Understanding the Cheng and Church Biclustering Algorithm

A capstone project report submitted in partial fulfillment of the requirement for

the degree of

Masters of Engineering

In the Graduate Program,

College of Engineering & Applied Science

04/18/2019

Nithyasri Babu

Contents

Abstract

Clustering is an unsupervised learning method that groups similar data points in a dataset. Most commonly known clustering algorithms such as K-Means, DBSCAN or hierarchical clustering algorithms generally form clusters with only one dimension at a time. Biclustering algorithms are used to form clusters with rows and columns simultaneously. One of the first developed and widely used algorithms for biclustering is the Cheng and Church Biclustering Algorithm. This algorithm was used to find patterns on gene expression data that indicated with a set of genes that are expressed for various conditions. The objective of this project is to understand how the Cheng and Church biclustering algorithm works. We will discuss an implementation of the algorithm in the MTBA library, a MATLAB Toolbox for Biclustering Analysis that was developed in the year 2013 and used for running different biclustering algorithms. We will explore the different input parameters that are required for the Cheng and Church algorithm and how we can modify them based on the dataset to get proper results. Furthermore, we will discuss the results from two datasets, a synthetic dataset and an Abalone dataset using summary metrics. This will include measuring the bicluster dimensions, mean, standard deviation and coefficient of variation for the values extracted using the biclustering algorithm. We will also discuss the resulting biclusters formed for different input parameters and how changing the input parameters will affect the results.

Introduction

Clustering is the most common method used for unsupervised learning in the field of data mining. The term clustering means to group the data points in such a way that the points in the same group are more like each other than the points in a different group. This is mostly measured using a distance function such as Euclidian distance function or a vector dot product. The main goal is to minimize the distance of data points within a given cluster and maximize the distance in between clusters. Over the years, there have been numerous algorithms that were developed for efficient clustering. There are different types of clustering algorithms based on how the clusters are formed.

A few common algorithms that are used widely are K-Means, DBSCAN, Fuzzy clustering, and hierarchical clustering algorithms. While these algorithms are widely used and applied in many fields to incur valuable insights, they can only cluster in a single dimension.

Biclustering algorithms are used to simultaneously cluster both rows and columns. These algorithms are also called Co-clustering or block clustering algorithms. In a biclustering algorithm, the result will contain a subset of rows and columns. This would typically be a sub-matrix formed using the selected rows and columns for the bicluster. It is expected the values of the sub-matrix formed to have similar values overall, row-wise or column-wise within a given cluster. The calculation of similarity differs from algorithm to algorithm.

The algorithm we will be discussing in this project report is called **Cheng and Church Biclustering** algorithm. The implementation used in this project to explore the algorithm was from a library called the **MTBA, MATLAB Toolbox for Biclustering Analysis**.

Cheng and Church Algorithm

Introduction

The Cheng and Church Biclustering algorithm, hereafter referred to as CC, was developed in the year 2000. It was one of the first biclustering algorithms that were developed and generalized for all types of data. Initially, it was used to extract clusters for Gene expression data. The main objective was to identify sets of genes that commonly occurred with sets of conditions. The term bicluster was used because the algorithm simultaneously extracted sets of rows and sets of columns.

One of the main concerns that led to the development of the algorithm was that the existing clustering algorithms only created clusters in which all conditions were given equal importance. The patterns that could be formed with better similarity measures with a subset of important conditions rather than using all conditions were lost in distance calculations.

Another major concern was that the clusters formed using other methods did not extract overlapping clusters. For example, suppose a cluster contained a set of genes for a set of conditions. The other algorithms generally extracted mutually exclusive biclusters that were usually constant. The authors of this method argue that there is a need to allow the overlapping of clusters so that a single gene could be clustered with different sets of conditions.

This led to developing the CC algorithm that would extract biclusters that would contain a subset of rows and columns that would have a high similarity score within the bicluster. Instead of considering the similarity measure to be a distance function in the CC algorithm, they choose to maximize the coherence within a the bicluster using a Maximum Mean Squared Residual Score that is provided to the algorithm. This value controls the variance of values within a bicluster formed using CC.

Given a dataset containing ‘M’ data points with ‘N’ feature values, a bicluster is formed such that a set of rows ‘I’ and a set of columns ‘J’ are selected. Let i Є I and j Є J, the residue of a single element in the submatrix bicluster that is formed is

$$a_{ij} - a_{iJ} - a_{Ij} + a_{IJ}$$

where the individual values are defined as below:

- a_{ij} – value at the ith row and jth column of the matrix
- a_{iJ} - row mean of the ith row with all selected columns values
- a_{Ij} – column mean of jth column with all selected row values
- a_{IJ} – overall mean of all values of the selected rows and columns

$$a_{iJ} = \frac{1}{|J|} \sum_{j} a_{ij}$$

$$a_{Ij} = \frac{1}{|I|} \sum_{i} a_{ij}$$

$$a_{IJ} = \frac{1}{|I||J|} \sum_{i,j} a_{ij}$$

Mean Squared Residual Score H (I, J) for a bicluster of rows I and columns J is defined as

$$H(I, J) = \frac{1}{|I||J|} \sum_{i \in I, j \in J} \left(a_{ij} - a_{iJ} - a_{Ij} + a_{IJ}\right)^2$$

The algorithm inputs a value delta (δ) that is to be the maximum residue score allowed for biclusters that are formed. This input value is used to control which rows and columns can be added to or removed from the bicluster formed. More about this is explained further in the report.

Input Parameters for CC

1. **Delta (δ)**

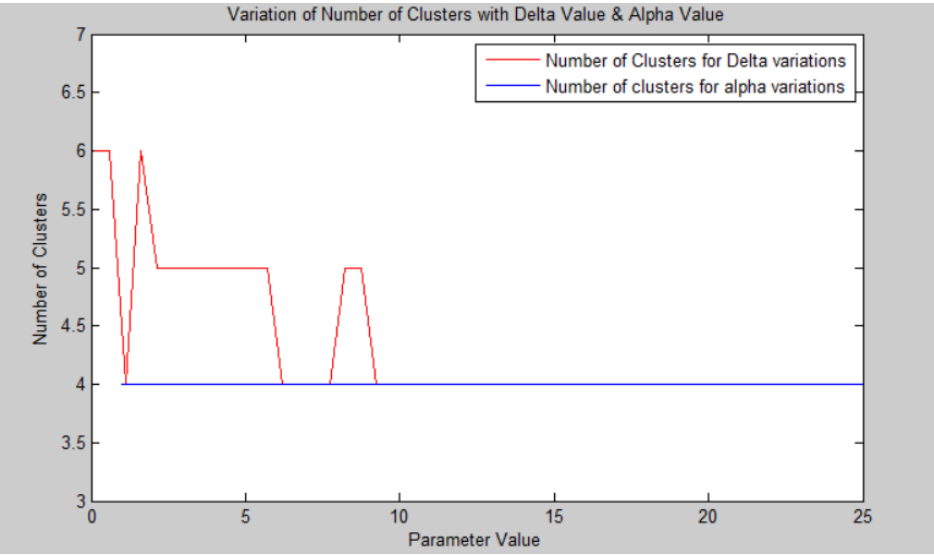
   As discussed earlier, the CC algorithm obtain clusters by controlling the variance within the bicluster. The algorithm inputs a value delta that acts as the maximum mean squared residue score that will be accepted for a bicluster that is formed. Delta is the only value that affects the quality and number of clusters formed from a dataset. Delta is required to be above 0.

2. **Alpha (α)**

   Alpha is the learning rate that is used to accelerate the cluster formation for large datasets. This is optional and usually has a default value of 1. Alpha does not affect the results of the algorithm. The value is used during the step of multiple node deletion that is discussed later.

3. **Number of Clusters**

   Though the algorithm inputs a value for the number of clusters, it is not always required that so many clusters be formed. This value indicates the maximum number of clusters to be formed. If at a point of the execution there are no new clusters, but the input value is still higher than the number of clusters formed so far, the algorithm stops anyway.



Plot to shows how delta δ and alpha α affect the number of clusters formed

How CC Works

It is interesting to note that the algorithm works in such a way that the largest cluster is formed first and in each of the next iterations smaller clusters are formed. Initially, each matrix element is considered a sub-matrix and could be the bicluster.

The CC algorithm consists of 4 steps that are repeated until all biclusters are found. Each of the steps is explained subsequently.

1. Multiple Node Deletion
2. Single Node Deletion
3. Node Addition
4. Mask Discovered Biclusters

Multiple Node Deletion:

This step comes before Single Node Deletion. The main goal of Multiple Node Deletion is to decrease the size of the input matrix significantly so that the next step can be performed on a lower number of nodes. This improves the efficiency of the algorithm significantly.

Suppose we have an input matrix 'M' with a set of rows 'I' and columns 'J', $\delta >= 0$ and $\alpha > 1$.

If H (I, J) <= δ, we do not have to delete any node. Else we compute the mean residue scores d(i) for each row i Є I and remove all rows with d(i) > α * H (I, J).

$$d(i) = \frac{1}{|J|} \sum_{j \in J} \left(a_{ij} - a_{iJ} - a_{Ij} + a_{IJ}\right)^2$$

Then we compute mean residue score d(j) for each column j Є J for the remaining rows and remove all columns with d(j) > α * H (I, J).

$$d(j) = \frac{1}{|I|} \sum_{i \in I} \left(a_{ij} - a_{iJ} - a_{Ij} + a_{IJ}\right)^2$$

Single Node Deletion:

With the input matrix size significantly reduced, we now remove one node at a time rather than multiple nodes. To form a bicluster that has a lower residue score than delta, either a row or column is removed from the input data matrix. If H (I, J) <= δ for an input matrix we can skip the deletion step, else we compute the mean residue scores d(i) for each row i Є I and d(j) for each column j Є J in the input matrix. A single row or column with the highest score is removed from the data matrix to get a lower residue score. This is continued until the input matrix H (I, J) <= δ.

Node Addition:

With the Multiple and Single Node deletions, the matrix may not be maximal. This means that there may be certain rows and columns that may be added to the input matrix such that the residue score is still below Delta. This is done by first computing the mean residue scores for columns not in the input matrix. Those columns that have a score less than previous H (I, J) are then added to the input matrix. The same is repeated for all rows that are not in the input matrix. For all rows still not in 'I', add the inverse if

$$\frac{1}{|J|}\sum_{j\in J}(-a_{ij}+a_{iJ}-a_{Ij}+a_{IJ})^2 \leq H(I,J)$$

These rows can be considered as co-regulated rows and cannot be added to the original data matrix as it will negate the row score.

Mask Discovered Biclusters:

At every iteration, if the input matrix is the original data matrix, we will get the same results. To prevent this, we mask the discovered bicluster rows at the end of each iteration. The masking is done by substituting the values of the discovered submatrix with random values. This step facilitates the discovery of other clusters.

Implementation

In this project, the MTBA library, MATLAB Toolbox for Biclustering Analysis, was used to study the CC algorithm. I was released in the year 2013 by the IIT Kanpur. This library contains implementations of various biclustering algorithm implementations of which some of them are mentioned below:

- Cheng and Church
- Bipartite Spectral Graph Partitioning
- Spectral Biclustering
- Bayesian Biclustering

The CC algorithm takes the same inputs that we have mentioned earlier and implements the algorithm explained in the previous section. The results of the biclustering algorithm is a structure containing 4 attributes that allow us to extract the biclusters formed.

- **RowxNum –**
    - Logical matrix of size = Total number of Rows x Number of Clusters
    - If RowxNum (i, j) = 1, then Row i belongs to j^{th} Cluster
- **NumxCol –**
    - Logical matrix of size = Number of Clusters x Total number of Rows
    - If NumxCol (i, j) = 1, then Column j belongs to i^{th} Cluster
- **ClusterNo –**
    - Number of Clusters formed in each dimension
- **Clust –**
    - Column vector containing structures = number of clusters formed
    - Clust (i) contains 2 values
        - rows – Row indexes in cluster i
        - cols – Column indexes in cluster i

Synthetic Dataset Results

A synthetic dataset of the dimensions 24 x 23was initially used to understand the workings of the MTBA library and the implementation of the CC algorithm. There was no need to use an alpha since the data is very small. The biclusters formed using the CC algorithm were expected to be similar and to have low variance within the bicluster. To choose the proper delta value was critical to get clusters as expected.

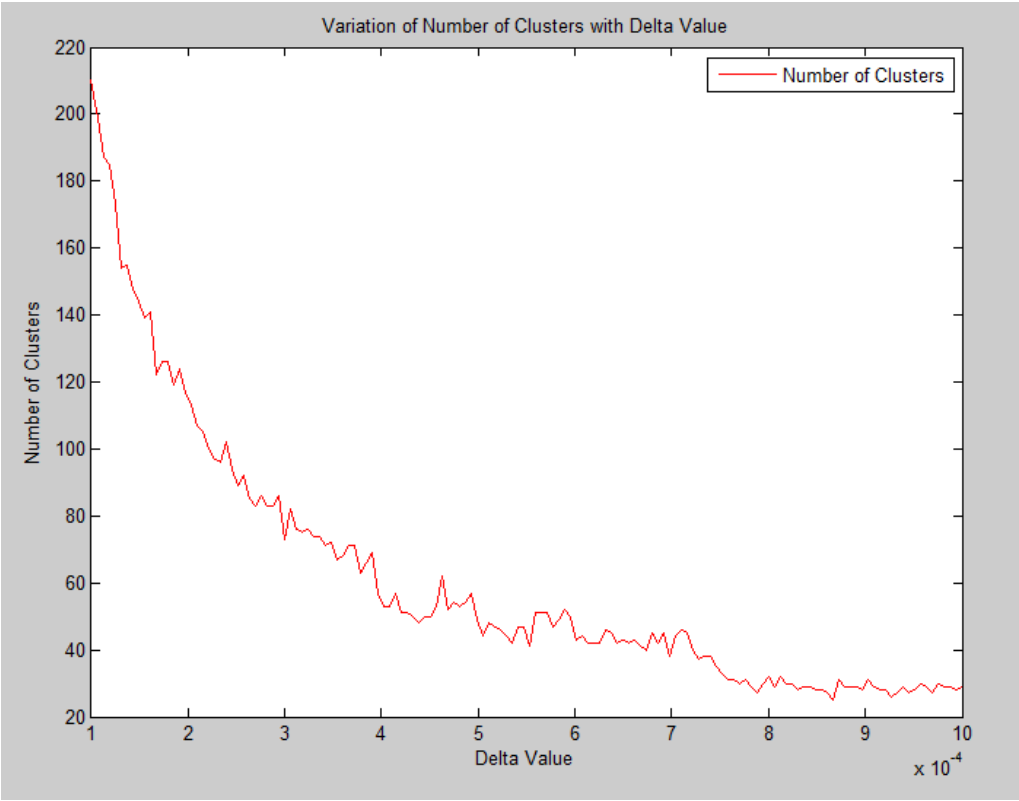
A range of delta values was given to check how the delta value affects the number of clusters formed and the values within the clusters were assessed based on metrics like Max, Min, Mean, Standard Deviation and Variance.

The biclusters formed with input delta ranging from 1.2 to 10 were not satisfactory as the variance within the bicluster was high and the delta value seemed to be much larger for the range of values in the input matrix. Much lower input deltas of range 0.0001 to 0.001 were tried. This led to better results in terms of cluster values. The clusters had more similar values and lower mean residual score.

Abalone Dataset Results

The abalone dataset consists of age values predicted based on physical measurements of abalones. The dataset is of dimensions 4177 x 8 and is continuous numerical dataset. The aim here was to use the abalone's physical measurements to execute CC biclustering algorithm and to extract the ages for the rows selected in a bicluster. It is expected that they will be similar ages and will have low variance.

The Input parameters chosen for the data set was in the range 0.001 to 0.01 as most of the values are less than 1 in the input matrix. The lower the delta value the more biclusters formed.



The metrics for this dataset is the ages attribute that is not included in the clustering process. For each bicluster formed the ages are extracted from of each row belonging to that cluster. The metrics include Max, Min, Mean, Standard Deviation and Coefficient of Variation.

To compare the results of delta values, let us consider biclusters formed with input 2 input deltas.

δ = 0.005: NumClust = 11:

Clusters	NumRows	NumColun	Rows	Columns	MaxAges	MinAges	MeanAges	AgesStD	AgesCV
C1	828	8	[1;4;9;13;1	[1 2 3 4 5 6 7 8]	26	6	10.64372	2.907668	0.273182
C2	930	8	[6;17;51;1	[1 2 3 4 5 6 7 8]	21	5	8.56129	2.20846	0.257959
C3	672	8	[7;8;10;23	[1 2 3 4 5 6 7 8]	23	6	11.07292	3.01052	0.271881
C4	367	8	[33;74;84;	[1 2 3 4 5 6 7 8]	23	7	11.44414	2.836436	0.24785
C5	282	8	[3;11;14;1	[1 2 3 4 5 6 7 8]	22	6	10.34752	2.702275	0.261152
C6	880	6	[2;5;12;19	[2 3 4 6 7 8]	29	2	8.519318	3.652679	0.428752
C7	105	8	[129;169;1	[1 2 3 4 5 6 7 8]	22	8	11.98095	2.605397	0.217462
C8	85	6	[34;130;16	[2 3 4 6 7 8]	24	9	12.21176	3.262683	0.267175
C9	17	6	[166;359;1	[2 3 4 5 7 8]	23	9	12.94118	3.151797	0.243548
C10	9	4	[82;164;16	[2 3 4 7]	27	8	15.22222	5.761462	0.37849
C11	2	5	[237;2052	[2 3 6 7 8]	8	1	4.5	4.949747	1.099944

δ = 0.0005: NumClust = 43: (Cropped picture below)

Clusters	NumRows	NumColun	Rows	Columns	MaxAges	MinAges	MeanAges	AgesStD	AgesCV
C1	157	8	[28;29;30;	[1 2 3 4 5 6	19	7	10.23567	2.361931	0.230755
C2	126	8	[23;26;27;	[1 2 3 4 5 6	19	7	10.47619	2.378955	0.227082
C3	981	5	[1;3;4;8;9;	[2 3 4 7 8]	23	6	10.49745	2.702086	0.257404
C4	393	8	[6;17;101;	[1 2 3 4 5 6	19	5	7.569975	1.720469	0.227275
C5	90	6	[37;94;188	[1 2 3 4 7 8	21	8	11.32222	2.543234	0.224623
C6	243	8	[51;422;53	[1 2 3 4 5 6	18	6	8.736626	1.622599	0.185724
C7	93	8	[163;186;2	[1 2 3 4 5 6	17	8	10.46237	1.710321	0.163474
C8	142	8	[11;15;18;	[1 2 3 4 5 6	18	6	10.57746	2.719313	0.257086
C9	307	5	[74;85;87;	[2 3 4 7 8]	23	6	11.71661	2.886947	0.246398
C10	352	8	[5;22;43;4	[1 2 3 4 5 6	13	3	6.053977	1.420219	0.234593
C11	95	8	[2;19;21;4	[1 2 3 4 5 6	15	4	8.694737	1.951942	0.224497
C12	70	5	[32;35;185	[1 2 3 4 7]	21	8	12.6	2.999517	0.238057

Points to note from the 2 summaries above:

1. The dimensions of the biclusters formed for a lower delta value is high while for a lower value more clusters are formed with a smaller dimension.
2. The distance between Max and Min ages of any given bicluster reduces with the decrease in the delta value.
3. The Standard Deviation within the clusters goes down with the decrease in delta value.

All of the above is in line with the concepts discussed previously in the report. With the decrease of Delta, the distance between points in a bicluster value goes down leading to closer knit biclusters.

Coefficient of Variation (CV):

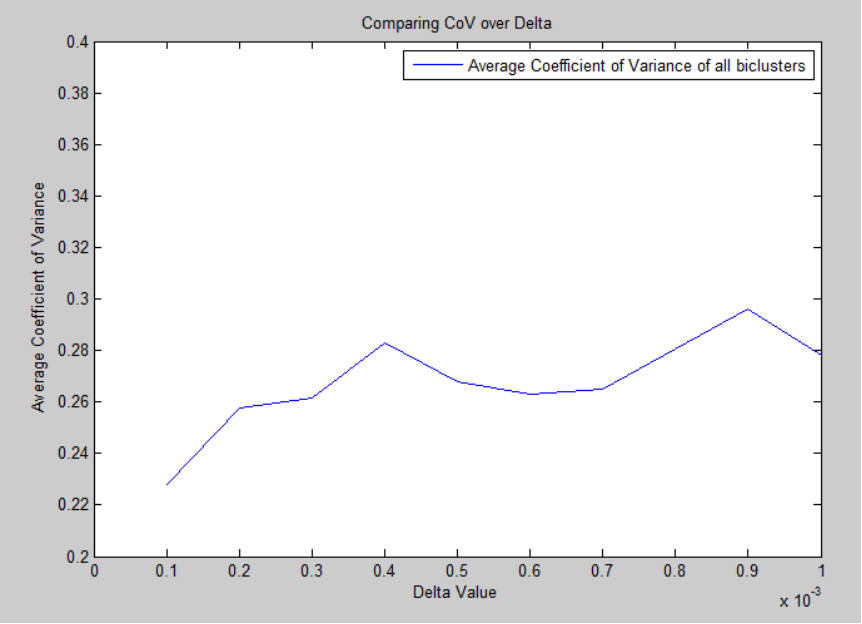
Coefficient of Variation is a statistical measure that represents the dispersion of data around the mean. It is useful to compare the degree of variation from one data series to another irrespective of the difference between the 2 data series.

It is calculated using the formula:

$$\text{Coefficient of Variation (CV)} = \frac{\text{Standard Deviation } (\sigma)}{\text{Mean } (\mu)}$$

Using this metric, we will now compare how the variation in delta affects the biclusters that are formed. We expect the CV to increase with the increase of the delta.

DeltaValue	NumClusters	AverageCoV
0.0001	210	0.22807425
0.0002	110	0.25752575
0.0003	89	0.2615297
0.0004	58	0.28293902
0.0005	43	0.26787507
0.0006	43	0.26321994
0.0007	39	0.26510446
0.0008	30	0.28030615
0.0009	31	0.29633683
0.001	29	0.27793746
0.002	19	0.27772385
0.003	15	0.25419938
0.004	12	0.31021696
0.005	11	0.35885414
0.006	10	0.30424808
0.007	11	0.28097422
0.008	10	0.27855072
0.009	11	0.2904704
0.01	10	0.29931247
0.05	3	0.28387494
0.1	3	0.26435212



As expected, the value of CV increases gradually with an increase in delta value indicating that the variance is higher for a higher value of delta. Code, plots and results for the information provided in this report can be found in https://github.com/NithyasriBabu/ChengChurchBiclustering

Bibliography

1. Cheng, Y. & Church, G. M. Biclustering of expression data. *ISMB* **8**, 93–103 (2000).
2. J. K. Gupta, S. Singh, and N. K. Verma, MTBA: MATLAB Toolbox for Biclustering Analysis, IEEE Workshop on Computational Intelligence: Theories, Applications, and Future Directions, IIT Kanpur, India, pp. 94-97, July 2013
3. UCI Machine Learning Repository, http://archive.ics.uci.edu/ml/machine-learning-databases/abalone/
4. http://www.kemaleren.com/post/cheng-and-church/