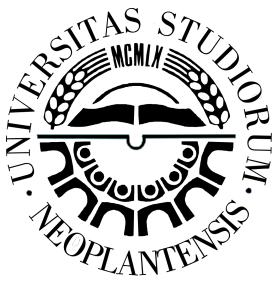

University of Novi Sad

MASTER THESIS

Clustering Gene Expression Data - Comprehensive Evaluation

Author:
Nataša Topić

Supervisor:
Tatjana Lončar-Turukalo, PhD

Faculty of Sciences
Department of Mathematics and Informatics

June, 2021

"Go down deep enough into anything and you will find mathematics."
Dean Schlicter

Abstract

Linking phenotype disease characteristics with biological data at cellular and sub cellular level, has the potential to uncover underlying causes of multiple diseases, including cancer. Gene expression relates to the process by which cells transfer their genetic information from deoxyribonucleic acid into a protein molecule with biological activity. Levels of gene expressions are measured most commonly using two microarray technologies (Affymetrix and cDNA) to analyze gene functions, regulatory mechanisms and their changes in disease. In this work 33 gene expression data sets of tissues with different types of cancer is analyzed using unsupervised clustering approaches. The thesis evaluates the performance of K-means, Spectral Co-Clustering and Spectral Bi-Clustering approaches, exploring the potentials of simultaneous clustering of samples and genes. Separate performance analysis was done on data sets from Affymetrix and cDNA chip platforms to examine the possible influence of the microarray technology.

Based on two validation criteria (Adjusted Rand Index and Silhouette Index) there is a noticeable difference between the results obtained over two chip platforms. The evaluated indices are commonly higher in case of Affymetrix data sets. Since a large collection of data sets and different chip technologies are used, no universal recommendation can be made. In some data sets partitions with high ARI, i.e. aligned with the ground truth labels, are achieved already with K-means clustering, while some data sets have very heterogeneous samples within classes that hampers clustering based on sample similarity metrics. Biclustering approaches offer possibility of grouping both samples and genes, and potential to uncover some clustering patterns of the genes typical of the certain cancer subtypes.

Acknowledgements

I would like to express my sincere gratitude to my supervisor prof. Tatjana Lončar-Turukalo for support, directions and insightful comments. Her knowledge and experience have encouraged me during this research.

I am also very grateful to my friends for the wonderful times we shared during studies.

Finally, I would especially like to thank my family for their love, unfailing support and help. I am grateful to my brother for always being there for me. I am forever indebted to my parents for their encouragement, without which I would never have enjoyed so many opportunities that have made me who I am. Nothing would be possible without my family, and I dedicate this thesis to them.

Contents

List of Tables

List of Figures

Chapter 1

Motivation and Introduction

1.1 Motivation

Some cancers are a preventable disease if they are diagnosed in the early stage. There are various types of cancer, affecting blood, liver, lung, brain and other tissues. Since some types of cancer can be better treated with certain drugs than others, gene expression profiling can allow the development of more appropriate personalized treatment plans for patients.

Gene expression relates to the process by which cells transfer their genetic information in deoxyribonucleic acid (DNA) into a protein molecule with biological activity through transcription and translation in the life process [1]. Levels of gene expressions are measured under specific experimental conditions to analyze cancer subtypes, gene functions and regulatory mechanisms. In order to collect large gene expression data sets microarray technology is used. Clustering is commonly used to identify genes with similar expressions across different conditions or for grouping patients (samples) based on the similar expressions of genes. However, traditional clustering analysis can not discover the gene expression pattern visible in only a subset of samples. The aim of this thesis is to explore methods that enable clustering of both genes and samples at the same time. This would result in identification of groups of samples whose subset of genes exhibits similar expression patterns and groups of genes that share similar expression patterns in a subset of samples. This information can be used to link phenotype disease characteristics with gene expression profiles, possibly leading to the discovery of new cancer subtypes. Biclustering [2] is used to find this local structure inside the gene expression matrix. This approach is well suited to gene expression data because genes are not related across all samples, and vice versa. Biclustering algorithms belong to a distinct class of clustering algorithms that perform simultaneous clustering of both rows and columns of the data matrix.

1.2 Introduction

1.2.1 Clustering

Clustering is an unsupervised method for grouping n objects with m features into k sets of disjoint groups called clusters, where similarity between objects within a cluster is maximized and similarity between objects in separate clusters is minimized. One of the major points in analysis of gene expression data is a need to cluster genes as well as samples. Biclustering is commonly used to achieve clustering of both column-wise (features) and row-wise (samples) at the same time.

1.2.2 Biclustering

Biclustering (co-clustering, block clustering, two-mode clustering) is a data mining technique which allows simultaneous clustering of the rows and columns of a data matrix. Unlike clustering methods, where all features are considered, biclustering methods can find local patterns in only a subset of features. Consider a data matrix $A \in \mathbf{R}^{n \times m}$, with a set of rows R, and a set of columns C, and where a_{ij} represents the element in i^{th} row and j^{th} column. A bicluster A_q is defined as $A_q = (R_q, C_q)$ with $R_q \subseteq R$ and $C_q \subseteq C$.

A number of biclustering algorithms have been proposed so far, but one of the earliest and most cited formulations was proposed by Hartigan [3]. There are many diverse areas where biclustering can be applied, and a typical one is gene expression analysis. With a microarray technology the mRNA levels can be measured in a huge number of genes, and produce a data set of gene expression profiles that is represented with a large data matrix. Columns of the data matrix represent genes, which usually outnumber rows representing different samples (patient tissue). In analysis of gene expression data, we can identify subsets of genes whose expression levels exhibit a coherent pattern under a subset of samples.

Bicluster Type

Different biclustering algorithms search for different types of biclusters. In the following, four major classes are presented:
1. Biclusters with constant values.
2. Biclusters with constant values on rows or columns.
3. Biclusters with coherent values.
4. Biclusters with coherent evolutions.

1. Biclusters with Constant Values

Natural way for a biclustering algorithm to find constant biclusters is to try to reorder columns and rows of the data matrix in order to group together columns and rows with similar values and form biclusters. Bicluster is a *perfect* constant bicluster if it is a submatrix of a data matrix $A_q = (R_q, C_q)$ where all values a_{ij} are equal, for all $i \in R_q$ and $j \in C_q$: $a_{ij} = \mu$ (Figure 1.1). This approach gives good results only in absence of noise in data. In

practice noise free data are rare, as e.g. the measurement process itself usually introduces some noise, so more sophisticated methods have to be used. Because real data are usually corrupted by noise, a_{ij} can be represented as $\eta_{ij} + \mu$, where η is the noise term. Both identification and evaluation of constant biclusters is based on the analysis of variance. After splitting the original data matrix into a set of biclusters, Hartgan's [3] algorithm, known as Block Clustering, uses variance to evaluate the quality of each bicluster (R_q, C_q):

$$VAR(R_q, C_q) = \sum_{i \in R, j \in C} (a_{ij} - a_{RC})^2, \tag{1.1}$$

where a_{RC} represents the mean of all elements in the bicluster. So, a *perfect* bicluster is a matrix with variance zero. An ideal bicluster might be one with just one element a_{ij}, to avoid this Hartigan assumes that there are k biclusters within the data matrix and after finding these k biclusters, the algorithm terminates. For evaluation in this case the overall variance of k biclusters is used:

$$VAR(R_q, C_q)_k = \sum_{l=1}^{k} \sum_{i \in R, j \in C} (a_{ij} - a_{RC})^2. \tag{1.2}$$

1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0

Figure 1.1: An example of Perfect Constant Bicluster

2. Biclusters with Constant Values on Rows or Columns

Perfect bicluster with constant rows is a submatrix of a data matrix $A_q = (R_q, C_q)$ (Figure 1.2) where all the values within the bicluster can be obtained using:

$$\begin{aligned} a_{ij} &= \mu + \alpha_i \\ a_{ij} &= \mu \times \alpha_i \end{aligned} \tag{1.3}$$

where μ is the typical value within the bicluster and α_i is the adjustment for row $i \in R_q$.

Perfect bicluster with constant columns is a submatrix of a data matrix $A_q = (R_q, C_q)$ (Figure 1.3) where all the values within the bicluster can be obtained using:

$$\begin{aligned} a_{ij} &= \mu + \beta_j \\ a_{ij} &= \mu \times \beta_j \end{aligned} \tag{1.4}$$

where μ is the typical value within the bicluster and β_j is the adjustment for column $j \in C_q$.

In identification of these biclusters the variance or similarities between the rows and columns of the data matrix can not be used, but instead normalization of the rows or the columns. The normalization is done using row

mean and column mean. In this way biclusters will be transformed (Figures 1.2 and 1.3) into constant biclusters (Figure 1.1). After this normalization step biclustering algorithm can be used.

1.0	1.0	1.0	1.0
2.0	2.0	2.0	2.0
3.0	3.0	3.0	3.0
4.0	4.0	4.0	4.0

Figure 1.2: An example of Bicluster with Constant Rows

1.0	2.0	3.0	4.0
1.0	2.0	3.0	4.0
1.0	2.0	3.0	4.0
1.0	2.0	3.0	4.0

Figure 1.3: An example of Bicluster with Constant Columns

3. Biclusters with coherent values

Bicluster with coherent values based on additive or multiplicative model is a subset of rows and a subset of columns, whose values a_{ij} are predicted using:

$$\begin{aligned} a_{ij} &= \mu + \alpha_i + \beta_j \\ a_{ij} &= \mu \times \alpha_i \times \beta_j \end{aligned} \tag{1.5}$$

where μ is the typical value within the bicluster, α_i is the adjustment for row $i \in R_q$ and β_j is the adjustment for column $j \in C_q$. The bicluster in Figure 1.4 is an example of biclusters with coherent values on rows and columns. Values of this bicluster can be described using an additive model. Figure 1.5 represents another biclustering approach, this is an example of a bicluster with coherent values on both rows and columns based on a multiplicative model. If we consider Equations (1.3) and (1.4) we can conclude that these two are special cases of Equation (1.5) where $\alpha_i = 0$ and $\beta_j = 0$ in additive case, or $\alpha_i = 1$ and $\beta_j = 1$ in multiplicative case, respectively.

1.0	2.0	5.0	0.0
2.0	3.0	6.0	1.0
4.0	5.0	8.0	3.0
5.0	6.0	9.0	4.0

Figure 1.4: An example of Bicluster with Coherent Values (addictive model)

1.0	2.0	0.5	1.5
2.0	4.0	1.0	3.0
4.0	8.0	2.0	6.0
3.0	6.0	1.5	4.5

Figure 1.5: An example of Bicluster with Coherent Values (multiplicative model)

A few biclustering algorithms assume either additive or multiplicative models. One of them is the biclustering algorithm introduced by Kluger et al. [4]. They looked for a hidden checkerboard structure in the data matrix. That kind of structure will be more obvious if the particular normalization is used first. After normalization it is assumed that values within the bicluster can be obtained using the multiplicative model (Equation(1.5)). This algorithm will be explained in detail in Chapter 2.

4. Biclusters with Coherent Evolutions

In this class, biclustering algorithms try to find biclusters with coherent behaviors regaradless of the exact numeric values in the data matrix. Elements of the data matrix are viewed as symbolic values. Problem of finding coherent evolutions can be observed on the entire bicluster (Figure 1.6), on the rows (Figure 1.7) or on the columns of bicluster (Figure 1.8).

S1	S1	S1	S1
S1	S1	S1	S1
S1	S1	S1	S1
S1	S1	S1	S1

Figure 1.6: An example of Bicluster with Overall Coherent Evolution

S1	S1	S1	S1
S2	S2	S2	S2
S3	S3	S3	S3
S4	S4	S4	S4

Figure 1.7: An example of Bicluster with Coherent Evolution on the Rows

S1	S2	S3	S4
S1	S2	S3	S4
S1	S2	S3	S4
S1	S2	S3	S4

Figure 1.8: An example of Bicluster with Coherent Evolution on the Columns

Another bicluster formulation is given by Ben-Dor et al.[5] where they represent bicluster as an order-preserving submatrix (OPSM). The bicluster is defined as a submatrix whose row values induce the same linear ordering across the columns. For a submatrix we can say that it is order-preserving if there is a permutation of its columns under which the sequence of values in every row is strictly increasing. In Figure 1.9 we can see one example of this kind of matrix.

70	13	19	10
49	40	49	35
40	20	27	15
90	15	20	12

Figure 1.9: An example of Order Preserving Sub-Matrix (OPSM)

Bicluster Structure

There are two classes of biclustering, based on its structure:
1. matrix has only one bicluster (Figure 1.10 (a))
2. matrix has r biclusters $A = \{A_1, ..., A_q, ..., A_r\}$.

In the second case there exists several subclasses (Figure 1.10 (b), (c), (d), (e), (f), (g), (h), (i)):

(b) Exclusive row and column biclusters (rectangular diagonal blocks after row and column reordering).
(c) Nonoverlapping biclusters with checkerboard structure.
(d) Exclusive-rows biclusters.
(e) Exclusive-columns biclusters.
(f) Nonoverlapping biclusters with tree structure.
(g) Nonoverlapping nonexclusive biclusters.
(h) Overlapping biclusters with hierarchical structure.
(i) Arbitrarily positioned overlapping biclusters.

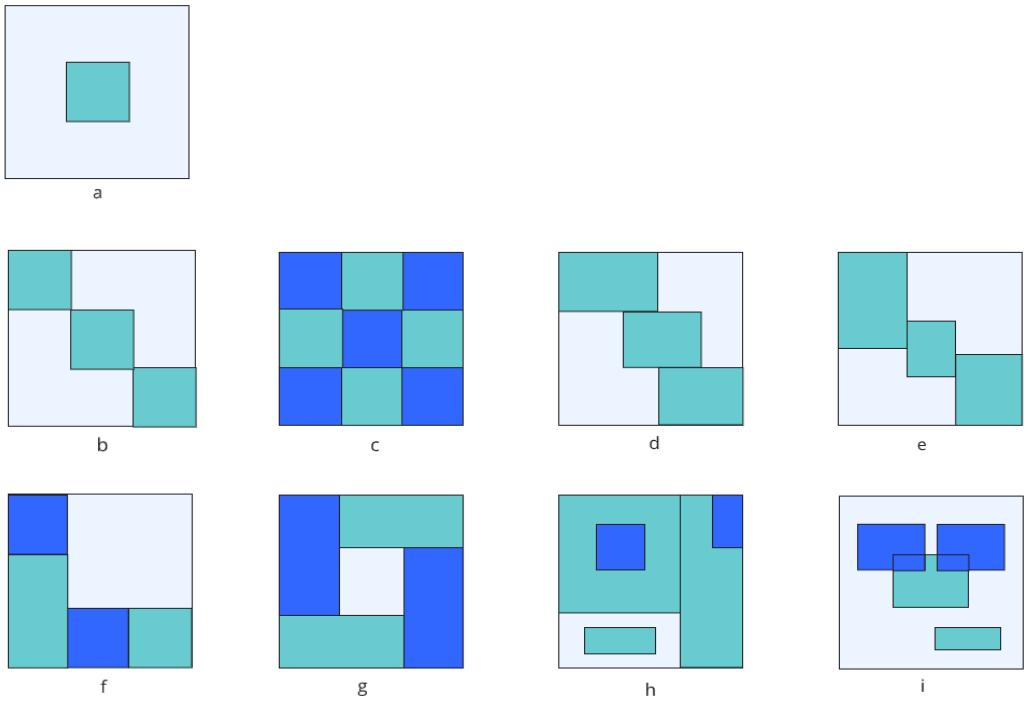

Figure 1.10: Bicluster structure. (a) Single bicluster, (b) exclusive row and column biclusters, (c) checkerboard structure, (d) exclusive rows biclusters, (e) exclusive columns biclusters, (f) nonoverlapping biclusters with tree structure, (g) nonoverlapping nonexclusive biclusters, (h) overlapping biclusters with hierarchical structure, and (i) arbitrarily positioned overlapping biclusters

One of the first approaches when extracting knowledge from gene expression data is to reorder rows and columns of the data matrix such that similar rows and similar columns are grouped together. In that way subsets of rows and subsets of columns with similar expression value are obtained. Ideal case of this reordering can be seen in Figure 1.10 (b). In this figure we can see that every row and every column in the data matrix belongs exclusively to one of the k biclusters ($k = 3$).

Previously mentioned ideal reordering is very rare in real data. Thus, next, we can consider case where rows and columns may belong to more than one bicluster. Assumption here is a checkerboard structure in the data matrix (see Figure 1.10 (c)). With this kind of reordering we get k nonoverlapping and nonexclusive biclusters. Every row and every column of the data matrix will belong to exactly k biclusters. The algorithm defined in Kluger et al. [4] that will be described in Chapter 2, assumes this bicluster

structure.

Next structure of bicluster is where rows can only belong to one bicluster, but columns can belong to several. Example of this structure can be seen in Figure 1.10 (d). If the algorithm uses the opposite orientation of the data matrix we get a different structure (see Figure 1.10 (e)). This structure assumes exclusive-columns biclusters, i.e. every column can only belong to one bicluster, but rows can belong to several.

Bicluster is called exhaustive if every row and every column belongs to at least one bicluster. Examples of this type are presented in Figure 1.10 (b), (c), (d), (e). Nonexhaustive biclusters allow that some rows and columns are not included in any bicluster. Additionally, there are biclusters that allow overlapping. This kind of structure allows that particular pair (row, column) belongs to more than one bicluster (Figure 1.10 (i)). These two properties exhaustion and overlapping are very common in real data (Figure 1.10 (i)).

Biclustering Algorithms

Biclustering algorithms aim to identify one bicluster or a given number of biclusters. There are few biclustering approaches: discover one bicluster at a time, discover one set of biclusters at a time, and discover all biclusters at the same time (simultaneous bicluster identification). Algorithms that will be discussed and used in this thesis discover all biclusters at the same time.

All biclustering algorithms can be grouped into five categories in terms of the techniques used for bicluster identification:

1. Iterative row and column clustering combination - using an iterative procedure row clusters are combined with column clusters by applying clustering algorithms separately to the rows and columns of the data matrix.
2. Divide and conquer - the problem is partitioned into smaller size problems, that are similar to the original and obtained solutions are combined to represent solution of the original problem.
3. Greedy iterative search - make locally optimal results that are chosen in hope that they might be optimal globally.
4. Exhaustive bicluster enumeration - enumerating all the possible biclusters which are hidden in data matrix; to make this search feasible constraints on bicluster dimension must be included.
5. Parametric estimation of distribution- assumption here is that biclusters follow some statistical model and parameters are identified to fit in the best way.

Chapter 2

Materials and Methods

2.1 Data Description

2.1.1 Microarray Technology

Way back in the 20th century, scientists analyzed genes in many ways, mapped them, sequenced them and made mutations in them. Flaw in their analysis is that they studied only one or maybe a few genes at a time, while humans have around 20000 genes and it would take a very long time to investigate each human gene one at a time. This problem is solved with advances in sequencing technology which led to remarkable development in genomics. Genomics [6] is an interdisciplinary field of biology that relates to analyses of thousands of genes or even every gene in an organism, all at once.

Tool for monitoring a huge number of genes in parallel is microarray technology. Microarray technology allows us to collect large gene expression data sets. One microarray experiment involves the hybridization of each mRNA molecule to the DNA template from which it originated. Thousands of DNA samples are used to construct an array. The number of mRNA molecules that are bound to each site in the array gives us the expression level of the various genes. In general, there are the two major types of microarray experiments, cDNA and oligonucleotide (developed by Affymetrix). Both experiments consist of three basic procedures [7]:
1. Chip manufacture - A single microarray is a chip with small dimensions where many molecules of DNA are attached in a usually rectangular grid.
2. Target preparation, labeling and hybridization - First, a test and a control sample of mRNA is reverse-transcribed into cDNA (targets). Next comes labeling using fluorescent dyes and finally hybridization with small fragments of DNA (probes) on the surface of the chip.
3. The scanning process - Chips are scanned to read the signal intensity that is emitted from the labeled and hybridized targets and this provides numeric matrix.
The main difference between two experiments is synthesization of polynucleotide chains. The reason why we should interpret measurements of these two technologies differently is that with cDNA molecules of DNA are hybridized from two tissues and with Affymetrix only one DNA is hybridized.

In cDNA microarray, the gene expression levels are measured as the ratio of the signal from mRNA target sample and the reference sample and Affymetrix data are estimates of the number of mRNA copies in a sample.

2.1.2 Data Sets

Microarray technology evaluates expressions of large number of genes (here features) taken from various subjects (here denoted samples).
In this analysis thirty-three publicly available microarray data sets (Table 2.1) [8] are used. Nineteen of those are Affymetrix data sets, and the other fourteen are cDNA data sets. Besides the type of chip, the data sets differ based on tissue type, the number of available samples (N), the class number (k), the sample distribution per classes, the original dimensionality (m) and dimensionality after feature reduction (d), as presented in Table 2.1. The data sets are available only with reduced dimensionality (from m to d), and modest information is available on the way dimensionality reduction is performed. In most of the data sets, where information could be found, the genes with little or no variance in expression levels across the samples were eliminated. The Affymetrix data are strictly positive, as defined by the explained measurement procedure. The values of gene expression can vary from 0 to a large number here limited to 16.000. Gene expression values lower than 10 are not taken into account. For this reason the possible values range from 10 to 16 000 and range normalization can be applied to scale the expression levels. The pre-processing procedures used before clustering procedures is normalization, as explained in detail in the next section.

Data set can be presented in the form of a matrix $A = \{a_{ij} | 1 \leq i \leq n, 1 \leq j \leq m\}$ with real valued entries (2.1). Columns of the matrix represent genes and rows represent samples. Element in i-th row and j-th column represents measured expression of i-th sample in j-th gene.

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix} \tag{2.1}$$

Dataset	Chip	Tissue	N	k	Sample distribution	m	d
Armstrong-V1	Affy	Blood	72	2	24, 48	12582	1081
Armstrong-V2	Affy	Blood	72	3	24, 20, 28	12582	2194
Bhattacharjee	Affy	Lung	203	5	139, 17, 6, 21, 20	12600	1543
Chowdary	Affy	Breast, Colon	104	2	62, 42	22283	182
Dyrskjot	Affy	Bladder	40	3	9, 20, 11	7129	1203
Golub-V1	Affy	Bone marrow	72	2	47, 25	7129	1877
Golub-V2	Affy	Bone marrow	72	3	38, 9, 25	7129	1877
Gordon	Affy	Lung	181	2	31, 150	12533	1626
Laiho	Affy	Colon	37	2	8, 29	22883	2202
Nutt-V1	Affy	Brain	50	4	14, 7, 14, 15	12625	1377
Nutt-V2	Affy	Brain	28	2	14, 14	12625	1070
Nutt-V3	Affy	Brain	22	2	7, 15	12625	1152
Pomeroy-V1	Affy	Brain	34	2	25, 9	7129	857
Pomeroy-V2	Affy	Brain	42	5	10, 10, 10, 4, 8	7129	1379
Shipp	Affy	Blood	77	2	58, 19	7129	798
Singh	Affy	Prostate	102	2	58, 19	12600	339
West	Affy	Breast	49	2	25, 24	7129	1198
Yeoh-V1	Affy	Bone marrow	248	2	43, 205	12625	2526
Yeoh-V2	Affy	Bone marrow	248	6	15, 27, 64, 20, 79, 43	12625	2526
Alizadeh-V1	cDNA	Blood	42	2	21, 21	4022	1095
Alizadeh-V2	cDNA	Blood	62	3	42, 9, 11	4022	2093
Alizadeh-V3	cDNA	Blood	62	4	21, 21, 9, 11	4022	2093
Bittner	cDNA	Skin	38	2	19, 19	8067	2201
Bredel	cDNA	Brain	50	3	31, 14, 5	41472	1739
Chen	cDNA	Liver	180	2	104, 76	22699	85
Garber	cDNA	Lung	66	4	17, 40, 4, 5	24192	4553
Khan	cDNA	Multi-tissue	83	4	29, 11, 18, 25	6567	1069
Lapointe-V1	cDNA	Prostate	69	3	11, 39, 19	42640	1625
Lapoint-V2	cDNA	Prostate	110	4	11, 39, 19, 41	42640	2496
Liang	cDNA	Brain	37	3	28, 6, 3	24192	1411
Risinger	cDNA	Endometrium	42	4	13, 3, 19, 7	8872	1771
Tomlins-V1	cDNA	Prostate	104	5	27, 20, 32, 13, 12	20000	2315
Tomlins-V2	cDNA	Prostate	92	4	27,20,32,13	20000	1288

Table 2.1: Description of Datasets; N-number of available samples, k-class number, m-the original dimensionality, d-dimensionality after feature reduction

2.2 Normalization

The feature values (gene expression levels) in the data set lie within different dynamic ranges. Transformation of these values would seem to be necessary in those cases where the similarity measure, such as Euclidean distance, is sensitive to differences in scales of the input values. That means that small values would have smaller influence than large ones. Here two feature transformations are analyzed.

All the values in the data sets are numerical, so there is no need for generalizing transformations to fit categorical or ordinal attributes. First transformation that is used is z-score normalization. Formula for calculating z-score is:

$$Z_i = \frac{X_i - \overline{X}}{s} \tag{2.2}$$

where X_i is the original data value, $\overline{X}$ the sample mean and s standard deviation. After this transformation features will have mean value 0, and a variance 1.

In Affymetrix data, since expressions are limited between 0 and 16.000 to normalize the values in different ranges, expression levels for each sample (each raw in data matrix) are divided by the sum of expression levels for that sample. Using the sample-wise normalization (SN), the sum of expression levels in each row is 1, analogous to probability mass function.

Another approach to normalize Affymetrix data is range normalization (RN). In this case normalization is performed column-wise. From each expression level minimum value of the column is subtracted and the new value is divided by the range (max-min) of that column.

2.3 Clustering Techniques

2.3.1 K-means algorithm

K-means is one of the nonhierarchical procedures that are made to group points into a collection of k clusters. This kind of method is used when high dimensional data are analyzed. If set $A = \{a_1, a_2, ..., a_n\}$ of n points is given in d-dimensional space, clustering into k clusters is done so as to minimize the sum of squared distances of each point to its cluster center.

How this algorithm works is described in three steps:
1. Choose k initial centers and cluster each point with the center nearest to it.
2. Find the new cluster centers and replace the old center with a new one.
3. Repeat steps 1. and 2. until centres converge (according to some criterion).

Still needs to be decided how the new center is determined. Common choice is using a centroid - average value over features for all the points in the cluster. Another popular choice is picking clustroid - the most central data point within the cluster, but following Lemma 2.3.1 and Corollary 2.3.1 shows that centroid is the most desirable choice.

Lemma 2.3.1. *Let $\{a_1, a_2, ..., a_n\}$ be a set of points. Sum of squared distances between randomly chosen point x and all datapoints a_i equals the sum of the squared distances to the centroid plus the number of points times the squared distance from the point x to the centroid. That is,*

$$\sum_i |a_i - x|^2 = \sum_i |a_i - c|^2 + n|c - x|^2 \tag{2.3}$$

where $c = \frac{1}{n}\sum_{i=1}^{n} a_i$ *is the centroid of the set of points.*

Proof.

$$\sum_i |a_i - x|^2 = \sum_i |a_i - c + c - x|^2 = \sum_i |a_i - c|^2 + 2(c-x)\sum_i (a_i - c) + n|c-x|^2 \tag{2.4}$$

Since c is centroid, $\sum_i (a_i - c) = 0$. *Thus,* $\sum_i |a_i - x|^2 = \sum_i |a_i - c|^2 + n|c-x|^2$

Corollary 2.3.1. *Let* $\{a_1, a_2, ..., a_n\}$ *be a set of points. Sum of squared distances between randomly chosen point x and all datapoints* a_i *is minimized when x is the centroid, namely* $x = \frac{1}{n}\sum_i a_i$.

In k-means algorithm the user needs to specify a desired number of clusters k as an input. Often, optimal choice is not known a priori, so we need a way to evaluate goodness of fit. Common approach is to run the algorithm with different values k and for each compute sum from Equation 2.3. If an elbow pattern is obtained, i.e. there is a sharp decrease of this score when transitioning from some value k to $k+1$, then $k+1$ is a good choice.

2.3.2 Spectral Clustering

Compared to traditional clustering algorithms such as k-means, spectral clustering often gives better results [9]. It is simple to implement and can be efficiently solved relying on linear algebra tools. Spectral clustering is a technique that exploits the connectivity approach for clustering. In other words, points that are next to each other or connected are assigned to the same cluster. Cluster as defined in this sense is a dense group of points; so that elongated clusters of any shape can be discovered. In this case two points that are far apart, but have dense cloud of data connecting them, will be clustered together, while two much closer points in disconnected neighborhoods will belong to different clusters. Data points scattered in the form of two concentric circles are presented in Figure 2.1. The points belonging to one ring should be clustered together. In the same Figure (b) the result of k-means and (c) spectral clustering are presented. K-means cannot detect this underlying structure because its minimizing the Euclidean distance to the cluster center neglecting the density and shape of point clouds, whereas the results of spectral clustering reveal the correct clusters.

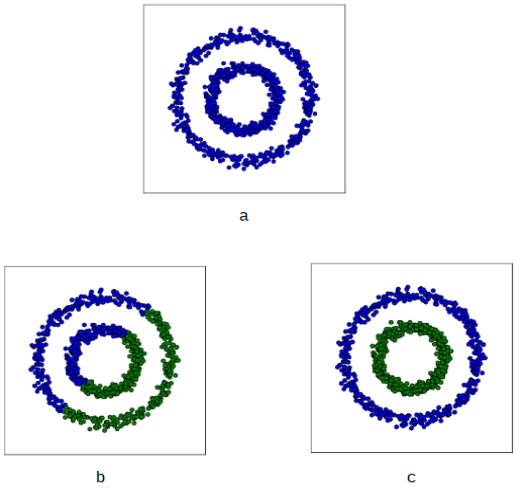

Figure 2.1: (a) data before clustering, (b) k-means, (c) spectral clustering

In spectral clustering the data points represent nodes of the graph. Thus, clustering can be framed as a graph partitioning problem. Next, the data are mapped to a low-dimensional space in which k-means gives better results.

Following lines explain the procedure of the spectral clustering [10].
Given a set of points $\{v_1, v_2, ..., v_n\}$ in $\mathbb{R}^m$ that we want to cluster into k clusters:
1. Build the affinity matrix $F \in \mathbb{R}^{n \times n}$ defined by $F_{ij} = \exp(-\|v_i - v_j\|^2/2\sigma^2)$ if $i \neq j$, and $F_{ii} = 0$ where σ^2 is the scaling parameter.
2. Compute normalized Laplacian matrix $L = D^{\frac{-1}{2}} F D^{\frac{-1}{2}}$, where D is a diagonal matrix whose (i,i)-element is the sum of F's i-th row.
3. Find $x_1, x_2, ..., x_k$ the k largest eigenvectors of L and form the matrix X by stacking the eigenvectors in columns.
4. Form the matrix Y from X by renormalizing each of X's rows to have unit lenght (i.e. $Y_{ij} = X_{ij}/(\sum_j {X_{ij}}^2)^{\frac{1}{2}}$).
5. Treating each row of Y as a point in $\mathbb{R}^k$, cluster them into k clusters via k-means or any other algorithm (that attempts to minimize distortion).
6. Finally, assign the original point v_i to cluster j if and only if row i of the matrix Y was assigned to cluster j.

2.3.3 Spectral Co-Clustering algorithm

We can observe a data matrix A as a bipartite graph, where rows would form one partition and columns the other. Each entry of the matrix corresponds to an edge between a row and a column. Spectral Co-Clustering algorithm approximates the normalized cut of this graph to find heavy subgraphs via generalized eigenvalue decomposition of the Laplacian of the graph. The resulting bicluster structure is block-diagonal, since each row and each column belongs to exactly one bicluster (Figure 2.2) [11].

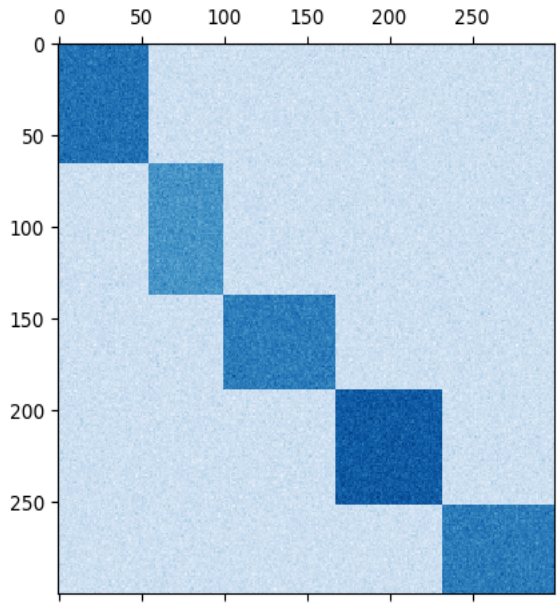

Figure 2.2: Rearranged data matrix to show biclusters

Described procedure is known as a co-clustering algorithm, and it was originally proposed for textual documents. If documents are arranged as rows of data matrix, and columns correspond to appearing words, the algorithm will cluster both simultaneously. This setting is quite similar to the one that is used here, so this algorithm can be generalized for clustering genes and samples.

Graph Theory Preliminaries

Definition 2.3.1. *An undirected graph G is an ordered pair (V, E) comprising:*
- V a set of vertices (also called nodes);
- E a set of edges, which are unordered pairs of vertices.

Definition 2.3.2. *A graph G is a simple graph if it does not contain loops or parallel edges.*

Definition 2.3.3. *With each edge e of G let there be associated a real number w(e), specifying its weight. Then G, together with these weights over its edges, is called a weighted graph.*

Definition 2.3.4. *A weighted adjacency matrix W of a simple weighted graph G is:*

$$W = \begin{cases} w_{i,j}, & \text{if there is an edge between } \mathrm{v_i} \text{ and } \mathrm{v_j} \\ 0, & \text{otherwise.} \end{cases} \tag{2.5}$$

Definition 2.3.5. *A degree d_i of a vertex $v_i \in V$ is defined as:*

$$d_i = \sum_{j=1}^{n} w_{i,j} \tag{2.6}$$

There are two different ways of measuring the "size" of a subset $V_i \in V$:

$$\begin{aligned} |V_i| :=& \text{ the number of vertices in V}_\text{i} \\ vol(V_i) :=& \textstyle\sum_{i \in V_i} d_i \end{aligned} \tag{2.7}$$

Definition 2.3.6. *A cut of a graph G = (V, E) is set of edges whose removal makes the graph disconnected.*

Definition 2.3.7. *For weighted graphs, cost of a cut is*

$$cut(V_1, V_2) = \sum_{i \in V_1, j \in V_2} W_{i,j}. \tag{2.8}$$

Extension of cut definition to k vertex subsets:

$$cut(V_1, V_2, ..., V_k) = \sum_{i<j} cut(V_i, V_j). \tag{2.9}$$

Definition 2.3.8. *An undirected bipartite graph is a triplet G = (R, C, E), where $R = \{r_1, r_2, ..., r_m\}$ and $C = \{c_1, c_2, ..., c_n\}$ are two sets of vertices and E is the set of edges $\{(r_i, c_j) : r_i \in R, c_j \in C\}$.*

In the data sets analyzed in this thesis, R would be a set of samples (rows), C would be a set of genes (columns) and an edge would be an association between a sample and a gene.

Consider data matrix A, with dimensions $n \times m$, the weighted adjacency matrix of the bipartite graph may be written as

$$W = \begin{bmatrix} 0 & A_{i,j} \\ A_{i,j}^T & 0 \end{bmatrix} \tag{2.10}$$

Simultaneous Clustering

The main idea behind this algorithm is that clustering of words induces document clustering and that document clustering induces word clustering. Consider document clusters $Q_1, ..., Q_k (\cup_i Q_i = Q$ and $Q_i \cap Q_j = \emptyset, i \neq j)$ and corresponding word clusters $H_1, ..., H_k (\cup_i H_i = H$ and $H_i \cap H_j = \emptyset, i \neq j)$, the induced word clustering is given by

$$H_m = \{h_i : \sum_{j \in Q_m} A_{ij} \geq \sum_{j \in Q_l} A_{ij}, \text{ for all l} = 1, ..., \text{k}\} \tag{2.11}$$

and the induced document clustering is given by

$$Q_m = \{q_i : \sum_{i \in H_m} A_{ij} \geq \sum_{i \in H_l} A_{ij}, \text{ for all } \mathrm{l} = 1, ..., \mathrm{k}\}. \tag{2.12}$$

When microarray data sets are analyzed the goal is to identify both clusters of genes that participate in common regulatory networks and clusters of samples associated with the effects of these genes. If clusters of genes are known in advance it can help in clustering samples and vice versa, as in the case with words and documents.

Graph Partitioning

First, let us convert our data matrix A into a graph G. Desired graph is a graph $G = (R, C, E)$ that is a bipartite graph with n rows (samples) vertices, m columns (genes) vertices and edges between a sample and a gene. For G we can say that it is bipartite because there are two disjoint sets of vertices with no edges within sets (and every gene is connected to every sample). Example of such graph can be seen in Figure 2.3.

Bisection problem of this graph is to find nearly equally-sized vertex subsets - clusters. The goal is to find 'good' clusters and that means that the number of within-cluster connections is maximized, and the number of the between-cluster connections is minimized.

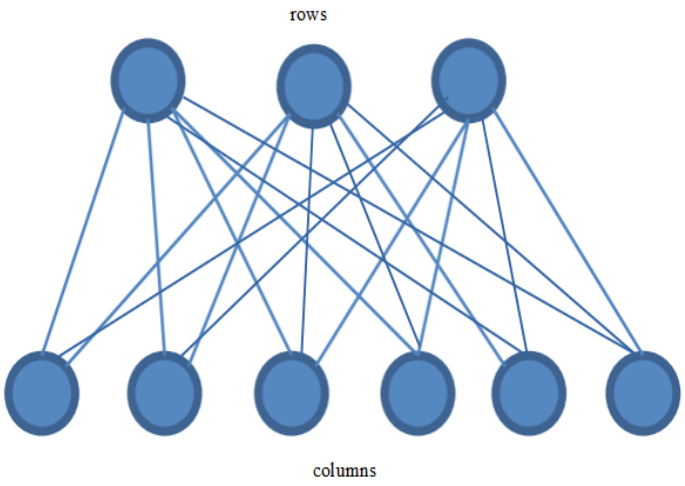

Figure 2.3: Bipartite graph

This section will introduce the Laplacian matrix L and some of its properties that will be used for deriving the clustering algorithm.

Definition 2.3.9. *The Laplacian matrix $L = L_G$ of G is an $n \times n$ symmetric matrix, with one row and column for each vertex, such that*

$$L_{ij} = \begin{cases} \sum_k w_{ik}, & \mathrm{i} = \mathrm{j} \\ -w_{ij}, & \mathrm{i} \neq \mathrm{j} \text{ and there is an edge i, j} \\ 0, & \text{otherwise.} \end{cases} \tag{2.13}$$

Theorem 2.3.1. *The Laplacian matrix $L = L_G$ of the graph G has the following properties:*
1. L = D - W, where W is the adjacency matrix and D is the diagonal "degree" matrix with $D_{ii} = \sum_k w_{ik}$.
2. $L = I_G I_G^T$.
3. L is a symmetric positive semi-definite matrix. Thus all eigenvalues of L are real and non-negative, and L has a full set of n real and orthogonal eigenvectors.
4. Let $e = [1, ..., 1]^T$. Then $Le = 0$. Thus 0 is an eigenvalue of L and e is the corresponding eigenvector.
5. If the graph G has c connected components then L has c eigenvalues that are equal to 0.
6. For any vector x, $x^T Lx = \sum_{i,j \in E} w_{ij}(x_i - x_j)^2$.
7. For any vector x and scalars α and β, $(\alpha x + \beta e)^T L(\alpha x + \beta e) = \alpha^2 x^T Lx$.

Proof. *1. Part 1 follows from the definition of L.*
2. This is easily verified by multiplying I_G and I_G^T.
3. Using property 2, $x^T Lx = x^T I_G I_G^T x = y^T y \geq 0$, for all x. This implies that L is symmetric positive semidefinite. All such matrices have non-negative real eigenvalues and a full set of n orthogonal eigenvectors.
4. Given any vector x, $Lx = I_G(I_G^T x)$. Let k be the row of $I_G^T x$ that corresponds to the edge $\{i,j\}$, then it is easy to see that

$$(I_G^T x)_k = \sqrt{w_{ij}}(x_i - x_j), \tag{2.14}$$

and so when $x = e$, $Le = 0$.
5. See [12]
6. This follows from equation (2.14).
7. This follows from property 4 above.

To see how powerful these properties are, let us consider data in Figure 2.1 (a). The graph should contain two connected components, each corresponding to one of the clusters we would like to find. The Laplacian matrix property that the smallest eigenvalue of Laplacian is 0 with eigenvector e will be used. Since there are no edges between two components in a graph, the adjacency matrix is block diagonal and that implies that L is block diagonal, as well. L consists of two blocks, two submatrices L_1 and L_2 which are also Laplacians but for two subgraphs. That means that both, L_1 and L_2 have eigenvalues 0 and eigenvectors 1. Thus, the smallest eigenvalue of L has multiplicity two, and its eigenspace is spanned by:

$$\begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{2.15}$$

where the number of 1 entries in each vector is equal to the number of vertices in that connected component.

In the real data the exemplified structure is rare. Graph G will not be so well separated into connected components, thus, we will assume that G consists of exactly one connected component. From property 5 of the Theorem (2.3.1) it can be seen that if G consists of exactly one connected component, the second smallest eigenvalue of the Laplacian is nonzero. This fact is interesting because it states that graph Laplacian spectrum, meaning its eigenvalues, tells us something about the underlying connectivity of the graph. The solution of this problem is to remove a few edges to create two connected components again. For this, graph cut is used (Definition 2.3.3) and our goal is to minimize the cut.

Consider graph $G = (V, E)$, whose set of vertices is partitioned into two sets V_1 and V_2. Let us define a vector p with respect to this partitioning. Each entry of p will correspond to a particular vertex. We will assign that entry a positive one if vertex lies in V_1 and a negative one if vertex lies in V_2.

$$p_i = \begin{cases} +1, & \text{i} \in \text{V}_1, \\ -1, & \text{i} \in \text{V}_2. \end{cases} \tag{2.16}$$

Theorem 2.3.2. *Given the Laplacian matrix L of graph G and a partition vector p, the Rayleigh Quotient is*

$$R(L, p) = \frac{p^T Lp}{p^T p} = \frac{1}{n} 4cut(V_1, V_2). \tag{2.17}$$

Proof. *Clearly* $p^T p = n$. *By property 6 of Theorem (2.3.1)* $p^T Lp = \sum_{i,j \in E} E_{ij}(p_i - p_j)^2$. *Thus edges within* V_1 *or* V_2 *do not contribute to the above sum, while each edge between* V_1 *and* V_2 *contributes a value of 4 times the edge-weight.*

From Theorem (2.3.2) we can see that the minimum of the cut is trivially achieved by setting all p_i to -1 (or $+1$). In other words, $V_1 = V$ and $V_2 = \emptyset$ (or vice versa). The problem with finding 'good' clusters is that cut only considers external cluster connections and does not consider internal cluster connectivity. Solution to this problem is to find an objective function that will ensure that each partition is approximately balanced.

The following objective function captures what we need:

$$Q(V_1, V_2) = \frac{cut(V_1, V_2)}{W(V_1)} + \frac{cut(V_1, V_2)}{W(V_2)}, \tag{2.18}$$

where $W(V_i) = \sum_{l \in_i} D_{ll}$ is the sum of the edge weights within a partition.

The smaller value of this function, more balanced partitioning. There are two kinds of objective function, RatioCut [13] and the normalized cut Ncut [14]. In RatioCut, the size of a subset V_i of a graph is measured by its number of vertices $|V_i|$, while in NCut the size is measured by the weights of its edges $W(V_i)$. The definitions are:

$$RatioCut(V_1, V_2) = \frac{cut(V_1, V_2)}{|V_1|} + \frac{cut(V_1, V_2)}{|V_2|}, \tag{2.19}$$

$$NCut(V_1, V_2) = \frac{cut(V_1, V_2)}{W(V_1)} + \frac{cut(V_1, V_2)}{W(V_2)}. \tag{2.20}$$

The both objective functions try to achieve that the clusters are "balanced", as measured by the number of vertices or edge weights, respectively. In this algorithm is used NCut. The normalized cut problem is NP-hard.

In the following text it is shown that the Rayleigh Quotient of the generalized partition vector q equals the objective function value (Equation 2.18).

Lemma 2.3.2. *Given graph G, let L and D be its Laplacian and vertex weight matrices respectively. Let $\eta_1 = W(V_1)$ and $\eta_2 = W(V_2)$. Then the generalized partition vector q with elements*

$$q_i = \begin{cases} +\sqrt{\frac{\eta_2}{\eta_1}}, & \mathrm{i} \in \mathrm{V}_1, \\ -\sqrt{\frac{\eta_1}{\eta_2}}, & \mathrm{i} \in \mathrm{V}_2, \end{cases} \tag{2.21}$$

satisfies $q^T De = 0$, and $q^T Dq = W(V)$.

Proof. *Let $y = De$, then $y_i = W(V_i) = D_{ii}$. Thus*

$$q^T De = \sqrt{\frac{\eta_2}{\eta_1}} \sum_{i \in V_1} W(V_i) - \sqrt{\frac{\eta_1}{\eta_2}} \sum_{i \in V_2} W(V_i) = 0. \tag{2.22}$$

Similarly,

$$q^T De = \sum_{i=1}^{n} D_{ii} q_i^2 = \eta_1 + \eta_2 = W(V). \tag{2.23}$$

Theorem 2.3.3. *Using the notation of Lemma (2.3.2),*

$$\frac{q^T Lq}{q^T Dq} = \frac{cut(V_1, V_2)}{W(V_1)} + \frac{cut(V_1, V_2)}{W(V_2)}. \tag{2.24}$$

Proof. *It is easy to show that the generalized partition vector q may be written as*

$$q = \frac{\eta_1 + \eta_2}{2\sqrt{\eta_1 \eta_2}} p + \frac{\eta_1 - \eta_2}{2\sqrt{\eta_1 \eta_2}} e \tag{2.25}$$

where p is the partition vector of (2.8). Using part 7 of Theorem (2.3.1), we see that

$$q^T Lq = \frac{(\eta_1 + \eta_2)^2}{4\eta_1 \eta_2} p^T Lp. \tag{2.26}$$

Substituting the values of $p^T Lp$ and $q^T Dq$, from Theorem (2.3.2) and Lemma (2.3.2) respectively, proves the result.

Theorem 2.3.4. *The problem*

$$\min_{q \neq 0} \frac{q^T Lq}{q^T Dq}, \quad \textit{subject to } q^T De = 0, \tag{2.27}$$

is solved when q is the eigenvector corresponding to the 2nd smallest eigenvalue λ_2 of the generalized eigenvalue problem,

$$Lz = \lambda Dz. \tag{2.28}$$

Proof. *This is a standard result from linear algebra [15].*

Consider Theorem (2.3.3) globally minimum solution of Equation 2.18 can be found using the generalized partition vector q. Thus, it turns out that the first k eigenvectors of the generalized eigenproblem $Lz = \lambda Dz$ provide the solution to the relaxed problem.

Spectral Co-Clustering Algorithm

From Theorem (2.3.4) it can be seen that problem of finding the minimum normalized cut can be solved with finding the second eigenvector of generalized eigenvalue problem:

$$Lz = \lambda Dz. \tag{2.29}$$

In the following text it is explained how to avoid working on the matrix $L_{(n+m)\times(n+m)}$, that is of larger dimensions than data matrix $A_{n\times m}$ and instead how one can work with A. Normalizing the matrix A in certainr way will give the desired partitions of the rows and the columns of A.

One of the properties of Laplacian matrix is that it can be written as $L = D - W$, and since bipartite graph is used, weighted adjacency matrix can be written as

$$W = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \tag{2.30}$$

Thus, Laplacian in this case will be

$$L = \begin{bmatrix} D_1 & -A \\ -A^T & D_2 \end{bmatrix} \tag{2.31}$$

where

$$D = \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix} \tag{2.32}$$

and $D_1(i,i) = \sum_j A_{i,j}$ is the sum of edge-weights incident on sample i, and $D_2(j,j) = \sum_i A_{i,j}$ is the sum of edge-weights incident on gene j. Further, generalized eigenvector problem can be written in the following way:

$$\begin{bmatrix} D_1 & -A \\ -A^T & D_2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \lambda \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$
$$\implies \begin{aligned} D_1 x - Ay &= \lambda D_1 x, \\ -A^T x + D_2 y &= \lambda D_2 y. \end{aligned} \tag{2.33}$$

Both D_1 and D_2 are nonsingular matrices, so we can rewrite (2.33):

$$\begin{aligned} D_1^{\frac{1}{2}} x - D_1^{\frac{-1}{2}} Ay &= \lambda D_1^{\frac{1}{2}} x, \\ -D_2^{\frac{-1}{2}} A^T x + D_2 \tfrac{-1}{2} y &= \lambda D_2^{\frac{1}{2}} y. \end{aligned} \tag{2.34}$$

If $u = D_1^{\frac{1}{2}} x$ and $v = D_2^{\frac{1}{2}} y$, we get:

$$\begin{aligned} D_1^{\frac{-1}{2}} A D_2^{\frac{-1}{2}} v &= (1-\lambda)u, \\ D_2^{\frac{-1}{2}} A^T D_1^{\frac{-1}{2}} u &= (1-\lambda)v \end{aligned} \tag{2.35}$$

The singular value decomposition (SVD) of an $n \times m$ matrix A is a factorization of the form $U\Sigma V^*$ where U is an $n \times n$ real or complex unitary matrix, Σ is an $n \times m$ rectangular diagonal matrix with non-negative real numbers on the diagonal, and V is an $m \times m$ real or complex unitary matrix. The diagonal entries σ_i of the Σ are the singular values of A and the columns of U and V are left-singular vectors and right-singular vectors of A, respectively. The equations (2.35) are equations that determine singular value decomposition of normalized matrix $A_N = D_1^{\frac{-1}{2}} A D_2^{\frac{-1}{2}}$, so (2.35) can be written as:

$$\begin{aligned} A_N v &= \sigma u, \\ A_N^T u &= \sigma v, \end{aligned} \tag{2.36}$$

where u and v are the left and right singular vectors of A_N respectively, while $\sigma = (1 - \lambda)$ is the corresponding singular value of A_N. This implies that instead of computing the eigenvector of the second smallest eigenvalue of (2.29), the left and right singular vectors corresponding to the second largest singular value of A_N, can be computed. Since $u = D_1^{\frac{1}{2}} x$ and $v = D_2^{\frac{1}{2}} y$, then the second eigenvector of L is given by:

$$z_2 = \begin{bmatrix} D_1^{\frac{-1}{2}} u_2 \\ D_2^{\frac{-1}{2}} v_2 \end{bmatrix}. \tag{2.37}$$

Bipartition Case

Now, when u_2 and v_2 are known, the main goal is to extract the optimal partition from these vectors. In other words we want to assign points to clusters, based on lower-dimensional representation based on eigenvector. The generalized partition vector q is two-valued, so the idea is to find bi-modal distribution in the values of u_2 and v_2.The optimal bipartitioning is obtained if $z_2(i)$ is assigned to the, earlier defined, bi-modal values m_1 and m_2 such that the following sum of squares criterion is minimized,

$$\sum_{j=1}^{2} \sum_{z_2(i) \in m_j} (z_2(i) - m_j)^2. \tag{2.38}$$

This can be done using k-means algorithm (2.38).

Now the co-clustering algorithm can be stated:

1. Given data matrix $A \in \mathbb{R}^{n \times m}$, form normalized matrix $A_n = D_1^{\frac{-1}{2}} A D_2^{\frac{-1}{2}}$.
2. Compute the u_2 and v_2, the second singular vectors of A_n and create the vector z_2 as in (2.37).
3. Run a one-dimensional K-means algorithm over z_2, to obtain desired bipartitioning.

Multipartition Case

This algorithm can be adapted for the general problem of finding k sample and gene clusters. There are two basic approaches to partition a graph into k clusters: recursive bi-partitioning and cluster multiple eigenvectors. Second approach was chosen here. Collection of the $l = \lceil \log_2 k \rceil$ singular vectors $u_2, u_3, ..., u_{l+1}$ and $v_2, v_3, ..., v_{l+1}$ often contains k-modal information about the data set. Then l-dimensional data set can be formed:

$$Z = \begin{bmatrix} D_1^{\frac{-1}{2}} U \\ D_2^{\frac{-1}{2}} V \end{bmatrix}, \tag{2.39}$$

where $U = [u_2, u_3, ..., u_{l+1}]$ and $V = [v_2, v_3, ..., v_{l+1}]$.

Now, when data are mapped to a low-dimensional space, each l-dimensional row, $Z(i)$, can be assigned to the l-dimensional points m_j, $(j = 1, ...k)$, such that the following sum of squares criterion is minimized

$$\sum_{j=1}^{k} \sum_{Z(i) \in m_j} ||Z(i) - m_j||^2. \tag{2.40}$$

Again, this can be done with k-means.

Co-clustering algorithm for problem of finding k columns and rows clusters:

1. Given data matrix $A \in \mathbb{R}^{n \times m}$, form normalized matrix $A_n = D_1^{\frac{-1}{2}} M D_2^{\frac{-1}{2}}$.
2. Compute $l = \lceil \log_2 k \rceil$ singular vectors of A_n, $u_2, u_3, ..., u_{l+1}$ and $v_2, v_3, ..., v_{l+1}$and create the matrix Z
3. Run the k-means algorithm on Z, l-dimensional data, to obtain the desired k biclusters.

One thing to underline is that n rows and m columns of data matrix are converted to the $n + m$ rows in matrix Z. Thus, both rows and columns are treated as samples and clustered together.

2.3.4 Spectral Bi-Clustering algorithm

Spectral Bi-Clustering algorithm assumes that the data matrix has a hidden underlying checkerboard pattern that can be discovered by permuting rows and columns. The problem of finding checkerboard structure is solved using eigenvectors. In order to make the pattern more obvious, the first step is normalization of data matrix. There are three possible approaches to normalize a data matrix: Independent rescaling of genes and conditions, Bi-stochastization and Log-interactions normalization. Afterwards, SVD is used to identify eigenvectors. While the Co-Clustering algorithm (Section 2.3.3) divides the genes and samples into the same number of clusters, Bi-Clustering algorithm has one advantage, it allows that the number of gene clusters and sample clusters can be different.

In analysis of cancer data sets the structure of data should be considered. Spectral Bi-Clustering algorithm is designed to be suitable for checkerboard structure. The reason this kind of structure might be suitable for a tumor classification problem analyzed in this thesis is that there exist subsets of highly active or almost completely inactive genes specific for the tumor type. Thus, under this assumption, the data matrix can be organized in a checkerboard-like structure with blocks of genes with high expression levels and low expression levels. In Figure 2.4 can be seen the original matrix that has hidden checkerboard structure and in Figure 2.5 can be seen the same matrix with rearranged rows and columns where checkerboard pattern is revealed.

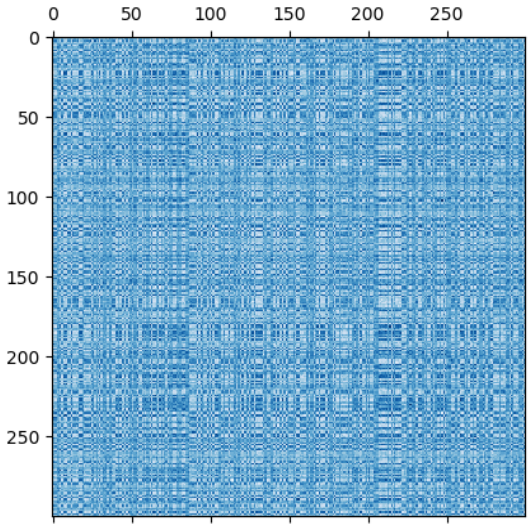

Figure 2.4: Original data matrix

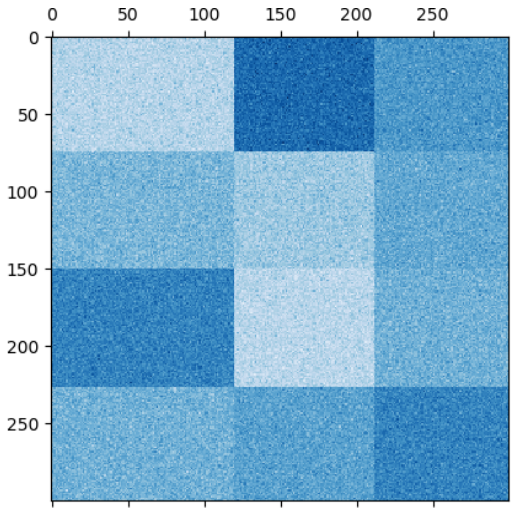

Figure 2.5: Rearranged data matrix to show biclusters

Eigenproblem and Checkerboard Structure

Eigenvectors are used to find a hidden checkerboard pattern in the data matrix, if it exists. It is an eigenproblem with matrix AA^T, where A is a matrix that has a checkerboard structure. Let us consider two classification vectors, one for the rows (samples) r and other for the columns c (genes). Each of these vectors has piecewise-constant values corresponding to the row and column partitions of the matrix. If matrix A is applied to a classification vector for genes c, new classification vector for samples r' is obtained, with the same block pattern as r and similarly, if A^T is applied to classification vector for samples r, as result new gene classification vector c' with the same block pattern as c is obtained:

$$\begin{aligned} Ac &= r' \\ A^T r &= c' \end{aligned} \tag{2.41}$$

and after substitution:

$$\begin{aligned} AA^T r &= r' \\ A^T Ac &= c' \end{aligned} \tag{2.42}$$

Therefore, it can be concluded that checkerboard structure of A is recognizable in piecewise constant structure of pair of eigenvectors r and c that solves coupled eigenvalue problem where eigenvectors r and c have the same eigenvalue λ^2:

$$\begin{aligned} A^T Ar &= \lambda^2 r \\ AA^T c &= \lambda^2 c \end{aligned} \tag{2.43}$$

From linear algebra it is known that solving an eigenproblem involving AA^T is equivalent to finding singular value decomposition (SVD) of A. In this decomposition, matrix A can be written as $A = U\Sigma V^T$. The columns of U and V are eigenvectors of the matrices AA^T and $A^T A$, respectively and they are singular vectors of A. To make pattern evident after finding eigenvectors, all that is needed is to reshuffle rows and columns. What makes uncovering the checkerboard pattern in matrix difficult is the different average amount of expressions associated with particular genes or samples. This problem can be solved by initial normalization of the data matrix A. This algorithm uses three different normalizations to set each gene on the same scale.

Normalization

As mentioned before, the first step in spectral bi-clustering algorithm is data matrix normalization. Kluger [4] introduced three possible normalization procedures: Independent rescaling of genes and conditions, Bi-stochastization and Log-interactions normalization.

Independent rescaling of genes and conditions

The similarity between expression levels of the two genes should be more obvious if each gene is scaled so that they have the same mean value. Similarly as explained in Co-clustering algorithm, here two diagonal matrices for scaling are used, matrix R whose diagonal elements r_{ii} represent the row sums of A and matrix C whose components are the column sums of data matrix

A. The matrix $R^{-\frac{1}{2}}$ is used for scaling rows and the matrix $C^{-\frac{1}{2}}$ is used for scaling columns. Thus, if data matrix A is normalized, $A_n = R^{-\frac{1}{2}}AC^{-\frac{1}{2}}$ is obtained and in that way checkerboard pattern would be more recognizable. Figure 2.6 shows an example of a perfect checkerboard structure, but where each row and column are multiplied by a random factor. After applying described normalization to this matrix, result in Figure 2.7 with almost uniform clusters are obtained.

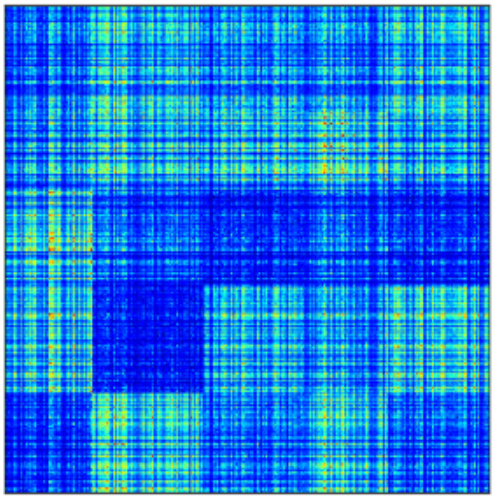

Figure 2.6: Checkerboard matrix before normalization

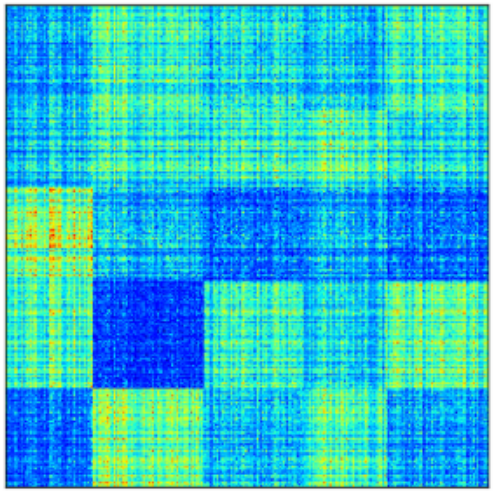

Figure 2.7: Checkerboard matrix after normalization (Independent rescaling)

Bi-stochastization

This is an approach that simultaneously normalizes genes and samples. Normalization yields a matrix that has doubly stochastic-like structure and it is called a bistochastic matrix. Such a matrix has a property that all rows sum to a constant and all columns sum to a different constant.

Sinkhorn [16] proved that every entry-wise positive matrix can be made doubly stochastic by multiplying with two diagonal matrices. Thus, normalized matrix can be computed by repeating *independent scaling of rows*

and columns iteratively until convergence. The procedure starts with $A_1 = R_0^{-\frac{1}{2}} A C_0^{-\frac{1}{2}}$, $A_2 = R_1^{-\frac{1}{2}} A_1 C_1^{-\frac{1}{2}}$, and general iteration is $A_{t+1} = R_t^{-\frac{1}{2}} A_t C_t^{-\frac{1}{2}}$. In the Figure 2.8 can be seen the matrix from Figure 2.6 after bistochastization and more obviously the checkerboard pattern.

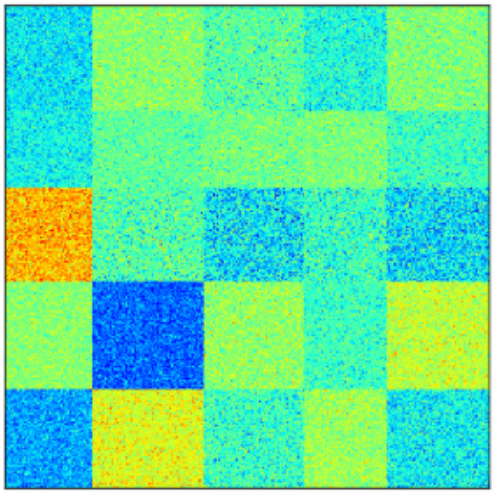

Figure 2.8: Checkerboard matrix after normalization (Bi-stochastization)

Log-interactions normalization

As a third method of normalization, log-interactions normalization is used. A useful process in transforming microarray data is taking logarithm as it results in data distribution closer to normal distribution. The idea here is to calculate the logarithm of the data and then extract the interactions between the genes and the samples. In the following formula the final matrix can be seen:

$$K_{ij} = L_{ij} - \overline{L_{i.}} - \overline{L_{.j}} + \overline{L_{..}} \tag{2.44}$$

$L_{ij} = \log A_{ij}$
$\overline{L_{i.}} = \frac{1}{m} \sum_{j=1}^{m} L_{ij}$ is the average of i-th row,
$\overline{L_{.j}} = \frac{1}{n} \sum_{i=1}^{n} L_{i,}$ is the average of j-th column and
$\overline{L_{..}} = \frac{1}{mn} \sum_{i=1}^{n} \sum_{j=1}^{m} L_{ij}$ is the average of the whole matrix.

The value K_{ij} captures interaction between row i and column j that can not be explained by systematic variability among rows, among columns, or within the entire matrix. The Figure 2.9 presents the matrix from Figure 2.6 after log-interactions normalization.

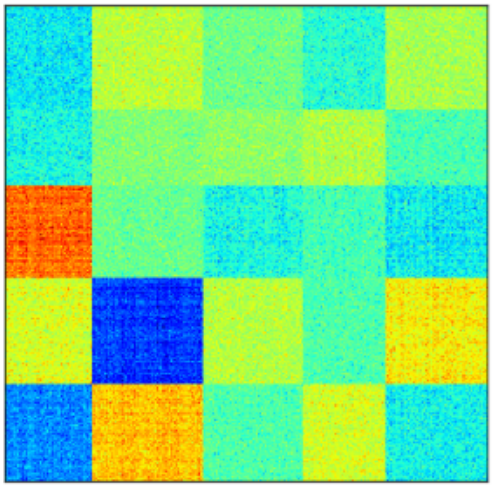

Figure 2.9: Checkerboard matrix after normalization (Log-interactions)

Partitioning vectors determination

When normalization of the data matrix is done SVD is applied to the normalized matrix to reveal the checkerboard structure. In that step p singular vectors of matrix A are obtained. Next step is to process these vectors to find partitions.

First, what is important to mention is that in the case of Independent rescaling and Bistochatizaton first (left and right) singular vectors are discarded. It is because they do not contain any partitioning information. Thus, if these two normalization techniques are used, the first singular vectors of interests will be u_2 and v_2. In the case of Log-interactions normalization, all singular vectors are meaningful.

Converting singular vectors into partitioning vectors is done as follows. In the first step singular vectors are approximated with piecewise-constant vectors. For this approximation is used one-dimensional k-means. As the best singular vector is chosen the one that yields the lowest norm difference from its peace-wise approximation. In this algorithm there are two approaches how to process these vectors further.

In the first approach, vectors of interest are the best left and the best right singular vectors, u' and v', respectively. u' determines the row partitioning and v' determines the column partitioning. In the end, by applying of the k-means algorithm to the u' and v' the assignments of rows and columns to biclusters can be determined.

In this thesis the second approach is used. Unlike the previous approach, where data are projected just to the one best singular vector, here data is projected to the subset of the best singular vectors. Let us denote the number of the best singular vectors with q ($q < p$). Next, the matrix U' whose columns are the q best left singular vectors and matrix V' whose columns are the q best right singular vectors are defined. In order to get row labels, the rows of A are projected to AV'. The matrix AV' has dimension $n \times q$. All of n rows are clustered using k-means and the desired row labels are obtained. Similarly, column labels are obtained. Columns of matrix A are projected to $A^T U'$, that has dimensions $m \times q$, and after clustering m

rows with k-means, column labels are obtained.

The co-clustering algorithm can be summarized as follows:

1. Given data matrix $A \in \mathbb{R}^{n \times m}$, form normalized matrix by using one of the three normalizations: Independent rescaling of genes and conditions, Bi-stochastization and Log-interactions normalization
2. Compute first p singular vectors and choose the best q to obtain U' and V'.
3. Project the rows of A to AV' and run k-means algorithm to obtain row labels.
Project the columns of A to $A^T U'$ and run k-means algorithm to obtain column labels.

2.4 Cluster Validation

The term cluster validation is used to refer to the procedure of evaluating the goodness of clustering algorithm results. This is important for comparing clustering algorithms, comparing two sets of clusters, comparing two clusters and to avoid finding patterns in random data. For measuring cluster quality there are three categorizations of measures:
1. External: Compare clustering against prior or expert-specified knowledge (i.e. ground truth) using certain clustering quality measures. Since the "true" cluster number is known in advance, this approach is mainly used for selecting the right clustering algorithm for a specific data set.
2. Internal: Evaluate the goodness of a clustering by considering how well the clusters are separated and how compact the clusters are.
3. Relative: Directly compare different clusterings, usually those obtained via different parameter settings for the same algorithm.

2.4.1 The Adjusted Rand Index

The Adjusted Rand Index is a measure of agreement between two partitions: one given by the clustering process and the other defined by external criteria. Given the data matrix $A = \{A_{ij}\}_{n \times m}$, where n is the number of rows (objects) and m is the number of columns (attributes), a partition of n objects in r groups can be formed such that union of all the groups from $P = \{p_1, ..., p_r\}$ is equal to the entire object set and the intersection of any two groups from P is empty. Given two partitions P and $L = \{l_1, ..., l_c\}$, with r and c groups, respectively, the contingency table Table 2.2 [17] can be formed to indicate groups overlap between P and L where t_{rc} indicates the total number of objects that simultaneously belong to r^{th} cluster and c^{th} class. Suppose that P is a clustering result and L is our external criteria.

Cluster\Class	l_1	l_2	$\cdots$	l_c	Sums
p_1	t_{11}	t_{12}	$\cdots$	t_{1c}	t_{1+}
p_2	t_{21}	t_{22}	$\cdots$	t_{2c}	t_{2+}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
p_r	t_{r1}	t_{r2}	$\cdots$	t_{rc}	t_{r+}
Sums	t_{+1}	t_{+2}	$\cdots$	t_{+c}	$t_{++} = n$

Table 2.2: Notation for the contingency table for comparing two partitions

Notation

- a - the number of pairs of objects that are in the same cluster in P and in the same class in L
- b - the number of pairs of objects that are in the same cluster in P but not in the same class in L
- c - the number of pairs of objects that are in the same class in L but not in the same cluster in P
- d - the number of pairs of objects in different classes and different clusters in both partitions

Alternatively representation of Table 2.2 based on this definition is 2×2 Table 2.3.

	L	
P	Pair in same group	Pair in different groups
Pair in same group	a	b
Pair in different groups	c	d

Table 2.3: 2×2 Contingency Table Representation of the Partition Co-Occurrence Table

a, b, c and d we can calculate in the following way:

$$a = \frac{\sum_{i=1}^{r} \sum_{j=1}^{c} t_{ij}^2 - n}{2}, \tag{2.45}$$

$$b = \frac{\sum_{i=1}^{r} t_{i+}^2 - \sum_{i=1}^{r} \sum_{j=1}^{c} t_{ij}^2}{2}, \tag{2.46}$$

$$c = \frac{\sum_{j=1}^{c} t_{+j}^2 - \sum_{i=1}^{r} \sum_{j=1}^{c} t_{ij}^2}{2}, \tag{2.47}$$

$$d = \frac{\sum_{i=1}^{r} \sum_{j=1}^{c} t_{ij}^2 + n^2 - \sum_{i=1}^{r} t_{i+}^2 - \sum_{j=1}^{c} t_{+j}^2}{2} \tag{2.48}$$

With these four numbers we can represent different indices and one of them is Rand index[18]. The Rand index is calculated by:

$$\frac{a+d}{a+b+c+d} \tag{2.49}$$

This index is in interval [0,1]. When two partitions agree perfectly, the Rand index is 1. However, there is an issue with this metric. The expected value of the index of two random partitions does not take a constant value and when we increase the number of clusters it increases as well. One attempt to overcome this problem is proposed by [19]:

$$ARI = \frac{\binom{n}{2}(a+d) - [(a+b)(a+c)+(c+d)(b+d)]}{\binom{n}{2}^2 - [(a+b)(a+c)+(c+d)(b+d)]} \tag{2.50}$$

.

ARI is in interval [-1, 1], where 1 means agreement between partitions, a value close to 0 is for random labeling independently of the number of clusters and samples and -1 is for disagreement between partitions. The expected value of ARI is 0 and it is a symmetric measure, i.e. $ari(a,b) = ari(b,a)$. This adjusted Rand index has been shown to be the most desirable index for measuring cluster recovery because it does not depend on algorithm or on number of clusters and has been used in several cluster validation studies.

2.4.2 Silhouette Index

Silhouette index is an internal measure for cluster validation. The Silhouette index is calculated using the mean intra-cluster distance and the mean nearest-cluster distance for each sample. The silhouette value is a measure of how similar a sample is to its own cluster compared to other clusters.

Let us assume that samples are clustered into k clusters. Take any sample i in the data set, and denote by C_i the cluster to which it has been assigned. When cluster C_i contains other objects apart from i, we can compute:

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i,j), \tag{2.51}$$

where $d(i,j)$ is the distance between samples i and j in the cluster C_i. It represents average dissimilarity of i to all other objects of C_i. Let $d(i,C)$ denote a mean distance between sample i and all the points in a cluster A that does not contain it. For a sample i one can compute the smallest mean distance from all the points in any cluster that does not contain i:

$$b(i) = \min_{j \neq i} \frac{1}{|C_j|} \sum_{j \in C_j} d(i,j). \tag{2.52}$$

The cluster that has the smallest mean dissimilarity is called neighbor of sample i, it is the second-best choice for sample i. Now, silhouette for a single data point i (Figure 2.10) can be defined:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}},\ if\ |C| > 1 \tag{2.53}$$

From 2.53 can be seen that $-1 \leq s(i) \leq 1$, when $s(i)$ is close to 1 then sample i is well-clustered, in the case where $s(i)$ is close to 0, it can be

considered as intermediate case, and in the case when $s(i)$ is -1, sample i is misclassified. The mean $s(i)$ over all data of the entire data set is called silhouette index:

$$SI = \overline{s}(k), \tag{2.54}$$

where $\overline{s}(k)$ is the mean $s(i)$ over all data of the entire data set for k clusters.

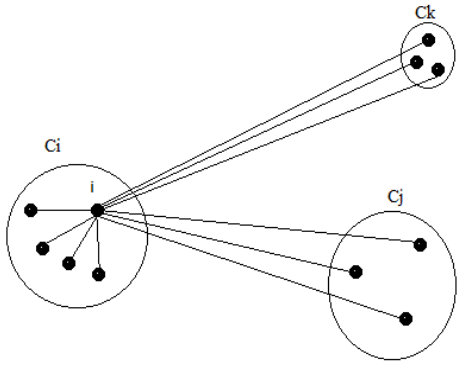

Figure 2.10: An illustration of the elements involved in the computation of $s(i)$, where the object i belongs to cluster C_i[20].

2.4.3 Experimental Setup

As mentioned before, some form of dimensionality reduction was already performed over original data excluding dimensions that did not show significant variability. In the first step in analysis of gene expression data sets three kinds of normalization are used: Z-normalization, sample-wise normalization (SN) and range normalization (RN). Z-normalization is used over feature values (genes) in cDNA data sets, so after this transformation each feature have mean value 0 and variance 1. SN is used over samples in Affymetrix data sets and after this transformation sum of values in each sample is 1. Finally, range normalization is used in Affymetrix data over the columns, transforming data values into $[0, 1]$ interval.
After data are normalized three clustering algorithms are performed: K-means, Co-Clustering algorithm and Bi-Clustering algorithm.
Important to mention is that K-means algorithm is used in two ways. First, as a part of Co-Clustering and Bi-Clustering algorithms where K-means is used to cluster data after mapping it in low dimensions. Second as independent clustering algorithm. As independent clustering algorithm it is performed over all data sets and in original data as well as normalized data. Co-Clustering algorithm is algorithm that is performed over all data sets and in original and normalized data as well.
The last algorithm Bi-Clustering is performed only over original data in all thirty-three data sets. As explained in Section 2.3.4, Bi-Clustering algorithm in order to make checkerboard pattern more obvious as first step uses three kinds of normalization, so there is no need to normalize data before feeding them to algorithm. Each of the three Bi-Clustering normalization is analyzed.
Different validation criteria (Adjusted Rand Index (ARI) and Silhouette

Index (SI)) are used to evaluated performances of the algorithms. Each algorithm was performed twenty times yielding twenty indices values which are later averaged to produce a single index value.

Implementation Details

All experiments were performed using Python programming language. It contains numerous machine learning libraries including `sklearn` which is used in this study. From this library all three clustering algorithms were imported: `KMeans, SpectralCoclustering, SpectralBiclustering`. Also, it contains both validation metrics: `adjusted_rand_score` and `silhouette_score`.

Chapter 3

Results

In this thesis different clustering techniques are evaluated in the task of discovering cancer subtypes based on gene expression data. Used data sets are thirty-three publicly available microarray data sets. Considered clustering techniques are Spectral Co-Clustering, Spectral Bi-Clustering and K-means. In order to enhance the clustering results, additional normalization techniques are included.

3.1 Data Preprocessing

3.1.1 Data Sets and Ground Truth Labels

Let us first inspect original data after corresponding normalization and Silhouette Index based on the ground truth class assignments. Heatmaps for pairs of cDNA and Affymetrix data sets are presented in Figures 3.1 - 3.3. Rows of each matrix are sorted according to ground truth labels. Although in some cases we can see some regularity in the row space, in general it is not obvious enough and there is no checkerboard pattern. Regarding Affymetrix data sets, both Sample wise and Range Normalization are presented and there is clear difference between the two.

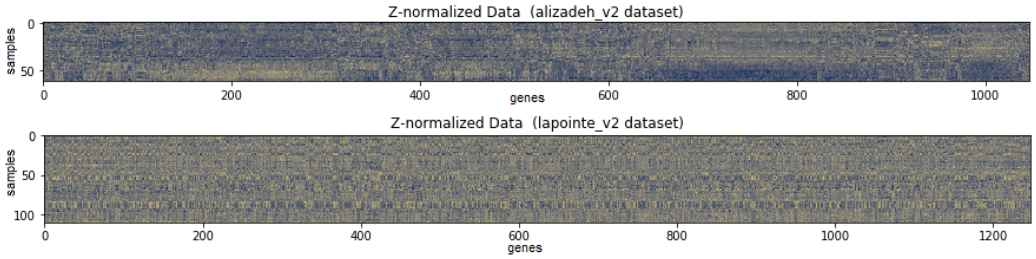

Figure 3.1: *Alizadeh_v2* and *Lapointe_v2* data sets after Z-normalization.

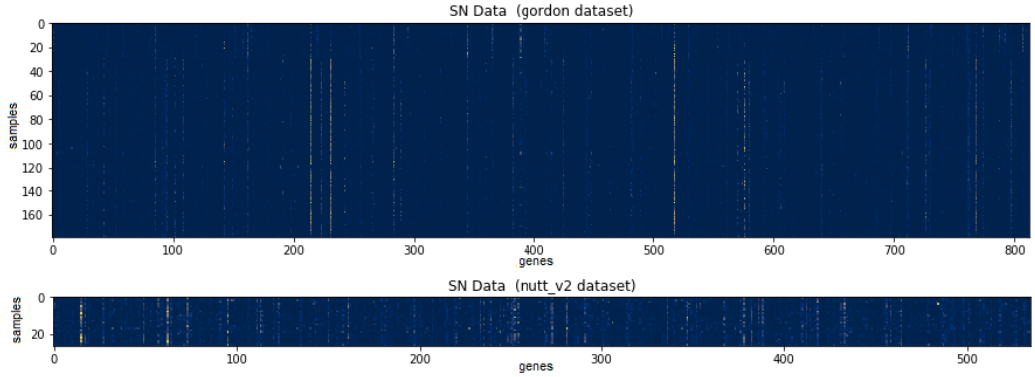

Figure 3.2: *Gordon* and *Nutt_v2* data sets after SN.

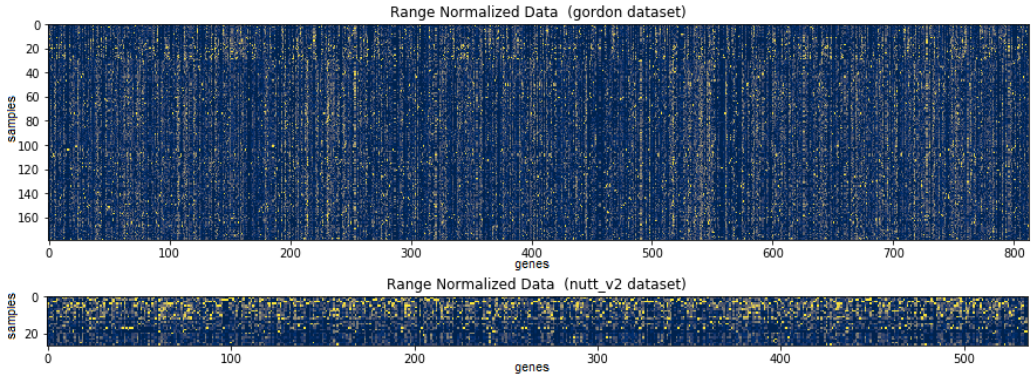

Figure 3.3: *Gordon* and *Nutt_v2* data sets after Range Normalization.

Silhouette Index over each normalized data set is presented in Figure 3.4. SI values indicate small within class similarity, especially in case of cDNA data sets. Also, there are negative values indicating high variability of gene expression in samples within the same class. For some Affymetrix data sets SI has more pronounced changes when different type of normalization is used. For example, data set *Nutt_v3* has almost the smallest value in case of SN, while in case of RN its value is the largest. This indicates that SI, as estimated based on Euclidean distance, might not be the optimal objective criteria when evaluating clustering partitions in this type of data. For these reasons we are estimating Adjusted Rand Index, as an external objective measure, directly comparing cluster and class labels. Moreover, this further supports the need to explore bi-clustering results looking for some better agreement between the subsets of genes and output labels.

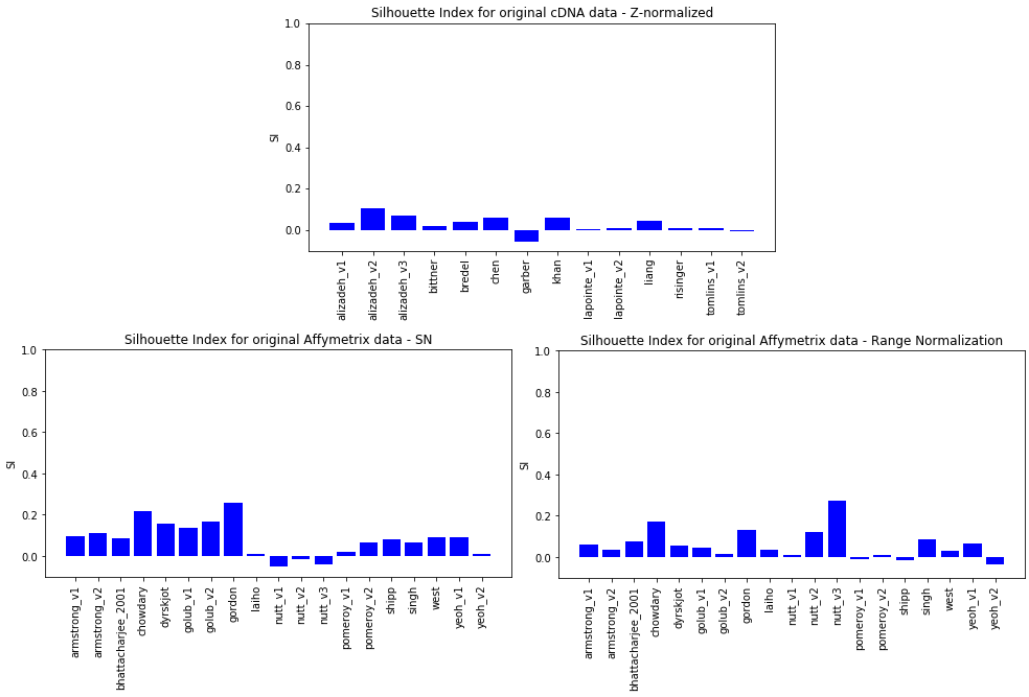

Figure 3.4: SI values for ground truth classes. First row: cDNA data sets after Z-normalization. Bottom left: Affymetrix data sets after SN. Bottom right: Affymetrix data sets after Range Normalization

3.1.2 Outliers

As a first step in preprocessing outliers in gene expression values are excluded.

If the distribution of the data is normal there is a standard procedure for detecting and handling outliers. In Figure 3.5 we can see distributions of cDNA data sets which resembles normal so gene expression values at distance larger than three standard deviations from the mean value were considered as outlier values. Detected points are then simply projected in the following manner: if the point value is larger than $\mu + 3\sigma$ its value is set exactly to $\mu + 3\sigma$ and if the point value is smaller than $\mu - 3\sigma$ its value is set to $\mu - 3\sigma$ where μ, σ are mean value and standard deviation of the feature, respectively.

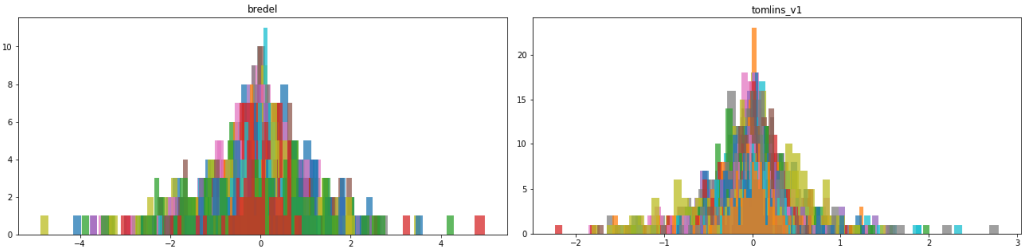

Figure 3.5: Distribution of *Bredel* and *Tomlins_v1* data sets.

In Figure 3.6 we see distributions for Affymetrix data. These do not resemble any known distributions so we can not use the same method for

handling outliers. Instead, we used Isolation Forest [21]. This is decision three based algorithm for detecting outliers for data do not follow any known distribution.

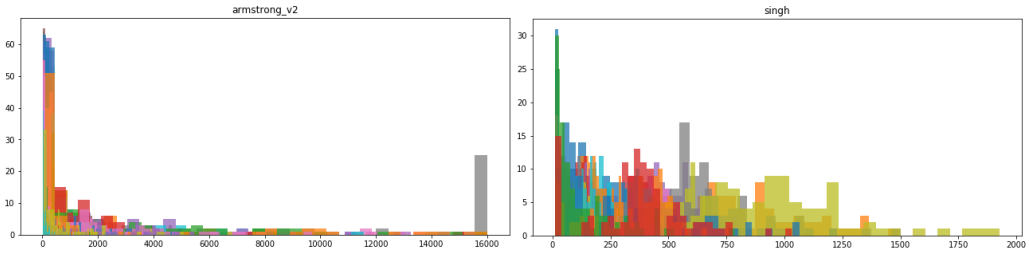

Figure 3.6: Distribution of *Armstrong_v2* and *Singh* data sets.

Percentages of outliers per data set can be seen in Table 3.1. Outliers for cDNA data were handled for each feature separately, and in the table only the average value for a set is printed. In general, cDNA sets exhibit lower number of outliers compared to Affymetrix.

Affymetrix Data Sets		cDNA Data Sets	
Data Set	%	Data Set	%
Armstrong-V1	1.40	Alizadeh-V1	0.51
Armstrong-V2	1.40	Alizadeh-V2	0.44
Bhattacharjee	1.50	Alizadeh-V3	0.44
Chowdary	1.96	Bittner	1.14
Dyrskjot	2.56	Bredel	0.79
Golub-V1	1.40	Chen	1.11
Golub-V2	1.40	Garber	1.39
Gordon	1.11	Khan	2.00
Laiho	2.77	Lapointe-V1	0.91
Nutt-V1	2.04	Lapoint-V2	1.02
Nutt-V2	3.70	Liang	0.74
Nutt-V3	4.76	Risinger	1.93
Pomeroy-V1	3.03	Tomlins-V1	0.90
Pomeroy-V2	2.43	Tomlins-V2	0.88
Shipp	1.31		
Singh	2.00		
West	2.08		
Yeoh-V1	1.22		
Yeoh-V2	1.22		

Table 3.1: Average percantages of outliers per each Affymetrix and cDNA data set

3.1.3 Normalization

Normalization technique used depends both on clustering algorithm and microarray experiments (Affymetrix and cDNA) of the considered data set.

Z-normalization is performed over cDNA data sets when using K-means or Co-Clustering algorithms. Figure 3.7 shows *Chen* data set before and

after Z-normalization (for the sake of the visualization outliers are excluded). When comparing two images, data after normalization exhibits more regular shape (in accordance with $\mathcal{N}(0,1)$ distribution). As expected the values stay within ± 3 standard deviation.

Sample-wise Normalization (SN) is used on Affymetrix data sets when using K-means and Co-Clustering algorithms. The SN rescales data into $[0,1]$ interval, row (sample) wise, so that the resulting row values are like a distribution of gene expressions within each sample. Figure 3.8 shows *Pomeroy_v1* data set before and after SN. Comparing the values after normalization, it can be noticed that the normalized expression ranges are quite balanced between the samples. Both, sizes of the boxes and of the whiskers are similar across samples.

Range Normalization (RN) is another normalization approach for Affymetrix data sets. Similar to SN, data are rescaled to $[0,1]$ interval but column-wise. In Figure 3.9 one can see *Chowdary* original data set and after RN with respect to genes (features).

In the Bi-Clustering algorithm original data are used as input. Normalization techniques in this case are contained in the algorithm itself (Independent rescaling of genes and conditions, Bi-stochastization and Log-interactions normalization) Section 2.3.4.

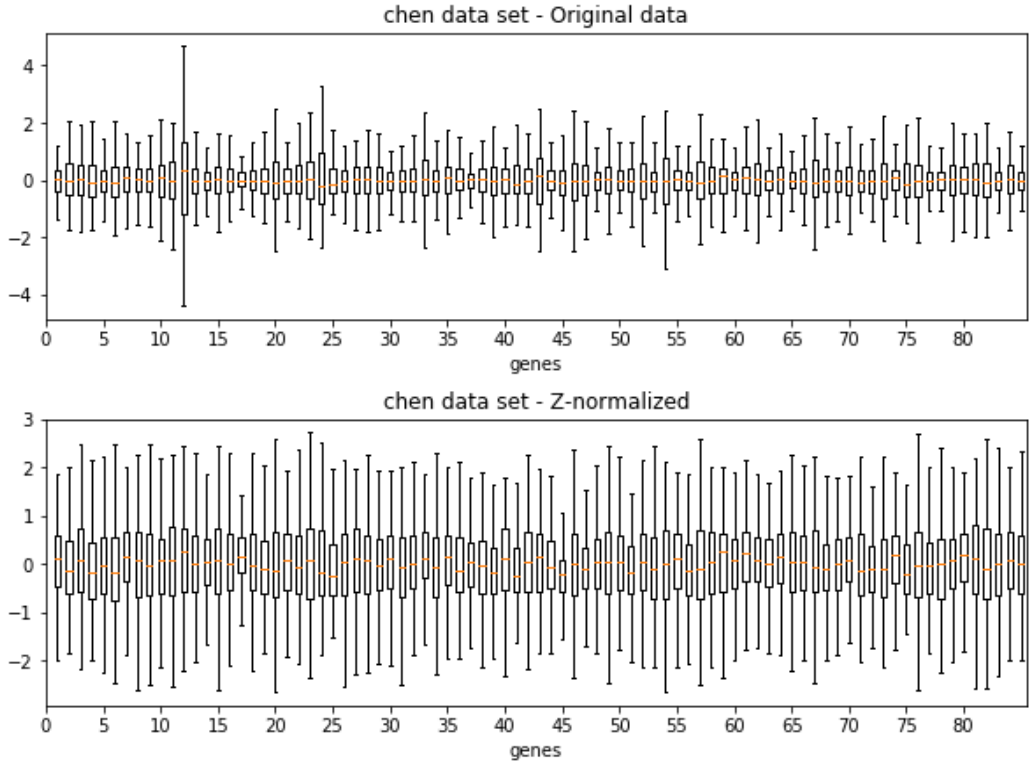

Figure 3.7: *Chen* data set before and after Z-normalization.

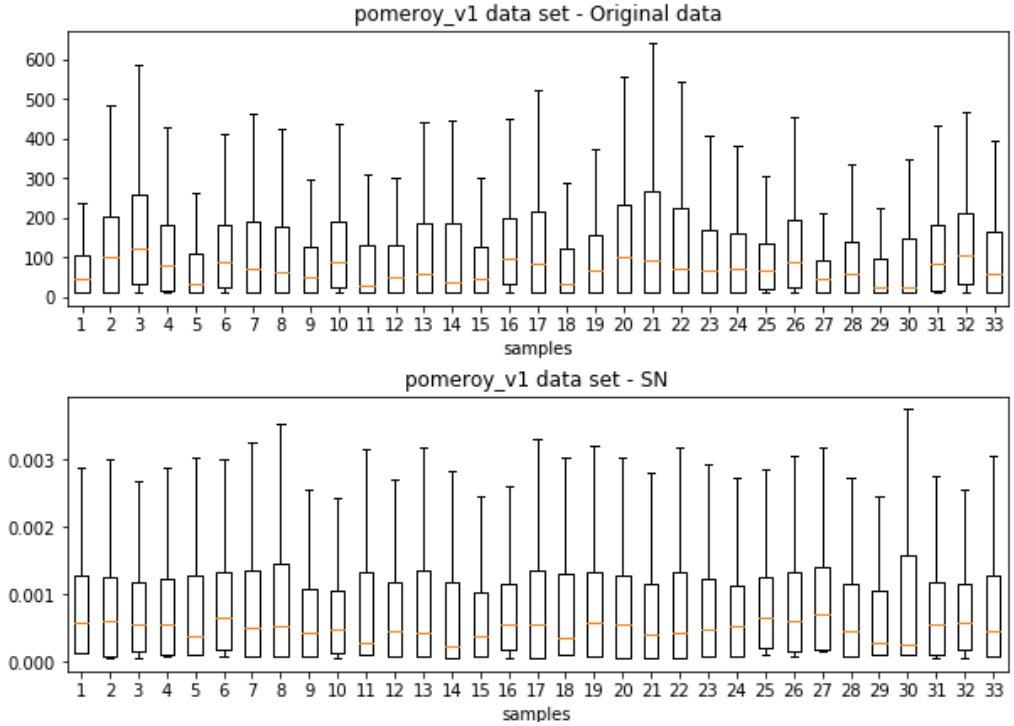

Figure 3.8: *Pomeroy_v1* data set before and after SN.

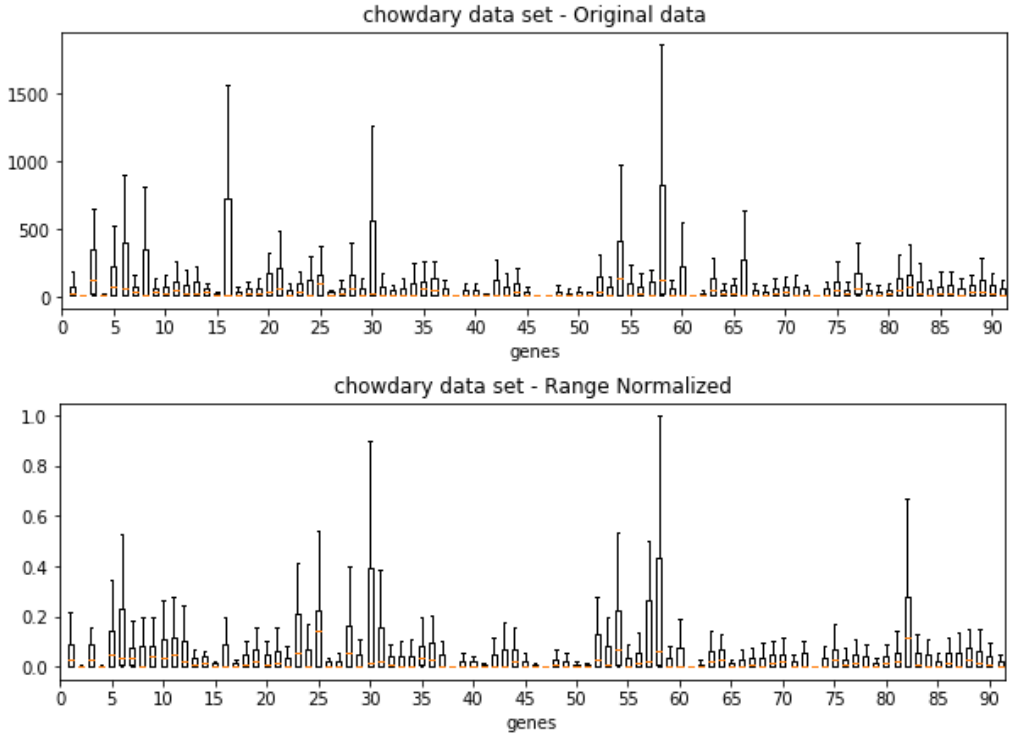

Figure 3.9: *Chowdary* data set before and after Range Normalization.

3.2 K-means

K-means is one of the most commonly used algorithms for data clustering. In this thesis K-means clustering approach was evaluated setting the num-

ber of clusters equal to the number of classes and the performance results measured by ARI and SI are presented in Figures 3.10 and 3.11, respectively. As in the initialization step K-means centroids are randomly selected, two runs of algorithm can produce different clustering results. For this reason in this work K-means was performed 20 times for each data set, hence the ARI values are presented in form of box plots. The randomly selected actual sample points have been used as initial centroids in each run.
ARIs for both cDNA (Z-normalized) and Affymetrix (SN and Range Normalized) data are presented in Figure 3.10. Affymetrix data under SN exhibit higher stability in predictions across different K-means runs compared with other two. Also, ARI values in cDNA datasets are low; and even in cases when some larger values are obtained, such as for *Alizadeh_v2* the output is not stable and largely depends on initializations. Difference in ARIs for two normalizations of Affymetrix sets is obvious. In SN case *Armstrong_v2*, *Chowdary* and *Yeoh_v1* perform the best with stable partitions corresponding closely to class distributions in these data sets. When Range Normalization is used case, where in data set *Gordon* ARI improves to indicate almost perfect match with ground truth. However, lack of consistency in ARI values, depending on normalization type is noticeable. No universal conclusions can be made, as even in the experiments with the same microarray and tissue type the ARI values do not follow the same trend. We can inspect if in the mentioned data sets (with high ARIs) biclustering results would also be better, and how the ARI values change for other data sets.

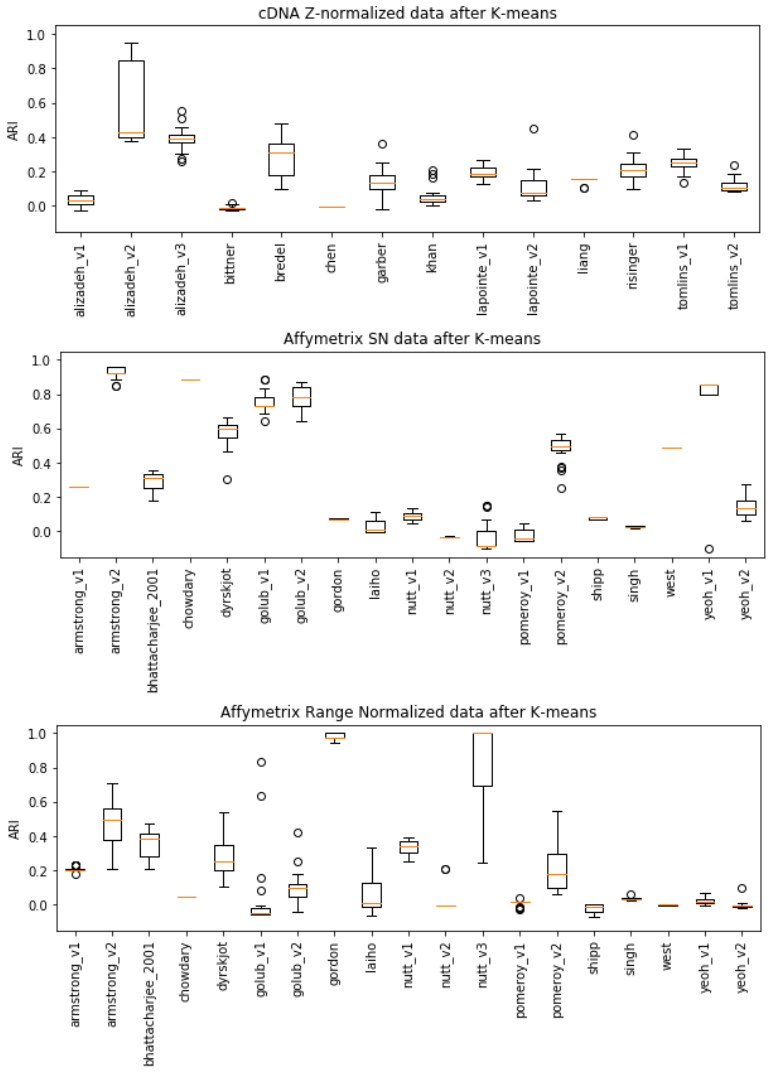

Figure 3.10: Box plots for K-means ARI over each data set. First row depicts cDNA data sets with Z-normalization. Affymetrix data sets are in the last two rows with SN and Range Normalization, respectively.

Besides ARI, results can also be examined based on SI, reflecting how similar is a sample to the samples within the same cluster, as opposed to other samples. The SI index is low even for ground truth class distribution. The SI values for cDNA are not very low, yet positive, indicating that in average samples have positive SI. However, SI values for cDNA are not very high either, telling us that clusters are not so obvious but also values are positive which means that points not assigned to the wrong clusters either. For Affymetrix data with SN normalization only *Singh* data set has high SI. In RN case average SI is a bit higher but again only one data set (*Chowdary*) stands out. These initial observations aligned with [22] where variability in performance of K-means clustering is noted, and the data sets are coarsely grouped into three categories based on the stability of the K-means results. In the same work it was shown that in general the ensemble based approach Evidence accumulation clustering, does improve results in data sets where K-means produces more stable partitions (initialization independent).

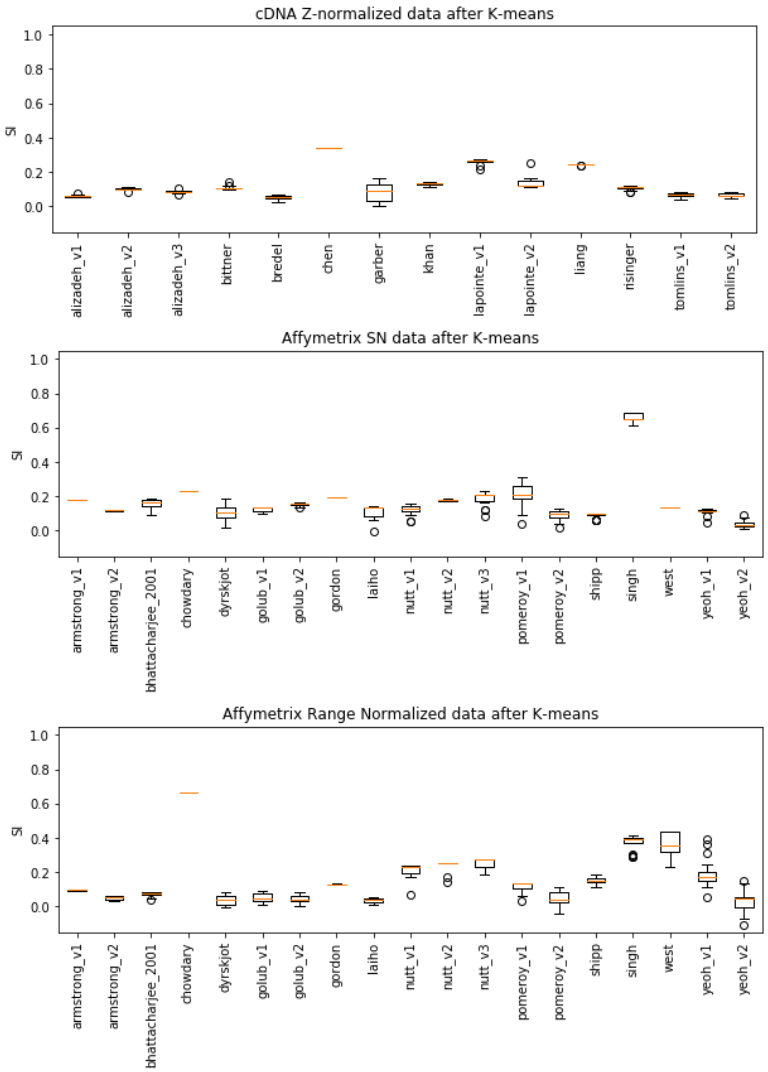

Figure 3.11: Box plots for K-means SI over each data set. First row depicts cDNA data sets with Z-normalization. Affymetrix data sets are in the last two rows with SN and Range Normalization, respectively.

3.3 Clustering Results

An intuitive insight into the clustering results for two Affymetrix and two cDNA is provided Figures 3.12-3.15. The heatmaps for Affymetrix data are organised as follows: first row represents original data matrix, second one is original matrix with rows and columns permuted according to results of Bi-Clustering algorithm, third row is data matrix after SN, forth is SN data matrix with rows and columns permuted according to results of Co-Clustering algorithm, fifth row is data matrix after RN, and the last one is RN data matrix with rows and columns permuted according to results of Co-Clustering algorithm. In case of cDNA data we have only four heatmaps since Co-Clustering considers only Z-normalization. For the sake of visualization only every other column is plotted, as suggested in [23].

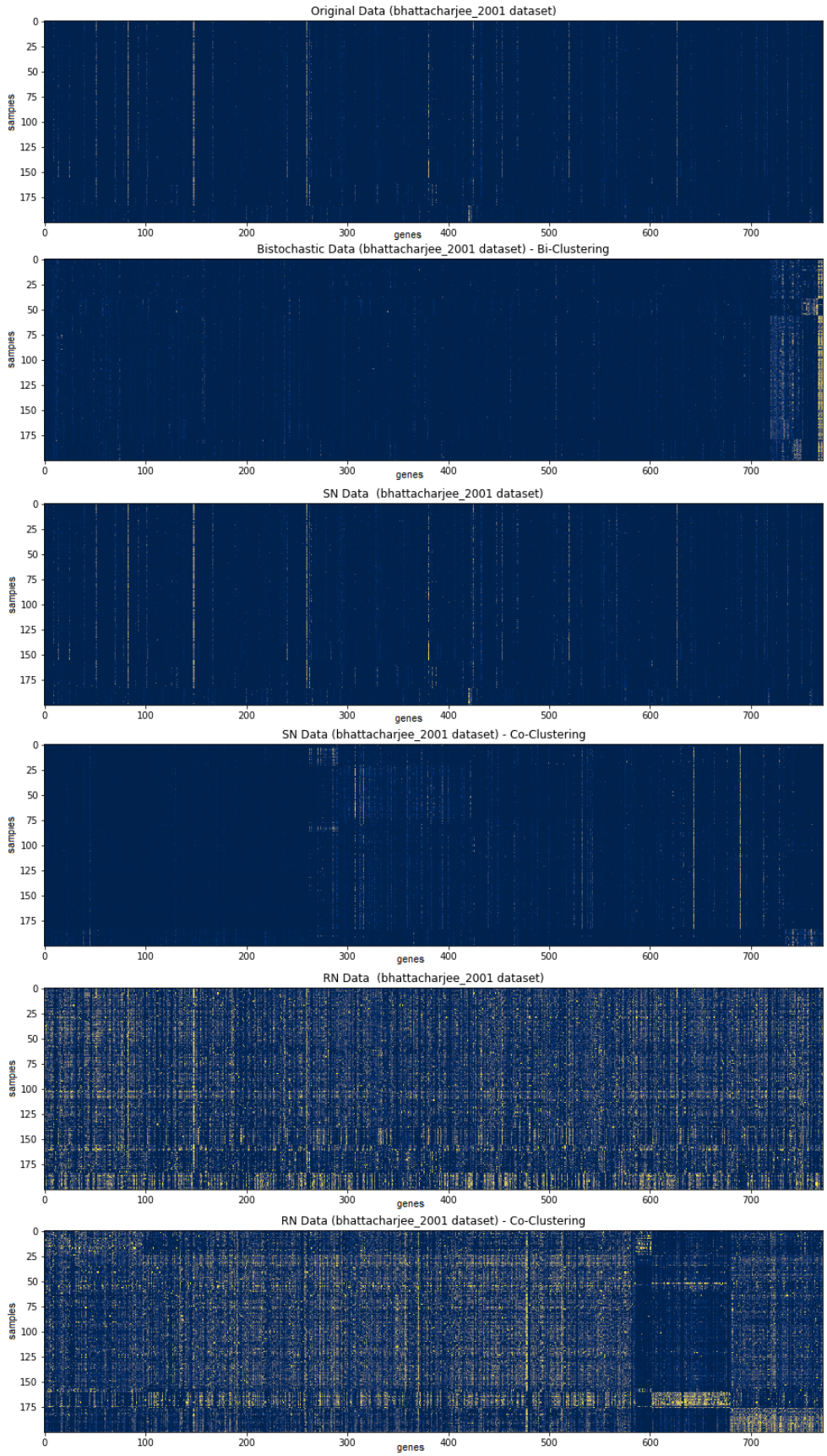

Figure 3.12: Heatmaps of *Bhattacharjee_2001* data set. First row: original data; second row: original data after Bi-Clustering; third row: data after Z-normalization; fourth row: SN data after Co-Clustering; fifth row: data after RN; last row: RN data after Co-Clustering.

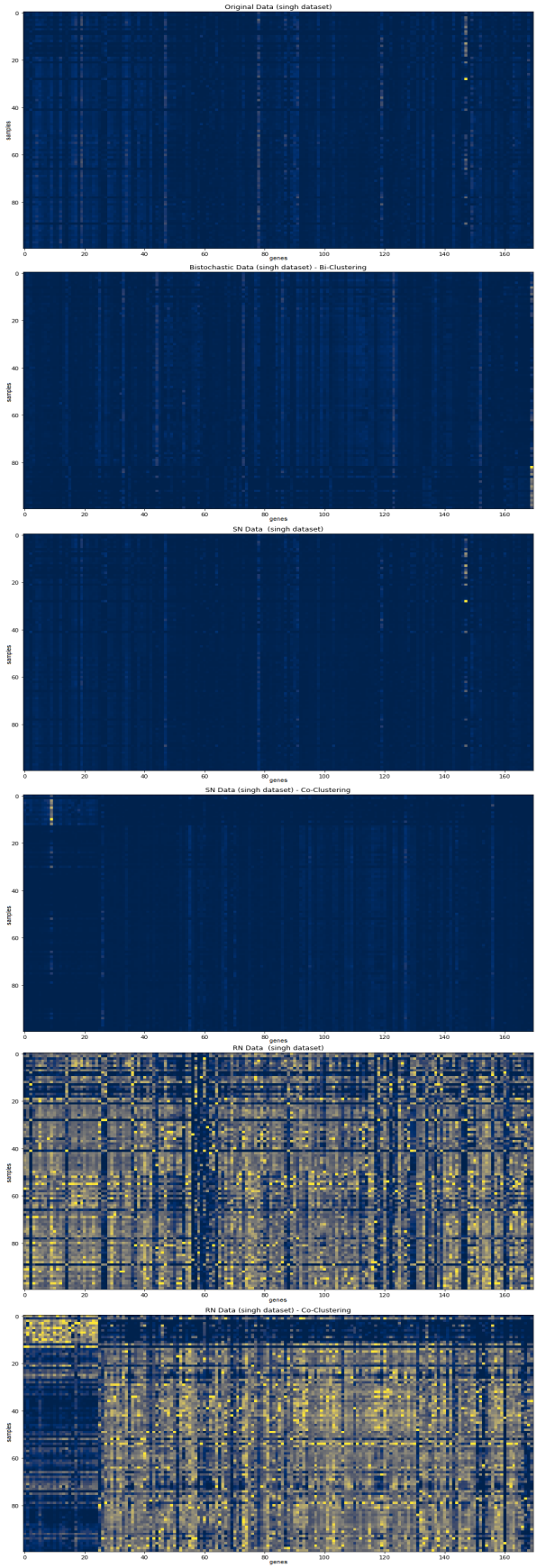

Figure 3.13: Heatmaps of *Singh* data set. First row: original data; second row: original data after Bi-Clustering; third row: data after Z-normalization; fourth row: SN data after Co-Clustering; fifth row: data after RN; last row: RN data after Co-Clustering.

The data sets presented in Figures 3.12 and 3.13 are Affymetrix. Figure 3.12 shows the clustering results for *Bhattacharjee_2001* data set. In the image with Bi-Clustering results there is a visible checkerboard pattern distinguishing biclusters on the right side of the figure. For the Co-Clustering with SN the pattern is not so obvious but we can see regularity in the middle of the matrix, with two defined coclusters. In the case of RN normalization, data are considerably different and coclusters are more obvious. There we have several well defined coclusters. In Figure 3.13, with *Singh* data set results of Bi-Clustering algorithm are not easily discernible, but Co-Clustering yields clear partitions. Again, when Range Normalization is applied the partitioning is more easily discernible.

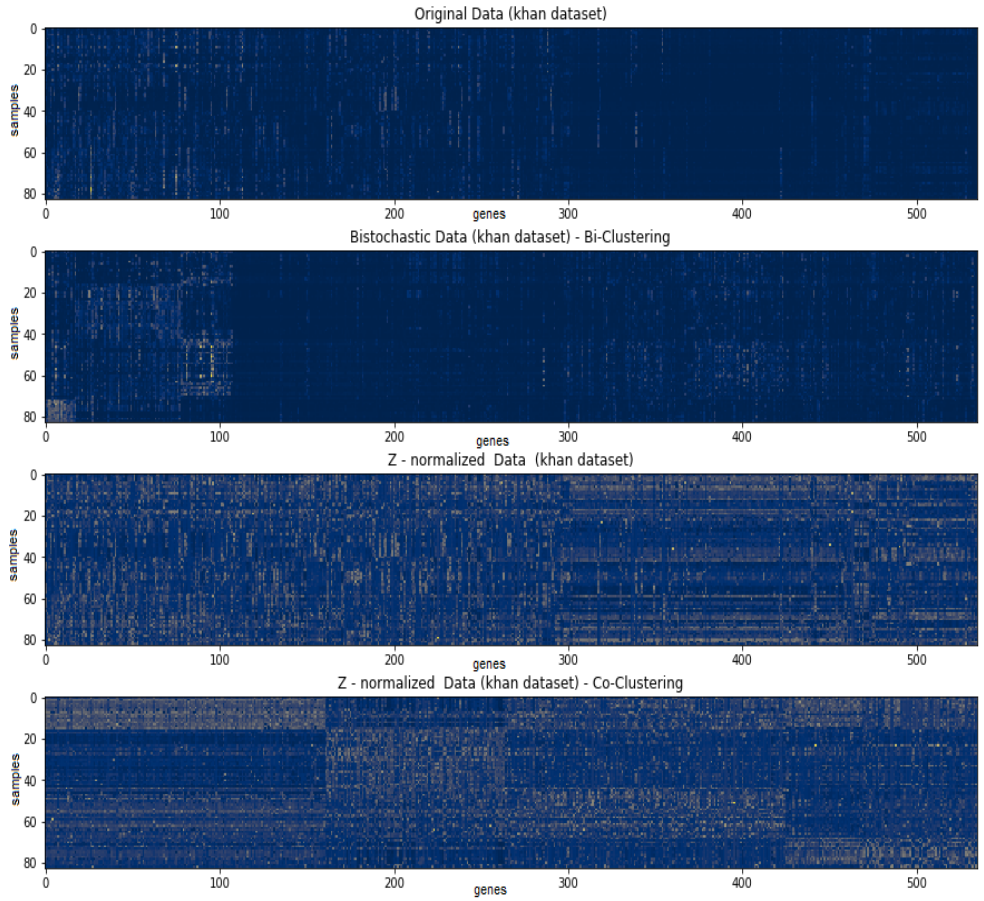

Figure 3.14: Heatmaps of *Khan* data set. First row: original data; second row: original data after Bi-Clustering; third row: data after Z-normalization; fourth row: Z-normalization data after Co-Clustering.

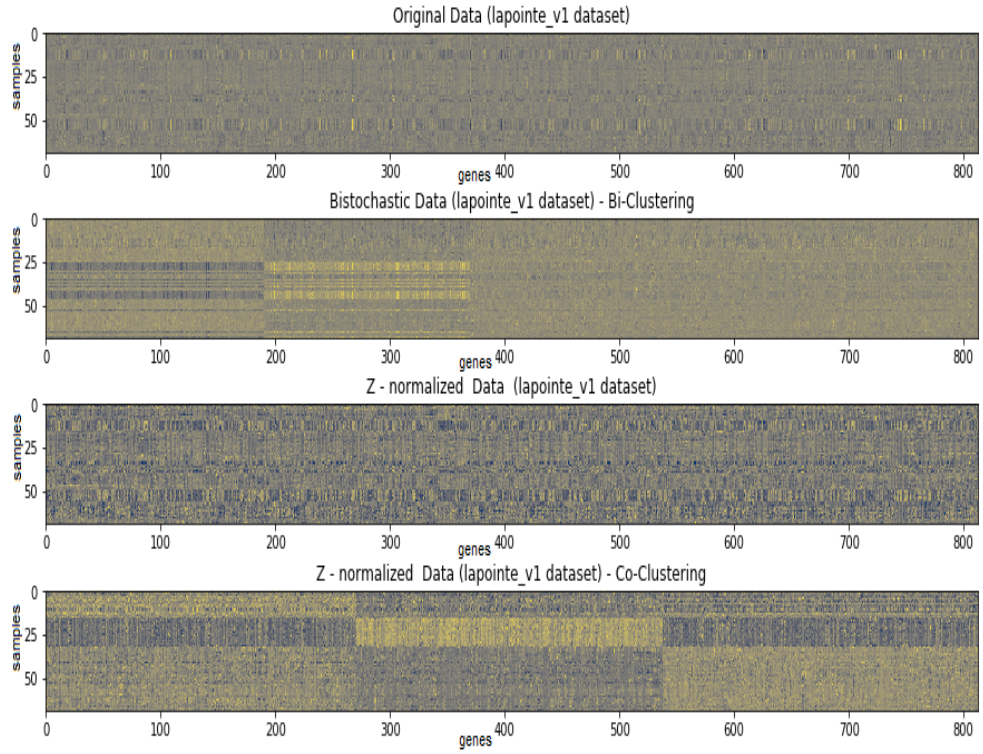

Figure 3.15: Heatmaps of *Lapointe_v1* data set. First row: original data; second row: original data after Bi-Clustering; third row: data after Z-normalization; fourth row: Z-normalization data after Co-Clustering.

In the next two figures (3.14 and 3.15) cDNA data sets are considered. In both, *Khan* and *Lapointe_v2* data sets Bi-Clustering and Co-Clustering discover compact genes/samples sets.

In order to evaluate clustering results we will consider ARI (Section 2.4.1) as an external measure of the clustering quality (compared to the ground truth) and SI (Section 2.4.2) over obtained results. Due to the random initialization of algorithms we performed twenty repetitions for each set and in the rest of this section presented average values of ARI and SI.

Average Adjusted Rand Index (ARI) for each data set and each of three clustering algorithms is presented in the Figure 3.16 separately for Affymetrix and cDNA data. None of the algorithms has consistently superior performance over others. In several data sets K-means yields higher ARI, but more often one of the biclustering algorithms (Co-Clustering or Bi-Clustering) is better. Also, it is obvious that in case of cDNA data results are in general lower. An overall performance of algorithms over certain type of microarray data can be visualized by aggregating ARIs in the form of box plots as presented in Figure 3.17. In top subfigure with cDNA data Bi-Clustering algorithm proved to be best. In the bottom left with SN Affymetrix data both median and lower quartile are highest for Co-Clustering so we can conclude that this algorithm wins. In the bottom right, with Range Normalized Affymetrix data, Bi-Clustering results seem to be the best, but still very similar with Co-Clustering scores.

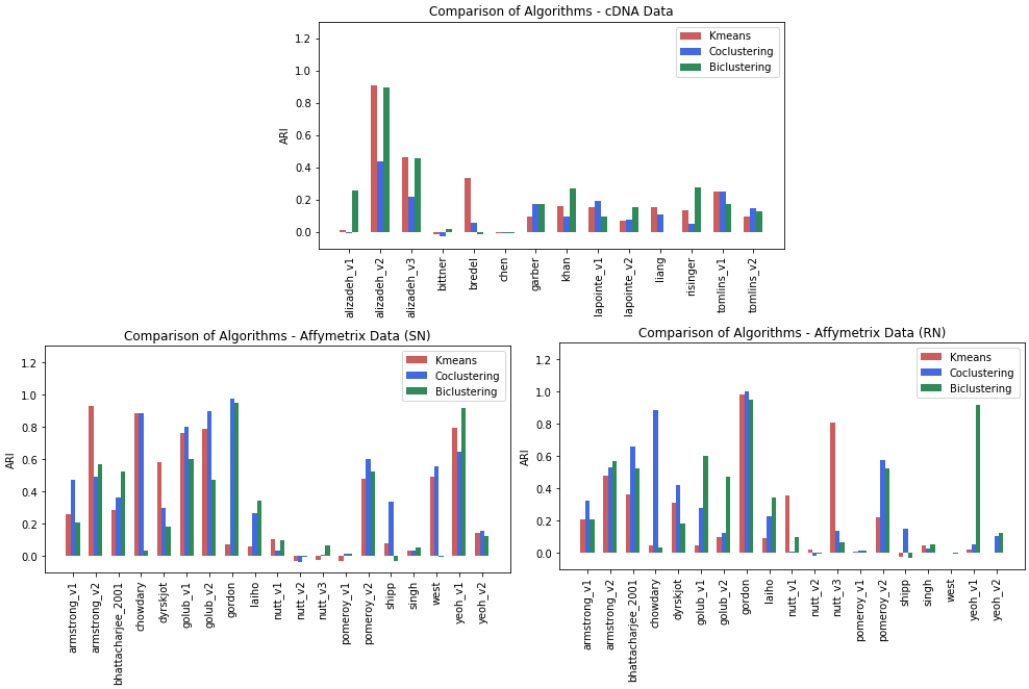

Figure 3.16: Mean ARI for different approaches for all datasets.

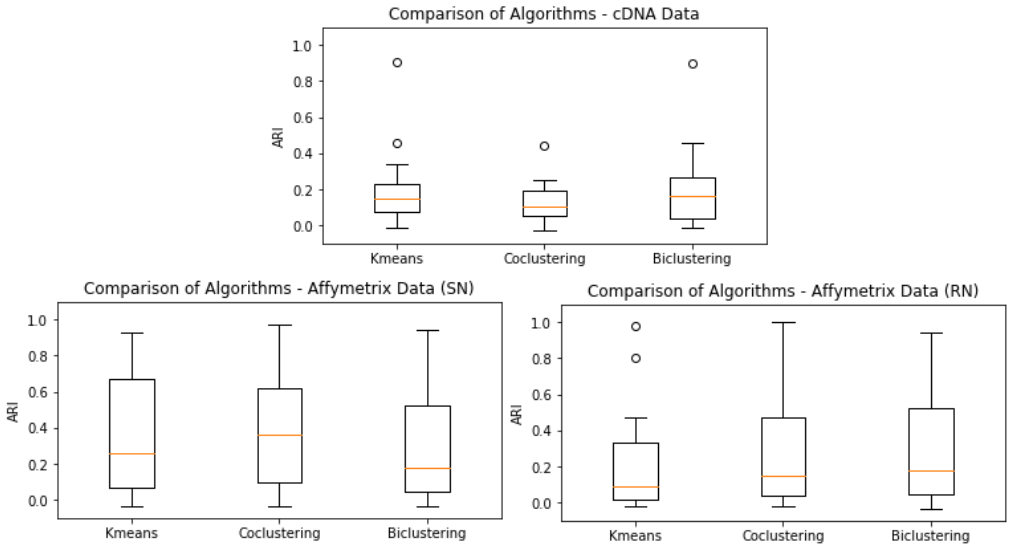

Figure 3.17: Box plots for mean ARI for different approaches aggregated over all datasets.

Next evaluation metric that is used is Silhouette Index (SI) and its results are presented in Figure 3.18. Due to the nature of this score we can also incorporate scores with ground truth labels. We can notice that for cDNA data all the values are low which indicates that the underlying structure in data is not well captured by the used similarity metrics and approaches. In the case with Affymetrix data, some sets have significant higher scores than others (e.g. *Chowdrary*). Again we consult box plots (Figure 3.19) to compare overall performances. In the upper subfigure for cDNA data, original labels have the lowest SI which indicates that ground truth labels are not followed by the consistent expression of genes in data samples. The difference between algorithms performances is negligible. Bi-Clustering stands

out in Affymetrix data for both normalizations (SN and RN). Original data again exhibits the lowest SI.

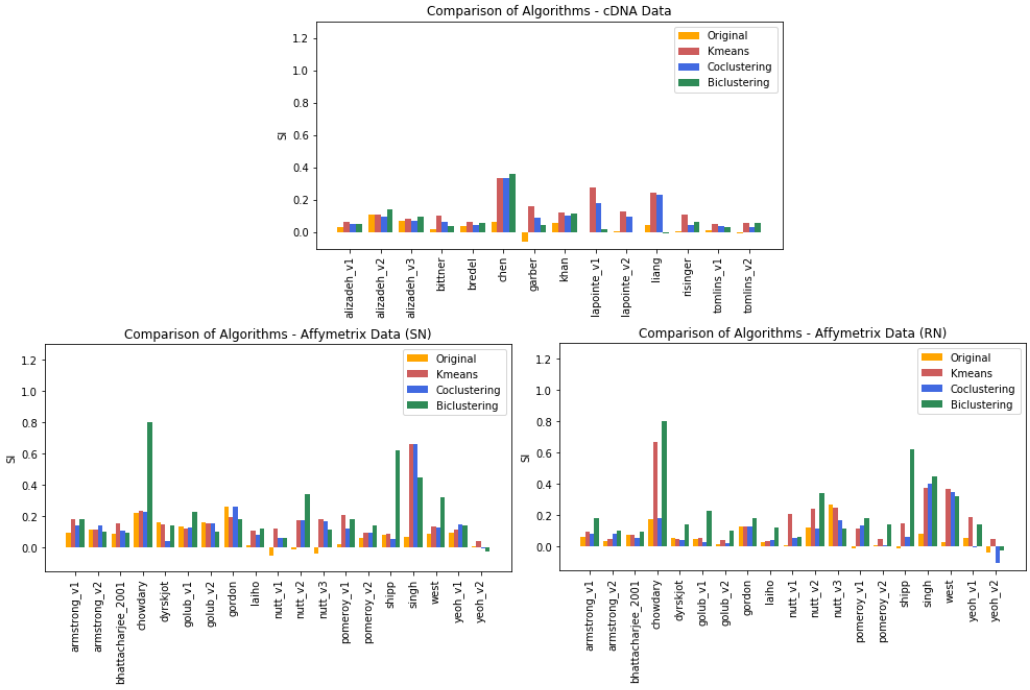

Figure 3.18: Mean SI for different approaches for all datasets.

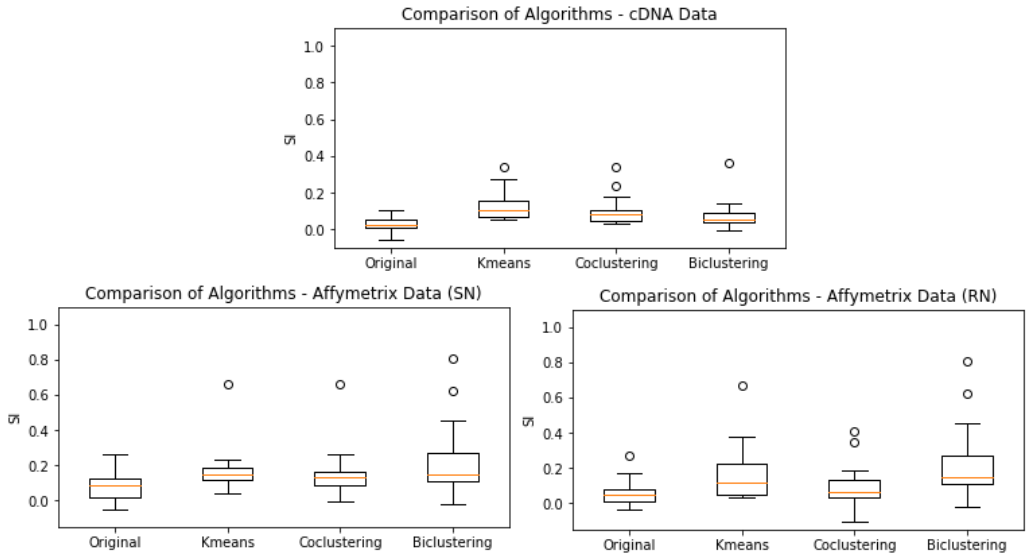

Figure 3.19: Box plots for mean SI for different approaches aggregated over all datasets.

Proposed biclustering algorithms (Co-Clustering and Bi-Clustering) produce both, row and column labels. Hence, we can also consider genes as inputs (each gene being represented by its values over all data samples). In order to compare how coherent are gene expression values for genes with the same cluster label, we have calculated the SI index for gene partitions. Results are presented in Figures 3.20 and 3.21. For cDNA data obtained SIs are quite low for both algorithms. On the other hand, Bi-Clustering

produces very high scores over the Affymetrix data for both normalization approaches, significantly beating Co-Clustering.

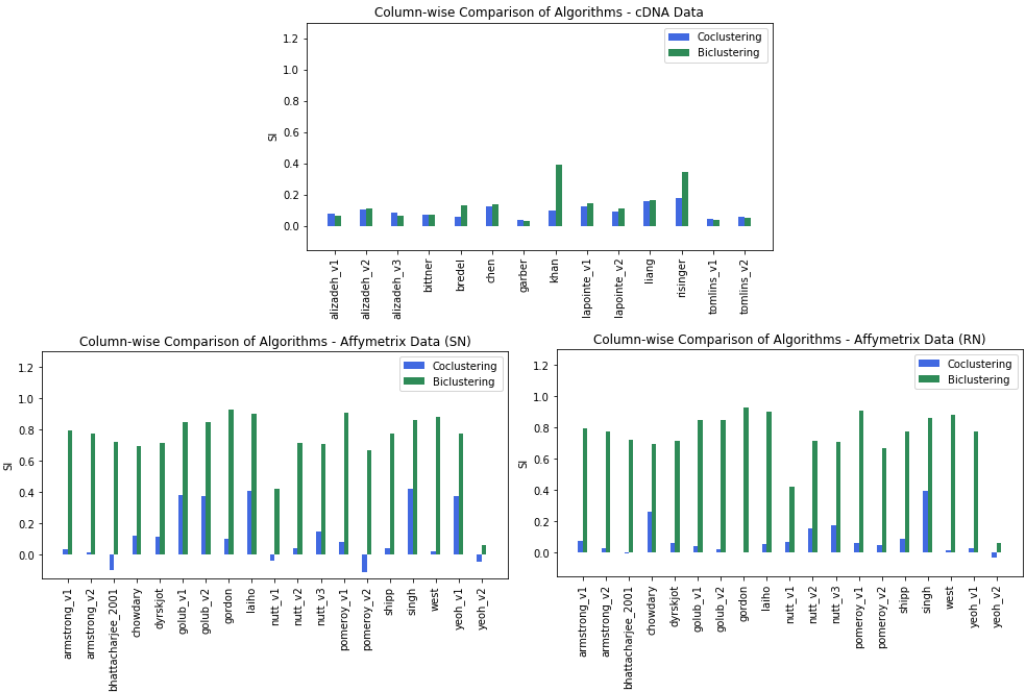

Figure 3.20: Column-wise mean SI for different approaches for all datasets.

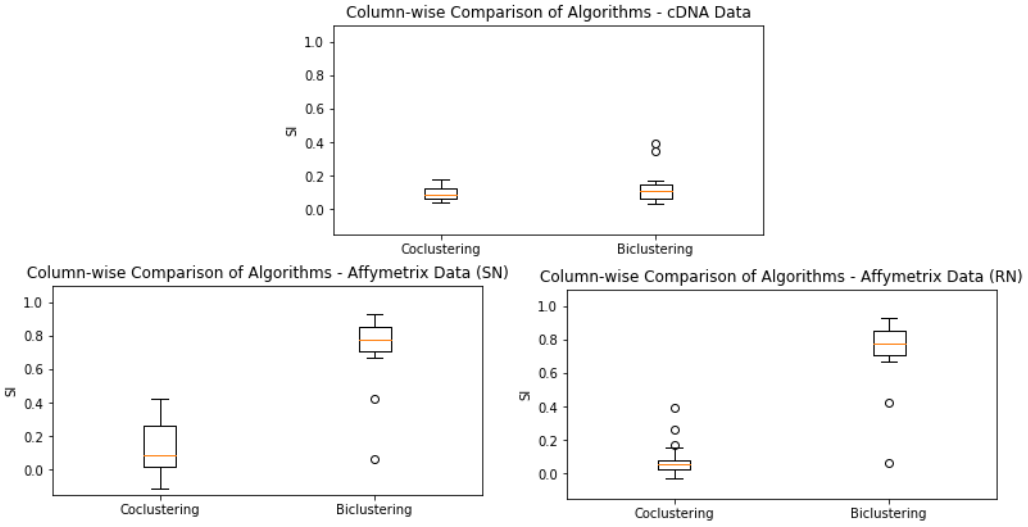

Figure 3.21: Box plots for column-wise mean SI for different approaches aggregated over all datasets.

Chapter 4

Conclusions

The aim of this thesis is to explore methods that enable simultaneous clustering of both genes and samples. In this analysis thirty-three publicly available microarray data sets are used. Nineteen of those are Affymetrix data sets, and the other fourteen are cDNA data sets. In this thesis we considered two appropriate clustering algorithms for this type of problem: Co-Clustering and Bi-Clustering. In order to illustrate performance of some conventional clustering algorithms on these data sets, K-means clustering was included. Two validation criteria used here are the ARI as an external criterion and SI as an internal evaluation criterion.

The analysis of samples within classes, as designated by the ground truth, has indicated samples' within class heterogeneity. The partition defined by the class distribution exhibits low SI. i.e. on average a small similarity of samples to the samples in the same class, as opposed to samples from the other classes. This diversity of gene expressions within the same class (same cancer type) already suggests potential difficulties in clustering of the samples, as clustering approaches exploit sample similarity to unveil the underlying data structure.

There is a noticeable difference between obtained results over two chip platforms (cDNA and Affymetrix), as the type of measurements values differs significantly. The use of normalization techniques is recommended in case of Affymetrix data, as high range of input values, gene expressions, might mask potentially relevant yet minor differences in expression of genes.

Visual inspection of data sets through heatmaps shows that expected structure (checkerboard pattern) is achieved upon clustering. In Affymetrix range normalized data sets the pattern is more obvious. As expected for the study that encompasses a larger collection of data sets of different origin and different microarray technologies, none of the examined approaches performs consistently well.

For cDNA data sets ARI values of K-means partitions are low, but the results are not very sensitive to initialization of cluster centroids. Likewise, for Bi-Clustering and Co-Clustering ARI does not improve significantly. SI values over cDNA data sets for the ground truth labels are very low, even negative in some data sets. K-means produces on average higher SI values. Bi-Clustering and Co-Clustering achieve similar low SI values. For biclustering algorithms we can also inspect gene clusters in order to evaluate the

clustering with respect to genes (features). It can be observed if the genes that are grouped together have similar expressions over all samples. In case of cDNA data sets results, this type of similarity is low as well, as measured by SI.

In Affymetrix data sets the effects of normalization as pre-processing step should be examined as well. K-means with SN data exhibits high variability of ARI values (i.e. different partitions, unstable clustering results), while in case of RN both ARI and its variability are reduced. ARIs for Biclustering algorithms are comparable for both types of normalization, SN and RN, while Co-Clustering is more sensitive to normalization type. The Silhouette Index of the partition defined by the ground truth labels yields low values. Bi-Clustering partitions have higher SI values in almost all Affymetrix sets regardless of normalization, as compared to Co-Clustering. When considering column-wise, gene clusters for Affymetrix data and both type of normalization Bi-Clustering has significantly higher values compared with Co-Clustering.

The results derived here are in accordance with those published in [8] and [22]. In these studies authors used same gene expression data sets and evaluated different clustering algorithms. The clustering results vary to a large extent depending on data set irrespective of the algorithm used. It was noticed that the performance and stability of K-means clustering on a certain data set is informative providing quick insight into needs to use more complex algorithms [22]. Another common conclusion is that due to the nature of the data, different data origins, tissue and cancer types and different data dimensionality approaches applied, some common guidance on algorithm selection could not be offered.

There are some ambiguities that should be addressed in the future work. The data sets originated from different laboratories and are provided with reduced dimensionality without availability of the original data sets. For these reasons further exploration of this relevant step in orginal data sets was not possible. A different dimensionality reduction technique might help in identification of the underlying data structure, as noise present in the data hampers the use of many similarity metrics. Also, it should be inspected if the ground truth labels are appropriate indicators of natural clusters. Some more refined labeling procedure involving cancer sub-types with substantial number of samples would benefit different machine learning approaches. The potentially promising approach is the use of semi-supervised learning.

Bibliography

[1] A. Brazma and J. Vilo. "Gene expression data analysis". In: *FEBS Letters* (2000). DOI: https://doi.org/10.1016/S0014-5793(00)01772-5.

[2] G. Govaert and M. Nadif. *Co-clustering: models, algorithms and applications*. John Wiley & Sons, 2013.

[3] J. A. Hartigan. "Direct Clustering of a Data Matrix". In: *Journal of the American Statistical Association* (1972). DOI: 10.1080/01621459.1972.10481214.

[4] Y. Kluger et al. "Spectral Biclustering of Microarray Data: Coclustering Genes and Conditions". In: *Genome research* 13.4 (2003), pp. 703–716. DOI: 10.1101/gr.648603.

[5] A. Ben-Dor et al. "Discovering Local Structure in Gene Expression Data: The Order-Preserving Submatrix Problem". In: *RECOMB '02: Proceedings of the sixth annual international conference on Computational biology* (2002), pp. 49–57. DOI: https://doi.org/10.1145/565196.565203.

[6] *Reaping the Benefits of Genomic and Proteomic Research*. 2006. URL: https://www.ncbi.nlm.nih.gov/books/NBK19861/.

[7] A. Tefferi et al. "Primer on Medical Genomics Part III: Microarray Experiments and Data Analysis". In: *Mayo Clinic Proceedings* 77 (2002), pp. 927–940. DOI: https://doi.org/10.4065/77.9.927.

[8] M.C. de Souto et al. "Clustering cancer gene expression data: a comparative study". In: *BMC Bioinformatics 9* 497 (2008). DOI: 10.1186/1471-2105-9-497.

[9] H. Jia et al. "The latest research progress on spectral clustering". In: *Neural Computing and Applications* 24.7 (2014), pp. 1477–1486.

[10] A. Y. Ng, M.I. Jordan, and Y.Weiss. "On Spectral Clustering: Analysis and an algorithm". In: *Advances in neural information processing systems* 2 (2002), pp. 849–856.

[11] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[12] M. Fiedler. "Algebraic connectivity of graphs". In: *Czechoslovak Mathematical Journal* 23 (1973), pp. 298–305. DOI: http://eudml.org/doc/12723.

[13] L. Hagen and A.B. Kahng. "New spectral methods for ratio cut partitioning and clustering". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 11 (1992), pp. 1074 –1085. DOI: 10.1109/43.159993.

[14] J. Shi and J. Malik. "Normalized cuts and image segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22 (2000), pp. 888 –905. DOI: 10.1109/34.868688.

[15] I.S. Dhillon. "Co-clustering documents and words using bipartite spectral graph partitioning". In: *KDD '01* (2001). DOI: 10.1145/502512.502550.

[16] R. Sinkhorn. "A Relationship Between Arbitrary Positive Matrices and Doubly Stochastic Matrices". In: *Ann. Math. Statist.* 35 (1964), pp. 876–879. DOI: 10.1214/aoms/1177703591.

[17] K.Y. Yeung and W. Ruzzo. "Details of the Adjusted Rand index and Clustering algorithms Supplement to the paper "An empirical study on Principal Component Analysis for clustering gene expression data" (to appear in Bioinformatics)". In: *Science* 17 (Jan. 2001).

[18] W.M. Rand. "Objective Criteria for the Evaluation of Clustering Methods". In: *Journal of the American Statistical Association* (1971), 846–850.

[19] L. Hubert and P. Arabie. "Comparing partitions". In: *Journal of Classification 2* (1985), 193–218. DOI: https://doi.org/10.1007/BF01908075.

[20] P.J. Rousseeuw. "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis". In: *Journal of Computational and Applied Mathematics* 20 (1987), pp. 53–65. DOI: https://doi.org/10.1016/0377-0427(87)90125-7.

[21] F.T. Liu, K. M. Ting, and Z. Zhou. "Isolation forest". In: *2008 eighth ieee international conference on data mining*. IEEE. 2008, pp. 413–422.

[22] I. Šašić et al. "Consensus Clustering for Cancer Gene Expression Data-Large-Scale Analysis using Evidence Accumulation Approach". In: *International Conference on Bioinformatics Models, Methods and Algorithms*. Vol. 4. SCITEPRESS. 2017, pp. 176–183.

[23] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. 2001.

Biography

Nataša Topić was born in Sombor on 24^{th} of February 1994. She finished elementary school "Nikola Tesla" in Kljajićevo. After that, she finished high school "Veljko Petrović" in Sombor. She received her Bachelor degree in Applied Mathematics in 2017 at Faculty of Sciences, University of Novi Sad, same year she continued her Master studies in the field of Data Science at the same faculty. She attended ECMI Mathematical Modelling Week in summer of 2018 where she was included on project "Diffusion and anomalous diffusion models: simulation and application to biological data". Nataša currently works at Synechron.

UNIVERZITET U NOVOM SADU
PRIRODNO-MATEMATIČKI FAKULTET
KLJUČNA DOKUMENTACIJSKA INFORMACIJA

Redni broj:
RBR
Identifikacioni broj:
IBR
Tip dokumentacije: monografska dokumentacija
TD
Tip zapisa: tekstualni štampani materijal
TZ
Vrsta rada: master rad
VR
Autor: Nataša Topić
AU
Mentor: dr Tatjana Lončar-Turukalo
MN
Naslov rada: Evaluacija algoritama klasterovanja na podacima genskih ekspresija
NR
Jezik publikacije: engleski
JP
Jezik izvoda: e
JI
Zemlja publikovanja: Republika Srbija
ZP
Uže geografsko područje: Vojvodina
UGP
Godina: 2021.
GO
Izdavač: autorski reprint
IZ
Mesto i adresa: Novi Sad, Trg Dositeja Obradovića 4
MA
Fizički opis rada: 4 poglavlja, 56 strana, 23 lit. citata, 41 figura, 4 tabele
FO
Naučna oblast: matematika
NO
Naučna disciplina: primenjena matematika
ND
Ključne reči: Klasterovanje, Spektralno Biklasterovanje, genske ekspresije
UDK
Čuva se: u biblioteci Departmana za matematiku i informatiku, Prirodno-matematičkog fakulteta, u Novom Sadu
CU
Važna napomena:

VN

Izvod: Tema ovog rada je analiza rezultata algoritama klasterovanje na 33 skupa koji sadrže podatke o ekspresijama gena raznih kancerogenih tkiva. Skupovi podataka se razlikuju po tehnologijama kojim su ekspresije procenjene (čip tehnologije engl. cDNA i Affymetrix) i vrsti tkiva, što zahteva različite pristupe u inicijalnoj obradi podataka. U tezi se evaluiraju i porede performanse: algoritma K srednjih vrednosti, spektralnog bi-klasterovanja i spektralnog ko-klasterovanja na ovim skupovima podataka.

IZ

Datum prihvatanja teme od strane NN veca:

DP

Datum odbrane:

DO

Članovi komisije:

KO

Predsednik: dr Dušan Jakovetić, vanredni profesor
Mentor: dr Tatjana Lončar-Turukalo, vanredni profesor
Član: dr Sanja Brdar, naučni saradnik

UNIVERSITY OF NOVI SAD
FACULTY OF SCIENCES
KEY WORDS DOCUMENTATION

Accession number:
ANO
Identification number:
INO
Document type: monograph type
DT
Type of record: printed text
TR
Contents code: master thesis
CC
Author: Nataša Topić
AU
Mentor: Tatjana Lončar-Turukalo, PhD
MN
Title: Clustering Gene Expression Data - Comprehensive Evaluation
XI
Language of text: English
LT
Language of abstract: e
LA
Country of publication: Republic of Serbia
CP
Locality of publication: Vojvodina
LP
Publication year: 2021.
PY
Publisher: author's reprint
PU
Publ. place: Novi Sad, Trg Dositeja Obradovića 4
PP
Physical description: 4 chapters, 56 pages, 23 references, 41 figures, 4 tables
PD
Scientific field: mathematics
SF
Scientific discipline: applied mathematics
SD
Key words: Clustering, Spectral Biclustering, Cancer Gene Expression Data
UC
Holding data: Department of Mathematics and Informatics's Library, Faculty of Sciences, Novi Sad
HD
Note:

N

Abstract: The topic of this thesis is the analysis of the results of clustering algorithms on 33 gene expression data sets of tissues with different types of cancer. Data sets differ in the chip technologies by which expressions were estimated (cDNA and Affymetrix) and tissue type, all of this requires different approaches in the initial data processing. The thesis evaluates and compares the performance of: K means algorithm, spectral bi-clustering and spectral co-clustering on these data sets. **AB**

Accepted by the Scientific Board on:

ASB

Defended:

DE

Thesis committee:

DB

Chair: Dušan Jakovetić, PhD, associate professor

Mentor: Tatajana Lončar-Turukalo, PhD, associate professor

Member: Sanja Brdar, PhD, research assistant professor