Time and Space Efficient Implementation of Robust Biclustering Algorithm (ROBA)

Muhammad Ibrahim\(^1\) Nasimul Noman\(^1\)

ibrahimmuhammad313@yahoo.com noman@univdhaka.edu

Hitoshi Iba\(^2\)
iba@iba.k.u-tokyo.ac.jp

\(^1\)Department of Computer Science and Engineering, University of Dhaka, Dhaka-1000, Bangladesh
\(^2\)Department of Electronics Engineering, University of Tokyo, Tokyo, Japan

Keywords: gene expression data, biclustering.

1. Introduction

Biclustering techniques perform clustering simultaneously along rows and columns of a matrix in order to find which rows are clustered under which columns and/or vice versa. A bicluster of a gene expression matrix is a subset of rows (genes) that exhibits similar behavior under a subset of columns (conditions).

2. Methods and Results

A biclustering algorithm named Robust Biclustering Algorithm (ROBA) \cite{2} has been used in a number of recent research works. The existing implementation of ROBA is not that time and space efficient. In this work, we develop a time and space efficient implementation of ROBA. We elicit some subtle properties of the base principle of ROBA to achieve these efficiencies. We reduce both time and space requirements by a factor of \(L\) where \(L\) is the number of distinct values present in the \(N \times M\) input data matrix. The comparison of time and space requirement between the existing implementation and our implementation is summarized in table 1.

Table 1: Comparison of Complexities between ROBA and proposed algorithm

<table>
<thead>
<tr>
<th>Normal</th>
<th>In terms of (N_b) (# of biclusters found)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ROBA</td>
</tr>
<tr>
<td>Time</td>
<td>(O(L^2N^2M))</td>
</tr>
<tr>
<td>Space</td>
<td>(O(LNM))</td>
</tr>
</tbody>
</table>

2.1 Description of the Data Set

We implemented our algorithm in Matlab on a computer having 1.8 GHz Dual Core processor, 1 GB RAM and running Windows Vista operating system. We used the much-used Yeast Saccharomyces Cerevisiae gene expression dataset \cite{4} to compare the performance between the existing implementation of \cite{2} and our algorithm. The data is organized in a 2884 x 17 matrix where rows and columns correspond to genes and conditions respectively. The entries are integer numbers in a range from -1 to 595 with 207 distinct values. -1 entry indicates missing value.

2.2 Result Analysis

We performed two types of comparisons between the implementation of \cite{2} and our algorithm. Firstly, like \cite{2}, we quantized the data matrix according to the technique suggested in \cite{2} which reduces the numbers of distinct values from 207 to 113. Then we run the implementation of \cite{2}
Table 2: Performance Comparison between the Implementation of [3] and our algorithm

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Number of distinct values in data matrix</th>
<th>Minimum # of rows</th>
<th>Minimum # of columns</th>
<th>Total number of biclusters found</th>
<th>Time taken (in seconds)</th>
<th>Gain of our algorithm over [2]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>113</td>
<td>3</td>
<td>2</td>
<td>10807</td>
<td>17144</td>
<td>10.71</td>
</tr>
<tr>
<td>2nd</td>
<td>207</td>
<td>3</td>
<td>2</td>
<td>10498</td>
<td>30307</td>
<td>12.83</td>
</tr>
</tbody>
</table>

and our algorithm to compare the performance. Secondly, we performed no quantization; rather, we run both the algorithms directly on the data matrix that contains 207 distinct values.

The result of both of the algorithms is summarized in table 2. We find that our algorithm is almost 11 and 13 times faster than the implementation of [2] for first and second test respectively.

As our target is to show the superiority of our implementation over the implementation of [2] in terms of computational time, we did not perform any quality measurement of the biclusters found – in fact, [2] showed that ROBA outperforms some classic works mentioned in [2] in terms of quality and number of biclusters found. We did not perform duplicate bicluster checking – so there can be duplicate biclusters in the result. However, we did compare both of the outputs i.e. set of biclusters found by the two implementations and noticed no difference; this checking was performed just to be experimentally sure that our algorithm yields exactly same biclusters as the implementation of [2]. Of course, the order of biclusters produced by the two implementations may be different. The presence of duplicate biclusters in the output poses no significant variation in time if their number is nominal. However, if they are large in number, then checking for duplicate biclusters and not reporting them more than once may reduce considerable time in the implementation of [2]. But this phenomenon should not significantly distort the gain we mentioned in the last column of table 2 because, in that case, our algorithm can also be modified to check the duplicate biclusters so as not to report them more than once and its time should also be reduced by almost same amount of time. As for the missing values, we simply treated missing values as other normal values because, again, in this paper, our target is to show the superiority of our implementation in terms of computational time and space. As [2] reported that their implementation of ROBA outperformed some classic works such as Cheng and Church [1] and Wang et al [3] in terms of time requirement, therefore our implementation outperforms them by even more degree.

3. Discussion

ROBA has established itself as an important biclustering algorithm, finding many applications in recent work. Therefore, time and space efficient implementation of ROBA will surely be useful to enhance the significance of the algorithm.

References


