I, Hema Subramanian, hereby submit this original work as part of the requirements for the degree of Master of Science in Computer Science.

It is entitled:
**Summarization Of Real Valued Biclusters**

Student's name: **Hema Subramanian**

This work and its defense approved by:

Committee chair: Raj Bhatnagar, PhD
Committee member: Yizong Cheng, PhD
Committee member: John Schlipf, PhD

University of Cincinnati
Summarization Of Real Valued Biclusters

A Thesis submitted to the
Graduate School
of the University of Cincinnati
in partial fulfillment of the
requirements for the degree of

Master of Science

in the Department of Computer Science
of the College of Engineering and Applied Science

by

Hema Subramanian

B.E, Anna University, India, 2006
M.S., University of Cincinnati, 2011

18 April 2011

Thesis Advisor and Committee Chair: Dr. Raj Bhatnagar.
Abstract

With an explosion in database sizes, there is an increasing need for mining relevant information from them. Subspace clustering has been applied in various fields for discovering patterns, and many such algorithms have been investigated for finding interesting biclusters from binary-valued datasets. Mining biclusters from real-valued datasets has gained significant importance in many of the recently emerging applications. The algorithms devised for mining such biclusters generally minimize an objective function, and hence the biclusters generated by each algorithm vary depending on the objective function used.

Due to the inherent size and density of the data sets, the algorithms generate a very large number of biclusters, making it difficult to select the useful ones from among them. To overcome this problem, it is important to design strategies to summarize these biclusters into few representatives of the main ideas embedded in the dataset. The objective of this thesis is to apply some statistical properties of the generated biclusters to identify some distinguished clusters that seek to summarize the large number of biclusters into few representative ones.

In order to achieve the above stated objective, similarity measures based on mutual information and standard deviation difference between biclusters are used to identify similar biclusters. These measures quantify the information shared (or the similarity) between two biclusters, and this helps in identifying potential biclusters that could be merged. The algorithm has been applied to a synthetic and two real world datasets and the results are presented. The information content and the variance in a bicluster are analyzed as the biclusters are progressively merged. The methodologies proposed in this thesis are compared to a baseline method to verify the quality of the biclusters and validate that our approach performs significantly well and has good merit.
I would like to thank my advisor, Dr. Raj Bhatnagar for helping me define my thesis problem and for his guidance and support throughout the course of this study. I would also like to thank Zhen Hu and Faris Alqadah, for their help in giving me valuable inputs on coming up with solutions to overcome roadblocks, and providing data sets to run my algorithms.

Additionally, I am very grateful to Dr. Yizong Cheng and Dr. John Schlipf for their valuable suggestions and feedback. And finally, this work is dedicated to my parents Mrs. Mala and Mr. N Subramanian, and I can only begin being grateful for their unconditional love and support.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>i</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>iii</td>
</tr>
<tr>
<td>Contents</td>
<td>iv</td>
</tr>
<tr>
<td>List of Figures</td>
<td>v</td>
</tr>
<tr>
<td><strong>1 Introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.1 Overview</td>
<td>1</td>
</tr>
<tr>
<td>1.1.1 Clustering</td>
<td>1</td>
</tr>
<tr>
<td>1.1.2 Full dimensional clustering</td>
<td>2</td>
</tr>
<tr>
<td>1.1.3 Subspace clustering</td>
<td>2</td>
</tr>
<tr>
<td>1.1.4 Subspace Clustering with constant valued and binary data</td>
<td>3</td>
</tr>
<tr>
<td>1.1.5 Formal Concept Analysis and Bi-Clustering</td>
<td>3</td>
</tr>
<tr>
<td>1.2 Problem Statement</td>
<td>5</td>
</tr>
<tr>
<td>1.3 Thesis Organization</td>
<td>6</td>
</tr>
<tr>
<td><strong>2 Related Research</strong></td>
<td>7</td>
</tr>
<tr>
<td><strong>3 Proposed Strategies</strong></td>
<td>10</td>
</tr>
<tr>
<td>3.1 Lattice Based Bi-Cluster Searching Algorithm</td>
<td>10</td>
</tr>
<tr>
<td>3.2 Relative Deviation Methodology</td>
<td>13</td>
</tr>
<tr>
<td>3.2.1 Overview</td>
<td>13</td>
</tr>
<tr>
<td>3.2.2 Design Approach - 1</td>
<td>15</td>
</tr>
<tr>
<td>3.2.2.1 Implementation using Graph Generation</td>
<td>15</td>
</tr>
<tr>
<td>3.2.2.2 Creating the Graph</td>
<td>15</td>
</tr>
</tbody>
</table>
3.2.2.3 Bicluster Reduction ................................................. 17
3.2.2.4 Limitation .......................................................... 19
3.2.3 Design Approach - 2 .................................................. 21
3.2.3.1 Implementation using Matrix .................................. 21
3.2.3.2 Complexity Analysis .............................................. 22
3.3 Mutual Information Methodology ...................................... 24
3.3.1 Overview ............................................................... 24
3.3.2 Computing the Entropy ............................................... 25
3.3.3 Design Approaches ................................................... 25
3.3.3.1 Approach-1 .......................................................... 25
3.3.3.2 Approach-2 .......................................................... 26
3.3.3.3 Complexity Analysis .............................................. 29
4 Analysis and Results .......................................................... 30
4.1 Application to Synthetic Data ........................................... 30
4.1.1 Sample Merging Results ............................................. 30
4.1.2 Comparison with Baseline Approach ............................... 32
4.1.3 Analysis ................................................................. 36
4.2 Experiments in Real Data ............................................... 42
4.2.1 Parkinsons Data Set .................................................. 42
4.2.1.1 Merge Results ...................................................... 42
4.2.1.2 Comparison with Baseline Approach ........................... 50
4.2.1.3 Analysis .............................................................. 54
4.2.2 Chip Sequence Data Set .............................................. 56
4.2.2.1 Merge Results ...................................................... 56
4.2.2.2 Baseline Comparison and Analysis .............................. 56
5 Conclusion ................................................................. 67
5.1 Summary ................................................................. 67
5.2 Future Work ............................................................... 69
6 References ................................................................. 70
# List of Figures

1.1 Objects-Attributes .................................................. 4  
1.2 Concept-Lattice ..................................................... 4  
3.1 Genes and Environment Biclusters ................................ 11  
3.2 Phase1 .............................................................. 12  
3.3 Phase2 .............................................................. 12  
3.4 Bicluster generated on merge ...................................... 15  
3.5 Graph before merging .............................................. 16  
3.6 Merge-1 ............................................................ 19  
3.7 Merge-2 ............................................................ 19  
3.8 Deviation matrix for 4 biclusters ................................. 21  
3.9 Deviation Matrix After Merge JPG ................................. 22  
3.10 Distribution For Bicluster B1 ...................................... 26  
4.1 Relative Deviation Approach: Merge-1 ............................. 31  
4.2 Relative Deviation Approach: L: Merge-2, R: Merge-3 ........ 32  
4.3 Mutual Information Methodology: Approach-2- Merge-1 ....... 33  
4.4 Relative Deviation vs. Approach-2 Approach: Merge-1 ........ 35  
4.5 (Baseline vs. Relative)Baseline Approach ....................... 37  
4.6 (Baseline vs. Relative)Relative Deviation Approach ........... 38  
4.7 (Baseline vs. Approach-2) Baseline Approach ................... 39  
4.8 Variations In Average Standard Deviation as Threshold increases 40  
4.9 Variations In Average Entropy as Threshold increases .......... 41  
4.10 Approach-1 vs Approach-2 ...................................... 41  
4.11 Merge Process ..................................................... 42
<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.12</td>
<td>Relative Deviation Approach: Merge-1</td>
<td>43</td>
</tr>
<tr>
<td>4.13</td>
<td>Relative Deviation Approach: Merge-2</td>
<td>44</td>
</tr>
<tr>
<td>4.14</td>
<td>Relative Deviation Approach: Merge-3</td>
<td>44</td>
</tr>
<tr>
<td>4.15</td>
<td>Approach-1-Merge Process</td>
<td>45</td>
</tr>
<tr>
<td>4.16</td>
<td>Approach-2-Merge Process</td>
<td>45</td>
</tr>
<tr>
<td>4.17</td>
<td>(Relative Deviation vs. Approach-2) Approach-2</td>
<td>46</td>
</tr>
<tr>
<td>4.18</td>
<td>(Relative Deviation vs. Approach-2) Relative deviation approach</td>
<td>47</td>
</tr>
<tr>
<td>4.19</td>
<td>(Approach-1 vs. Approach-2) Approach-2</td>
<td>48</td>
</tr>
<tr>
<td>4.20</td>
<td>((Approach-1 vs. Approach-2) Approach-2)</td>
<td>49</td>
</tr>
<tr>
<td>4.21</td>
<td>(Baseline vs. Relative)Baseline Approach</td>
<td>50</td>
</tr>
<tr>
<td>4.22</td>
<td>(Baseline vs. Relative)Relative Deviation Approach</td>
<td>51</td>
</tr>
<tr>
<td>4.23</td>
<td>(Baseline vs. Approach-2)Baseline Approach2</td>
<td>52</td>
</tr>
<tr>
<td>4.24</td>
<td>(Baseline vs. Approach-2) Approach-2</td>
<td>53</td>
</tr>
<tr>
<td>4.25</td>
<td>Variations In Average Standard Deviation as Threshold increases</td>
<td>54</td>
</tr>
<tr>
<td>4.26</td>
<td>Variations In Average Entropy as Threshold increases</td>
<td>55</td>
</tr>
<tr>
<td>4.27</td>
<td>Approach-1 vs. Approach-2</td>
<td>56</td>
</tr>
<tr>
<td>4.28</td>
<td>Comparison of Merge Process between Baseline and Deviation Process</td>
<td>57</td>
</tr>
<tr>
<td>4.29</td>
<td>Relative Deviation Approach: Merge-1</td>
<td>58</td>
</tr>
<tr>
<td>4.30</td>
<td>Mutual Information Methodology: Approach-1-Merge-1</td>
<td>59</td>
</tr>
<tr>
<td>4.31</td>
<td>Mutual Information Methodology: Approach-2-Merge-1</td>
<td>60</td>
</tr>
<tr>
<td>4.32</td>
<td>(Baseline vs. Relative)Relative Deviation Approach</td>
<td>61</td>
</tr>
<tr>
<td>4.33</td>
<td>(Baseline vs. Relative)Baseline Approach</td>
<td>62</td>
</tr>
<tr>
<td>4.34</td>
<td>(Baseline vs. Approach-2)Baseline Approach</td>
<td>63</td>
</tr>
<tr>
<td>4.35</td>
<td>(Baseline vs. Relative)Approach-2 Approach</td>
<td>64</td>
</tr>
<tr>
<td>4.36</td>
<td>Variation of deviation in approaches on merging</td>
<td>65</td>
</tr>
<tr>
<td>4.37</td>
<td>Comparison of Entropy among Mutual Information Based Approaches</td>
<td>65</td>
</tr>
<tr>
<td>4.38</td>
<td>Variation of entropy in approaches on merging</td>
<td>66</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

In this information age, with the rapidly growing population and technology, available data has begun increasing enormously. Particularly, research in genetics and financial markets needs to deal with tons of data in order to carry out analysis. There have been a number of algorithms written in the field of "Data Mining" that have facilitated the process of mining the required information based on the type of data and data sparseness. An important technique that is predominantly used in data mining is Clustering. Clustering helps us find related patterns and there are different algorithms and applications for clustering. Due to its continually emerging applications, differences in the goals, and varying types of data, there is a growing need for different types of clustering algorithms.

1.1 Overview

1.1.1 Clustering

Clustering is one of the most commonly used unsupervised data mining methods that explores the hidden structures in the data set [Tan et al, 2006]. Traditionally, clustering is the process of partitioning the data points into mutually exclusive groups (or) clusters, such that data points in a cluster are similar to each other than to data points in other clusters. The dissimilarity between data points is usually measured using a distance metric defined on the differences between the values of their attributes. Clustering can be broadly classified into two groups i.e. Full dimensional clustering and Subspace clustering.
1.1.2 Full dimensional clustering

Full dimensional Clustering algorithms can be classified mainly into two types: partition based methods and hierarchical methods. In both these methods, the distances between the objects are computed considering all the attributes in the data set. Partition based methods define some objective functions $\zeta$ to measure the quality of the current clustering results. They try to optimize (either maximize or minimize) the objective function. K-means clustering algorithm, a representative partition based method, partitions all objects into $k$ groups, where $k$ is a pre-determined value of the total number of clusters to be found. Initially, $k$ objects are selected as the cluster centers, and all other objects are assigned to their corresponding nearest cluster centers. Then the $k$ cluster centers are recalculated based on the objects assigned into this cluster in the first iteration. This process is continued until, the centers do not change or the termination condition is met.

Hierarchical clustering methods do not form all the clusters in a single step, unlike partition based methods. These methods are also broadly classified into two types: the divisive methods and the agglomerative methods. The divisive methods start from considering all objects to be within a single cluster. More clusters are found by iteratively choosing a partition that divides the original cluster into two clusters. On the contrary, the agglomerative clustering methods start with each object being an individual cluster. The number of clusters is reduced by merging two very similar existing clusters [Kaufman et al, 1990].

1.1.3 Subspace clustering

Subspace clustering simultaneously clusters along both dimensions of a data table. In other words the output of a bi-clustering algorithm is a set of objects along with the subspace of attributes in which this cluster appears in. In high-dimensional data traditional clustering techniques that only cluster along one dimension of the table have proven inadequate due to the curse of dimensionality and noisy features.

A subspace clustering problem is a search algorithm for interesting subsets of objects and their associated subsets of attributes [Dhillon et al, 2003]. Since the first subspace clustering algorithm for data mining was proposed [Agrawal et al, 1998], many different algorithms have been presented. These algorithms can also be classified into two groups, Partition based [Aggarwal et al, 1999] and grid based approach [Cheng et al, 1999].

In partition based approach the objects are partitioned into two mutually exclusive groups. Each group along with the subset of dimensions, where these subsets of objects show greatest similarity is classified as a subspace cluster. Here as well, the algorithms try to minimize the objective function. In grid based
approach, the data matrix is treated as a high dimensional grid and the clustering process is a search for dense regions in the grid. Each dimension is partitioned into \( \epsilon \) number of intervals of equal-length, and a \( n \)-dimensional unit is the intersection of intervals from \( n \) distinct dimensions. A data point is contained in a unit if the values of all its attributes fall in the intervals of the unit for all dimensions of the unit. A unit is dense if the fraction of the total data points contained in it exceeds an input parameter \( \tau \).

1.1.4 Subspace Clustering with constant valued and binary data

As already expressed in the previous section, there are quite a few algorithms proposed to mine biclusters and each of the biclustering algorithms has a way of defining its biclusters. The data matrix could be binary, that is if an object has that attribute in it, it will be represented by a 1 else it will be represented as a 0. In binary data a bi-cluster corresponds to a maximal-edge bi-clique, which in turn corresponds to exactly to a closed item set [Bian et al., 2007]. If the data matrix is real valued, then a variety of algorithms exist to mine algorithms based on various properties of the data [Dhillon et al., 2003], and as already suggested, the definition of a subspace varies from algorithm to algorithm.

1.1.5 Formal Concept Analysis and Bi-Clustering

As is shown in [Bian et al., 2005] the formal biclusters located within a context correspond exactly to the bi-clusters. Formal Concept Analysis is a branch of applied mathematics and is an important approach in data analysis. FCA method basically takes in input in the form of a matrix of objects and its properties (attributes) and finds clusters of objects and attributes such that all objects in a cluster have the same attributes and also all attributes in the cluster are possessed by the same objects. These clusters of rows and columns (objects and attributes) are called as Formal Biclusters.

To further explain the idea of FCA, consider a matrix of objects and attributes, and every cell in the matrix represents a 1 or 0 indicating if the attribute holds of an object. The above example is to illustrate a concept lattice and is presented by Prof. David Eppstein. Here, the object set is a set of integers 1,2,3,4,5,6,7,8,9,10 and the attribute set or properties that these integers have are Composite, Even, Odd, Prime, Square.

The representation in Figure 1.1 is one representation of what is known as formal context. The clustering of the objects and properties of the formal context lead to formal biclusters that can be arranged in a concept lattice, that is shown in Figure 2. Mathematically, we define FCA as follows:
<table>
<thead>
<tr>
<th>Composite</th>
<th>Odd</th>
<th>Even</th>
<th>Prime</th>
<th>Square</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>8</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

Figure 1.1: Objects-Attributes

Figure 1.2: Concept-Lattice
A context $\kappa=(G,M,I)$ consists of two sets $G,M$ and a relation $I$ between $G$ and $M$. The elements of $G$ are referred to as objects and the elements of $M$ as attributes and $G \cap M = \emptyset$. The context is represented as a binary matrix $|G \times M|$. Let $\text{mat}(\kappa)$ is the matrix representation defined as

$$\kappa(i,j)= \begin{cases} 
1 & \text{if } g_i I m_j; \\
0 & \text{if otherwise.}
\end{cases}$$

For an object set $A \subseteq G$, we define the following as the objects of $M$ common to the objects in $A$,

$$A' = \{m \in M | g I m \forall g \in A\}$$

For an attribute set $B \subseteq M$, we define the following as the attributes of $G$ common to the attributes in $B$,

$$B' = \{g \in G | g I m \forall m \in B\}$$

Hence a concept (Bicluster) is defined as a pair $C=(A,B)$ with $A \subseteq G, B \subseteq M$ such that $A'=B$ and $B'=A$.

### 1.2 Problem Statement

Subspace Clustering algorithms usually define a *interestingness* measurement and search the data space based on the measurement, and prune subspaces that do not contain interesting clusters. One of the most defined interestingness measures is minimum density measurement, and we have many other properties defined on target clusters based on the domain of the data set.

Considering subspace clustering for real valued data, it is important to find interesting biclusters in large datasets that are correlated with statistical properties of the data, and also find biclusters that may overlap with some others in their attributes or in objects. But, one of the major shortcomings is the large number of biclusters that typically arise in datasets that are dense. Such a large number of generation of biclusters make the task of visualization and also rule generation and comprehension very tedious.

To overcome this shortcoming, it is important to develop formalisms and methods to segment, categorize and cluster biclusters. This helps us search for distinguished clusters and overcome redundant ones. The very first step in achieving these aims is to define a metric that computes how close two biclusters
are and numerically define the information that they share.

In this thesis we propose three metrics based on information theory and on the widely used statistical property of data i.e. deviation in a cluster. We have proposed design approaches based on these metrics, investigate the computational complexity of computing them, observe how the average deviation and entropy (the metrics defined) vary with reduction in generated biclusters, and compare their results on synthetic data and two real word data sets.

1.3 Thesis Organization

The remainder of the thesis contains the following five chapters.

Chapter 2 summarizes some related work in the area of subspace clustering of binary and real valued data, and work based on mutual information based biclustering, and similarity measures involved in identifying distinguished biclusters.

Chapter 3 elaborates the main idea behind this work and the design approaches. Algorithms for approaches based on the three metrics, comparison between the matrix and graph based approach, and why particularly the graph based approach did not give best results are explained.

Chapter 4 details on the different simulations done to evaluate the strategies. The design approach is evaluated for all three metrics based on 2 real world data sets and one synthetic data set. This is used to verify the validity of the theoretical idea that was arrived for the three metrics.

Finally, the conclusions of the completed research and some directions for its continuation in future research are described in Chapter 5.
Chapter 2

Related Research

This thesis proposes a new strategy for dealing with one of the drawbacks encountered in biclustering of real valued data; wherein, the number of biclusters generated tends to be quite large in dense data sets rendering it difficult to make interpretations about the bicluster. In order to overcome this, it is essential to develop methods to merge (or) cluster biclusters that are more or less the same, and project the most distinguished ones. These issues have been addressed marginally by a few unrelated works [Blachon et al., 2007].

The clustering algorithms presented in [Blachon et al., 2007] for biclusters generated for boolean datasets makes use of set theory particularly, set difference as a dissimilarity measure and compute the distance function between biclusters. The author carries out the concept reduction in two phases. He preprocesses the biclusters after concept generation i.e. applies a selection procedure such as size/ some biological condition to filter the biclusters, and further reduces the biclusters by hierarchical clustering of pseudo concepts as suggested in [Robardet et al., 2004], where in each cluster is associated with a pseudo concept summing up the main characteristics of the formal concepts it contains.

In [Formica, 2007] the author defines a single similarity measure based on information content approach between biclusters. But the similarity measure uses the information content computed only between concept intents i.e. attributes in a bicluster (inspired by maximum weighted matching problem in bipartite graphs). The approach proposed by Formica is more oriented to the Semantic Web and domain ontologies where in, the intensional components of biclusters are emphasized and can be defined without the extensional components and makes extensive usage of prior knowledge in the form of lexicographical databases to help evaluate the bicluster similarity.
Algorithms presented by [Belohlavek et al.] focus on reduction of formal biclusters of datasets having fuzzy attributes. The paper computes a factor lattice directly from the data instead of computing it from the concept lattice. The elements of the factor lattice are collections of clusters which are pair-wise similar in degree at least \( a \), a user-specified similarity threshold. On the contrary, the approach presented in this thesis is deterministic rather than being based on fuzzy attributes which is a separate line of research by itself. Similarity measures have also been studied in concept analysis of ontological data, and they can be either string-based or graph-based or knowledge-based. String-based measures employ techniques of edit distance, prefix, suffix, etc. while graph-based measures take into account the tree structure of the ontologies and integrate graph similarity along with concept similarity [Melnik et al., 2002]. Finally, knowledge-based similarity utilizes external knowledge sources to calculate similarities.

One paper that comes close to the work this thesis is focusing on is [Alqadah et al., 2009]. This paper extends well-known similarity measures Jaccard index, Sorensen coefficient and symmetric difference to cluster formal concepts. The algorithm is deterministic and is applicable to binary data. Alqadah applies weighted concept similarities to the existing mentioned measures. Also, the author introduces the idea of zero-induced similarity, which is the ratio of the number of non-zero cells to total number of cells that projects the quality of the clusters to an extent. Alqadah shows that the weighted concept similarities have two major shortcomings.

1. Setting the weights greatly affects the concept similarity and
2. The measures consider only the cardinality of the sets and do not explicitly consider the amount of information shared between the biclusters.

Also, both weighted similarity measure and the zero-induced similarity cannot be extended to real-valued data sets, which is what this thesis focuses on. Again, in another paper [Alqadah et al., 2009] also brings out similarity measures between biclusters to retrieve the most distinguished one, by growing a minimum spanning tree using Prims algorithm. Weights are assigned to the edges \((C2, C1) \in E \) if \( C1 \succ C2 \) of the directed graph constructed from the concept lattice based on a metric defined as "change in shape" of the corresponding concepts in the lattice. The minimum cost spanning tree thus constructed leads to the idea of measuring distinction between the biclusters. This is not directly applicable to biclusters based on real-valued, because we take only structural information between biclusters to compute the distinction.

One key strategy explored in computing similarities between biclusters in our approaches is based on mutual information which is extended from the work done by [Dhillon et al., 2003]. Dhillon treats the co-clustering problem as an optimization problem, and employs information theory in finding the optimal co-clustering that maximizes the mutual information between the clustered random variables subject to constraints on the number of row and column clusters. But, this work focuses on hard clustering and does
not apply to soft clustering, and explores biclusters from the data, that are not overlapping.

[Ye et al, 2009] extends Dhillon’s paper, by proposing a weighted approach to identify distinguished features, thereby improving the quality of the clusters as well, but this also is applicable only for hard clustering. Information based methods for semantic similarity has been detailed in number of proposals, and the main crux is based on the amount of information content of the common ancestors of two given bicluster-nodes. In one of such works [Resnik et al., 1999] the information content of a bicluster is defined as the negative logarithm of the probability of encountering an instance of the bicluster in the given corpus. For example, similarity between two biclusters, c1 and c2, is defined as

\[
sim(c_1, c_2) = \max_{\text{Subsume}(c_1, c_2)} [-\log P(c)],
\]

where Subsume(c1, c2) is the set of biclusters that subsume c1 and c2, and P(c) is the probability of encountering an instance of bicluster, c, in a corpus. But again prior information is required in computing the frequency information.

[Kim et al., 2006] proposed a vector based approach for dealing with concept similarity in ontological data that does not require prior frequency information, but used the hierarchical ordering of biclusters and quantified the degree of generality of a concept-node relative to another node in a hierarchy. The author measures generality of one bicluster to another by computing the \textit{share} of the individual biclusters, that is defined as a ratio of one over the total number of biclusters in its ancestry path, and further take the ratio of the two biclusters.

Now that we have seen parallel works, we can continue to explore the idea behind the approach adopted in this thesis. Our approach computes the similarity between biclusters by taking into consideration the statistical properties of the inherent data, and measures the variance in relative deviation and entropy of the clusters as cluster reduction proceeds.
Chapter 3

Proposed Strategies

In this chapter, we introduce various algorithms that we have proposed to solve the problem described earlier. Since we work on the biclusters mined from real valued data, we will need to briefly discuss the working behind the search for such biclusters, before we delve into the approaches proposed in this thesis.

3.1 Lattice Based Bi-Cluster Searching Algorithm

Many algorithms have been investigated for discovering biclusters, but fail to discover overlapping biclusters, or the biclusters discovered are not correlated with the statistical properties of the underlying real data. This work by Hu Zhen [Zhen et al., 2010] introduces an algorithm using search spaces defined by lattices and discovers interesting and overlapping biclusters despite the very large sizes of the lattices.

Zhen introduces an objective function and conducts the search on the lattice based on this function. Before, we describe the theory behind the objective function, we illustrate the idea on mining biclusters from real-valued data matrix by considering an example of gene-environment table (rows: genes, columns: environments). Shown below is the data matrix, where each cell represents the response of a gene under an environment. The biclusters shown below the table are a group of genes which have similar responses under certain groups of environment conditions. Biclusters shown in the figure are such that the values in the corresponding cells are below the standard deviation threshold "0.5" on the contrary to binary datasets, where in the theory of formal biclusters analysis treats all maximum sized sub matrices containing only 1’s as biclusters and arranges them in a partially ordered lattice.

Zhen’s approach extends the idea in [Hian et al., 2007][Alqadah et al., 2008] by considering a lattice of partially ordered closed biclusters and models it as a search space for his algorithm to look out for
potential good and overlapping biclusters. In his algorithm all maximum sized sub matrices are declared as biclusters, if the standard deviation of all included cell values in the sub matrix, is below a defined threshold and is also bound by a constraint on size, say, a bicluster should have a minimum of two rows and two columns. Further, the mined biclusters need to be closed under closure, and it is verified to be closed if the bicluster does not maintain its deviation below the threshold when a row/column is added to it. While most of the greedy search [Bryan et al., 2005] [Alqadah et al., 2008] based algorithms and pattern recognition techniques [Conlon et al., 2003] do seem to find overlapping and such closed biclusters, the author claims that they miss out on finding the best quality ones.

The objective function defined in the search algorithm is a metric "range". The author employs this metric instead of considering standard deviation directly, since deviation is not a monotonic property. The range of a bicluster with N data elements is defined as

\[
\text{max (N data elements)} - \text{min (N data elements)}
\]

It monotonically increases as the size of the sub matrix increases (either a row or column is added). Range for a bicluster is derived based on the following two lemmas:

(a) If every row and column in a bicluster is bounded by the range threshold \( \delta \), then the entire bicluster is bounded by a range threshold of \( 2*\delta \).

(b) If the range for each row and column of a bicluster is less than \( \delta \) then it is mathematically proven that the standard deviation of the bicluster is less than \( 2*\delta^2 \).

Hence, if the search algorithm wants to find biclusters with some bound on the standard deviation

<table>
<thead>
<tr>
<th></th>
<th>c1</th>
<th>c2</th>
<th>c3</th>
<th>c4</th>
</tr>
</thead>
<tbody>
<tr>
<td>g1</td>
<td>2.9</td>
<td>3.1</td>
<td>3.3</td>
<td>4.6</td>
</tr>
<tr>
<td>g2</td>
<td>3.1</td>
<td>2.8</td>
<td>2.8</td>
<td>1.7</td>
</tr>
<tr>
<td>g3</td>
<td>5.0</td>
<td>5.1</td>
<td>5.4</td>
<td>1.3</td>
</tr>
<tr>
<td>g4</td>
<td>2.2</td>
<td>5.3</td>
<td>4.9</td>
<td>.9</td>
</tr>
</tbody>
</table>
of the real-valued data, it only needs to look for adding rows and columns that are bounded by the range.

The entire search process is viewed as two independent phases put together. Prefix based equivalence classes have been used to formulate many search algorithms, and this technique has been employed in this paper. Prefixes can be formed either from column headings or from row headings. Zhen uses column headings, divides the search space into independent subspaces. In the first phase, children candidates are generated by adding columns to each parent bicluster, updating the range for each row in the context of newly added columns, and removing those rows whose range values exceed the specific threshold. In the second phase, all the generated biclusters are reexamined and the range for each column is checked if it is within the bound and if not, the row that adds up the most to the range is removed. Figure 3.2 and 3.3 clearly explains the phase 1 and 2.
In figure 3.2, the top level of the two branches are the column headings of the input data matrix. Consider the branch for column a. Since this approach follows prefix based equivalence classes, the columns that could be at level 2 are b or c or d. We enumerate \{a,b\}\{g1,g2,g3\}, and further go deep and enumerate \{a,b,c\}\{g1,g2,g3\}. If we examine clearly, the two biclusters \{a,c\}\{g1,g2,g3\} and \{a,d\}\{g1,g2,g3\} are crossed out. The reason the former is pruned out is, the bicluster \{a,b,c\}\{g1,g2,g3\} was generated already (we do a depth first rather than a breadth first search) and it contains \{a,c\}\{g1,g2,g3\}, and the latter is crossed out because it violates the size constraint.

Zhen states that this algorithm looks for interesting biclusters. Even though a number of pruning strategies based on containment, size, similarity and redundancy has been implemented to mine interesting biclusters, the resulting biclusters are still very huge in number. Also, the similarity pruning strategy defined in this paper is a bounded ratio of common rows and columns to the total number of rows and columns respectively, where in the threshold is user defined, loses out on certain biclusters because it doesn’t take into consideration the inherent statistical properties of the data in the sub matrix.

The total number of biclusters and the quality of the biclusters depends highly on the range threshold chosen. A very low threshold, with increasing row size limit of each bicluster, could generate lesser biclusters, while increasing range would generate more biclusters. The paper adopts the strategy of keeping top k clusters for the most interesting biclusters, but the objective of my thesis is to keep the information about all the biclusters, yet project the most distinguished ones and still maintain the information content. In order to produce only the most distinguished biclusters, we need a measure to compare two existing biclusters, and observe how similar \dissimilar are those biclusters, and further decide on opting which one out or merging them. Keeping this basic idea in mind, we have proposed three new methodologies and how certain parameters vary with the reduction in biclusters.

### 3.2 Relative Deviation Methodology

This strategy is based on the standard deviation of an N data element bicluster.

#### 3.2.1 Overview

Given any two biclusters, this methodology computes how different the two biclusters are, by considering the variance in the values enclosed by the biclusters. If the difference is huge or they are not overlapping when considering their structures, then they are individually important biclusters, and incase the difference is less, the information could still be contained in it by merging the biclusters.

Consider C1 <\{o1,o2,o3,o4\},\{a1,a2,a3\}> and C2 <\{o1,o2,o3\},\{a1,a2,a3,a4,a5\}> as biclusters of
real-valued data generated by the above algorithm, with mean and standard deviation \((\mu_1, \sigma_1)\) and \((\mu_2, \sigma_2)\) respectively. Observing further, we see that an addition of object o4 to C2 leads to the loss of two attributes \(<a4,a5>\), and forms bicluster C2. We decide on merging the two biclusters if the differing information between them is bounded by a threshold \(\delta\). If they are to be merged, then the resulting bicluster would be C3 \(<o1,o2,o3,o4,a1,a2,a3,a4,a5>\), with a standard deviation and mean as \((\mu_3, \sigma_3)\). In order to figure out how similar C1 and C2 are, we calculate the deviation in the values of the to-be merged bicluster with respect to the mean of C1

\[
\sigma_{3a} = \left( \frac{\sum_{i=0}^{n} (x_i - \mu_1)^2}{N-1} \right)
\]

(1)

and with respect to the mean of C2

\[
\sigma_{3b} = \left( \frac{\sum_{i=0}^{n} (x_i - \mu_2)^2}{N-1} \right)
\]

(2)

We define the metric \(\sigma_{diff}\) as follows.

\[
\sigma_{diff} = \max\left( |\sigma_{3a} - \sigma_1|, |\sigma_{3b} - \sigma_3| \right)
\]

(3)

If the maximum difference in deviation with respect to C1 and C2 is below the threshold \(\delta\) then, the biclusters C1 and C2 can be merged, indicating that despite the merging, the deviation of C3 would not significantly rise because majority of the values in biclusters C1 and C2 are the same, or their variance doesn’t significantly rise.

Further explaining it diagrammatically, consider the below figure for the above mentioned example. In Figure 3.4, we see C2 has 2 additional attributes and is down two objects. C1 and C2 are merged to generate C3 because, we do see that majority of C1 and C2’s values do overlap and on merging, the values in C2 do not affect the values much in C1 and vice versa. And although the small greener area on lower rightmost corner does have some outliers, since it does not affect the to-be-merged bicluster significantly, the merging can only be beneficial. In case of binary data, it would be evident, that we are adding 0’s to the merged bicluster, but since its real-valued data, the smaller area is almost neglected, because the variance of the bicluster C3 does not increase heavily.
3.2.2 Design Approach - 1

The above idea is implemented using two approaches. Due to limitations in the first approach, the second has been adopted for further analysis. The algorithms take as input the biclusters generated by [Zhen et al., 2010] for various a definite range.

3.2.2.1 Implementation using Graph Generation

The strategy behind this approach is to design a graph for the biclusters, where in the nodes are arranged in a hierarchical way, and apply the idea mentioned above on the graph in an iterative fashion until we reach a steady state.

3.2.2.2 Creating the Graph

In this step, we scan through the biclusters, and generate the graph with the biclusters that seem arranged in a hierarchical manner. This idea is extended from the paper on building a recommendation system [Narayanaswamy et al., 2008]. CreateBiclusterGraph() procedure takes in the biclusters generated using [Zhen et al., 2010]. First, the biclusters are sorted based on their attribute lengths. Further SplitByColLength() procedure groups all the biclusters together based on their column lengths. To explain further, let us say we had the following biclusters with column and rows as \(<\{a\},\{g1,g2,g3,g4\}>\), \(<\{b\},\{g2,g3,g4,g5\}>\), \(<\{a,b\},\{g1,g2,g3\}>\), \(<\{a,b,c\},\{g1,g2\}>\), \(<\{a,b,d\},\{g1,g3\}>\). Referring to Algorithm 2, SplitByColLength() generates 3 groups

\(<\{a\},\{g1,g2,g3,g4\}>\), \(<\{b\},\{g2,g3,g4,g5\}>\).
\(<\{a,b\},\{g1,g2,g3\}>\)
\(<\{a,b,c\},\{g1,g2\}>\), \(<\{a,b,d\},\{g1,g3\}>\)

Once the biclusters are grouped, in order to create the bicluster graph, we have to relate biclusters
based on their structure. Referring to Algorithm 3, CombineLevels() procedure, starting from the group of biclusters of least column length, progressively, adds edges between the biclusters in that group and biclusters in other groups. A directed edge is added between biclusters U and V (from U to V) in groups G1, G2 where G1.ColumnLength > G2.ColumnLength, indicating U is a parent of V and V is a child of U, if rows of U is the superset of rows of V and columns of U are a subset of Columns of V. We extend this property inherent in a bicluster lattice generated in FCA for binary data, so that we know which two biclusters are structurally overlapping.

Figure 3.5 below is a graph, connecting the above biclusters. As already explained, the biclusters are grouped and presented hierarchically based on increasing column lengths. Further, based on the column-subset and row-superset constraint we place edges between nodes.

```plaintext
input : Bicluster Sets B
output: Biclusters arranged in a Graph (V,E), with V={B1,B2, B3..,B1,B2,B3∈B} and E={(B1,B2), (B1,B2)∈ B}

1 begin;
2 Sort the biclusters in increasing order of bicluster column length;
3 maxLength←maximum column length in the biclusterSet;
4 biclustersByColLength[][]←SplitByColLength(B);
5 G←// G is a graph structure that contains a struct node, with each node being a Bicluster from biclustersByColLength[][]
6 for i←1 to maxLength do
7    | G←combineLevels(G,biclustersByColLength,i);
8 end
9 return G;
10 end;
```

**Algorithm 1: CreateBiclusterGraph**

Figure 3.6 and 3.7 shows the merging stages. We pick the edge with the least relative difference and merge them. The red boundary around the two edges indicate those two nodes are being merged, and further from Figure 3.7, we see that after merge, the node <{a,b,c}, {g1,g2}> loses its edge with the existing
node, and it gains an edge with node that had an edge with the existing node and is still has an edge with the merged node.

In order to merge biclusters, we initially calculate the mean and deviation of every bicluster. When we add directed edges between biclusters, we simultaneously compute the parameter "edgeDeviation". The edgeDeviation is nothing but $\sigma_{diff}$ defined in equation 3 above. "edgeDeviation" indicates how similar two biclusters are.

3.2.2.3 Bicluster Reduction

After running the procedure CombineLevels(), we pass the graph to the reduceBiclusters() procedure in order to get the most distinguished bicluster. Like already explained, we pass the threshold and the graph as inputs to the algorithm. Starting from least "edgeDeviation" (computed among all the edges), we merge the biclusters; if it is less than the defined threshold. The merged biclusters' deviation and variance are updated. Also, if the edge $C1 \leftarrow C2$ is least, we combine C1 and C2 and now the edges incident on C1 and edges leaving C2 will be retained on the newly merged node C1C2, in other words the parents and the children of the merged bicluster are recalculated and adjusted across the graph. Apart from this, the edges leaving from C2, and edges incident on C1, need to be verified if the edge condition still satisfies on the newly merged node (Row superset and Column Subset constraint). If the constraint satisfies, they hold true on the merged node, else those edges are deleted, before the nodes C1 and C2 are deleted from the graph.
**input**: Graph $G$, Biclusters partitioned based on column lengths $\text{biclustersByColLength}$, $\text{ColSize}$ is the current set of biclusters of column size $\text{ColSize}$

**output**: Graph with edges added between the nodes

1. $\text{numberOfBiclusters} \leftarrow \text{number of biclusters in biclustersByColLength}[\text{ColSize}]$;
2. $\text{levelBiclusters} \leftarrow []$;
3. $\text{for } i \leftarrow 1 \text{ to } \text{numberOfBiclusters} \text{ do}$
   4. // For every bicluster create a new node in $G$ that comprises of a nodeid, column no.s and row no.s in a bicluster
   5. $\text{newNode} \leftarrow \text{biclustersBySize}[\text{ColSize}][i]$;
   6. $\text{currentLevel} \leftarrow \text{ColSize} - 1$;
   7. $\text{elusiveBiclusters_Column} \leftarrow \text{newNode} \rightarrow \text{Column}$;
   8. $\text{checkContains_Column} \leftarrow \text{newNode} \rightarrow \text{Column}$;
   9. $\text{while } \text{elusiveBiclusters_Column is not empty and currentLevel is greater than 0} \text{ do}$
      10. $\text{numBiclusters} \leftarrow \text{number of biclusters in currentLevel}$;
      11. $\text{for } j \leftarrow 1 \text{ to } \text{numBiclusters} \text{ do}$
         12. $\text{levelBiclusters} \leftarrow \text{biclustersBySize}[\text{currentLevel}][j]$;
         13. $\text{checkRemaining} \leftarrow (\text{levelBiclusters} \rightarrow \text{col} \subseteq \text{newNode} \rightarrow \text{col}) \text{ and } (\text{newNode} \rightarrow \text{row} \subseteq \text{levelBiclusters} \rightarrow \text{row})$ and $\text{(checkRemaining is not empty)}$
            14. $\text{add directed edge from levelBiclusters to newNode}$;
            15. $\text{newBicluster} \leftarrow \text{levelBiclusters} \cup \text{newNode}$;
            16. $\sigma_{3a} \leftarrow \text{relative deviation of newBicluster with respect to } \mu_{\text{levelBiclusters}}$;
            17. $\sigma_{3b} \leftarrow \text{relative deviation of newBicluster with respect to } \mu_{\text{newNode}}$;
            18. $\text{edgeDeviation} \leftarrow \max (\sigma_{\text{newBicluster}} - \sigma_{3a}, \sigma_{\text{newBicluster}} - \sigma_{3b})$;
            19. $\text{elusiveBiclusters} \leftarrow \text{elusiveBiclusters} \cap (\text{newNode} - \text{levelBiclusters})$;
      16. $\text{end}$
      17. $\text{checkContains} \leftarrow \text{elusiveBiclusters}$;
      18. $\text{currentLevel} \leftarrow \text{currentLevel} - 1$;
   19. $\text{end}$
4. $\text{end}$

Algorithm 3: CombineLevels
Hence, taking the above example, when we merge \(<\{a,b,c\},\{1\}\>\) and \(<\{a,b\},\{1,2\}\>\), we get \(<\{a,b,c\},\{1,2\}\>\) and we remove the edge incident on \(<\{a,b\},\{1,2\}\>\) from \(<\{a\},\{1,2,3\}\>\) and add the edge from \(<\{a\},\{1,2,3\}\>\) to \(<\{a,b,c\},\{1,2\}\>\). And further compute how close is this bicluster to the other biclusters it is connected to, and redo the process of merging again until the threshold condition is not satisfied or no more structurally overlapping biclusters exist.

### 3.2.2.4 Limitation

One of the main drawbacks with this implementation is that, we are missing out on merging some biclusters, due to the condition on establishing edges between nodes in the graph. It is important to notice that the graph could have nodes that are just standalone, and are not connected to any other node. Such nodes are not taken into consideration while merging, since it could be categorized as a distinguished cluster. The merge process using this implementation results in number of standalone nodes. e.g. in the above section we took an example, where in we merged \(<\{a\},\{1,2,3\}\>\) and \(<\{a,b\},\{1,2\}\>\), which resulted in \(<a,b,1,2,3>\). Let us say, we chose to merge \(<\{a\},\{1,2,3\}\>\) with the child \(<\{a,b\},\{1,2\}\>\) than \(<\{a,c\},\{1,3\}\>\), because the former biclusters were closer than the latter. But if we notice we are losing the other child of \(<\{a\},\{1,2,3\}\>\)
Algorithm 4: MergeBiclusters

```plaintext
input : Graph G, threshold δ
output: Merged Nodes and altered edges in the Graph

begin;
while smallest edgeDeviation of all edges e∈E≤δ do
merge u,v where (e=u,v) u and v are the biclusters;
Compute σ for the mergedBicluster;
make edges incident on u to be incident on merged bicluster;
make edges leaving from v to leave from the merged bicluster;
for ∀e=(u,u1)∈E do
  if u1 ⊆ merged bicluster then
    add directed edge from u1 to merged bicluster;
  else
    add directed edge from merged bicluster to u1;
  end
  Update edgeDeviation for new edge;
delete edges leaving from and incident on u;
delete u;
end
for ∀e=(v1,v)∈E do
  if v1 ⊆ merged bicluster then
    add directed edge from v1 to merged bicluster;
  else
    add directed edge from merged bicluster to v1;
  end
  Update edgeDeviation for new edge;
delete edges leaving from and incident on v;
delete v;
end
return G;
end;
```
which is $<\{a,c\},\{1,3\}>$, after merge since $<a,c,1,3>$ is no longer the child of the merged bicluster. In this case, we could have merged all the three nodes if they shared more common information leading to $<\{a,b,c\},\{1,2,3\}>$.

### 3.2.3 Design Approach - 2

#### 3.2.3.1 Implementation using Matrix

Due to the drawbacks in the implementation using graph, the relative standard deviation approach is implemented by constructing a $N \times N$ deviation matrix, where $N$ is the total number of biclusters generated using the lattice based search algorithm [Zhen et al., 2010]. We need to compute only the upper diagonal matrix.

For every pair of biclusters ($G_1, G_2$) in the upper diagonal, we check how much do they structurally overlap. In this algorithm, we examine if any two biclusters have at least two overlapping rows and columns. If so, we compute the deviation of the to be merged cluster $G_1G_2(G_1 \cup G_2)$ with respect to mean of $G_1$ and $G_2$ using equation 1,2 and 3 explained in the beginning of this chapter. Figure 3.5 below is the deviation matrix for 3 biclusters.

![Deviation matrix for 3 biclusters](image)

For beginning the merge process, we always start with the most similar biclusters (smallest deviation value) and examine if the deviation is less than the defined threshold $\delta$. If it satisfies the threshold constraint, we merge them into cluster $G_1G_2(G_1 \cup G_2)$. Now, we further compute the similarity between the newly merged bicluster and the rest of the biclusters and continue the merging process, until the threshold condition is not met or the overlapping condition is not met at all for any of the pair of biclusters.

Referring to Figure 3.5, we can see $G_1$ and $G_2$ have the least deviation; hence they are more
similar, thus G1 and G2 are merged. Once they are merged, the deviation matrix is shrunk, meaning both G1 and G2 are taken out of the context, and the deviation is computed between G1G2 and the rest of the biclusters, and the merge process is continued all over again. Figure 3.6 explains the change in deviation matrix after a merge.

![Figure 3.9: Deviation Matrix After Merge.JPG](image)

Procedure MergeBiclusters() takes two input parameters, real valued data set and the biclusters. It computes the mean $\mu$ and $\sigma$ for all the biclusters to start with. Further, for all pairs of biclusters, the similarity matrix is computed by calling procedure CalculateSmlrtypairwise(), that takes biclusters as one input parameter, whose similarity is to be computed, and returns the similarity value, based on which the merging is done. If the val $\leq$ $\delta$ (the threshold), then the biclusters are merged, the two biclusters are removed from the bicluster set, and also the existing similarity values computed between the two biclusters and the rest of biclusters in bicluster set, are erased from the deviation matrix.

### 3.2.3.2 Complexity Analysis

The complexity of the merge algorithm is computed analyzing every step. Initially in order to calculate the similarity matrix, it would be $(N \times N - 1)/2$, where $N$ is the number of Biclusters, since we are calculating the upper diagonal of the matrix. Hence this step is of $O(N^2)$. Further to retrieve the closest similarity ones, we maintain a multimap, that takes $O(N \log N)$ where $N = (N \times N - 1)/2$, and deleting and adding is again $O(N \log N)$. On deciding the biclusters to merge, the similarity matrix need to be recomputed, with deleting the two biclusters compute similarity values from the matrix, which takes constant time. Now, updating the similarity matrix by computing the similarities between newly merged bicluster with other biclusters takes after every merge, for $N$ concepts say until one concept, will take $(N-2)+(N-3)+(N-4)+(N-5)+...+(N-(N-1))$ that is again $O(N^2)$. Hence the $O(N^2)+O(N^2)+O(N \log N)$ would be the total time.
input : Bicluster Set $B=R, C$, Input Data Set $D$
data : $\beta, M_\text{SimrtyValBetween}((B_i, B_j))|((B_i, B_j)) \in B$
output: Merged BiClusters

begin;

for $b_i \in B$ do

$\mu_i \leftarrow \text{Mean}(b_i, D)$;
$\sigma_i \leftarrow \text{Deviation}(b_i, D)$;
Entropy $i \leftarrow \text{Entropy}(b_i, D)$;

end

for $(b_i, b_j) \in B$ do

$M \leftarrow \text{CalculateSmlrtyPairwise}(b_i, b_j)$;

end

for smallest $\text{simrtyVal}$ between $(b_i, b_j) \in M$ do

if $\text{simrtyVal} \leq \beta$ then

$B_{n+1} = ((r_i \cup r_j), (c_i \cup c_j))$;
$\mu_{n+1} \leftarrow \text{Mean}(B_{n+1}, D)$;
$\sigma_{n+1} \leftarrow \text{Deviation}(B_{n+1}, D)$;
Entropy $n+1 \leftarrow \text{Entropy}(B_{n+1}, D)$;
$B \leftarrow B \cup B_{n+1}$;
$B \leftarrow B \setminus (b_i, b_j)$;
for $b_i \in B$ do

$M \leftarrow \text{CalculateSmlrtyPairwise}(B_{n+1}, b_i)$;

end

end
else

return $B$;
end

end

end;

Algorithm 5: Merge Biclusters

input : Bicluster Sets $B1=(R1,C1)$ and $B2=(R2,C2)$ with mean,deviation,size as $(\mu_1, \sigma_1, n1)$ and $(\mu_2, \sigma_2, n2)$
output: similarityVal

begin;

mergedBicluster $=((R1 \cup R2), (C1 \cup C2))$;
n3=sizeOf (mergedBicluster);
$\text{stddev}_{\text{mergedBicluster1}} \leftarrow \text{calcStdDev(mergedBicluster values, } \mu_1)$;
$\text{stddev}_{\text{mergedBicluster2}} \leftarrow \text{calcStdDev(mergedBicluster values, } \mu_2)$;

similarityVal $= \max (|\text{stddev}_{\text{mergedBicluster1}} - \text{stddev}_{1}|, |\text{stddev}_{\text{mergedBicluster2}} - \text{stddev}_{2}|)$;
return similarityVal;
end;

Algorithm 6: CalculateSmlrtyPairwise
complexity. Hence it is polynomial time in the number of biclusters. The time complexity would further be increasing in polynomial order depending on how big the data set is.

Computing the similarity value requires us to compute the mean and variance of the to-be merged bicluster beforehand. For a total of \((n^*(n+1))/2\) sets i.e. starting from the first cluster and pairing it up with \(n-1\) biclusters, second bicluster with \(n-2\) biclusters and so on till \(n+1\)st bicluster, we need to calculate the mean and variance of the to be merged biclusters, which takes approximately \((r_1+r_2)*(c_1+c_2)\) computation time for every pair with total number of rows and columns as \((r_1,c_1),(r_2,c_2)\). Hence, we make use of the existing mean and variance of the original two biclusters, and compute the merged biclusters mean and variance, which hence reduces the time complexity. In \(2(m+n-1)\) comparisons, we compute the newly added rows and columns and further compute their mean and second moments. Hence with this and the existing mean and second moment of the two biclusters, we can compute the deviation and mean of the to be merged biclusters.

The iterative merging process is executed for various threshold values, to see the change in number of biclusters vs. the average deviation of total number of biclusters at every threshold level.

### 3.3 Mutual Information Methodology

In this section, we have proposed 2 approaches using the concept of information theory in bicluster reduction, one involving information theory as a similarity measure itself. We are going to explore how information content varies with bicluster reduction.

#### 3.3.1 Overview

**Information Theory** is a branch in mathematics, based on probability theory and statistics that teaches us how to quantify information. It was developed by Claude E. Shannon to find fundamental limits on signal processing operations, and his theory has also been extended to many other branches in engineering. One of the key concept in information theory is "Entropy".

Entropy quantifies the uncertainty involved in a random variable. For e.g.: Suppose one transmits 100 bits (0s and 1s). If these bits are known ahead of transmission, logic dictates that no information (entropy is 0) has been transmitted. If, however, each is equally and independently likely to be 0 or 1, 100 bits have been transmitted with respect to information bits.

Hence mathematically defining it, If \(X\) is the set of all messages \(x_1,\ldots,x_n\) that \(X\) could be, and
p(x) is the probability of X given \( x \in X \), then the entropy of X is defined as

\[
H(X) = - \left( \sum_{x \in X} p(x) \log p(x) \right)
\]

3.3.2 Computing the Entropy

Given a bicluster \( B_1(R_1,C_1) \), we find the range \( \Delta \) of the bicluster. The range of the bicluster is nothing but, the maximum value minus the minimum value in the data grid enclosed by the bicluster i.e. \( \text{max}(B_1) - \text{min}(B_1) \). Starting from the \( \text{min}(B_1) \), in increments of \( \Delta/10 \) until \( \text{max}(B_1) \), we count the number of values in the data grid enclosed by the bicluster that falls in each of the range until \( \text{max}(B_1) \). Further, after getting the frequency of every value in the data grid, we find the probability of their occurrence. Hence the pmf under the curve can be plotted.

Let us say we have a bicluster \( B' \) with 3 rows and 2 columns, with values 1.3, 1.4, 1.3, 1.5, 1.9, 1.7, we compute the range \( \Delta = 1.9 - 1.3 = 0.6 \). Now starting from lowest, 1.3 in increments of .6, we plot the X axis range. For every range of values in the X axis, we compute the number of values enclosed by the bicluster that falls in that range. The reciprocal of this count will give the frequency. Also, we can calculate the probability of that value occurring, from the frequency. Hence we can compute the PMF for the bicluster. This methodology paves way to find out the entropy that quantifies the information gain for this bicluster using the equation stated in the above section. Hence, we compute the product of the probability of every range defined on the X axis occurring in the bicluster and their logarithm until the max range, and further compute the sum of the products. The negative of this value gives us the entropy of the bicluster.

3.3.3 Design Approaches

We are implementing the idea using the matrix generation methodology.

3.3.3.1 Approach-1

In this approach, we are going to monitor the change in information content as the merging takes place. Here, the bicluster reduction methodology entails the same described for the relative deviation methodology. The algorithm MergeClusters holds for this approach as well. The similarity measure adopted in this section is a very basic yet, intuitive measure. We compute the statistical mean for all the biclusters, i.e. ratio of summation of all values enclosed by the bicluster to the total size. The intuition behind this
approach is that, the biclusters whose means are very close also have values centered around those means quite similar. And hence, the information gain on merge is expected to be controlled since the values are around very close mean values. If the difference in mean between two biclusters is below the defined threshold, then the biclusters are merged.

Reiterating with an example, let us assume biclusters G1, G2, G3 and G4. We compute the statistical means for all 4 biclusters and compute the upper diagonal $4 \times 4$ deviation matrix, storing the difference in their means, if they are structurally overlapping to some degree (say 2 biclusters should have at least 2 rows and 2 columns overlapping). If the difference satisfies the threshold $\delta$ defined, then we merge them. And as described in the previous approach, we re-compute the mean for the merged cluster and update the similarity matrix by computing the difference in mean between the newly merged bicluster and the existing ones. The merging process continues iteratively, until threshold condition does not satisfy, or there are no more biclusters left over that are structurally overlapping.

Before, the merging process begins; one additional factor that we compute for all the biclusters is the entropy for the bicluster. After every merge, we re-compute the entropy of the merged cluster. When the threshold condition is no longer met for the merging to continue, we observe some interesting patterns in the merged biclusters and also the variation in average information gain, before and after merge.
### 3.3.3.2 Approach-2

Approach-2 makes use of entropy as a similarity measure in comparing biclusters. The algorithm designed for merging is similar to Relative Deviation Methodology i.e. matrix methodology described in the previous section. The main variation in this approach is the procedure to calculate the similarity between biclusters.

**Algorithm 7: CalculateSmlrtyPairwise**

```plaintext
input : Bicluster Sets \( B1=(R1,C1) \) and \( B2=(R2,C2) \) with mean, entropy, size as \((\mu_1, H(X_1, n1))\) and \((\mu_2, H(X_2, n2))\)
output: smlrtyval
begin;
smlrtyval = |\mu_1 - \mu_2|;
end;
return smlrtyval;
end;
```

**Algorithm 8: Merge Biclusters**

```plaintext
input : Bicluster Set \( B=R,C \), Input Data Set \( D \)
data : \( \beta, M=SmlrtyValBetwe en((Bi,Bj))|((Bi,Bj) \in B \)
output: Merged BiClusters
begin;
for \( bi \in B \) do
    \( \mu_i \leftarrow \text{Mean}(b_i, D) \);
    \( \text{Entropy}_i \leftarrow \text{Entropy}(b_i, D) \);
end
for \( (bi,bj) \in B \) do
    \( M \leftarrow \text{CalculateSmlrtyPairwise}(bi,bj) ; \)
end
for smallest smlrtyval between \( (bi,bj) \in M \) do
    if smlrtyval \( \leq \beta \) then
        \( B_{n+1} = ((r_i \cup r_j),(c_i \cup c_j)) ; \)
        \( \mu_{n+1} \leftarrow \text{Mean}(B_{n+1}, D) ; \)
        \( \text{Entropy}_{n+1} \leftarrow \text{Entropy}(B_{n+1}, D) ; \)
        \( B \leftarrow B \cup B_{n+1} ; \)
        \( B \leftarrow B \setminus (b_i, b_j) ; \)
        for \( bi \in B \) do
            \( M \leftarrow \text{CalculateSmlrtyPairwise}(B_{n+1}, b_i) ; \)
        end
    end
else
    return \( B ; \)
end
end;
```

Instead of comparing means between the biclusters, we include entropy in computing the simi-
input: Bicluster Sets $B1=(R1,C1)$ and $B2=(R2,C2)$ with mean,entropy, size as $(\mu_1, H(X_1,n1))$ and $(\mu_2, H(X_2,n2))$
output: smlrtyval
begin;
mergedBicluster = $((R1 \cup R2),(C1 \cup C2))$;
$H(X_{temp}) = \text{CalcEntropy}(\text{mergedBicluster})$;
smlrtyval = $|H(X_{TEMP}) - (H(X_1 + H(X_2)/2)|$
end;
return smlrtyval;
end;

Algorithm 9: CalculateSmlrtyPairwise

We compute the entropy differences between the merged cluster and the biclusters before merge.

$$Distance Measure = H(X_3) - (H(X_1) + H(X_2)/2)$$

The intuition behind the above definition is how much is the information gain of the merged bicluster when compared to the average information gain of both the biclusters. Meaning, if there are very few unrelated values that are getting added up in the merged cluster then the information gain difference would not be very high, but if there are more number of unrelated values on merging, then the gain will be a whole lot. We are interested in a not so very high information gain, because we are looking at merging biclusters that have almost the same set of values, and those very few unrelated values do not matter much on merging.

In the below figure, we have elucidated the theory behind this idea. G1 and G2 are two biclusters that have been extracted using the lattice based search algorithm for a range less than 0.5. Now, when we are trying to merge based on the entropy difference, we need to verify the information gain. We see G1 and G2 have only $3 \times 3$ matrix of values overlapping. If we had merged the said biclusters into a $8 \times 7$ matrix, we see the bicluster on merging has a $5 \times 3$ matrix of unrelated values, that significantly add up to the information content, though there are quite a few values that still could be merged. Hence defining the right threshold is important in this case, so that we merge biclusters without having to compromise much on the information content.

Similar to the above approach, we compute the entropy differences between biclusters that are structurally overlapping to a user defined extent, and fill up the upper diagonal of the deviation matrix before initiating the merging process. Picking up the least distance value, we check if it is less than the defined threshold $\delta$ as done before, and if the threshold constraint satisfies, then we merge the two biclusters sharing the least distance. Further, we re-compute the similarities between the other biclusters and the newly
merged bicluster and further continue with the merging process, until we reach the threshold condition.

We gather the average entropy information, once the threshold condition has been reached, to observe the variation in entropy before and after merge.

### 3.3.3.3 Complexity Analysis

The complexity of the main merge algorithm is computed already in the previous section analyzing every step. As already stated in the previous section it takes polynomial time in the number of biclusters. The time complexity would further be increasing in polynomial order depending on how big the data set is.

Approach-1’s similarity measure calculation takes linear time $O(P)$ where $P$ is the size of the bicluster, for computing the difference in mean, and Approach-2 for computing the pdf, we perform a sort of the values i.e. $O(n \log n)$, first and compute the probability of occurrence of the set of values that falls in a range starting from $\min_{val} + (\max_{val} - \min_{val})/10$ to $\max_{val}$, which takes constant time. Hence this approximately takes $O(n \log n)$. Hence approach1’s complexity would $O(n^2) + O(P)$, and the second would take $O(n^2) + O(n \log n)$. 

![Diagram](image.png)
Chapter 4

Analysis and Results

This chapter focuses on the evaluation part of the research. All experiments were performed on Windows on a Pentium D 2.8 GHz, 3.68 GB of RAM, and all implementations were programmed in C++. The analysis of the proposed strategies is carried out using

(A) A synthetic data set

(B) 2 real world data sets, a chip sequence dataset [Chen et al., 2008] and Parkinsons data set [Little et al., 2007] Further, the approaches presented are compared with a baseline approach and the variation is analyzed.

4.1 Application to Synthetic Data

The algorithm MergeClusters is applied on a synthetic data set having 11 attributes and 25 objects. The data sets are prepared in a way that it has dense areas of biclusters to be found. Initially, the lattice based search algorithm [Zhen, 2010] is applied on the data set first, and the biclusters are retrieved with a given range of 1.0. The algorithm retrieves a total of 84 biclusters, that has a range of attribute and object sizes, given the minimum row and column size constraint to be 4 and 2 respectively.

4.1.1 Sample Merging Results

The algorithm is run on the synthetic data set, with all the three defined CalculateSmlrtyPairwise methods for the approaches based on relative deviation and mutual information. Shown below is a sample merging process as a result of applying relative deviation methodology. Figures have been titled Merge-1, Merge-2 and so on indicating that these merges take place at different threshold levels. Merge-1 indicates
that few values have got added on after merging. (C3) $8 \times 3$ and (C72) $7 \times 4$ biclusters are merged into (C3C72)$8 \times 4$ bicluster that has a dark blue colored strip indicating that a value got added that does not maintain the original range of the bicluster. But then, apart from the out of range value, the rest of the values i.e. $8 \times 4 \ominus 1$ do not affect the range or the deviation. Progressively merging generates the bicluster (C3C72C2) as shown in Merge-2 where it is observed that the deviation increases further because some more out of range values are further added. On Merge-3, this bicluster c3c72c2 is combined with another bicluster that yields to more number of noise values, leading to a pretty high deviation.

After every merge, it is observed that the deviation of the merged cluster is greater than the deviation of the biclusters that are merged. A merged cluster could get merged again with another potential bicluster at the current threshold itself, or may get merged at higher threshold level.

On applying the approach-2 based on mutual information, the merging results in biclusters which exhibits the same nature as above. Fig 4.3 shows the biclusters before and after merge. Also, similar to the deviation, as the merging progresses, it is observed that the out of range values keep getting added on while the deviation is kept below the threshold.

As with the standard deviation of the merged bicluster, the entropy of the merged biclusters is
always greater than entropy of the individual biclusters, i.e. for any two biclusters $B_1$ and $B_2$, $E(B_1 \cup B_2) > E(B_1)$ and $E(B_1 \cup B_2) > E(B_2)$. This statement is pretty intuitive because, if there was no gain of information, then it would have to be a maximal bicluster that the lattice based algorithm would have already generated. Hence as and when the merging continues, there is always an information gain.

Applying the approach-1 based on mutual information methodology, merged biclusters get generated at every threshold level, and as with the other approaches the entropy increases as merging proceeds.

On comparing the biclusters that the two mutual information based approaches generated; it can be clearly observed that approach-2 works better than approach-1. In the above sample merging examples of approach-1 and approach-2, the entropy of the merged bicluster $c_{17c18}$ is lesser than the merged bicluster $c_{17c40}$. Hence merging $C_{17}$ with $C_{18}$ seems to be a better choice over merging $C_{17}$ and $C_{40}$. Approach-2 results in a lesser quality bicluster than the approach-1 that takes entropy into consideration while computing the similarity.

Observing the same bicluster $C_{17}$ under relative deviation methodology based merging; it is evident that this approach 4 makes a better quality oriented merges than approach-2. $C_{17}$ merges with $C_{14}$ that yields $C_{17C14}$ that has a lower deviation than $C_{17C18}$ produced by approach-2. Relative deviation approach seem to be providing better merged biclusters than approach-2 that can be confirmed by comparing the average deviation and the average entropy of the clusters generated by both approaches.

### 4.1.2 Comparison with Baseline Approach

Besides comparing the effectiveness of each of the approaches with respect to each other, it is also mandatory to exhibit the effectiveness of the approaches presented in this thesis over a merge that does
Figure 4.3: Mutual Information Methodology: Approach-2- Merge-1
Information Methodology: Approach-1 - Merge-1
Figure 4.4: Relative Deviation vs. Approach-2 Approach: Merge-1
not employ the characteristics of the inherent data. In order to examine this, a baseline approach has been employed to merge the biclusters, and further compare the mentioned strategies with the baseline approach.

The baseline methodology adopted in this thesis, has been used in a number of biclustering algorithms for binary data, to compute the quality of the biclusters. The similarity measure in the baseline approach is computed using only the cardinality of the attributes, instead of using the statistical properties of data. The measure is defined as follows:

$$sim_{G1,G2} = \left( \frac{|A1 \cap A2|}{|A1 \cup A2|} \right)$$ (1)

Application of the baseline approach to the above synthetic data set results in merges that pretty much exhibit how the approaches defined in this thesis works better and gives satisfactory results. Shown below is a sample merge of two single biclusters using the relative deviation and the baseline approach. Considering bicluster c66 it is observed that this gets combined with C24 in baseline, and it gets combined with c67c68 in relative deviation as seen below.

On further scrutiny, it can be seen from the figures that relative deviation approach is giving a better merged bicluster than the baseline approach. For real valued data, considering only cardinality seem to result in a wrong merging. It can be proved by looking at biclusters C24 and C66. Though they may have totally overlapping attributes, C24 has values ranging from 6 to 7 and C66 having values ranging between 1 and 3. Baseline approach is basically merging two totally non overlapping biclusters, although they may fall under the same subspace, which is not the expected outcome.

Also, the mutual information based approach leads to a merging much better than the baseline approach, as with the relative deviation. Figure 4.8 below explains this further. C17 got clubbed with C18 under relative deviation methodology that is shown in the previous section. Under baseline approach, C17 bicluster merges with C62, producing C17C62 having a much higher entropy than C17C18.

Comparing the entropies of the final merged cluster and also looking at the higher ranged value getting included in the merged cluster of the baseline approach, it’s pretty obvious that the approach-2 evaluates better than the baseline approach.

### 4.1.3 Analysis

Listed below are the plots of both average standard deviation and average entropy vs. the total number of biclusters at every threshold. Referring to Figure 4.9, it can be observed that the average
Figure 4.5: (Baseline vs. Relative)Baseline Approach
Figure 4.6: (Baseline vs. Relative) Relative Deviation Approach
Figure 4.7: (Baseline vs. Approach-2) Baseline Approach
standard deviation of the total number of biclusters at every threshold level increasing as the biclusters reduce in number. 25 biclusters have an average deviation of 0.74 while, the same number of biclusters have a deviation of .42 under the relative deviation approach. This explains clearly that the relative deviation approach performs quite well as compared to baseline. Also, though the approach-2 does not perform as good as the relative deviation approach, it does seem to do better than baseline.

Figure 4.10 also conveys the same as Figure 4.9. The relative deviation methodology performs better in terms of information content also in every bicluster. But here, we see that the mutual information content approach is as good as relative deviation methodology. And the baseline approach still produces lower quality biclusters on merge, and has the worst case performance. Figure 4.11 compares the average entropy of the biclusters produced by the approach-2 and approach-1 based on mutual information methodology at every threshold level. Approach-1 was experimented with the intuition that the smaller the statistical mean difference between two biclusters are, the information gain would still be smaller. This comparison brings out the fact that approach-2 works better than approach-1, as in the mutual information approach where entropy is included while computing similarity measures, performs much better than the statistical mean approach.
4.1 Application to Synthetic Data

Figure 4.9: Variations In Average Entropy as Threshold increases

Figure 4.10: Approach-1 vs Approach-2
4.2 Experiments in Real Data

In order to measure the performance and the quality of the biclusters on merging, the algorithms are evaluated using real-world datasets.

4.2.1 Parkinsons Data Set

Parkinsons data set has 112 objects and 22 attributes [Little et al., 2007]. The data set is normalized before applying the lattice based search algorithm with range .5 for retrieving biclusters.

4.2.1.1 Merge Results

CalculateSmlrtyPairwise algorithm formulated for relative deviation approach is applied first, and below is one sample merging that takes place. It is observed that, at threshold .03 and threshold 0.1, merging of two biclusters take place. Further at a much higher threshold, the merged biclusters again merge further with another bicluster leading to a new one, with a much higher variance. On further scrutiny, it can be seen that the deviation is constantly increasing at every threshold level, as already observed in synthetic data. The biclusters data set at every level is presented below to show the variation.

Figure 4.11: Merge Process
4.2 Experiments in Real Data

Figure 4.12: Relative Deviation Approach: Merge-1
Figure 4.13: Relative Deviation Approach: Merge-2

Figure 4.14: Relative Deviation Approach: Merge-3
Results similar to the above and also as that of synthetic data are produced while merging using the mutual information approaches. The entropy continues to increase as merging takes place progressively. Below is a flow diagram indicating a bicluster after merge, indicating merges at different threshold levels. The blocked rectangles on the top indicate the threshold value, and the left most set of biclusters mentioned, are the original biclusters, including information about row count, column count and standard deviation of the bicluster. The bicluster after every merge is written under the blocked rectangle that is indicating the threshold value.

Figure 4.15: Approach-1-Merge Process

Figure 4.16: Approach-2-Merge Process
In order to illustrate the differences in merging between the approaches, C15, a single bicluster is analyzed for merge under 2 different approaches in Fig 4.19 and Fig 4.20. It is observed that C15 bicluster clubs with C16 under relative deviation approach and clubs with c11 in approach-2. Higher range of values gets added amongst other in-range values in Fig 4.19 when compared to the bicluster in Fig. 4.20. The relative deviation produces a better quality than approach-2, in this case, but then it can affirmatively state it only when considering the average deviation and entropy of the total number of biclusters at every threshold value.

Similar to the above comparison, approaches based on mutual information methodology are being compared below, using a sample merge of bicluster C42. As with synthetic, approach-2 works better than approach-1 in this case.
Figure 4.18: (Relative Deviation vs. Approach-2) Relative deviation approach
Figure 4.19: (Approach-1 vs. Approach-2) Approach-2
Figure 4.20: ((Approach-1 vs. Approach-2) Approach-2)
4.2.1.2 Comparison with Baseline Approach

Similar to the comparison done in synthetic data, this section continues to compare the baseline approach described in the previous section with the approaches presented in this thesis. A sample of a bicluster merge at the first level is taken from all the three approaches and presented here for comparison. The figures clearly are self-explanatory and reveal that other approaches perform much better in this merge than the baseline approach.
Figure 4.22: (Baseline vs. Relative) Relative Deviation Approach
Figure 4.23: (Baseline vs. Approach-2) Baseline Approach2
Figure 4.24: (Baseline vs. Approach-2) Approach-2
4.2.1.3 Analysis

Having compared sample bicluster merges with the baseline approach; presented here are the summary of the results. Shown below are the two graphs plotting the variation in the average standard deviation and entropy of the total number of biclusters generated at that threshold for all the three approaches.

![Variation in Average Standard Deviation](image)

Figure 4.25: Variations In Average Standard Deviation as Threshold increases

According to the graphs, all three approaches perform better than the baseline approach. But relative deviation approach gives out better quality results both in terms of controlling the deviation and also the information content on merging. Mutual Information Approach-2 performs as same as relative deviation approach while only analyzing the information content, almost the same as observed in synthetic data set. But the average standard deviation for biclusters generated by approach-2 does shoot up quite higher when compared to similar merges in deviation approach. We see that as we increase the threshold, the deviation and the entropy increases, and we may want to choose the right threshold, in order to avoid very high variation in variance while merging. An approach that controls both mutual information and standard deviation on merge is the ideal merge that we expect.
Something worth mentioning in information content approach would be, the type of biclusters formed by both relative deviation and mutual information theory approach. Most of the cluster merges are similar in nature until one level of merging i.e. where in only two single biclusters are involved in the merge process. Also in this case, after merge we would notice that either the object size or the attribute size increases but not both for many biclusters. This indicates that the two biclusters are almost overlapping to its complete size, except a few data points.

Another interesting result is observed when comparing the baseline approach to the relative deviation approach. We do notice that 10% of biclusters after merging, (considering merges at all the thresholds) do match. But then for many bicluster merges where in the number of biclusters merged are greater than 2, the order in which the merging takes place is different. Hence, if we are deciding on an objective function of minimizing the variance in a bicluster after merge, then certainly the relative deviation approach has an upper hand than the baseline approach. Shown below is sample order difference in the merge process.
4.2.2 Chip Sequence Data Set

This is the second data set on which merge algorithm has been evaluated. The data set has 18936 objects and 12 attributes. It is normalized before applying the lattice based search algorithm with range .001, with a constraint on the minimum number of rows and columns as 20 and 3 for retrieving biclusters.

4.2.2.1 Merge Results

As done for the synthetic and parkinsons data set, CalculateSmIrtyPairwise() algorithm built for all three approaches is executed on this data set. And the following are some samples of biclusters generated by the three approaches.

It is observed that as in the previous results, even in this data set, the avg. entropy increases as we merge and is always greater than the clusters that we merge. The same applies to average standard deviation too. In depth analysis has been avoided on comparison between the approach-2 and the relative deviation approach because it is consistent with this data set as well.

4.2.2.2 Baseline Comparison and Analysis

Baseline approach applied to this data set generates merged biclusters with a very high variance. And it is extremely obvious from the graphs in Fig. 4.36, which relative deviation methodology performs better than the baseline approach, considering both the example of the merging in Fig. 4.32 below as well as from the graph plotting the average standard deviation vs. the total number of clusters. Also Fig 4.34
shows the difference between average entropy variation between method 3 and baseline approach.

The results we infer from the below graph are consistent with the synthetic and parkinsons data set. The relative deviation approach gives better quality biclusters than the other approaches, in terms of the variance of the bicluster. Also the graph in Fig 3.9 indicates that the approach-2 performs better than approach-1 in terms of the information content in the bicluster, and also better than the baseline approach as indicated in Fig 3.8
4.2 Experiments in Real Data

Figure 4.29: Relative Deviation Approach: Merge-1

(a) Pathways_M1

(b) Pathways_M1
Figure 4.30: Mutual Information Methodology: Approach-1- Merge-1
Figure 4.31: Mutual Information Methodology: Approach-2- Merge-1
Figure 4.32: (Baseline vs. Relative) Relative Deviation Approach
Figure 4.33: (Baseline vs. Relative)Baseline Approach
Figure 4.34: (Baseline vs. Approach-2) Baseline Approach
Figure 4.35: (Baseline vs. Relative) Approach-2 Approach
Figure 4.36: Variation of deviation in approaches on merging

Figure 4.37: Comparison of Entropy among Mutual Information Based Approaches
Figure 4.38: Variation of entropy in approaches on merging
Chapter 5

Conclusion

5.1 Summary

Algorithms that seek to generate all low-variance overlapping biclusters from a real-valued datasets tend to output a very large number of biclusters satisfying the specified size and variance thresholds. Many of these biclusters overlap significantly and can be merged to form larger biclusters for a very little increase in the variance of the resulting bicluster. This type of merging also provides a summarization of all the originally generated biclusters in terms of a few larger biclusters that subsume a large number of similar biclusters. This summarization provides a much better insight into the structures embedded in the dataset. We have presented a methodology for generating summary biclusters such that the increase in variance due to merging is minimized. We have proposed and analyzed three different strategies based on the statistical properties of the data for summarizing the low-variance biclusters generated from the real-valued datasets. The approach based on relative deviation and mutual information methodologies generates good quality summary biclusters by successive merges as long as the specified variance bound is controlled. The former approach is shown to be more effective than the latter, in terms of both the information content and the variance of a summary bicluster after a few merges. When only the variance of the merged biclusters is considered, the first approach performs much better than the second, and when the information content, measured by entropy, of the bicluster is taken into consideration, both the approaches perform equally well in producing good quality summary biclusters after merges for every specified threshold level.

Both the approaches clearly outperform the baseline approach considered in terms of the increase in the variance of the resulting biclusters. This validates our belief that measures based on cardinality and sets of the attributes alone do not measure the closeness of two real-valued biclusters as well as it can be
done by our proposed measures. The results emphasize the need to use measures that take into account the properties of the data in biclusters for making the decisions for merging into summaries. Our approaches use the mutual information and/or the standard deviation for making merging decisions and thus generate very good summaries for the numerous real-valued biclusters.
5.2 Future Work

The work presented in this thesis allows for many variations in the scenario that can make the problem even more interesting. A few of these are listed below,

- The algorithm presented in this thesis computes the similarity between every concept to every other concept, as long as they overlap over a bare minimal set of objects and attributes. If we were to relate biclusters of real valued data with FCA, and further relate the biclusters as we relate concepts in a concept lattice, we could employ our measure over biclusters that are structurally related, hence reducing the computational cost, and also produce better quality results.

- The threshold parameter defined in both the approaches does not have an upper bound. It would be interesting to gain an insight to get closer to this threshold value by looking into the data, rather than by progressively incrementing the threshold to decide on the upper bound for merging where we should stop merging the summary biclusters.
Chapter 6

References


F. Alqadah, R. Bhatnagar."Similarity Measures in Formal Concept Analysis",2009


P. Resnik."Semantic Similarity in a Taxonomy: An Information-Based Mea-

71


[Chen et al., 2008] Chen et al., "Integration of external signaling pathways with the core transcriptional network in embryonic stem cells", in Cell, pp. 11061117, 2008.