scoremin and output the cluster set Clustermin corresponding to it.

**RESULTS AND CONCLUSION**

We used the data set of the Yeast cell cycle in which activity was measured at 18 time points. We used two subsets each consisting of 500 genes for testing the algorithm. Cluster validation was done using figure of merit score (FOM). We compared the proposed algorithm with k-means, random, and our algorithm without using GO distances. Fig. of Merit score (FOM) is defined as in (Yeung et al., 2001) suppose e is the left out condition of the m experimental conditions present in the data set, let there be k clusters C1,C2,...Ck, and let Eg,e be the expression level of g under condition e. Let μCi(e) be the average expression level in condition e for genes of cluster Ci

\[
\text{FOM}(e,k) = \frac{1}{n} \sum_{1 \leq c \leq k} \sum_{s \in C_c} (E_{g,e} - \mu_{C_c}(e))^2 \quad \text{and} \quad \text{FOM}(k) = \sum_{1 \leq e \leq m} \text{FOM}(e,k).
\]

We use FOM(k) to assess the quality of clusters obtained from different algorithms for a given number of clusters k. It is basically a leave one out approach, where clustering is performed using all but one of the experimental conditions in the data set. The left out condition is used to assess the predictive power of the clustering algorithm. The FOM score represents scatter from the actual value at test condition; thus lower the FOM score higher is the predictive power of the algorithm and better is the quality of clusters obtained.

The performance graph for the four algorithms on the two data sets is shown in Fig. 1a, b. The following can be observed from the plots of Fig. 1a: (a) The proposed algorithm outperforms k-means and the random algorithm for greater than 10 clusters. (b) The proposed algorithm without GO distances outperforms k-means after 20 clusters. (c)
The proposed algorithm without GO distances outperforms random algorithm right from the beginning. (d) The proposed algorithm outperforms the one without GO distances right from the beginning. Performance on the second data set (Fig. 1b) reveals the following: (a) The proposed algorithm with and without GO distances outperforms k-means and the random algorithm right from the beginning. (b) The proposed algorithm outperforms the one without GO distances right from the beginning. We conclude from these observations that the proposed algorithm outperforms k-means and the random algorithm and the usage of GO distances improves cluster quality.

Figure 1. Performance analysis on dataset1 (a); performance analysis on dataset2 (b).

REFERENCES

www.geneontology.org/GO.docs.html