Biclustering with δ-pCluster

John Tantalo

1. Introduction

The subject of biclustering is chiefly concerned with locating submatrices of gene expression data that exhibit shared trends between genes. That is, from one attribute to another, each gene's expression in the bicluster shifts approximately the same amount. The goal is to determine sets of related genes that are affected similarly by different experimental procedures. In the terminology of biclustering, genes are called *objects* and the independent variables of the microarray analysis are called *attributes.*

This problem has classically been approached by casting objects and attributes as vertices of a bipartite graph and assigning weights to edges representing the corresponding expression level. Based on findings with this perspective, researchers have generally agreed that the biclustering problem is NP-hard.

In this report, I will briefly describe a recent approach to biclustering, the δ-pCluster method[1]. The discussion includes my effort to implement the algorithm and a modification I designed to solve some problems I experienced with this method.

2. Problem Statement

Let d_{ij} be the matrix element of object i under attribute j. Let O and T be a subset of objects and attributes respectively. *(O, T)* forms a δ-pattern cluster if the absolute difference of the differences of the attribute values of two objects is less than a threshold δ for every pair of objects and attributes. More formally:

$$\forall x, y \in O, \forall a, b \in T, \left|(d_{xa} - d_{xb}) - (d_{ya} - d_{yb})\right| \leq \delta \rightarrow (O, T) \text{ is a } \delta\text{-pCluster}$$

The problem is to find all δ-pClusters *(O, T)* with no less than a minimum number of objects and attributes.

3. Related Work

Cheng and Church[2] introduced the bicluster model based on mean squared reside scores with a threshold δ. They also offer a reduction from the balanced complete bipartite subgraph problem to finding the largest square δ-bicluster. This demonstrates a special case of biclustering is NP-hard, but leaves the general question of the complexity of finding the largest rectangular δ-bicluster unsolved. Cheng and Church also offer a polynomial-time randomized greedy algorithm to calculate biclusters.

Yang et al.[3] expanded on the work of Cheng and Church by introducing the concept of α-occupancy, which allows missing values in a bicluster up to a threshold. The residue score of a missing value is then defined to be zero. They present FLOC, a polynomial-time move-based randomized algorithm, to compute biclusters much more efficiently than the Cheng-Church algorithm. Yang et al. also suggest that biclustering is a generalized case of traditional subspace clustering (i.e., the CLIQUE algorithm). The FLOC algorithm is explored in detail in Yang et al.[4] and it is shown to locate better results faster than the Cheng-Church algorithm.

Each of these methods use mean squared residues to evaluate the score of a bicluster. Wang et al. point out that, when this formula is used, it is common to find submatrices of a δ-bicluster that are not a δ-bicluster. While seemingly contradicting the basic premise of a bicluster, this property also inhibits the design of efficient algorithms for biclustering.

Many other biclustering algorithms have been developed since Cheng and Church introduced the concept. Madeira and Oliveira[5] catalogued 19 different algorithms with approaches such as iterative row and column clustering combination, divide and conquer, greedy iterative search, exhaustive enumeration, and distribution parameter identification. Several biclustering applications are also described.

4. Methods

Wang et al. present δ-pClusters to solve the problems related to mean squared resides. The first advantage of the pCluster model is that any submatrix *(O', T')* of a δ-pCluster *(O, T)* is also a δ-pCluster. This property is fundamental to the pCluster algorithm, which locates δ-pClusters by first identifying two-object and two-attribute pClusters and incrementally builds larger sets. The pCluster algorithm is also deterministic and will not miss any qualified biclusters, unlike Cheng-Church and FLOC, which only provide approximations of the full bicluster set.

The key concept behind pCluster is the maximum dimension set (MDS), which are maximum length contiguous subsequences of the sorted values of the difference of two objects across all attributes bounded by the threshold δ. In the following example, the rows of D_{xy} are objects x and y and the columns are attributes a through e. The vector S_{xy} is the difference of the objects for each attribute, and it is sorted in $\vec{S}$. The set of maximal dimension sets with threshold $\delta = 2$ contains every maximal subvector of $\vec{S}$ that spans δ.

$$D_{xy} = \begin{bmatrix} a & b & c & d & e \\ 1 & 2 & 3 & 5 & 7 \\ 0 & 5 & 5 & 6 & 2 \end{bmatrix}$$

$$S_{xy} = \begin{bmatrix} a & b & c & d & e \\ -1 & 3 & 2 & 1 & -5 \end{bmatrix}$$

$$\vec{S}_{xy} = \begin{bmatrix} e & a & d & c & b \\ -5 & -1 & 1 & 2 & 3 \end{bmatrix}$$

$$MDS_{\delta}(D_{xy}) = \{(a,d),(d,c,b)\}$$

Figure 1: Maximum dimension sets.

The paper presents algorithms for building maximum dimension sets for each object pair and attribute pair, pruning these sets, and building a prefix tree of the object-pair maximum dimension sets. These algorithms are conceptually simple and easy to implement. The result is a rooted tree in which each node *O* under a path *T* corresponds to a candidate bicluster of objects *O* and attributes *T*.

The final step is to prune nodes of the prefix tree and discover valid biclusters in a postorder traversal. This technique is given in figure 2.

```
for each node n encountered in the post-order traversal do
    $\mathcal{O}$ := objects in node $n$;
    $\mathcal{T}$ := columns represented by node $n$;
    for each $a, b \in \mathcal{T}$ do
        find column-pair MDSs: $\mathcal{C}$ = $pairCluster(a, b, \mathcal{O}, nr)$;
        remove from $\mathcal{O}$ those objects not contained in any MDS $c \in \mathcal{C}$;
    Output $(\mathcal{O}, \mathcal{T})$ ;
    Add objects in $n$ to nodes which has one less column than $n$;
```

Figure 2: Prefix tree pruning and bicluster discovery (Wang et al.).

5. Advantages and Disadvantages

The pCluster method is demonstrably improved over CLIQUE, and Yoon et al.[6] describe an enhanced version of pCluster that performs better than the Cheng-Church method, but I could not find any documented comparisons between pCluster (enhanced or otherwise) and FLOC, the enhanced version of Cheng-Church. Since these two methods are closely related in the world of biclustering, I think a comparison is due.

The form of pCluster given by the paper is said to perform poorly on very large data sets, e.g., greater than 1000 genes. Wang et al.[7] address this problem by introducing a new measurement technique and counting tree algorithm, which performs better than pCluster on large data sets. It may be feasible to sacrifice a degree of approximation for faster results and redesign pCluster as a randomized algorithm, more akin to the Cheng-Church or FLOC methods.

6. Improvements

I originally implemented the δ-pCluster exactly as given, but I quickly ran into a reoccuring problem with the prefix tree pruning and bicluster discovery method. Consider the case of a node with objects (*w, x, y, z*) under the path of column (*a, b, c*). If the maximum dimensions sets of the column pairs are each {(*w, x, y)*, (*x, y, z)*}, then the objects *w* and *z* are not removed from the node and the bicluster (*w, x, y, z*), (*a, b, c*) is discovered. This does not seem correct, however, since under no column pair was the set of objects (*w, x, y, z)* fully supported. A more intuitive bicluster is (*x, y*), (*a, b, c*), since objects *x* and *y* are fully supported by each maximum dimension set.

I designed an implemented an alternative to the prefix tree pruning and discovery algorithm. This is presented in figure 3.

```
for each node $n$ encountered in the post-order traversal:
    $O$ = objects in $n$
    $T$ = columns represented by $n$
    $C = \cup_{a,b \in T} pairClusters(a,b)$
    For each $a,b \in T$ :
        For each $c \in C$ :
            $c = \max_{MDS \in pairClusters(a,b)} \{c \cap MDS\}$
    For each $c \in C$ :
        $If\ |c| \geq nr$ :
            Output $(O,T)$
    Add $O$ to nodes which have one less column than $n$
```

Figure 3: Alternative pruning and discovery method.

This method relies on the intuition that every bicluster embedded at a node must be a subset of some maximum dimension set of each column-pair of the node's path. First, the candidates *C* are collected from the maximum dimension sets of each column pairs. Then, each candidate *c* is pruned by replacing it with the maximal intersection of *c* and the maximum dimension sets of each column pair. This ensures that each candidate is fully supported by each column pair. The remaining candidates of sufficient size are the biclusters under the node's columns; no verification step is necessary.

7. Evaluation & Conclusion

I implemented the original algorithm and modified algorithm in Python using simulated data with single embedded biclusters. Test showed repeatedly that the original algorithm incorrectly computed biclusters in the manner described in section 6. This was determined by evaluated each discovered bicluster against the threshold parameter δ. The modified algorithm did not report any false biclusters. Moreover, it always successfully located the embedded bicluster. Approximately 2000 experiments with 20x20 data containing 5x5 biclusters were performed.

In conclusion, the prefix tree pruning and bicluster discovery method described in this paper is adequate to discover biclusters in candidate sets.

8. References

[1] Wang H, Wang W, Yang J, Yu P. "Clustering by Pattern Similarity in Large Data Sets." *SIGMOD*. 2002.
[2] Cheng Y, Church G. "Biclustering of epression data." *ISMB*, 2000.

[3] Yang J, Wang W, Wang H, Yu P. "δ-Clusters: Capturing Subspace Correlation in a Large Data Set." *ICDE*. 2002.

[4] Yang J, Wang H, Wang W, Yu P. “Enhanced biclustering on expression data.” *BIBE*. 2003.

[5] Madeira SC, Oliveira AL. “Biclustering Algorithms for Biological Data Analysis: A Survey.” *TCBB*. 2004.

[6] Yoon S, Nardini C, Benini L, De Micheli G. “Enhanced pClustering and Its Applications to Gene Expression Data.” *BIBE*. 2004.

[7] Wang H, Chu F, Fan W, Yu P, Pei J. “A Fast Algorithm for Subspace Clustering by Pattern Similarity.” *SSDBM*. 2004.