CHALMERS
UNIVERSITY OF TECHNOLOGY

Co-clustering of Tensor Data Using Sparse Tensor Factorisation

Master's thesis in Engineering Mathematics and Computational Science

SELMA TABAKOVIC

Department of Mathematical Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

Master's thesis 2020

Co-clustering of Tensor Data Using Sparse Tensor Factorisation

SELMA TABAKOVIC

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Chalmers University of Technology
Gothenburg, Sweden 2020

Co-clustering of Tensor Data Using Sparse Tensor Factorisation
SELMA TABAKOVIC

Supervisor: Felix Held, Department of Mathematical Sciences
Examiner: Marina Axelson-Fisk, Department of Mathematical Sciences

Master's Thesis 2020
Department of Mathematical Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Description of the picture on the cover page (if applicable)

Typeset in LaTeX
Gothenburg, Sweden 2020

Co-clustering of Tensor Data Using Sparse Tensor Factorisation

SELMA TABAKOVIC
Department of Mathematical Sciences
Chalmers University of Technology and University of Gothenburg

Abstract

With the ever increasing amounts of data generated from new sources and scientific methods, e.g. high throughput genome sequencing methods in bioinformatics, powerful tools for exploratory data analysis are required. One such tool is clustering, i.e. grouping together coherent observations in data, which is important for categorising vast amounts of observations into a more manageable format for further analysis. However, this task is subject to new challenges as tensor data, i.e. multidimensional data, has become a frequent occurrence in many applications. For tensor data, a clustering approach called co-clustering in particular has recently attracted research attention. Co-clustering means that the clustering is performed on all of the tensor dimensions simultaneously, which enables the detection of joint data expressions that only occur under special circumstances.

In this thesis, two methods for co-clustering of tensor data using sparse CP decompositions are proposed. The motivation behind using a tensor factorisation with enforced sparsity is that it can enable the extraction of the most relevant data from the tensor, whilst reducing noise. The first method, called the sCP-S, considers the sign pattern in the vectors, obtained from a sparse CP decomposition, to determine the clustering. The second method instead uses hierarchical clustering on the sparse CP decomposition vectors, and is named sCP-HC. The two methods were compared on simulated data and the more flexible sCP-HC was tested thoroughly on more advanced simulated data sets. The types of predefined co-clusters that can be detected, and the stability of co-cluster detection under perturbations of the input data, were both investigated prior to applying the sCP-HC on real data. These evaluations have been performed through computer simulations on simulated data sets, along with application on a real genomic tensor data set.

The obtained results from the simulations show that the sCP-HC has the potential to detect several types of additive coherent co-clusters. Additionally, the stability simulations show that the sCP-HC is quite consistent in its co-clustering, even in the presence of considerable noise. Applying the sCP-HC to real genomic data, several interesting co-clusters were obtained, which can be used for further analysis. As such, this work concludes that the sCP-HC is a useful tool for detecting coherent co-clusters in tensor data, and for exploratory data analysis.

Keywords: Co-clustering, Tensor decomposition, CANDECOMP/PARAFAC decomposition, Sparsity, Agglomerative hierarchical clustering.

Acknowledgements

Firstly, a heartfelt thanks is given to my supervisor Felix Held who has given his full support throughout the project. He has provided me with many insights and much help during this project. Secondly, I also wish to thank Rebecka Jörnsten for helpful discussions during the course of this thesis, and Marina Axelson-Fisk for being my examiner. Final thanks goes to Nelander Lab for providing me with the genomic tensor data set.

Selma Tabakovic, Gothenburg, August 2020

Contents

List of Figures

List of Tables

1

Introduction

In this chapter, the background of the thesis will be given, elaborating on the context of the study. The chapter will then present some related work, including work that has inspired this thesis along with other approaches to tensor clustering. Finally, the aims and limitations of this thesis will be clarified.

1.1 Background

In recent years the amounts of data that are generated, processed and stored have increased rapidly, up to volumes of tens of zettabytes each year [1]. Much of the stored data can be classified as big data, i.e. large data sets with many observations or variables. These data sets have become more common due to new methods, e.g. high throughput genome sequencing methods in bioinformatics [2], and from new types of data being stored, e.g. metadata captured from sensors [3]. Big data sets are often problematic due to having missing values, having data coming from different sources, and being noisy.

Most commonly data is stored in matrices, but in a several fields, e.g. neuroscience [4], bioinformatics [5], data mining [6], and signal processing [7], the data is often stored in multi-dimensional arrays, called *tensors*. Tensors may be thought of a generalisation of matrices to higher dimensions. An example of a typical tensor data set found in bioinformatics is in Figure 1.1. The tensor shows gene expression measurements across different tissues in a group of patients in a three-dimensional fashion. One axis represents genes, the other the patients and the third the different types of tissues.

A tool commonly used in analysis of big data sets is matrix factorisation. It is used to decompose a matrix into a product of matrices, which may enable an easier interpretation of the data, or be used for dimensionality reduction. Some examples of well-known matrix factorisation methods are singular value decomposition (SVD), QR decomposition and LU decomposition. Analogously, tensors can be decomposed into a product of often smaller tensors. Popular tensor decompositions are the CANDECOMP/PARAFAC (CP) decomposition, Tucker decomposition [8], and tensor-train decomposition [9]. The foundation of tensor factorisation was laid in the beginning of the 20th century, but just recently tensor factorisation has become a topic of great interest in many different fields. This renewed interest might

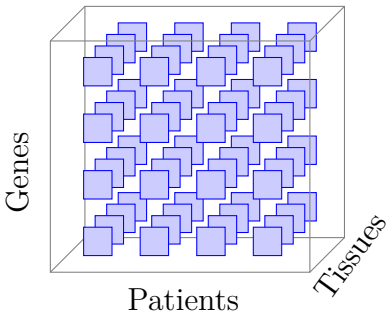


Figure 1.1: *Example of three dimensional tensor data found in bioinformatics. It contains gene expression measurements of different tissues in a group of patients.*

be explained by, for example, the increase in computer power and advancements within machine learning [8].

Due to the noisy nature of big data, the assumption of low rank is often made. This means that one assumes that the data in reality has a simpler structure than the observed data might suggest. One strategy for extracting the underlying "true" data is using a low-rank approximation. For matrices, rank is uniquely defined, and can be obtained by the SVD. Moreover, the best rank-K approximation is given by the truncated SVD. For tensors, however, there are multiple definitions of rank, meaning that there are multiple ways of defining what low-rank means. Another strategy for handling the noise and finding structure, is to enforce sparsity in the decomposition. By enforcing sparsity, the true, sparse, structure of the data can be isolated from the noise.

In many big data applications, it is interesting to investigate if the data forms clusters. While most clustering algorithms are generally good at identifying global structures, they can have difficulties in finding localised groups, i.e. patterns seen only in a localised part of the data matrix [10]. This issue is addressed by *bi-clustering*, which is a method that clusters the columns and rows of a matrix simultaneously, which enables it to find more localised patterns. For example, bi-clustering can be used to discover combinations of significantly expressed genes and varying conditions [11]. Analogously, for tensors this technique goes under the name of *co-clustering* [10].

SVD-based approaches for detecting bi-clustering have been broadly used [12]. One such method for bi-clustering, that utilises the structure of a sparse SVD, is described in Lee et. al [11]. With that paper as a starting point, and motivated by the increasing interest in tensor data, this thesis instead studies the feasibility of co-clustering on sparse tensor decompositions. The details of this study will be elaborated in the following subsection, on the thesis' aim and limitations.

1.2 Related Work

There are generally two approaches in co-clustering: Either by clustering on the tensor directly, or by clustering the different modes of either the tensor or its decomposition. In this section several different works within these categories will be

highlighted, using the terminology commonly used within the field. This terminology will be more formally defined in Chapter 2, so readers that are unfamiliar with the terminology are suggested to read there before engaging in this section.

In Zhao et al. [10], a co-clustering method based on linear groupings in the factor matrices of the Tucker decomposition or CP decomposition is proposed. Their work highlights several key aspects that are also used in this work, e.g. the low-rank approximation for the tensor decomposition in order to reduce noise. However, the two main differences between their method, and the two methods developed in this thesis, are that Zhao et al.'s methods do not use sparse tensor decompositions, and they use another procedure to find co-clusters in the factor matrices, namely via the the linear grouping algorithm [13].

An interesting approach to co-clustering, using constrained multi-linear decompositions with sparse latent factors, is considered in Papalexakis et al. [14]. Their work differs from this thesis in e.g. the way a co-cluster is determined. In both works, sparse CP decompositions are made to produce an approximation of the original data tensor with the help of factors vectors. However, their work determines co-clusters from the product of these vectors, i.e. the reconstructed tensor, whereas this work first considers the factor vectors individually, and then combines information across factor vectors, to determine the final co-clusters.

A different approach to tensor clustering is found in Jegelka el al. [15], where an approximation algorithm that clusters on the data tensor directly is used. Their method does not involve tensor decompositions or enforced sparsity. However, their work holds likeness to this thesis in that they also consider clustering per tensor mode, whereafter these clusterings are combined into co-clusters with a Cartesian product.

In Lee et al. [11], bi-clustering is performed on matrices via sparse SVD. Although not performed on tensor data, this paper laid the foundation for, and inspired, later publications on co-clustering via sparse factorisations, e.g. [14, 16]. The work of Lee et al. is as such a significant source of inspiration for this thesis.

Whilst on the topic of biclustering, Yang et al. [17] used SVD in conjunction with a biclustering method based on agglomerative hierarchical clustering. Whereas their work concerns matrix data, the application area and methodology also share similarity to this thesis.

1.3 Aim and Limitations

The aim of this thesis, is to investigate if matrix bi-clustering in the style of Lee et al. [11] can be successfully generalised to tensors using sparse tensor decomposition methods. Two sparse tensor decomposition methods, one based on Tucker decomposition and the other one based on CP decomposition, as described in Allen [18] have been considered. However, in the end the sparse CP decomposition was deemed more suitable, since it is easier to interpret. Therefore the co-clustering methods were only developed for this sparse decomposition. The presentation in this thesis

will focus on the CP decomposition.

In this work, two co-clustering methods have been developed. The first method is called sCP-S and is based on considering the sign of the factors that make up the CP decomposition. The second method is called sCP-HC and involves hierarchical clustering of the same factors. The two methods have been evaluated through a series of simulation studies. The first simulation study aimed to clarify what kind of co-clusters can be found, and was performed on a selection of possible pre-defined co-cluster patterns. The second study compared the differences between sCP-S and sCP-HC. In the third type of simulations, the clustering stability was investigated. Lastly, the sCP-HC was applied on real world data.

2

Theory

In this chapter, tensors will be defined and common tensor operations will be reviewed. Moreover, a few tensor and matrix decomposition methods will be presented. Finally, different categories of co-clusters will be defined, along with a description of two scoring functions for evaluating clusters. Much of this chapter is based on Kolda and Balder [8], especially when it comes to theory concerning the operations of matrices and tensors.

2.1 Basics on Tensors

A tensor can be represented as a multidimensional array, and may be thought of as a generalisation of a matrix. More formally, a tensor $\boldsymbol{\mathcal{X}}$ is defined as an element of the product of N real vector spaces, more precisely

$$\boldsymbol{\mathcal{X}} = \{x_{i_1 i_2 \ldots i_N},\ i_k = 1, \ldots, I_k\} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}, \tag{2.1}$$

where N is known as the *order*, *mode* or *way*. A tensor of order N is therefore sometimes referred to as an N-way tensor, or an Nth-order tensor. However, for $N \leq 2$, it is more natural to refer to them are as matrices, vectors or scalars, in the cases $N = 2$, $N = 1$ and $N = 0$, respectively. Tensors of order three, or higher, on other other hand are often simply called higher-order tensors. In Figure 2.1 some examples of tensors are illustrated.

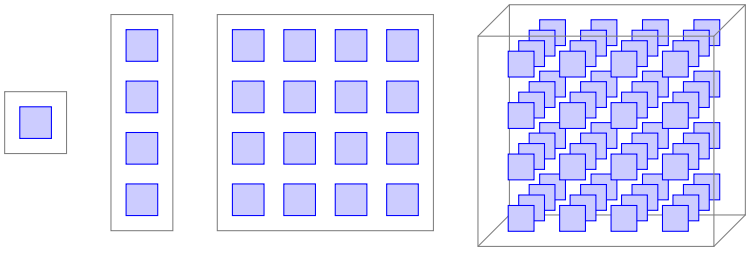


Figure 2.1: *Examples of tensors of different order. A scalar, a vector, a matrix, and a three-way tensor are illustrated, in order from left to right.*

The notation of the tensor depends on the order, as summarised in Table 2.1. This notation is adopted from Kolda and Balder [8]. Higher order tensors are denoted in bold Euler script letters, matrices are denoted in bold capital letters, vectors

are denotes as bold lowercase letters, and scalars are denoted as lowercase letters. The elements of a tensor $\mathcal{X}$ are denoted as x_{ijk}, x_{ij} and x_i for third-order tensors, matrices and vectors, respectively.

Table 2.1: *Notation for tensors of different orders N.*

Tensors of order N	**Notation**
Higher-order tensor, $N \geq 3$	$\mathcal{X}$
Matrix, $N = 2$	$\mathbf{X}$
Vector, $N = 1$	$\mathbf{x}$
Scalar, $N = 0$	x

By fixing one or more indices of the tensors, *subtensors* are created. A colon is used to indicate that all elements in that dimension are included. For example, $\mathbf{x}_{i:}$ is the ith row of a matrix $\mathbf{X}$. A *fiber* is defined by fixing all indices in a tensor but one. A mode-n fiber is a fiber that extends in the nth dimension. In the case of matrices, the mode-1 fibers are the rows and mode-2 fibers are the columns of a matrix. For tensors, the mode-1, mode-2 and mode-3 fibers are called rows, columns and tubes, respectively. These are, in the same ordering, denoted by $x_{:jk}$, $x_{i:k}$ and $x_{ij:}$. Similarly, *slices* are obtained by fixing every index but two. The different types of slices in $\mathcal{X}$ are called horizontal, lateral and frontal depending on which dimension is kept constant. They are given as $\mathbf{X}_{i::}$, $\mathbf{X}_{:j:}$ and $\mathbf{X}_{::k}$, respectively. In Figure 2.2 and Figure 2.3 all different fibers and slices, respectively, are seen in the case of a three-way tensor.

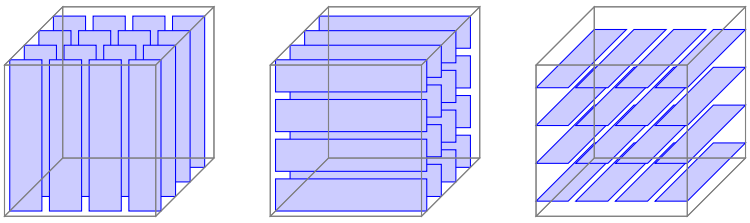


Figure 2.2: *Fibers for a three-way $4 \times 4 \times 4$ tensor, in order column, row and tube fibers. These correspond to mode-1, mode-2 and mode-3, respectively.*

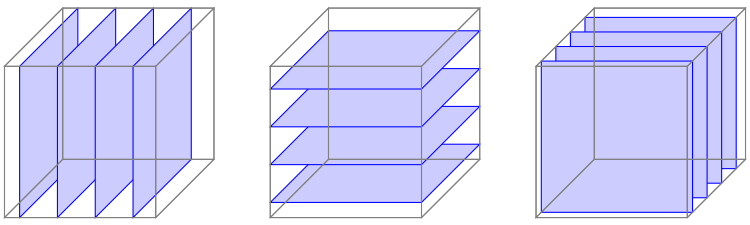


Figure 2.3: *Slices for a three-way $4 \times 4 \times 4$ tensor, in order lateral, horizontal and frontal fibers.*

2.2 Tensor Operations

In this section, a review of common tensor operations will be presented. Some of them are natural extensions from matrices and vectors to tensors of higher order. For a more extensive description of tensor operations, see Kolda and Balder [8].

2.2.1 Matricisation

Matricisation, also flattening, or unfolding, is an operation on a tensor that reorders its element into a matrix. It may be regarded as a generalisation of vectorisation, which is a matrix operation that converts matrix into a vector. In this project, only mode-n matricisation is considered, which is a special case of matricisation. For a more general description see [19].

The *Mode-n matricisation* of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is denoted by $\mathbf{X}_{(n)}$, where $\mathbf{X}_{(n)} \in \mathbb{R}^{I_n \times (I_1 \times I_2 \times \ldots I_{n-1} \times I_{n+1} \times \ldots I_N)}$. The operation can be described as a mapping where the tensor element on position $(i_1, i_2, \ldots, i_N)$ is mapped to a matrix element on position (i_n, j), where

$$j = 1 + \sum_{\substack{k=1 \\ k \neq n}}^{N} (i_k - 1) J_k \quad \text{and} \quad J_k = \prod_{\substack{m=1 \\ m \neq n}}^{k-1} I_m \,. \tag{2.2}$$

This means that the mode-n fibers of the $\mathcal{X}$ are rearranged, so that the resulting matrix $\mathbf{X}_{(n)}$ has them as columns. An illustration where this is apparent, in the case of a order-three tensor, is provided in Figure 2.4.

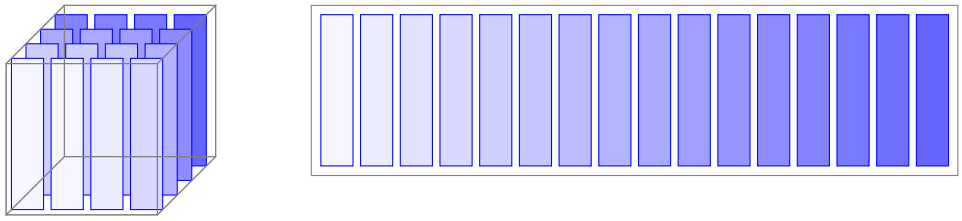


Figure 2.4: *Mode-1 matricisation for a three-way $4 \times 4 \times 4$ tensor to a 4×16 matrix. The tensor's mode-1 fibers (columns) are arranged into the resulting matrix's columns. The colours clarify the ordering of the columns.*

There are different conventions of the ordering of the fibers. However, in general the ordering does not matter as long as the permutations of the fibers are consistent through the calculations.

2.2.2 Inner Product and Norm

Given two tensors $\mathcal{X}$ and $\mathcal{Y}$, where $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, the *inner product* between them is defined as [8]

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{i_1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} x_{i_1 i_2 \ldots i_N} y_{i_1 i_2 \ldots i_N} \,. \tag{2.3}$$

Note the similarity between this definition and the standard inner product of vectors. When $\langle \mathcal{X}, \mathcal{Y} \rangle = 0$ the tensors are said to be orthogonal.

The *norm* of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is given by

$$\|\mathcal{X}\| = \sqrt{\langle \mathcal{X}, \mathcal{X} \rangle} = \sqrt{\sum_{i_1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} x_{i_1 i_2 \ldots i_N}^2} \,. \tag{2.4}$$

In the case of a matrix, this is equivalent to the Frobenius norm.

2.2.3 Outer Product of Vectors

The outer product of two vectors $\mathbf{a}$ and $\mathbf{b}$ results in a matrix $\mathbf{X}$, and is conventionally written as $\mathbf{X} = \mathbf{a}\mathbf{b}^\top$, where $\mathbf{b}^\top$ denotes the transpose of $\mathbf{b}$. To generalise this operation to higher dimensions, a new notation is needed. Let therefore the symbol $\circ$ denote the outer product.

The *outer product* of the vectors $\mathbf{a}^{(n)} \in \mathbb{R}^{I_n}$ $(n = 1, \ldots, N)$ is denoted

$$\mathcal{X} = \mathbf{a}^{(1)} \circ \mathbf{a}^{(2)} \circ \cdots \circ \mathbf{a}^{(N)} \,, \tag{2.5}$$

where $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, and is element-wise defined as

$$x_{i_1 i_1 \ldots i_N} = a_{i_1}^{(1)} a_{i_2}^{(2)} \ldots a_{i_N}^{(N)} \,. \tag{2.6}$$

This means that every element of the tensor is decomposed as a product of the corresponding vector elements. Note that the order of the tensor is given by the number of vectors involved in the outer product. This is illustrated in Figure 2.5, where a three-way tensor is obtained from the outer product of three vectors.

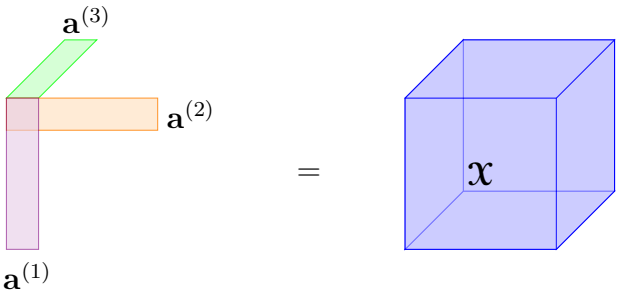


Figure 2.5: *Outer product of three vectors* $\mathbf{a}^{(1)}$, $\mathbf{a}^{(2)}$, *and* $\mathbf{a}^{(3)}$ *resulting in a three-way tensor* $\mathcal{X}$, *i.e.* $\mathbf{a}^{(1)} \circ \mathbf{a}^{(2)} \circ \mathbf{a}^{(3)} = \mathcal{X}$. *Elements in the tensor are given by* $x_{ijk} = a_i^{(1)} a_j^{(2)} a_k^{(3)}$.

2.2.4 The n-Mode Product

The n-mode product is a multiplication that is defined for both a tensor and a matrix, and a tensor and a vector. In this section both definitions will be given.

The *n-mode product* of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \ldots I_N}$ and a matrix $\mathbf{U} \in \mathbb{R}^{J_n \times I_n}$ is denoted by $\mathcal{Y} = \mathcal{X} \times_n \mathbf{U}$, and defined element-wise as [8]

$$(\mathcal{X} \times_n \mathbf{U})_{i_1 \ldots i_{n-1} j_n i_{n+1} \ldots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \ldots i_{n-1} i_n i_{n+1} \ldots i_N} u_{j_n i_n} \,. \tag{2.7}$$

The resulting tensor $\mathcal{Y} \in \mathbb{R}^{I_1 \times \ldots I_{n-1} \times J_n \times I_{n+1} \times \cdots \times I_N}$, i.e. it is of order N.

Another interpretation of the n-mode product is that each mode-n fiber of $\mathcal{X}$ is multiplied with the matrix $\mathbf{U}$. This implies that it also can be expressed in terms of mode-n matricisations of $\mathcal{X}$, i.e.

$$\mathcal{Y} = (\mathcal{X} \times_n \mathbf{U}) \quad \Longleftrightarrow \quad \mathbf{Y}_{(n)} = \mathbf{U}\mathbf{X}_{(n)} \,. \tag{2.8}$$

There are two notable properties of the n-mode product [20]. Let $\mathbf{A} \in \mathbb{R}^{J_n \times I_n}$ and $\mathbf{B} \in \mathbb{R}^{J_m \times I_n}$, and $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$. If $n \neq m$, then

$$(\mathcal{X} \times_n \mathbf{A}) \times_m \mathbf{B} = (\mathcal{X} \times_m \mathbf{B}) \times_n \mathbf{A} = \mathcal{X} \times_n \mathbf{A} \times_m \mathbf{B} \,, \tag{2.9}$$

i.e. it is commutative. If the modes are the same ($n = m$), then the following holds

$$(\mathcal{X} \times_n \mathbf{A}) \times_n \mathbf{B} = \mathcal{X} \times_n (\mathbf{B}\mathbf{A}) \,. \tag{2.10}$$

The *n-mode (vector) product* of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and a vector $\mathbf{v} \in \mathbb{R}^{I_n}$ is denoted by $\mathcal{Y} = \mathcal{X} \times_n \mathbf{v}$, and defined element-wise as

$$(\mathcal{X} \times_n \mathbf{v})_{i_1 \ldots i_{n-1} i_{n+1} \ldots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \ldots i_{n-1} i_n i_{n+1} \ldots i_N} v_{i_n} \,. \tag{2.11}$$

The resulting tensor $\mathcal{Y} \in \mathbb{R}^{I_1 \times \ldots I_{n-1} \times I_{n+1} \times \cdots \times I_N}$, i.e. it is of order $N - 1$. This operation can be interpreted as that the inner product is taken between each mode-n fiber and the vector $\mathbf{v}$.

Unlike the n-mode product for matrices, the n-mode product for vectors is not commutative. It does, however, possess another interesting property. Assume that $\mathbf{a} \in \mathbb{R}^{I_n}$, $\mathbf{b} \in \mathbb{R}^{I_m}$, and $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, then

$$(\mathcal{X} \times_n \mathbf{a}) \times_m \mathbf{b} = (\mathcal{X} \times_m \mathbf{b}) \times_{n-1} \mathbf{a} = \mathcal{X} \times_n \mathbf{a} \times_m \mathbf{b} \,, \tag{2.12}$$

for $n > m$. The operation is not defined for the case $n = m$.

2.3 Tensor Decomposition

There are many different decomposition methods for matrices and higher-order tensors. In this section the well-known matrix decomposition singular value decomposition (SVD) will be reviewed, and one of the more famous higher-order tensor decompositions will be presented, namely the CANDECOMP/PARAFAC (CP) decomposition. Furthermore, a sparse CP decomposition developed by Allen [18] will also be described. With regard to the application in the rest of this thesis, the higher-order tensor decompositions will be presented mainly for tensors of order three. As mentioned in the introduction, unlike matrices there are several definitions of the rank of higher-order tensors. In this section, a definition of the rank originating from the CP decomposition will be given.

2.3.1 Singular Value Decomposition

The Singular Value Decomposition (SVD) is a decomposition method that decomposes a matrix into a product of three matrices. It is widely used in many different applications, e.g. data compression and high-dimensional data visualisation.

The *SVD* of a $I \times J$ matrix $\mathbf{X}$ is of the form

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}} = \sum_{r=1}^{R} \sigma_r \mathbf{u}_r \circ \mathbf{v}_r \,, \tag{2.13}$$

where R is the matrix rank, $\mathbf{U} = (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_R)$ is an orthonormal $J \times R$ matrix, $\mathbf{\Sigma} = \mathrm{diag}(\sigma_1, \ldots, \sigma_R)$ is a diagonal matrix with positive values, and $\mathbf{V} = (\mathbf{v}_1, \mathbf{v}_1, \ldots, \mathbf{v}_R)$ is an orthonormal $I \times R$ matrix. The σ_r's are called the singular values of $\mathbf{X}$, and the columns of $\mathbf{U}$ and columns of $\mathbf{V}$ are referred to as the left-singular vectors and right-singular vectors of $\mathbf{X}$, respectively.

The SVD as described above is not unique. Often the singular values are ordered in descending order and assumed to be positive, i.e. $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_R$. Then $\mathbf{\Sigma}$ is uniquely determined by $\mathbf{X}$ [21]. Henceforth these assumptions will be made as well.

By considering the last equality in (2.13), it is clear that the SVD decomposes a matrix by a sum of rank-one matrices of the form $\sigma_r \mathbf{u}_r \mathbf{v}_r^{\mathrm{T}}$. Each $\sigma_r \mathbf{u}_r \mathbf{v}_r^{\mathrm{T}}$ is called an *SVD layer*. The first SVD layers, corresponding to the largest singular values, explain more of the structure of the data, while the smaller values are often considered to represent the noise in the data.

A *rank-K approximation* $\hat{\mathbf{X}}$ of a $n \times d$ matrix $\mathbf{X}$ is obtained by keeping the $K < R$ SVD layers, i.e.

$$\hat{\mathbf{X}} = \sum_{k=1}^{K} \sigma_k \mathbf{u}_k \circ \mathbf{v}_k \,. \tag{2.14}$$

With respect to the Frobenius norm, this is the closest rank-K approximation of $\mathbf{X}$, that is

$$\hat{\mathbf{X}} = \underset{\mathbf{X}^* \in \mathcal{A}_K}{\mathrm{argmin}} \, \|\mathbf{X} - \mathbf{X}^*\| \,, \tag{2.15}$$

where $\mathcal{A}_K$ is the set of all $n \times d$ matrices of rank K. This is known as the Eckart-Young-Mirsky theorem [22, 23].

An interesting note to make concerning the SVD is its connection to *Principal Component Analysis* (PCA). Assume that the data matrix $\mathbf{X}$ is centred, i.e. that each column has zero mean. Then, it can be shown that the right-singular vectors of $\mathbf{V}$ are equal to the eigenvectors of $\mathbf{X}^{\top}\mathbf{X}$, and these are in fact the principal components used in PCA [24]. Simultaneously, the PCA score matrix can be given from the SVD by UΣ.

2.3.2 CANDECOMP/PARAFAC Decomposition

The idea behind the CANDECOMP/PARAFAC (CP) decomposition was originally proposed by Hitchcock in 1927 as a "polyadic form of a tensor" [25]. It

has since been re-introduced several times in different applications under different names. The names canonical decomposition (CANDECOMP) [26] and parallel factors (PARAFAC) [27] have their origin in psychometrics. In this thesis this method will be referred as CANDECOMP/PARAFAC Decomposition (CP).

The *CP decomposition* splits a tensor into a finite sum of *rank one tensors*, i.e. tensors that can be written as an outer product of N vectors, as shown in (2.5). The problem of finding the CP decomposition of a three-way tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I \times J \times K}$ may be formalised as

$$\min_{\hat{\boldsymbol{\mathcal{X}}}} \|\boldsymbol{\mathcal{X}} - \hat{\boldsymbol{\mathcal{X}}}\| \quad \text{where} \quad \hat{\boldsymbol{\mathcal{X}}} = \sum_{r=1}^{R} \lambda_r \, \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r \,, \tag{2.16}$$

where R is a positive integer, λ_r is a scaling factor, and $\mathbf{a}_r \in \mathbb{R}^I$, $\mathbf{b}_r \in \mathbb{R}^J$ and $\mathbf{c}_r \in \mathbb{R}^K$ for $r = 1, 2 \ldots R$, and are all normalised to length one. The $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_R]$, $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_R]$ and $\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_R]$ are called *factor matrices*. The decomposition is illustrated in Figure 2.6. Note the similarities between the SVD in (2.13) and CP in (2.16).

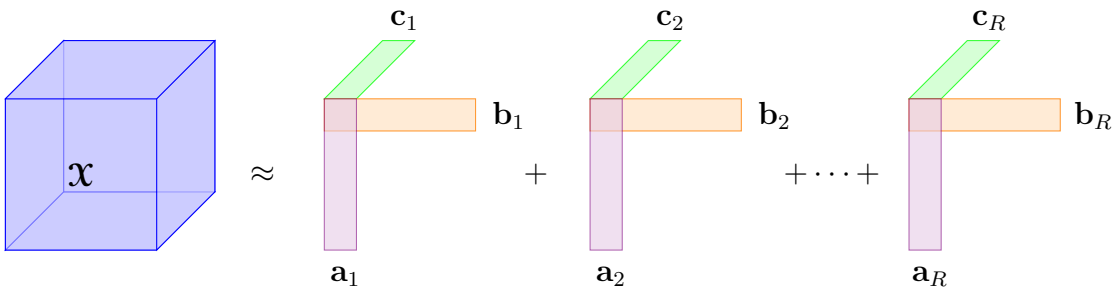


Figure 2.6: *CP decomposition of a three-way tensor $\boldsymbol{\mathcal{X}}$ into a sum of rank one tensors. The rank one tensors are of the form $\mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$, where $\mathbf{a}_r$, $\mathbf{b}_r$ and $\mathbf{c}_r$ for $r = 1, \ldots, R$ are vectors and R is a positive integer.*

The CP decomposition can be generalised for an N-way tensor. For $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ the CP decomposition is on the form

$$\hat{\boldsymbol{\mathcal{X}}} = \sum_{r=1}^{R} \lambda_r \, \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \cdots \circ \mathbf{a}_r^{(N)} \,, \tag{2.17}$$

where R is a positive integer, λ_r is a scaling factor, and $\mathbf{a}_r^{(n)} \in \mathbb{R}^{I_n}$ for $n = 1, \ldots, N$ and for $r = 1, \ldots, R$. The $\mathbf{a}_r^{(n)}$s are normalised to length one.

Matrix decompositions are in general not unique. The uniqueness of the SVD is due to the addition of orthogonality constraints and ordering of the singular values [8]. In contrast, the the CP decomposition is unique under weaker conditions, with the exception of scaling and permutations. One well-known result for three-way tensors was discovered by Kruskal [28]. This condition is expressed in terms of the *k-rank*. The k-rank is, for a matrix $\mathbf{A}$, defined as the maximum number k resulting in that any k chosen columns of $\mathbf{A}$ are mutually linearly independent. This k-rank is

denoted by $k_{\mathbf{A}}$. Using the k-rank, the CP decomposition in (2.16) can be shown to be unique, up to scaling and permutations, under the following condition:

$$k_{\mathbf{A}} + k_{\mathbf{B}} + k_{\mathbf{C}} \geq 2R + 2\,, \tag{2.18}$$

where the matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are the factor matrices for the decomposition. For a review of other bounds for higher-order tensors, see [8].

Another notable difference between the SVD and the CP decomposition is the difference in rank-K approximation. As mentioned in section 2.3.1, the best rank-K approximation is given by a sum of the leading K SVD layers. This is not true for the CP decomposition, and the best rank-K approximation might not even exist [8].

There are several different algorithms for fitting a CP decomposition for a fixed R. Some of the more popular ones are Jennrich's algorithm, alternating least squares (ALS) and tensor power method [29].

2.3.2.1 Tensor Rank

The tensor rank that originates from the CP decomposition, is simply named *rank* and denoted as $\text{rank}(\mathcal{X})$. It is defined as the minimum number of rank one tensors that exactly sums up to the tensor $\mathcal{X}$, i.e. the minimal R that fulfils (2.16) and where $\min_{\hat{\mathcal{X}}} \|\mathcal{X} - \hat{\mathcal{X}}\| = 0$. A tensor decomposition where $R = \text{rank}(\mathcal{X})$ is called a *rank decomposition*.

Even if the tensor rank is an exact analogue to the matrix rank it possesses some different properties. One remarkable property is that the rank of a higher-order tensor depends on the considered field. This means, for example, that whether a tensor is considered in $\mathbb{R}$ or in $\mathbb{C}$ can affect the rank of the tensor. In [8] an example of this situation is given. Note that this property does not hold for matrices.

As mentioned in Section 2.3.1, the matrix rank can be retrieved from the SVD. The tensor rank, however, cannot be obtained as easily. The problem of finding the rank is in fact NP-hard [8]. In practice, often a sequence of fits is made for $R = 1, 2, \ldots,$ and the best fit is chosen. A fit that gives $\|\mathcal{X} - \hat{\mathcal{X}}\| = 0$ is difficult to find due to the noisy nature of real data.

For tensors of a certain order, or structure, there does however exist some knowledge about the rank; either the maximum value of the rank, or values that the rank can attain with a probability greater than zero. One such boundary for a tensor of order three, $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ is [30]

$$\text{rank}(\mathcal{X}) \leq \min\left(IJ, IK, JK\right). \tag{2.19}$$

Other known bounds of the rank are summarised in [8].

2.3.3 Sparse CP Decomposition

In a paper by Allen [18] an iterative algorithm for Sparse CP decomposition is proposed. This algorithm encourages sparsity, i.e. strives to set small numerical

values to zero instead. One motivation as to why this is an attractive property is that it can serve to reduce noise, if the data is consistent with the low-rank assumption. The sparse CP decomposition is summarised in Algorithm 2.1 for a tensor of order three, $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$.

Algorithm 2.1 Sparse CP Decomposition

1: Initialise $\hat{\mathcal{X}} = \mathcal{X}$
2: **for** $r = 1, \ldots, R$ **do**
3: **repeat**
4: $\hat{\mathbf{a}}_r = S\left(\hat{\mathcal{X}} \times_2 \mathbf{b}_r \times_3 \mathbf{c}_r, \xi_{\mathbf{a}}\right)$
5: $\mathbf{a}_r \leftarrow \begin{cases} \hat{\mathbf{a}}_r / \|\hat{\mathbf{a}}_r\|, & \|\hat{\mathbf{a}}_r\| > 0 \\ 0, & \text{otherwise} \end{cases}$
6: $\hat{\mathbf{b}}_r = S\left(\hat{\mathcal{X}} \times_1 \mathbf{a}_r \times_3 \mathbf{c}_r, \xi_{\mathbf{b}}\right)$
7: $\mathbf{b}_r \leftarrow \begin{cases} \hat{\mathbf{b}}_r / \|\hat{\mathbf{b}}_r\|, & \|\hat{\mathbf{b}}_r\| > 0 \\ 0, & \text{otherwise} \end{cases}$
8: $\hat{\mathbf{c}}_r = S\left(\hat{\mathcal{X}} \times_1 \mathbf{a}_r \times_2 \mathbf{b}_r, \xi_{\mathbf{c}}\right)$
9: $\mathbf{c}_r \leftarrow \begin{cases} \hat{\mathbf{c}}_r / \|\hat{\mathbf{c}}_r\|, & \|\hat{\mathbf{c}}_r\| > 0 \\ 0, & \text{otherwise} \end{cases}$
10: **until** convergence of $\mathbf{a}_r$, $\mathbf{b}_r$ and $\mathbf{c}_r$
11: $\lambda_r \leftarrow \hat{\mathcal{X}} \times_1 \mathbf{a}_r \times_2 \mathbf{b}_r \times_3 \mathbf{c}_r$
12: $\hat{\mathcal{X}} \leftarrow \hat{\mathcal{X}} - \lambda_r \, \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$
13: **end for**

The Sparse CP decomposition is based on the tensor power method [18]. The main difference between them is that sparsity is introduced in the vectors $\mathbf{a}_r$, $\mathbf{b}_r$ and $\mathbf{c}_r$ (line 4–9 in Algorithm 2.1) through the soft-thresholding operator S defined as

$$S(x, \xi) = \text{sign}(x)(|x| - \xi)_+ \,, \tag{2.20}$$

where $(x)_+ = x$ if $x > 0$, otherwise $(x)_+ = 0$. The parameter $\xi > 0$ is called the bandwidth parameter and controls the degree of sparsity. Several methods for choosing the bandwidth parameters are mentioned in Allen [18], but the author suggests using the Bayesian Information Criterion (BIC). This criterion is derived by noting that for each update the Sparse CP solves a l_1-norm penalised regression problem. Using BIC, the bandwidth parameter is obtained by

$$\xi_{\mathbf{a}}^* = \underset{\xi_{\mathbf{a}}}{\text{argmin}} \, BIC(\xi_{\mathbf{a}}) \,, \tag{2.21}$$

where

$$BIC(\xi_{\mathbf{a}}) = \log\left(\frac{\|\mathcal{X} - \lambda \, \mathbf{a} \circ \mathbf{b} \circ \mathbf{c}\|^2}{IJK}\right) + \frac{\log(IJK)}{IJK} |\{\mathbf{a}\}| \,. \tag{2.22}$$

Here the $|\{\mathbf{a}\}|$ indicates the number of non-zero elements in $\mathbf{a}$. Analogous expressions apply for $\mathbf{b}$ and $\mathbf{c}$.

In PCA, the amount of explained variance is a commonly used measure for quantifying the goodness of fit of a rank-K SVD approximation. This measure can also be used in the case of a sparse CP decomposition to explain the amount of achieved dimension reduction. Define the projection matrix $\mathbf{P}_r^{(\mathbf{A})} = \mathbf{A}_r(\mathbf{A}_r^\top \mathbf{A}_r)^{-1}\mathbf{A}_r^\top$, and analogously for $\mathbf{P}_r^{(\mathbf{B})}$ and $\mathbf{P}_r^{(\mathbf{C})}$. These matrices have precisely r eigenvalues equal to one. Then the *cumulative proportion of variance* (CPV) for the sparse CP decomposition is given by

$$\frac{\|\mathcal{X} \times_1 \mathbf{P}_r^{(\mathbf{A})} \times_2 \mathbf{P}_r^{(\mathbf{B})} \times_3 \mathbf{P}_r^{(\mathbf{C})}\|^2}{\|\mathcal{X}\|^2}. \tag{2.23}$$

Here, the numerator corresponds to the projection of the tensor $\mathcal{X}$ on the subspace that is spanned by the first r higher-order principal component factors. A proof of this result can be found in [18]. If the tensor $\mathcal{X}$ is centred, then the denominator of (2.23), represents the variance of all elements in the tensor, whilst the numerator represents the variance of the projection. Thus, CPV is the ratio of these two variances.

2.4 Co-clustering

In this section, different types of co-clusters will be defined for tensors of order three. Additionally, scoring functions will be introduced, and the usage of scoring functions will be discussed.

2.4.1 Definition of Co-cluster

A co-cluster can be represented as a subtensor corresponding to a coherent pattern in the tensor. Let $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$, and denote the row, column and tube indices of $\mathcal{X}$ as $\mathcal{I} = \{1, 2, \ldots, I\}$, $\mathcal{J} = \{1, 2, \ldots, J\}$, and $\mathcal{K} = \{1, 2, \ldots, K\}$, respectively. With these indices defined, the tensor can alternatively be described by $\mathcal{X} = \mathcal{X}_{\mathcal{IJK}} \in \mathbb{R}^{I\times J\times K}$. This notation allows for an easier definition of a co-cluster in relation to the tensor. Considering a subtensor $\mathcal{D}$, representing a co-cluster of $\mathcal{X}$, this can now be expressed as $\mathcal{D} = \mathcal{X}_{\mathcal{PQR}} \in \mathbb{R}^{P\times Q\times R}$, where $\mathcal{P} = \{i_1, i_2, \ldots, i_P\} \subseteq \mathcal{I}$, $\mathcal{Q} = \{j_1, j_2, \ldots, j_Q\} \subseteq \mathcal{J}$, and $\mathcal{R} = \{k_1, k_2, \ldots, k_R\} \subseteq \mathcal{K}$.

The co-clusters can be categorised into different categories, and commonly different clustering methods focus on one or several of these categories. Let π, α_p, β_q and γ_r, for $p \in \mathcal{P}$, $q \in \mathcal{Q}$ and $r \in \mathcal{R}$, be constants. The co-cluster categories that are considered in this report are described by additive models, and can be expressed as follows [10]:

1. **Constant values:** $d_{pqr} = \pi$.
2. **Constant values on the slices:** $\mathbf{D}_{p::} = \pi+\alpha_p$, $\mathbf{D}_{:q:} = \pi+\beta_q$, or $\mathbf{D}_{::r} = \pi+\gamma_r$.
3. **Constant values on the mode-n fibers:** $\mathbf{d}_{:qr} = \pi+\beta_q+\gamma_r$, $\mathbf{d}_{p:r} = \pi+\alpha_p+\gamma_r$, or $\mathbf{d}_{pq:} = \pi + \alpha_p + \beta_q$.
4. **Coherent values on rows, columns and tubes**: $d_{pqr} = \pi + \alpha_p + \beta_q + \gamma_r$.

There are other ways to categorise co-clusters, discussed in e.g. [10]. The last three categories can, for example, be expressed by a multiplicative model.

2.4.2 Scoring Function

In order to find co-clusters, a scoring function can be used to measure the coherence of the clusters [10]. Similarly, it is possible to perform filtering via the scoring functions, i.e. reject co-clusters that have scores above a certain threshold. Two well-used scores for bi-clustering are the mean square residue (MSR) and the mean sum of squares (MSQ). These measures have been extended to co-clusters, and can be used in a similar manner.

The *mean square residue* (MSR) is defined for co-clusters of order three as following [31]

$$\mathrm{MSR}(\mathcal{D}) = \frac{1}{PQR} \sum_{p\in\mathcal{P},q\in\mathcal{Q},r\in\mathcal{R}} (d_{pqr} - d_{p\mathcal{Q}\mathcal{R}} - d_{\mathcal{P}q\mathcal{R}} - d_{\mathcal{P}\mathcal{Q}r} + 2d_{\mathcal{P}\mathcal{Q}\mathcal{R}})^2, \tag{2.24}$$

where $d_{p\mathcal{Q}\mathcal{R}}$, $d_{\mathcal{P}q\mathcal{R}}$ and $d_{\mathcal{P}\mathcal{Q}r}$ denotes the mean of the pth row, qth column and kth tube, respectively, and $d_{\mathcal{P}\mathcal{Q}\mathcal{R}}$ denotes the mean of the subtensor $\mathcal{D}$. This score is able to detect co-clusters described by an additive models, i.e. co-clusters of type 1–4 described in Section 2.4.1, and measures the residual of data in relation to such a model. Data that fits any of those models perfectly, will yield zero MSR.

The *mean sum of squares* (MSQ) is for co-clusters defined as [32]

$$MSQ(\mathcal{D}) = \frac{1}{PQR} \sum_{p\in\mathcal{P},q\in\mathcal{Q},r\in\mathcal{R}} (d_{pqr} - d_{\mathcal{P}\mathcal{Q}\mathcal{R}})^2, \tag{2.25}$$

using the same notation as in (2.24). The MSQ is able to detect constant co-clusters, i.e. co-clusters of type 1 in Section 2.4.1, and measures the residual of data in relation to an assumed constant co-cluster model.

3

Methods

In this chapter, the two developed methods for co-clustering will be described. The methods are based on either clustering of sign patterns or using agglomerative hierarchical clustering, on the sparse CP components, and are called sCP-S or sCP-HC, respectively. Moreover, the simulation setups used for testing the co-clustering methods will be explained. The co-clustering methods will also be applied on real data. A description of this data set is included at the end of the chapter.

3.1 Co-clustering on Sparse CP Decomposition

The first developed method, sCP-S, is inspired by a bi-clustering method using sparse SVD (SSVD) described in Lee et al. [11], and the second developed method, sCP-HC, is a continuation of the first. In this section the sCP-S and the sCP-HC will be described in detail. A simple flowchart, summarising them, can also be seen in Figure 3.1. Note that it is only the second step that is different between the two methods. Post-processing for both methods is identical and will be described in a separate section after the description of the co-clustering methods. As before, the methods will be described for tensors of order three, but they are generalisable to higher orders.

3.1.1 Co-clustering by Sign Patterns (sCP-S)

As mentioned, this method is inspired by a bi-clustering method using sparse SVD described in Lee et al. [11]. In their method a sparse SVD factorisation is performed, where the sparsity is imposed in the left-singular vectors and right-singular vectors using adaptive lasso penalties. Thereafter bi-clusters are formed by considering groups of positive, negative and zero values in $\mathbf{U}$ and $\mathbf{V}$, or the first sparse SVD layers.

The sCP-S also utilises the structure of a sparse decomposition, by clustering on the sparse CP vectors given by the sparse CP decomposition. The co-clustering analysis is done layerwise, where each layer typically carries different kinds of information. The first layers contain the most information about the original data. How many layers to consider depends on the data set at hand. Typically larger and more complex data set will need more layers. The boundary in (2.19) can be used as a reference for the maximal amount of layers needed. The cumulative propor-

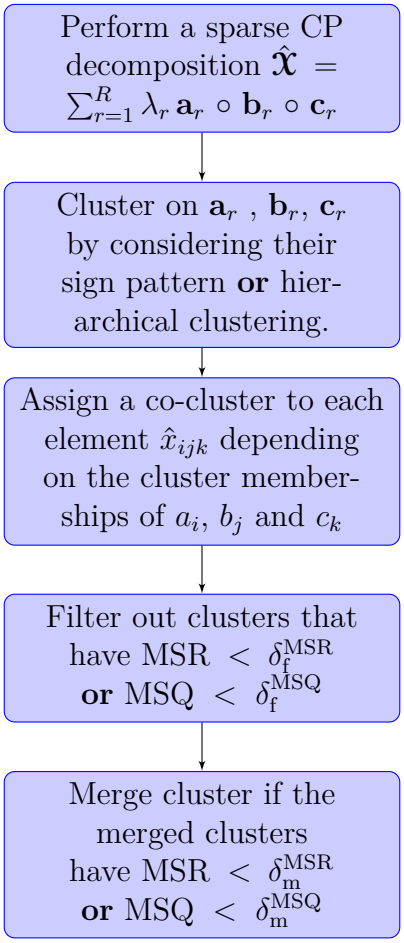


Figure 3.1: *Description of the sparse CP co-clustering based on either considering the signs of the factor vectors* $\mathbf{a}_r$, $\mathbf{b}_r$, $\mathbf{c}_r$ *(sCP-S), or using hierarchical clustering on them (sCP-HC). The* δ_f*'s and* δ_m*'s are hyperparameters, described in Section 3.1.3.*

tion of variance, found in (2.23), is used as a measure of the fit of the sparse CP decomposition, and as a tool to decide on the number of layers.

For a considered layer r, the sparse CP vectors $\mathbf{a}_r$, $\mathbf{b}_r$ and $\mathbf{c}_r$ are assigned a cluster corresponding to positive, negative or zero values. Then co-clusters are created by combining the clusters via a Cartesian product. This means that each element of the layer, $\hat{x}_{ijk} = \lambda a_i b_j c_k$, is a member of a co-cluster represented by a triplet of the form (S_i, S_j, S_k), where $S_i = \text{sign}(a_i)$, $S_j = \text{sign}(b_j)$, $S_k = \text{sign}(c_k)$. For example, an element with $a_i > 0$, $b_j > 0$ and $c_k < 0$ belongs to the co-cluster $(1, 1, -1)$. However, because co-clusters with at least one zero in the triplet results in $\hat{x}_{ijk} = 0$, these elements are combined into one *zero co-cluster*. Therefore, in total there are nine different co-clusters, where one of them corresponds to the zero co-cluster.

Note that as a consequence of the non-uniqueness of the CP decomposition in regards to scaling, as mentioned in Section 2.3.2, the vectors $\mathbf{a}_r$, $\mathbf{b}_r$ and $\mathbf{c}_r$ are not unique with respect to multiplication of -1, as long as two vectors are multiplied. This means that we should not place any meaning in whether the elements of a sparse CP vector have positive or negative value, since their signs can change arbitrarily.

3.1.2 Co-clustering by Hierarchical Clustering (sCP-HC)

This method is similar to sCP-S, with the difference being how the clustering on the sparse CP vectors is performed. Here the actual values of the entries of the sparse CP vectors are considered by using *agglomerative hierarchical clustering*. Agglomerative hierarchical clustering is a bottom up clustering method that starts with each data point in its own cluster and successively merges clusters based on a cluster-to-cluster distance measure, called linkage. See, for example, [33] for more details. The main reason for using this clustering algorithm is because of its flexibility, and that the number of clusters is not predefined.

The co-cluster analysis is performed layerwise. To take advantage of the sparse structure of the sparse CP vectors, the hierarchical clustering is only performed on the non-zero values of $\mathbf{a}_r$, $\mathbf{b}_r$ and $\mathbf{c}_r$. The obtained clusters are then combined via a Cartesian product, resulting in co-clusters. Thus, the co-cluster membership of each element of the layer, $\hat{x}_{ijk} = \lambda a_i b_j c_k$, depends of the cluster memberships of a_i, b_j, and c_k. For example, if C_i, C_j and C_k denotes the cluster memberships of a_i, b_j and c_k, respectively, then $\hat{x}_{ijk} = \lambda a_i b_j c_k \in (C_i, C_j, C_k)$. All zero value elements of the layer, corresponding to at least one zero value in a_i, b_j or c_k, are assigned to the zero co-cluster, similarly to the sCP-S.

In the hierarchical clustering, Ward linkage and Euclidean distances are used. Ward linkage is chosen since it is known to typically perform best amongst classical linkage measures [34]. However, it is worth noting that Ward linkage can be biased towards finding spherical clusters of about equal size [35], though that will not be an issue here since the clustering is performed on one dimensional data. Another interesting thing to note is that hierarchical clustering with Ward linkage has similarities to k-means. It will create clusters of roughly the same size and is sensitive to outliers, just like k-means, but is deterministic, i.e. not dependent on the starting values [36].

The number of clusters is chosen manually by inspection of the dendrogram obtained from the clustering algorithm. A horizontal line in the dendrogram, at a given height, indicates that two clusters can be merged with a similarity corresponding to the height. As such, if many data points are about equally similar, many mergings will occur within a small range of heights. However, if there is a visible gap in the height between different mergings, this shows that the similarity requirement has to be relaxed quite a bit before any new data points and clusters can be added together. The gap therefore indicates that a cluster is relatively stable, i.e. that no new data points will be added to it unless the cluster similarity requirement is relaxed quite a bit. From this reasoning, it is appropriate to take both the similarity values and the height gaps into consideration when choosing the number of clusters from a dendrogram. However, a larger number of clusters are preferred, since the number of co-clusters can be reduced using post-processing, and that a level of detail can be achieved using a larger number of clusters.

3.1.3 Post-Processing

The post-processing consists of two steps: Filtering and merging. Since the clustering is made on the sparse CP vectors, and the co-clusters are created based on those clusters, a large number of co-clusters is obtained. For example, having 10 clusters, excluding the zero cluster, in **a**, **b** and **c**, respectively, would result in 1 001 co-coclusters. This makes post-processing a crucial step in many cases. By using filtering, the co-clusters that are not sufficiently coherent can be filtered out. Thereafter, related co-clusters can be merged. If the post-processing steps would be used in the opposite order, i.e. merge and then filter, then it is possible that a co-cluster with a small score would be merged with a co-cluster with a high score, but clusters with a low score are considered to be relevant. This is avoided if filtering is performed first, because the co-clusters with a higher score are filtered out.

3.1.3.1 Filtering

Both the MSR and MSQ, found in (2.24) and (2.24), respectively, will be used as filtering scores, since they evaluate different kinds of clusters. Clusters with a score above the threshold δ_{f} are rejected. The threshold is chosen by first plotting the sorted scores for all the co-clusters, versus the co-cluster indices. In this plot, the sorted scores typically show a drastic increase after reaching a certain index value, similar to an elbow plot. From studying this plot, the threshold is chosen as to exclude clusters with indices that are located after the sudden increase.

3.1.3.2 Merging

For two co-clusters in a three-way tensor to be merged they need to have at least two dimensions in common. This means that they need to have either their rows and columns, rows and tubes, or columns and tubes in common. The coherence of the merged co-cluster is evaluated by a scoring function. The MSR or a MSQ, from (2.24) and (2.25), respectively, are used as a scoring function.

To merge the found co-clusters, two computational steps are made. First, the algorithm attempts to find all the possible merging alternatives that can be made from the found co-clusters. The scores of these possible merging alternatives are plotted, in ascending order. A merging threshold is then determined from this plot, in order to keep the good co-cluster merging alternatives that are presented but discard the rest. These two steps will be described in more detail here below.

To find all the possible merging alternatives, the algorithm runs iteratively. In each iteration, the algorithm first finds all the possible alternatives for merging the co-clusters that are currently available. A co-cluster merging score is calculated for each possible merged co-clustering alternative. Then, the algorithm proceeds to merge co-clusters, in order of which co-cluster alternatives that have the best scores. Once a co-cluster has been merged with another, it is excluded from further merging in the current iteration. In each iteration, the merged co-clusters and their corresponding scores are saved. Before a new iteration is made, the available co-clusters that are to be merged further are updated. The iterations stop when no more mergings are

possible.

As the second computational step, the scores of the co-cluster merging alternatives are plotted in ascending order. A horizontal line is drawn in this plot to indicate the filtering threshold for the found co-cluster mergings, as a reference for choosing the merging threshold. This plot can be interpreted similarly to an elbow plot, and in the same fashion it is appropriate to choose a merging threshold δ_{m} that corresponds to about the elbow of the curve in this plot. However, the filtering threshold should also be considered, with the following reasoning. The merging threshold is appropriately chosen to be slightly higher than the filtering threshold, but not very much so. This in order to allow for co-cluster mergings to slightly degrade their score, at the benefit of reducing the number of clusters. Once a merging threshold is chosen, only the co-cluster mergings below this threshold are kept. If one merged co-cluster is a subset of another merged co-cluster, and they both have scores below the merging threshold, then the smaller of the two co-clusters will be discarded.

3.2 Simulation Description

In order to evaluate the developed co-clustering methods sCP-HC and sCP-S, they were tested on several different data sets; both simulated data sets and a real data set. In this section the different data sets will be described, and the motivation behind each simulation will be given. The tests were primarily performed using sCP-HC, since the methods are alike and the sCP-HC offers more flexibility.

3.2.1 Simulated Data Sets

The simulated data sets were used for two studies. The first study aimed to investigate which of the co-cluster types, described in Section 2.4.1, the sCP-HC could find. In the second study, the sCP-HC and sCP-S were both applied to the same data set and a comparison of their co-clusterings was made.

For the simulation regarding what kinds of co-clusters the sCP-HC can find, four different data sets were used, each corresponding to one of the categories described in Section 2.4.1. The first data set contained constant co-clusters. The data set was generated by creating three vectors $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$, of unit length, and taking their outer product, i.e.

$$\boldsymbol{\mathcal{X}} = \rho\, \mathbf{u} \circ \mathbf{v} \circ \mathbf{w}\,, \tag{3.1}$$

where ρ is a scaling constant. For the three remaining co-cluster categories, the co-clusters were modelled following the definitions in Section 2.4.1. Then they were inserted in an empty three-way tensor. Finally, all the tensor elements were randomly permuted in each dimension.

In the second simulation, where the sCP-HC and sCP-S were compared, the first data set is used once more, i.e. the data set described by (3.1). Moreover, this data set is centred, meaning that the respective means were subtracted from each mode-1 co-cluster.

3.2.2 Genomic Data Set

This data set is a three-way tensor of size $17500 \times 123 \times 10$. The rows represent different genes, the columns represent different drugs, and the tubes represent different cell lines. A cell line is a cell culture which can be maintained over a longer period of time, and cloned so that all cell lines derive from the same common ancestor cell [37]. The tensor contains DESeq2 [38] normalised estimates of the fold changes for the genes under different circumstances, i.e. different drugs and different cell lines. The names of the involved genes, drugs and cell lines will not be shown due to confidentiality.

The genomic data set was used for two different studies. The first study used a subset of the genomic data set, and intended to assess the stability of co-cluster detection of the sCP-HC under pertubations of the input data.

This was done by altering the data set by adding noise, and comparing with the results on the data set with no noise. In the second study, a considerably larger subset of the genomic data set was used. The goal of the analysis was to find groups of genes whose expression levels behave in a coherent fashion across subsets of drugs and cell-lines.

In the stability test, the 200 genes with the highest variance, over the drugs and cell lines, were selected. This was mainly done in order to have a smaller, but more relevant, data set to work on. Normally distributed noise with different values of the standard deviation σ was then added to the data set, i.e.

$$\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{X}}_{\text{real data}} + \boldsymbol{\mathcal{E}}\,, \tag{3.2}$$

where $\boldsymbol{\mathcal{E}} \sim N(0, \sigma)$. The standard deviation σ was set to

$$\sigma = \sqrt{\frac{\|\boldsymbol{\mathcal{X}}_{\text{real data}}\|^2}{IJK}} \Big/ \gamma\,, \tag{3.3}$$

where γ is the *signal-to-noise ratio* (SNR). In the simulations, $\gamma = 1, 2, \ldots, 5$ were tested.

The similarity between the two clusterings, i.e. the no noise co-clustering and with noise co-clustering, were then measured by the *adjusted Rand index* (ARI). The ARI is at most equal to 1, which indicates that the clusterings are the same, but can taken on negative values if the expected value of the rand index is higher than the calculated one. Lower values imply worse cluster agreement. The ARI has previously been used in e.g. gene expression cluster analysis [39, 40], motivated partly by [41] wherein the ARI was concluded the best amongst five classical external clustering criteria. The ARIs were calculated by using the no noise clusterings as ground truth, and averaged over 10 times to get a more stable measure.

Since five values of γ were to be tested, and the simulations would run 10 times due to the averaging, this would have resulted in 150 simulations. To manually choose the thresholds $\delta_{\text{f}}^{\text{MSR}}$, $\delta_{\text{m}}^{\text{MSR}}$, $\delta_{\text{f}}^{\text{MSQ}}$ and $\delta_{\text{m}}^{\text{MSQ}}$ would require an overwhelming amount of work. Moreover, it would be difficult to choose the thresholds in an equal manner

for every simulation. To address these problems, the thresholds were chosen once, based on the data set with no noise. Then, the same thresholds were used for all the noisy simulations.

For the application of the sCP-HC on real data, a subset containing 5 000 genes, with all of the drugs and cell lines, was selected. These 5 000 genes were the ones with the highest variance over the drugs and cell lines. The reason for not using the whole data set was mainly due to time constraint, due to the large size of the data set.

In both studies, the co-clustering analysis was performed for the first three sparse CP layers. Three layers did not suffice to fully capture the information in the data set, i.e. attain a large value of CPV. However, in order to keep the amount of figures for presentation and analysis to a reasonable number, more layers were not used.

3.3 Implementation

The implementation of Allen's sparse CP decomposition, found in Algorithm 2.1, and the developed co-clustering methods sCP-S and sCP-HC are implemented in Python. To organise the code, the code for the sparse CP decomposition was made into its own package whereas the hierarchical clustering uses the Scikit-learn package [42]. The tensor operations are done using the package Tensorly [43]. The choice is motivated by the package's user-friendliness and computational speed. The matricisation algorithm, for example, is exceptionally fast compared to other implementations [44]. Visualisations are made using Matplotlib [45] and Seaborn [46].

4

Results

In this chapter various simulations will be performed to test the developed clustering methods, sCP-S and sCP-HC. The chapter is divided into four parts. Firstly, the kinds of co-clusters that the sCP-HC can find will be investigated. Secondly, a comparison between sCP-HC and sCP-S is made. Thirdly, a stability test is performed. Lastly, the sCP-HC will be applied on the real data set, described in Section 3.2.2.

4.1 Types of Co-clusters

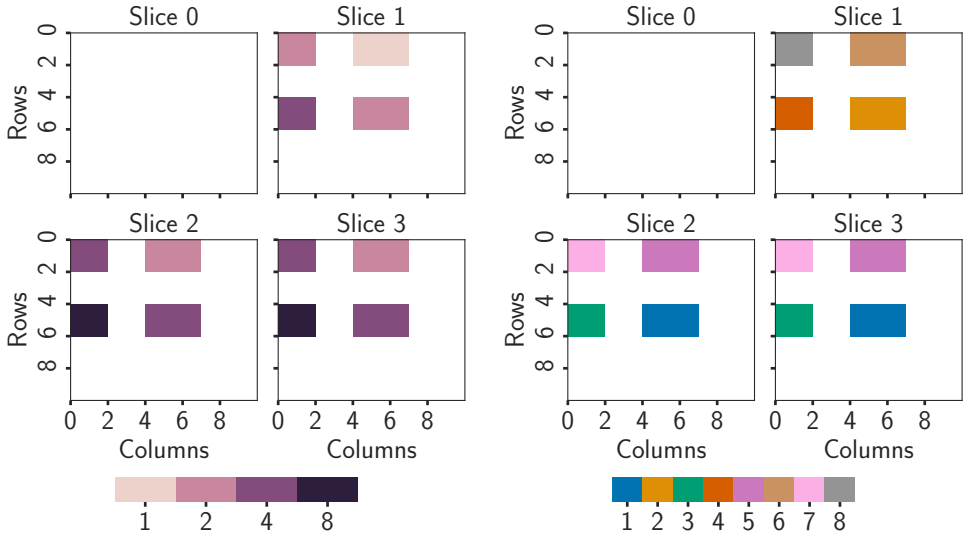
In this section the sCP-HC will be tested on different kinds of co-clusters, described in Section 2.4.1. From these tests, the method's ability to find different co-clusters types will be investigated. This assesses the usefulness of the method, and facilitates analysis of later results, when the method is applied on real data. Complementary results to the ones presented in this section can be found in Appendix A.

4.1.1 Constant Co-clusters

In this simulation, the sCP-HC is applied to co-clusters of type 1, described in Section 2.4.1. The following parameter values are used to generate the data set:

$$\begin{aligned} \hat{\mathbf{u}} &= (1,1,0,0,2,2,0,0,0,0), \quad \mathbf{u} = \hat{\mathbf{u}}/\|\hat{\mathbf{u}}\| \\ \hat{\mathbf{v}} &= (2,2,0,0,1,1,1,0,0,0), \quad \mathbf{v} = \hat{\mathbf{v}}/\|\hat{\mathbf{v}}\| \\ \hat{\mathbf{w}} &= (0,1,2,2), \quad \mathbf{w} = \hat{\mathbf{w}}/\|\hat{\mathbf{w}}\| \\ \rho &= \|\hat{\mathbf{u}}\|\,\|\hat{\mathbf{v}}\|\,\|\hat{\mathbf{w}}\|\,. \end{aligned} \tag{4.1}$$

The data set is shown slicewise in Figure 4.1a. Notice that the data set consists of four constant groups. However, these groups do not correspond to four co-clusters. As described in Section 2.4.1, co-clusters have a cuboid shape. This means that the two groups of value 2 in slice 1 do not form a co-cluster, because they do not share the same rows or columns. Likewise, in slice 2 and 3 the two groups of value 4 do not form a co-cluster. They do however create a co-cluster stretching from slice 2 to slice 3. Thus, the total number of constant co-clusters in this data set is eight.



(a) *Original data set.* **(b)** *Found co-clusters.*

Figure 4.1: *Slicewise illustrations of, (a) original data set containing eight co-clusters of constant values, and (b) the found co-clusters before post-processing. The colours correspond to the values of the data set, and the found co-clusters, respectively. The found co-clusters correspond to the eight constant co-clusters.*

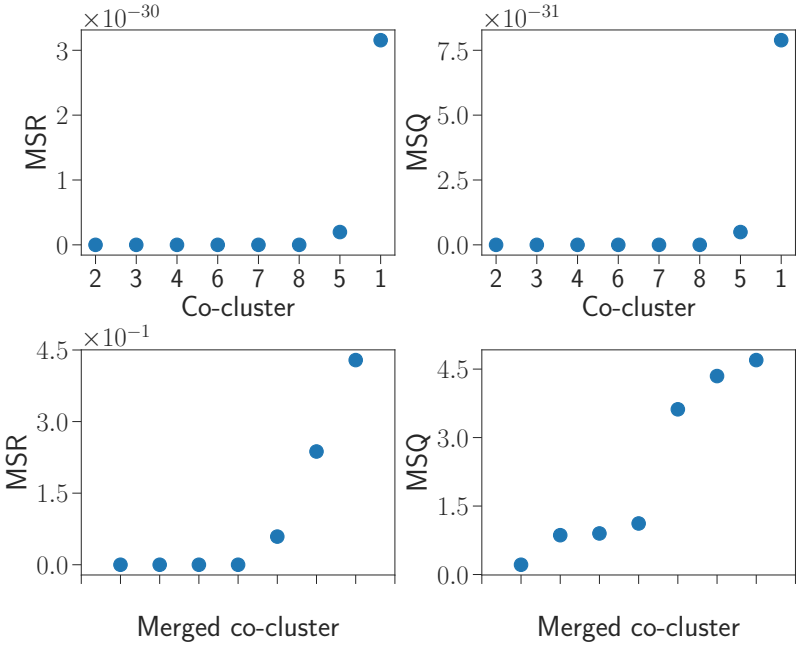


Figure 4.2: *Sorted MSR and MSQ for found co-clusters, and merged co-clusters. The upper plots are used for selection of filtering thresholds, and lower plots are used for selection of merging thresholds. The data set consists of eight constant co-clusters.*

The first layer of the sparse CP decomposition is almost identical to the original data, with a CPV of 99.99%. The number of clusters in **a**, **b** and **c** is set to 2, respectively. The corresponding dendrograms are seen in Figure A.1, Appendix A.

The obtained co-clusters, before any pre-processing, are shown in Figure 4.1b. The found co-clusterings are seen to correspond precisely to the ones present in the data. In Figure 4.2, the sorted MSR and MSQ for all found co-clusters are presented on the first row, along with the sorted MSR and MSQ for all found co-cluster mergings on the second row. Because of the small values of the scores, no filtering is done for the found co-clusters. On the second row, the MSR and MSQ of the merged co-clusters are seen to increase drastically in comparison to the scores of the found co-clusters on the first row. From this large discrepancy, it is appropriate to draw the conclusion that no co-cluster merging should be performed and that the originally obtained co-clusters are the best. Additionally, one might consider MSQ to be the most appropriate scoring function for this data. That is because the MSQ increases much more when co-cluster mergings are evaluated, whereas the four lowest MSR mergings have relatively low scores still.

4.1.2 Constant Values on the Slices

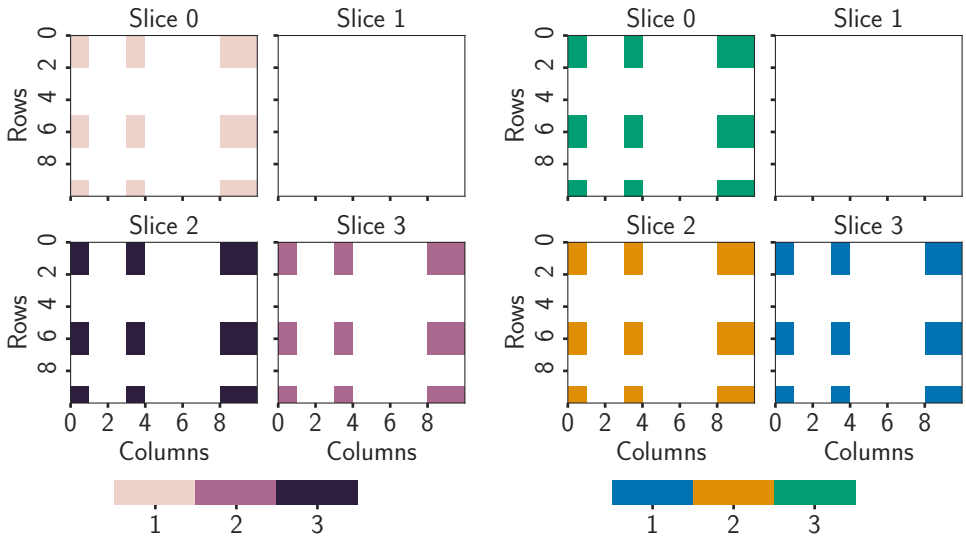
Here, the sCP-HC is tested on a co-cluster of type 2, defined in Section 2.4.1. The co-cluster is of size $5 \times 4 \times 3$, and is generated by the following parameter values

$$\pi = 0, \quad \gamma_r = (1, 2, 3)\,. \tag{4.2}$$

This corresponds to constant co-clusters on three of the frontal slices, as illustrated in Figure 4.3a.

Using one layer to approximate the data set with sparse CP is enough to reach a CPV of 99.98%. The number of clusters in **a**, **b** is set to 1, while the number of clusters in **c** is set to three. See Figure A.2 in Appendix A for an illustration of the dendrograms. The received co-clusters, before post-processing, are shown in Figure 4.3b. The three co-clusters are separated by the frontal layers, giving it a appearance similar to the original data set, seen in Figure 4.3a.

The MSR and MSQ of the original co-clusters and their potential mergings are seen in Figure 4.4. Since the MSR and MSQ values are seen to be very small for the original co-clusters, seen on the first row, no filtering is deemed necessary. Furthermore, the MSR for the merged co-clusters are about equally small as the original co-clusters. Therefore, the MSR plots would suggest that it might be suitable to merge all of the co-clusters. However, there is a large discrepancy between the MSQ values of the original co-clusters and the merged ones, which would contrarily suggest that merging is not a suitable alternative here. This demonstrates the MSQ measure's tendency to find constant co-clusters, when the data here in fact consists of a single additive co-cluster that varies across the slices. The final result, after merging using MSR, can be found in Figure A.5 in Appendix A.



(a) *Original data set.* **(b)** *Found co-clusters.*

Figure 4.3: *Slicewise illustrations of, (a) original data set containing one co-clusters of constant values on the frontal slices, and (b) the found co-clusters before post-processing. The colours corresponds to the values of the data set, and the found co-clusters, respectively.*

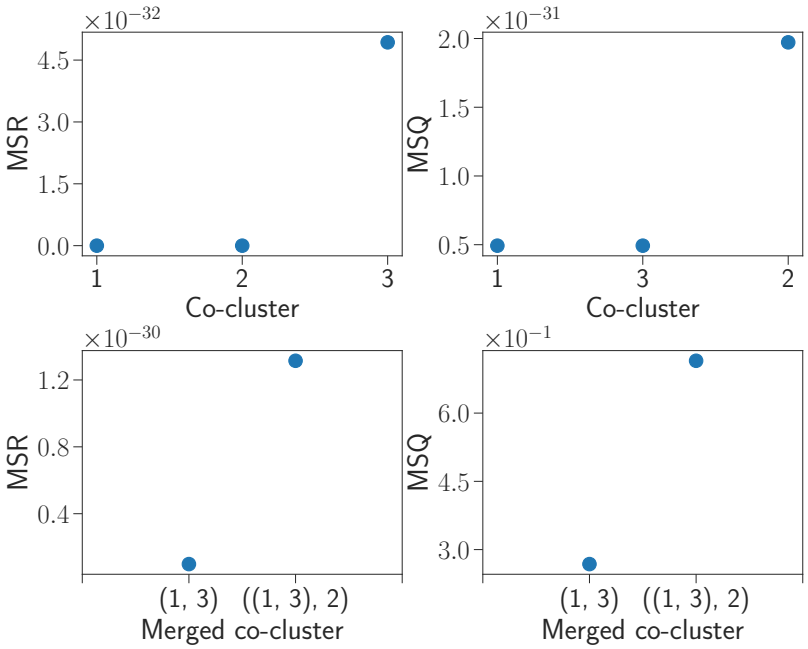


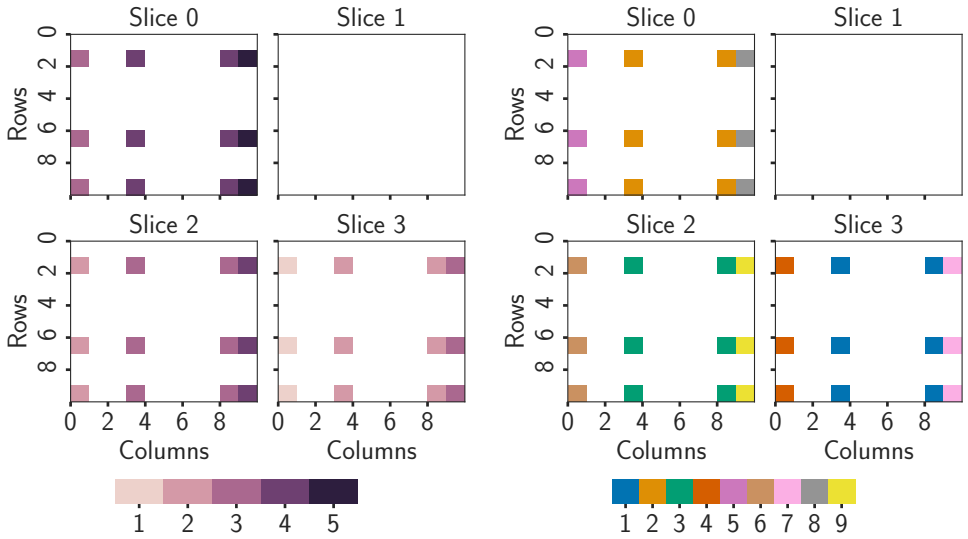
Figure 4.4: *Sorted MSR and MSQ for found co-clusters, and merged co-clusters. The upper plots are used for selection of filtering thresholds, and lower plots are used for selection of merging thresholds. The data set consists of one co-cluster of constant frontal slices.*

4.1.3 Constant Values on the Mode-n Fibers

In this simulation, a co-cluster of type 3 is used, see Section 2.4.1. The data set consists of one co-cluster of size $3 \times 4 \times 3$, and corresponds to constant values on the mode-1 fibers, i.e. the columns. The co-cluster is modelled using the following parameter values:

$$\pi = 1, \quad \beta_q = (1, 2, 1, 0), \quad \gamma_r = (2, 0, 1)\,. \tag{4.3}$$

A slicewise representation of the data set is shown in Figure 4.5a.



(a) *Original data set.* **(b)** *Found co-clusters.*

Figure 4.5: *Slicewise illustrations of, (a) original data set containing one co-cluster of constant values on the columns, and (b) the found co-clusters before post-processing. The colours corresponds to the values of the data set, and the found co-clusters, respectively.*

The sparse CP approximation of the data set using one layer has a CPV of 99.66%. The dendrograms obtained from the hierarchical clustering on **a**, **b** and **c** are displayed in Figure A.3 in Appendix A. The number of clusters for **a**, **b** and **c** is chosen as 1, 3 and 3, respectively. The found co-clusters before post-processing are shown in Figure 4.5b.

In Figure 4.6, the MSR and MSQ are seen on the first row for the found co-clusters, and on the second row for the merged co-clusters. Looking at the first row of scores, the MSR and MSQ values suggest that coherent co-clusters have been found. Moving on to the second row however, the MSR suggests that merging the co-clusters maintains coherence whilst the MSQ suggests otherwise. This in likeness to Figure 4.4, and a similar conclusion can be drawn. However, with Figure 4.5a in mind and knowing that the data in fact consists of a single co-cluster, the merged co-cluster MSR plot shows that the merging algorithm in this case fails to fully merge all of the found co-clusters into one. This highlights the difficulty of merging

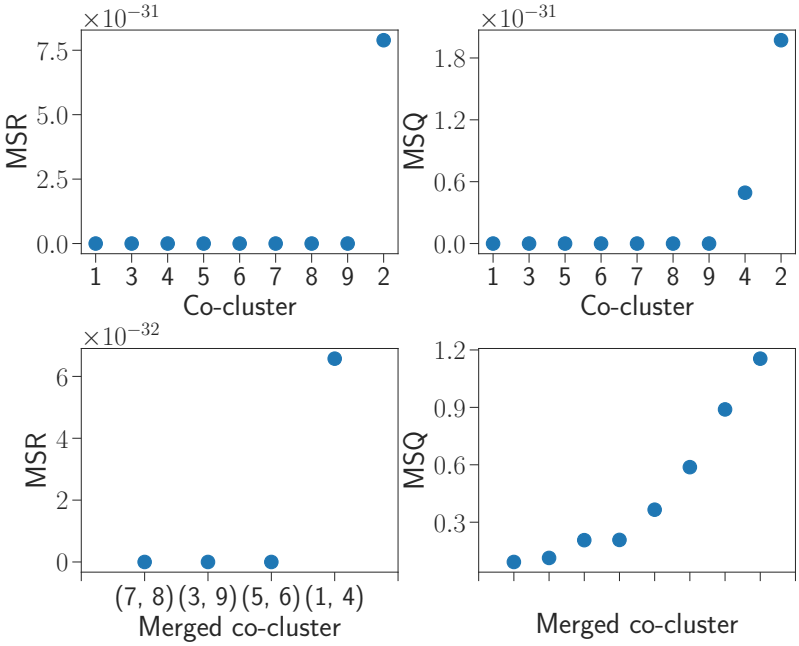


Figure 4.6: *Sorted MSR and MSQ for found co-clusters, and merged co-clusters. The upper plots are used for selection of filtering thresholds, and lower plots are used for selection of merging thresholds. The data set contain one co-cluster of constant mode-1 fibers.*

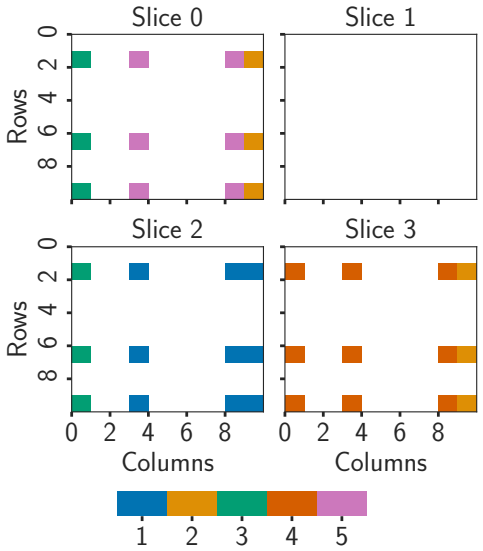


Figure 4.7: *Slicewise illustration of co-clusters after co-cluster merging using MSR, for a data set consisting of a co-cluster of constant columns. Four co-cluster mergings was performed. The data set before post-processing is seen in Figure 4.5b.*

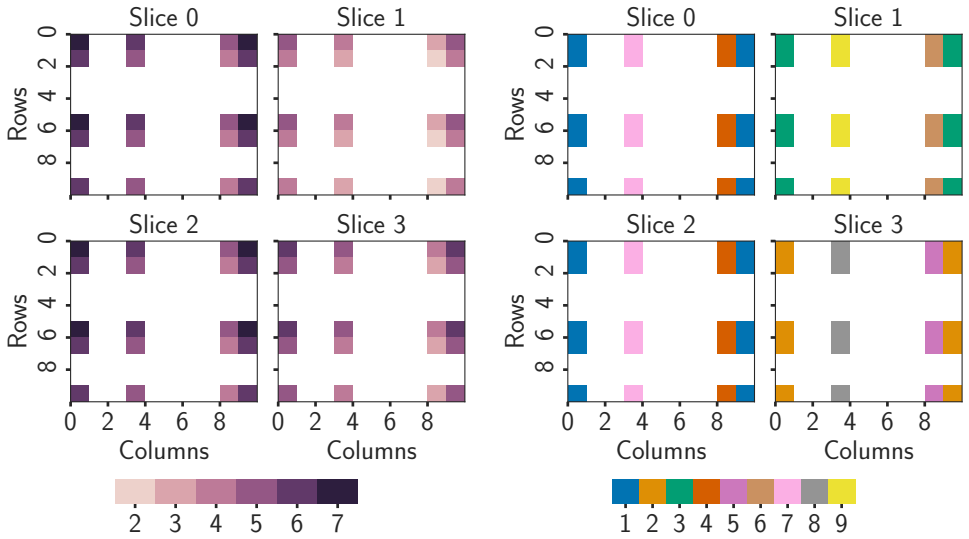
co-clusters due to their multidimensionality. The co-clusters after merging, using MSR, are shown in Figure 4.5b. From there it is evident that due to the order in which the mergings were performed, none of the resulting co-clusters have two dimensions in common, and therefore they cannot be further merged.

4.1.4 Coherent Values on the Mode-n Fibers

For this simulation, the sCP-HC is evaluated on the co-cluster type 4, seen in Section 2.4.1. A co-cluster of size $5 \times 4 \times 4$, with coherent values on the rows, columns and tubes, is used. The parameter values used to generate this co-cluster are:

$$\pi = 1, \quad \alpha_p = (1, 1, 1, 2, 2), \quad \beta_q = (0, 2, 1, 2), \quad \gamma_r = (2, 1, 2, 0)\,. \tag{4.4}$$

The data set is shown in a slicewise manner in Figure 4.8a.



(a) *Original data set.* **(b)** *Found co-clusters.*

Figure 4.8: *Slicewise illustrations of, (a) original data set containing one co-cluster of coherent values on the rows, columns and tubes, and (b) the found co-clusters before post-processing. The colours correspond to the values of the data set, and the found co-clusters, respectively.*

With one sparse CP layer, the CPV attained a value of 99.87%. The number of clusters were set to 1, 3 and 3, for **a**, **b**, and **c**, respectively. The dendrograms for the corresponding to these clustrings can be found in Figure A.4 in Appendix A.

A comparison of the MSR and MSQ, between the found co-clusters and the merged ones, can be seen in Figure 4.9. On the first row, the MSR values are very low, indicating that the found co-clusters are very coherent and in no need of filtering. The MSQ values on the first row however are much higher in comparison to the MSR. This suggests that the data likely consists of an additive co-cluster, but not one that is constant. On the second row, the MSR is seen to drastically increase by merging the co-clusters, suggesting that co-cluster merging is unsuitable. The MSQ

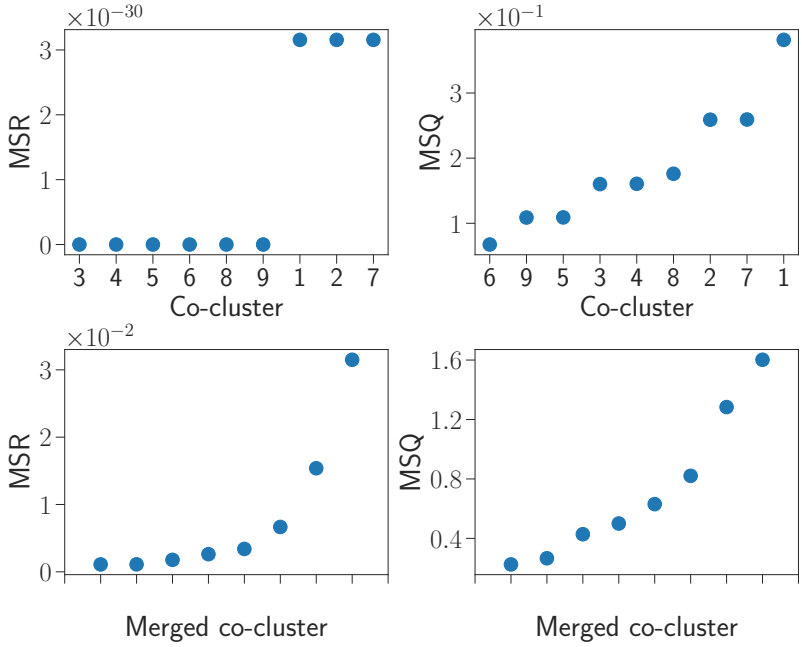


Figure 4.9: *Sorted MSR and MSQ for found co-clusters, and merged co-clusters. The upper plots are used for selection of filtering thresholds, and lower plots are used for selection of merging thresholds. The data set contain one co-cluster of coherent values.*

for the second row shows no improvement in comparison to the first row, suggesting that no co-cluster merging results in a constant co-cluster either.

4.2 Comparison of Co-clustering Methods

In this simulation the sCP-S and sCP-HC will be applied to the same data set as in Section 4.1.1, i.e. the data set containing eight constant co-clusters, to illustrate the difference between the co-clustering methods.

As seen from (4.1), the vectors $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$ that build up the data set $\mathcal{X}$ only contain positive values. This, along with $\rho > 0$, results in a tensor with positive values, but of different magnitudes. The sCP-HC considers this magnitude difference, and succeeds to find the true co-clusters, as seen in Figure 4.1b. The sCP-S, however, does not distinguish between the different positive values. Instead, it only finds a single cluster, corresponding to the positive values of the sparse CP vectors. This is illustrated in Figure 4.10.

Note however that if a centring of the data is performed, so that the data set is not limited to positive values, the sCP-S is able to find more structure of the data. This is illustrated in Figure 4.11. There, the centring has been performed over the mode-1 fibers, i.e. the mean over the mode-1 fibers have been subtracted from them. The sCP-S is seen to find two co-clusters instead of one. The found co-cluster 1 corresponds to the four smallest valued constant co-clusters, whereas the found co-cluster 2 corresponds to the four largest valued constant co-clusters.

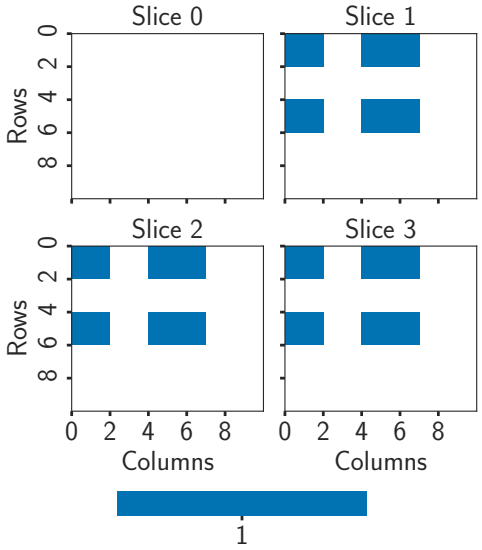
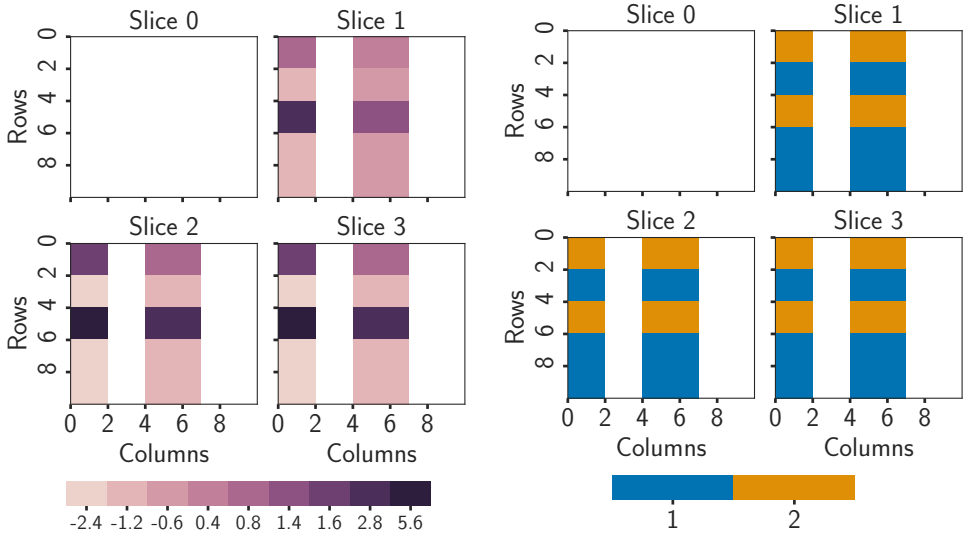


Figure 4.10: *Slicewise illustration of co-cluster obtained using sCP-S on a data set containing eight co-clusters of constant values, seen in Figure 4.1a.*



(a) *Centred data set.* **(b)** *Found co-clusters.*

Figure 4.11: *Slicewise illustrations of, (a) data set after centring containing containing eight constant co-clusters, and (b) the found co-clusters using sCP-S. The colours correspond to the values of the data set, and the found co-clusters, respectively.*

4.3 Co-clustering Stability

In this simulation, the sCP-HC is run ten times for each SNR value and each layer. In order to make the results more comparable, all needed parameters for each layer are selected once for the data set with no added noise. These parameters are then used for all the data sets with added noise.

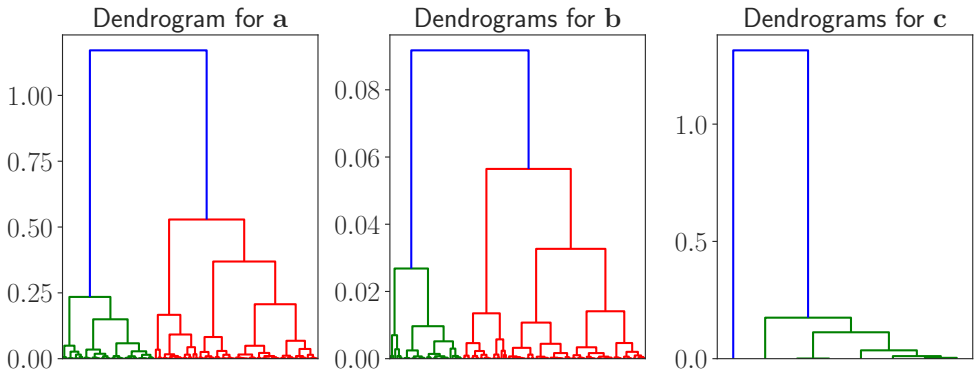


Figure 4.12: *Dendrograms obtained from agglomerative hierarchical clustering on the sparse CP vectors* **a**, **b** *and* **c**. *The data set has no added noise, and corresponds to the first sparse CP layer.*

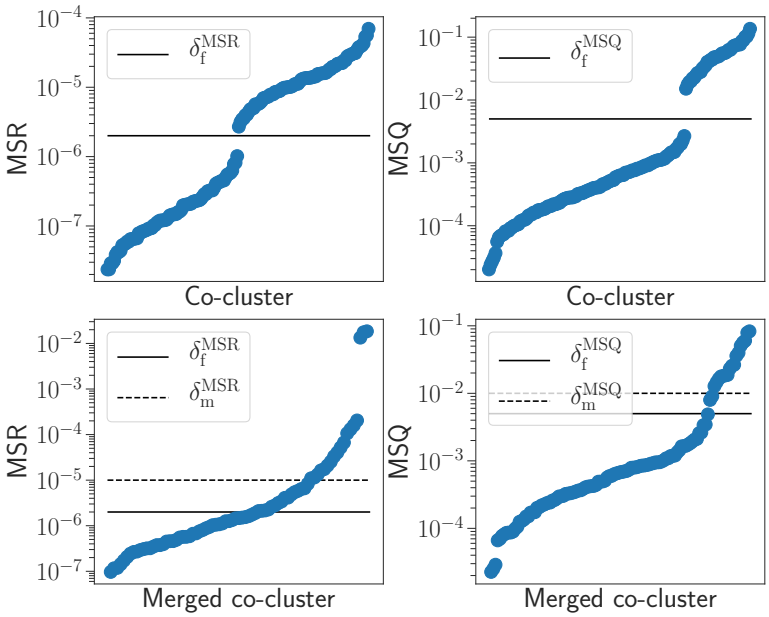


Figure 4.13: *Sorted MSR and MSQ for found co-clusters, and merged co-clusters. The upper plots are used for selection of filtering thresholds* δ_f^{MSR} *and* δ_f^{MSQ}, *and lower plots are used for selection of merging thresholds* δ_m^{MSR} *and* δ_m^{MSQ}. *The data set has no added noise, and corresponds to the first sparse CP layer.*

For the data set with no added noise, 11.71% CPV is attained using one layer. The dendrograms obtained from clustering on this layer are presented in Figure 4.12. The number of clusters in the sparse CP vectors **a**, **b**, and **c**, are set to 8, 5 and

Table 4.1: *Filtering thresholds δ_f and merging thresholds δ_m using MSR and MSQ, and the number of clusters for the sparse CP vectors* **a**, **b** *and* **c**, *for layer 1, 2 and 3.*

Layer	$\delta_{\mathbf{f}}^{\text{MSR}}$	$\delta_{\mathbf{f}}^{\text{MSQ}}$	$\delta_{\mathbf{m}}^{\text{MSR}}$	$\delta_{\mathbf{m}}^{\text{MSQ}}$	**#clusters**
1	2×10^{-6}	5×10^{-3}	1×10^{-5}	1×10^{-2}	$8, 5, 4$
2	2×10^{-4}	2×10^{-2}	5×10^{-4}	3×10^{-2}	$7, 7, 3$
3	7×10^{-5}	5×10^{-3}	3×10^{-4}	2×10^{-2}	$10, 4, 4$

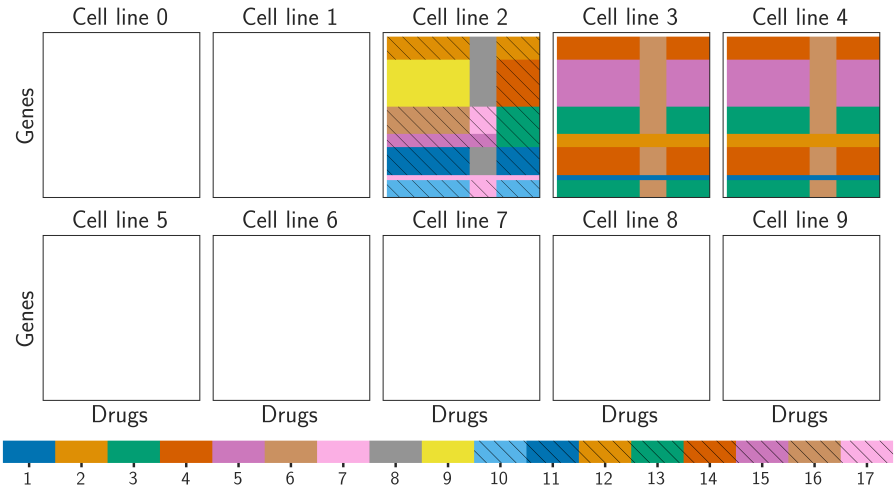
4, respectively. In Figure 4.13 the sorted MSR and MSQ values for the original co-clusters are found on the first row. Both curves are slightly S-shaped, and display a clear gap in the middle of the S. The thresholds are chosen around the middle of their respective gaps, i.e $\delta_{\text{f}}^{\text{MSR}} = 2 \times 10^{-6}$ and $\delta_{\text{f}}^{\text{MSQ}} = 5 \times 10^{-3}$, and are seen as solid lines in the figures. On the second row in Figure 4.12 sorted MSR and MSQ for potential mergings after filtering is seen. The merging thresholds are chosen slightly above the filtering thresholds, as illustrated by the dashed lines. The values of the thresholds are $\delta_{\text{m}}^{\text{MSR}} = 1 \times 10^{-5}$ and $\delta_{\text{m}}^{\text{MSQ}} = 1 \times 10^{-2}$.

The final co-clustering, after post-processing, is shown in Figure 4.14a using MSR, and Figure 4.14b using MSQ. For visual reasons, the tensors are sorted. The sorting is based on the co-clusters before post-processing. To start, the first non-zero row is selected. Then, this is sorted in order of ascending co-cluster number. The rest of the rows in the tensor are re-arranged according to this sorting. After that, the columns and tubes are sorted in a corresponding manner. This with the purpose of giving more closely located co-clustering elements.

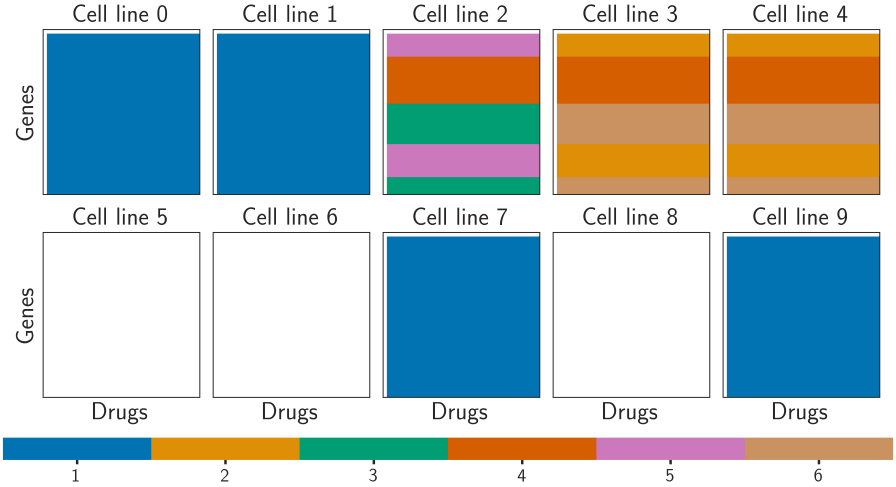
The second and third layer have a CPV of 17.74% and 22.04%, respectively. The parameters for these layers are chosen in a similar fashion as the first layer, and are summarised in Table 4.1. All the figures that underlie the parameter decisions, and the final co-clusters for the second and third layer, are found in Appendix B. Note that each layer has a different sorting.

The mean ARI scores, using both MSR and MSQ as scoring functions, for all three layers and the considered SNR values $\gamma = 1, \ldots, 5$ are presented in Table 4.2. The mean CPV for sparse CP decompositions of different number of layers are also included in the table. One observation worth noting is that the overall scores increase with higher values of the SNR γ. This is an expected result. However, compared to the first and third layer, the increase of the scores for higher γ are considerably lower in the in the second layer. Another observation is that the mean MSQ's tend to be higher than the mean MSR's. One exception of this was found in the first layer for $\gamma = 5$. There the mean MSR attains its highest value, and is notably higher than the mean MSQ.

To see an example of how a noisy simulation can look like, in the case $\gamma = 3$ specifically, a few key results will be shown. To start, in Figure 4.15 sorted MSR and MSQ for co-clusters and merged co-clusters are provided. The thresholds, indicated by horizontal lines, are the same as in Figure 4.13. Comparing Figure 4.15 and Figure 4.13, a few interesting observations can be made. On the first row of the respective figures, the unmerged co-cluster scores are found. Firstly, for the MSR,



(a) *Final co-clusters using MSR.*



(b) *Final co-clusters using MSQ.*

Figure 4.14: *Final co-clusters using MSR (a), and using MSQ (b), on the second layer of the data set with no noise. Cell line 5 and cell line 6 are set to 0 in the sparse CP decomposition.*

Table 4.2: *Mean adjusted Rand index (ARI) for co-clustering analysis performed using the MSR and MSQ, and mean cumulative proportion of variance (CPV), for different signal-to-noise ratios (SNR). The mean is taken over ten iterations. The values are shown for co-clustering analysis performed on the first, second and third layer of the sparse CP factorisation.*

	SNR	Mean ARI, MSR	Mean ARI, MSQ	Mean CPV (%)
Layer 1	1	0.33	0.54	5.881
	2	0.57	0.74	9.370
	3	0.71	0.67	10.54
	4	0.68	0.73	11.02
	5	0.85	0.69	11.27
Layer 2	1	0.22	0.24	8.914
	2	0.28	0.37	14.21
	3	0.31	0.41	15.99
	4	0.30	0.40	16.73
	5	0.29	0.39	17.08
Layer 3	1	0.18	0.15	10.85
	2	0.47	0.50	17.63
	3	0.59	0.63	19.85
	4	0.60	0.66	20.78
	5	0.67	0.69	21.20

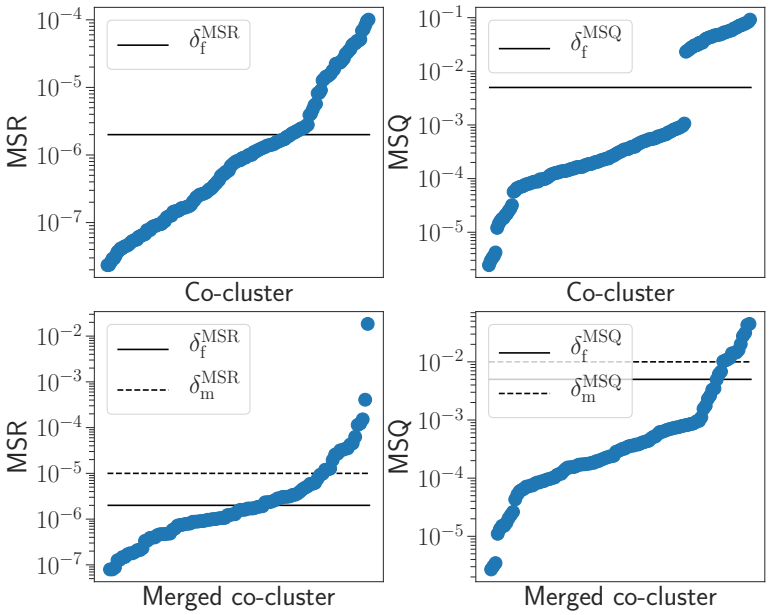
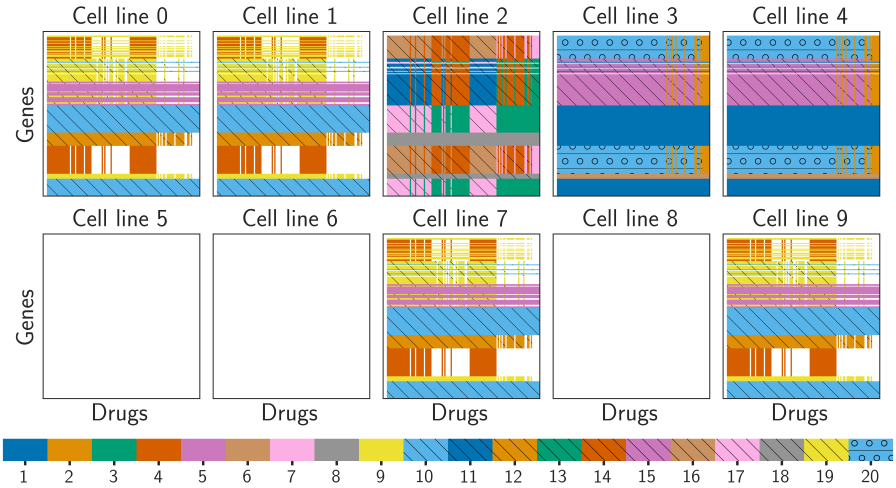
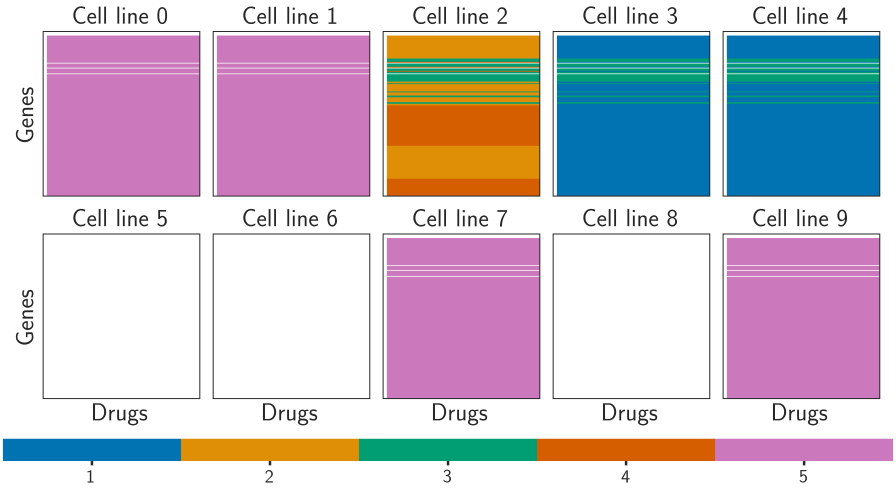


Figure 4.15: *Sorted MSR and MSQ for found co-clusters, and merged co-clusters. The data set has added noise of $\gamma = 3$, and the co-clustering is performed on the first sparse CP layer. The thresholds δ_f^{MSR} and δ_f^{MSQ} are selected from the simulation with no added noise.*



(a) *Final co-clusters using MSR.*



(b) *Final co-clusters using MSQ.*

Figure 4.16: *Final co-clusters using MSR (a), and using MSQ (b), on the first layer of the data set with $\gamma = 3$. Cell line 5 and 6 are set to zero in the sparse CP decomposition. The adjusted rand index compared to the run with no added noise is 0.36 for (a), and 0.92 for (b).*

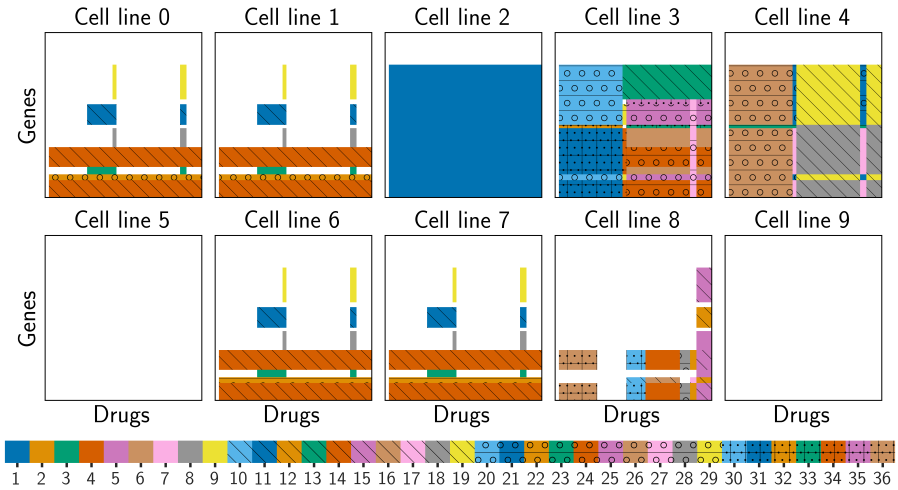
it can be noted that the S-shaped curve is flattened when noise is added. Secondly, the gap disappears for the MSR plot when noise is added. For the MSQ however, the gap is instead seen to widen when adding noise. Still concerning Figure 4.13 and Figure 4.15, it can be seen for the merged co-clusters that the noise is not that impactful on the general appearance of the curves. An obvious difference between the MSQ figures can however be seen. In Figure 4.13, the thresholds $\delta_{\mathrm{f}}^{\mathrm{MSQ}}$ and $\delta_{\mathrm{m}}^{\mathrm{MSQ}}$ intersects the curve slightly above the elbow, whereas in Figure 4.15 the thresholds intersects the curve higher up.

For the $\gamma = 3$ case, the final co-clusterings using MSR and MSQ can be seen in Figure 4.16. The co-clustering is performed on the first layer, and is sorted in the same manner as in Figure 4.14 to enable easier comparisons. The CPV of this specific example is 10.50%, and the ARI using MSR is 0.36, and 0.92 when using MSQ. For the final co-clusterings using MSR, the following observations can be made. Comparing the case with no noise in Figure 4.14a, to the case $\gamma = 3$ in Figure 4.16a, some co-clusters patterns are seen to remain, in cell lines 2–4, but also be disrupted by the noise forming new co-clusters. Additionally, new co-clusters are added in cell lines 0, 1, 7 and 9. Note that cell lines 5 and 6 are zero, for both MSR and MSQ, due to that corresponding elements in $\mathbf{c}_1$ are set to zero in the sparse CP decomposition. Another interesting remark that the MSR and MSQ co-clusterings have in common, is that cell line 8 is without new co-clusters despite the noise. For the MSQ final co-clusterings in Figure 4.16b, the noise is also seen to disrupt the original co-clusterings notably. Few number of co-clusters are obtained for the final co-clustering with MSQ. Additionally, the cell lines 0, 1, 7 and 9 are relatively unchanged and consists almost only of a single large co-cluster. This results in that the ARI is very high for this co-clustering. Figures for another run using the same parameters are found in Appendix B, in Figure B.7. This run had ARI of 0.016 for using MSR, and 0.64 using MSQ.

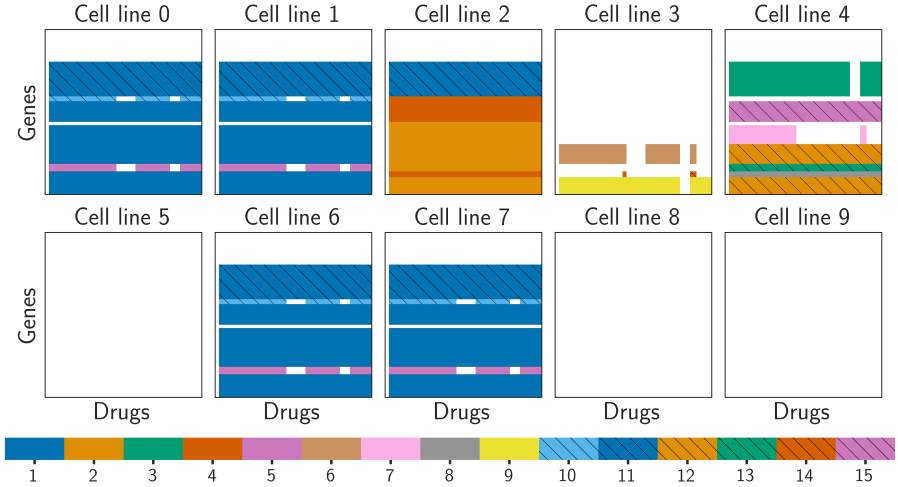
4.4 Co-clustering on Genomic Data Set

In this section, the sCP-HC is applied on a larger subset of the genomic data set. The section is structured by describing the co-clusterings of each layer, one by one, and for the different scores, MSR and MSQ. Similarities and differences between the co-clusterings using different scores are highlighted. The MSR and MSQ plots, used for selection of the filtering and merging thresholds, for all clusterings are provided in Appendix C. What is of special interest to observe in the following figures are groups of genes whose expression levels behave in a coherent fashion across subsets of drugs and cell-lines. This means that co-clusters that span over all drugs, are not very interesting.

Before presenting the layers, it can be noted that the relative contributions of each layer to the CPV are similar in magnitude. Therefore, the first layer is not vastly more important than the following layers. Explicitly, the first layer accounts for roughly 6% of the CPV, whereas the second and third layers accounts for about 4% each. The relative importance of the layers can also be noted from the values of d,



(a) *Final co-clusters using MSR.*



(b) *Final co-clusters using MSQ.*

Figure 4.17: *Final co-clusters using MSR (a), and using MSQ (b), on the first layer.*

i.e. the scaling factor of the sparse CP decomposition, where in fact the third layer has a slightly higher value of d than the second layer.

In Figure 4.17a, the first layer of the final co-clustering of the real genomic data is shown, using MSR as a scoring function. A total number of 36 co-clusters is obtained in this layer. Cell lines 5 and 9 are seen to be empty of co-clusters. This is due to the sparse CP decomposition, in which $\mathbf{c}_1$ is zero in the elements corresponding to these layers. Cell line 2 is seen to consist of a single co-cluster, making it relatively uninteresting. The same co-clusters span over the cell lines 0, 1, 6 and 7, whereas cell lines 3, 4 and 8 each have multiple co-clusters that are unique to each cell line. Especially interesting here are the drugs which display a different behaviour across the genes, i.e. the drugs that contain the co-clusters 3, 8, 11 and 9. These vertical lines appear for drug 35, 42, 44, 87, 94, 98, 110 and 116. In cell line 3 and 4, the

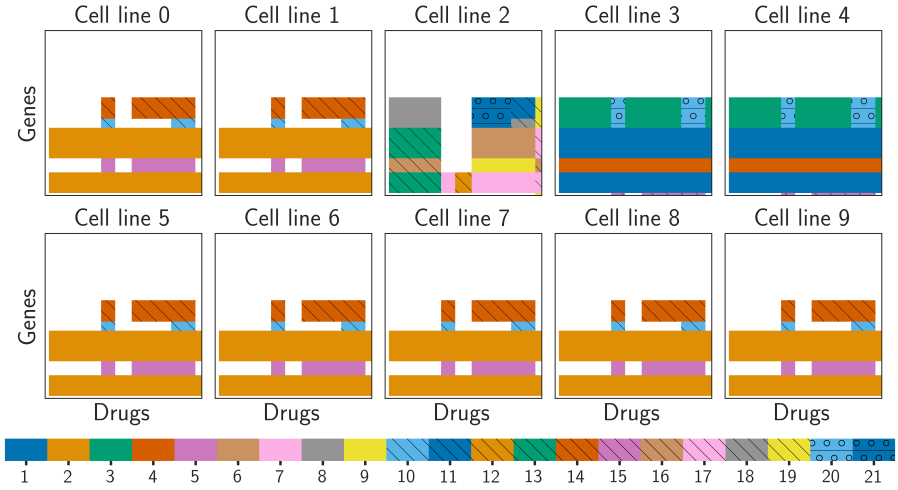
same drugs display a unique response relative to the other drugs in the same cell line. Cell line 3 overall displays a diverse range of responses for the various drugs across the genes.

In Figure 4.17b, the corresponding final co-clusterings are seen, but using MSQ as a scoring function. Here, only 15 different co-clusters are found. As before, cell lines 5 and 8 are empty, due to the sparse CP decomposition, whereas cell line 9 is filtered away. The same co-clusters span across cell lines 0, 1, 6 and 7. Since these co-clusters almost span over all the drugs, they are not very interesting. The cell line 3, 4 and 7 consists of co-cluster unique to each cell line, as was the case for the MSR co-clustering. Cell line 2 only consists of co-clusters spanning over all genes, making it less interesting. Cell lines 3 and 4 display a slightly more noteworthy behaviour. Interestingly, the small co-cluster 14 in cell line 3, involves the same drugs as discussed previously. They correspond perfectly to co-cluster 21 in cell line 4 in Figure 4.17a. There are other similarities between the to co-clusterings. Many of the co-clusters which are constant over the drugs in the MSQ plots, are also co-clustered together in a similar fashion in the MSR plots. However, whereas these co-clusters are uninterrupted as blocks in the MSQ plot, they get partly disrupted by other co-clusters, in the MSR plot. For example, compare co-cluster 24 in Figure 4.17a with co-cluster 6 in Figure 4.17b.

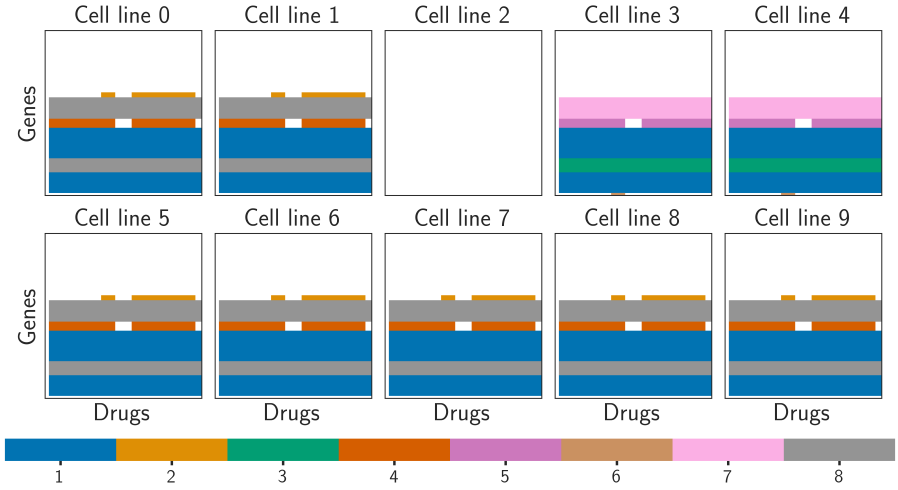
In Figure 4.18a, the final co-clustering of the second layer is seen, with MSR as a scoring function. Here, the cell lines have dominantly a common response, as seen by that the same co-clusters spans across cell lines 0, 1 and 5–9. Interestingly enough, the same co-clusters span across cell lines 3 and 4 in this layer. Cell line 2 is the only unique cell line in this layer, having many unique co-clusters over the drugs and genes. In total, the MSR co-clustering identifies 22 different co-clusters. However, only the genes in cell line 2 show a varied response for the different drugs, i.e. multiple variations of the co-clusters across the columns. For example, see the bottom of slice 2, where co-cluster 13, 7, 12, and 17 are present.

Figure 4.18b shows the corresponding co-clustering, using MSQ. Similarly to the MSR co-clustering, the same co-clusters span across cell lines 0, 1 and 5–9. A majority of the co-clusters span over all the drugs, or almost all the drugs. Several of these co-clusters are also present in the MSR co-clustering, in Figure 4.18a. Cell line 2, which in the MSR case contained a larger variety of co-clusters, was filtered away in the MSQ co-clusterings. Overall, the MSR co-clustering does not differentiate the response of any drug in the cell lines, and as such is not very rewarding.

Moving on to the third layer, the final co-clusterings using MSR are seen in Figure 4.19a. This co-clustering contains the most co-clusters out of all the co-clusterings, considering all layers and both scoring functions. A checkerboard-like co-clustering pattern is seen to span the cell lines 0–2 and cell line 8. Another set of co-clusters span across cell lines 6 and 9. Cell line 5 is found empty, due to the sparse CP decomposition setting the corresponding element to zero in $\mathbf{c}_3$. Likewise, the top part of the cell lines are empty due to the sparse CP desomposition. Cell line 2, 4 and 7, on the other hand, have unique co-cluster patterns. Making a broad grouping of the genes that exhibit a similar response across different drugs and cell lines,

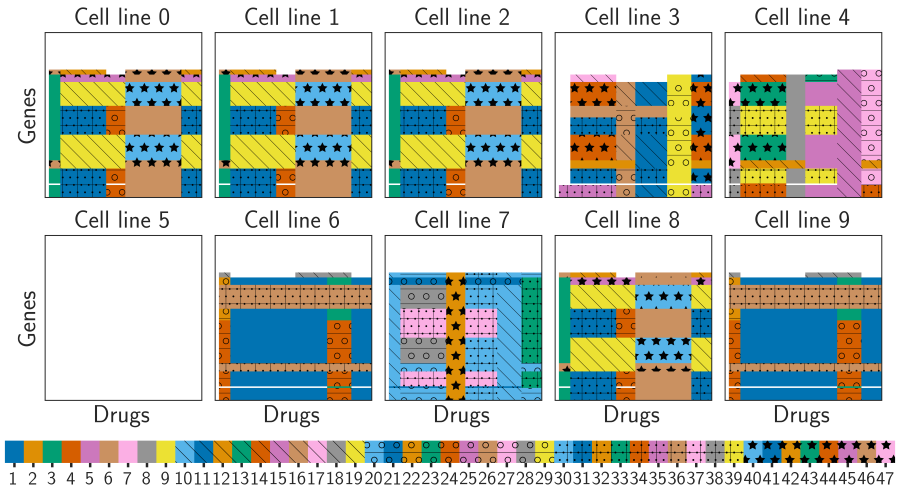


(a) *Final co-clusters using MSR.*

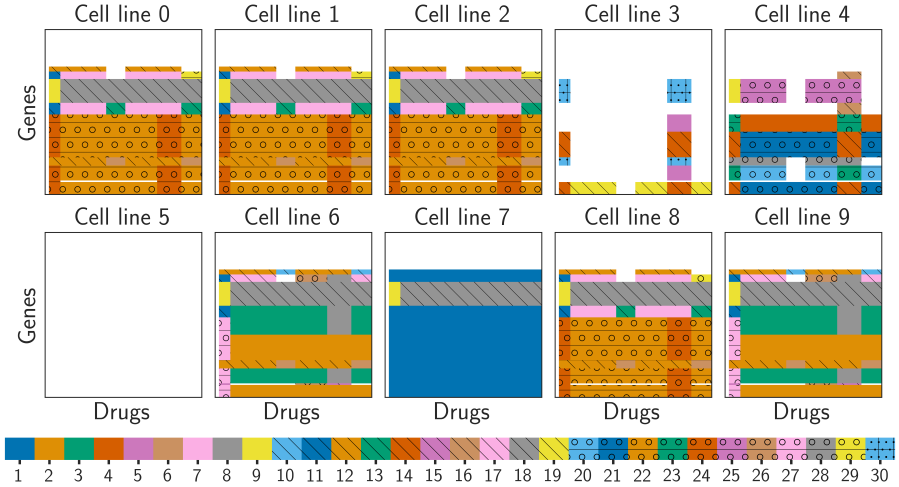


(b) *Final co-clusters using MSQ.*

Figure 4.18: *Final co-clusters using MSR (a), and using MSQ (b), on the second layer.*



(a) *Final co-clusters using MSR.*



(b) *Final co-clusters using MSQ.*

Figure 4.19: *Final co-clusters using MSR (a), and using MSQ (b), on the first layer.*

there seem to be mainly three groups of genes. The first such group are the genes that in e.g. cell line 3, looking from left to right, starts and ends with co-cluster 44. Observing the same genes for different cell lines, it can be seen that their responses are typically similar for various drugs. The second group consists of the genes that in cell line 3 starts with co-clusters 6 and 31. Note that the genes starting with co-cluster 31 occur on two different row levels. The third group concern the bottom rows in each cell line.

Some similarities, regarding which cell lines that share co-clusters, can be found when using MSQ to determine the final co-clusterings, as found in Figure 4.19b. Here, cell lines 0–2 and cell line 8 also share co-clusters. However, the patterns in cell lines 0–2 and cell line 8 across drugs and genes are not the same when comparing the co-clusterings for the two scoring functions. Some similarities can be found, e.g.

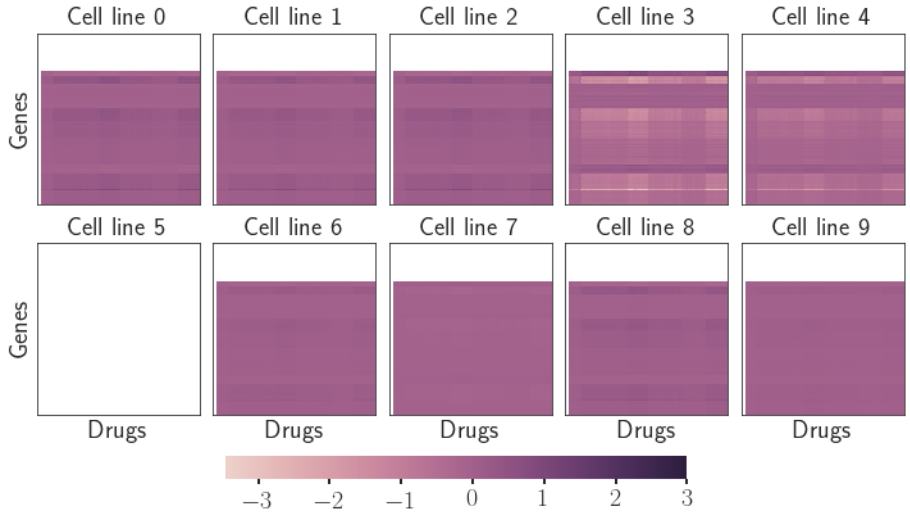


Figure 4.20: *Third layer of sparse CP decomposition, sorted in the same manner as Figure 4.19.*

in the upper rows and in the most left-hand side columns of cell lines 0–2 and cell line 8. However, the remaining co-cluster patterns in cell lines 0–2 and cell line 8 differ when comparing the co-clusterings of the two scoring functions.

Still concerning Figure 4.19b, cell lines 6 and 9 are seen to share co-clusters also, in similarity with the co-clustering using MSR. The upper rows, and the most left-hand side columns, show a similar co-clustering pattern as was found using MSR. Cell line 7 is seen to be unique, but to contain few, and not very informative, co-clusters. Cell lines 3 and 4 are, once again, seen to be the most diverse in terms of the number of co-clusters and the co-cluster patterns. Some similarity in the co-clusterings can be found between cell line 4, for the two different scores, e.g. on the most left-hand side columns and the row structure. Cell line 3 however, is vastly different for the MSQ co-clustering compared to the MSR co-clustering. Most of the co-clusters were filtered away in the MSQ case.

For comparison, the third layer of the sparse CP decomposition, sorted in the same manner as Figure 4.19 is provided in Figure 4.20. Of all layers, this layer contained the most interesting structures. This is not surprising considering that the co-clusterings on this layer also contained the most complex structures. Some similarities between the figures can be seen. Looking closely in Figure 4.20, it can be discerned that cell line 0, 1, 2, and 8 displays the same structure, and cell line 6 and 7 seem to display similar patterns as well. Contrarily, cell line 3, 4 and 7 appear to be unique. The similarities between the cell lines occur in a corresponding way for the MSR and MSQ clusterings, in Figure 4.19. Moreover, looking at specific cell lines, we can also see that the co-clusterings correspond to some structures in the data. The most obvious example of this, is cell line 3 in Figure 4.19a and Figure 4.20. To highlight some of the likeness, consider the following examples. Firstly, the brown co-clusters, co-cluster 16 and 26, create a vertical structure that is seen as a vertical light band in Figure 4.20. Secondly, it is also seen that the co-cluster that are removed in the filtering process in the left part of cell line 3 in Figure 4.19a,

have a less obvious structure in Figure 4.20.

5

Discussion

In this chapter, the simulation results will be discussed here in a corresponding order. The discussion will start on the topic of selecting a scoring function, after which it will proceed to give a brief discussion on the limitations of the sCP-S. After that, some interpretation to the results of the stability investigation will be given, followed by a section that comments the findings from applying the sCP-HC on the real genomic tensor data set. Finally, future work is discussed to wrap up the chapter.

5.1 Selection of Scoring Function and Thresholds

From Section 4.1 it is evident that the sCP-HC has the potential of finding the correct number of clusters. The potential to find the correct number of co-clusters relies firstly on that the sparse CP decomposition works well. Secondly, it also relies on that the sCP-HC results in co-clusters that are either correct from the beginning, or have the potential to be merged into the correct co-clusters. However, there are two main difficulties that impact the final results, related to the interpretation of the MSR and MSQ plots. One difficulty is to find a reasonable threshold for the MSR and MSQ, whereas another difficulty is to decide which of these scores that is more suitable.

To start, the choice of the scoring function will be discussed. In Figure 4.1, the results of investigating constant co-clusters are presented. Here, the correct co-clusters were found directly by the sCP-HC, without need of co-cluster merging or filtering. This is confirmed by both the MSR and MSQ scores, in the first row of the figure. In addition, they both indicate that merging is inappropriate by the increasing scores on the second row. As such, both the MSR and MSQ in this case confirm that these measure can detect constant co-clusters. For co-clusters that are constant on slices however, in Figure 4.3b, co-cluster merging was necessary to detect the complete co-cluster. In this figure, both the MSR and MSQ plots show agreement in that the unmerged co-clusterings are coherent, which means that no filtering was necessary. However, only by means of the MSR is it suggested that co-cluster merging is suitable. Therefore, it is seen constant slice co-clusters can be found by using MSR as a scoring function, whereas that is not the case using MSQ.

From looking at Figure 4.5b, a similar conclusion as for the constant slice co-clusters

can be drawn. The MSR measure is, to a greater extent, able to detect co-clusters that are constant on mode-1 fibers. This is seen by the low MSR scores for the merged co-clusters, whereas the merged MSQ scores indicate that no co-cluster merging is suitable. However, a difficulty unrelated to the choice of scoring function is also observed here, namely the fact that not all co-cluster mergings are suggested for the MSR. This is due to the fact that a greedy merging algorithm was used. The algorithm proceeded to merge the all co-clusters pairwise, in order of which had the best co-cluster scores, but limited each individual co-cluster to one merging per iteration. As such, if two different co-cluster merging alternatives were suggested but involved a common co-cluster, only the best merging was performed. The limitation of one merging per iteration was necessary to impose, to avoid invalid mergings. However, only merging the best co-cluster mergings per iteration meant that co-cluster mergings that in the current iteration were inferior, but that in a later iteration could yield better results, were discarded. To address this short-coming of the merging algorithm, a more extensive algorithm could have been implemented. This could have evaluated each possible co-cluster merging alternative, in each iteration, in parallel branches and not only the best merging option per iteration. Whilst more complex, such an approach could have found more mergings. However, this could computationally taxing, since listing all possible merging trajectories would result in a tree structure that might grow exponentially.

The last co-cluster type that was investigated was a coherent co-cluster, as seen in Figure 4.8b. Here it can be observed that the found co-clusters by the sCP-HC are all very coherent by the MSR measure, but not by the measure of the MSQ. From this it is seen that the MSQ will not be able to detect coherent co-clusters. However, there were difficulties with detecting coherent co-clusters also by the MSR measure since none of the co-cluster mergings were comparably coherent. Considering that the data set follows an additive model, and that the other simulations on data sets described by additive models have low MSR's, we would expect lower MSR's in this plot as well. One plausible explanation, as to why this is not the case here, is that the sparse CP decomposition did not reflect the original data set well enough. The co-clustering is based on the vectors of the sparse CP decomposition. Thus, if the decomposition is not sufficiently good, the co-clusters in the approximated tensor will not be entirely described by an additive model. Unfortunately this explanation becomes less likely for this particular case, when considering that the CPV here is very high.

These simulations indicate that the MSR has the potential to detect many different types of co-clusters, whereas the MSQ is limited to ones that are constant. As such, the MSR is seemingly a more potent scoring function to use for detecting more complex co-clusters in data. Regardless of which score is chosen though, a second difficulty must be addressed, namely the difficulty of choosing thresholds for filtering and merging.

The choice of the filtering and merging thresholds is not always easily made, and can introduce arbitrariness to the clustering procedure. This can be especially true for noisy data, which can be exemplified by comparing Figure 4.13 to Figure 4.15. Here, the curve without added noise looks more like a knee plot, whereas the noise

flattens the curve. As such, the choice of the threshold becomes more difficult for noisy data.

The implications of choosing the thresholds poorly can be quite considerable. As a first consideration, the choice of the thresholds will determine the final number of co-clusters. Reducing the final number of co-clusters can simplify the analysis, but at the risk of loosing valuable information. Going into more detail, one can consider the extremes. If the thresholds are chosen too high, poor co-clusters can be merged and presented in a final co-clustering despite having poor cluster coherence. This can be problematic, given that the final co-clustering plot does not make any distinction between which co-clusters that have high or low coherence. As such, reliability could become an issue, and a seemingly interesting co-cluster pattern could turn out to be merely noise. On the other hand, choosing the thresholds too low can also be problematic since it can cause interesting co-clusters to be removed in filtering. This could for example concern a co-cluster that gives meaningful insights into the data, but that is subject to higher noise than the less interesting observations. With all of this in mind, an appropriate procedure for choosing the threshold could be to first set its value around the knee of the plot. Then, after observing the resulting final co-clustering the threshold can be adjusted, as necessary, without deviating far from the knee.

5.2 Limitations of the sCP-S

In Section 4.2, the main difference of sCP-S and sCP-HC was highlighted. Due to the sCP-S not distinguishing the magnitudes of values of the same sign, it was not able to find the correct co-clusters. Still, in some applications where the magnitudes are not of great interest, this might not pose a problem. Furthermore, it was shown that if centring was performed beforehand, the sCP-S was able to detect some structure of the data. This is in likeness to the SSVD in Lee et al. [11]. There the SSVD was applied to a data set in order to detect significantly expressed genes for a set of subjects, but before the application of SSVD, the data set was centred and scaled. In contrast to the sCP-HC, the sCP-S needed pre-processing of the data set to find the correct number of co-clusters. This study therefore also highlights the need of pre-processing of the data depending on what co-clustering method is chosen.

5.3 Interpretation of Stability Test

In Section 4.3, the results of investigating the sCP-HC co-clustering stability are presented. In this section, a number of the results that were obtained will be discussed, with the overall goal of assessing the stability of the sCP-HC.

In Table 4.2, the mean ARI for both the MSR and MSR co-clusterings are presented as a function of the SNR. This result gives indication of how stable the co-clusterings are in the presence of various degrees of noise, i.e. if the co-clusterings change when noise is added. Here, it was observed that the mean ARI for the MSQ was higher than for the MSR. This is not a completely unexpected result. Considering that the

MSR can detect a variety of additive co-cluster types, it could also be more sensitive to misinterpreting noise as a type of additive pattern in the data. However, an additional finding was that the mean ARI for the MSR exceeded the mean ARI for the MSQ, in the first layer when $\gamma = 5$. This might be explained by that if the layer data follows an additive model, the MSR scoring function could better represent the true co-clustering and thus give a more correct co-clustering when the noise is low.

Another aspect of Table 4.2 to consider is that the second layer had much lower mean ARI's for both the MSR and MSQ. The low ARI values indicates that the second layer's co-clustering was more sensitive to noise. An explanation as to why this would be the case was not discovered, though it could perhaps be connected to that the proportion of variance for this layer, which was around 6.0%, was lower than for the first and third layers, which were about 11.7% and 7.4%, respectively.

It is found in Figure 4.13, compared to Figure 4.15, that the MSR and MSQ curves for varying unmerged co-clusters change their appearance in the presence of noise. For these co-clusters, seen on the first row of the respective figures, the MSR curve was seen to flatten when noise was added, whereas the gap between co-clusters widened for the MSQ. A possible explanation for the change of the MSR curve could be that when noise is added, new additive patterns could be generated in the data as a consequence. This might result in that new co-clusters are found by the algorithm, which have MSR values in the gap. Additionally, if more co-clusters are found in the noisy data, compared to previously, that would stretch the x-axis to make the plot look more flat. For the MSQ however, the gap might instead have increased due to co-clusters near the gap being deteriorated by the noise and having jumped up above the gap.

Overall, it is seen from the stability simulations that noise can have a considerable impact on the performance and consistency of the co-clusterings. However, from the first and third layers in Table 4.2, it is seen that a large proportion of the co-clusters are consistent even in the presence of considerable, but not extreme, noise, i.e. $\gamma = 3, 4, 5$. This demonstrates the ability of the sCP-HC to find consistent co-clusters, even in noisy data

5.4 Application on Real Data

In this section, the resulting co-clustering of the sCP-HC when applied on real genomic data will be discussed. The goal of the chapter is to discuss what kinds of co-clusters that could be found, and the overall usefulness of the method.

In Figure 4.17a, co-clustering of the first layer of the sparse CP decomposition was made, using MSR as the scoring function. When studying these co-clusterings, certain patterns are of special interest. One such pattern to look for in the data is which drugs that yield unique responses on different cell lines and genes, i.e. the behaviour on the columns. From the MSR co-clustering, the drugs 35, 42, 44, 87, 94, 98, 110 and 116 were found to result in unique responses across cell lines 0, 1, 3, 4, 6 and 7. This demonstrates that the co-clustering is capable of differentiating

unique responses in the data, and co-cluster these responses in a plausible manner, which lays a basis for further analysis. However, comparing the MSR and MSQ co-clustering, the MSR co-clustering is much more informative in this first layer. As seen for the MSQ co-clustering in Figure 4.17b, looking only for constant co-clusters in the data did not considerably differentiate the response of any few drugs in comparison to the others.

The second layer, seen in Figure 4.18a, using MSR as the scoring function shows a complementary view of the results found in the first layer. Here, a dominantly common response is found across all cell lines with the exception of cell line 2. This is interesting, since cell line 2 had no information of interest when viewed in the first layer. This demonstrates how each layer can complement each other in finding different patterns in the data. Once again though, the co-clustering of the second layer using MSQ, as found in Figure 4.18b, yielded a quite homogeneous response across the cell lines, and did not differentiate any drug considerably.

From Figure 4.19a, the third layer's co-clustering using MSR resulted in a vast amount of co-clusters. There are several larger groups of drugs and genes that are co-clustered together. One co-cluster in particular that could be highlighted is co-cluster 24, which gives a coherent response on different drugs across different cell lines, as seen by comparing the co-cluster occurrence in cell lines 0, 1, 2 and 8 compared to cell lines 6 and 9. The usefulness of such a co-clustering results will depend on the study in particular, but this result is interesting to highlight merely to show the possibilities of finding a coherent response across three dimensions.

Finally, it could also be rewarding to look at the similarities between the co-clustering of the third layer, compared to the third sparse CP layer data itself, i.e. compare Figure 4.19 to Figure 4.20. In comparing these, some of the co-clusters that are found on this layer can actually be seen from mere visual inspection of the data. Considering for example cell line 3 of the data, which shows the most clear checkerboard pattern of the cell lines, a corresponding spatial structure is seen for the co-clusters. This can be viewed as a sanity check of that the found co-clusterings are really based on structures that are present in the data.

Overall, the sCP-HC is found able to distinguish several different types of co-clusters. Additionally, it is found that each layer can give a complementary information of the patterns in the data. As such, the method could have usefulness in exploratory data analysis, in conjunction with good domain knowledge of the data.

5.5 Future Work

There are several aspects of this project that could be extended or further investigated. Two of them are connected to the hierarchical clustering method used on the sparse CP vectors. In this project Ward linkage and euclidean distance was used. The impact of this choice has not been investigated, and it could be interesting to study other linkages and distance measures. The other potential improvement, is to develop a method for the selection of clusters on the sparse CP vectors that does

not rely on manual inspection of the dendrograms. For example, validation indices, such as silhouette width or Dunn index, could be used to access the goodness of the clustering.

Another relevant extension of this thesis would be to compare how the sCP-S and sCP-HC compare with other co-clustering methods, such as Zhao et al. [10]. Furthermore, it would be interesting to formally investigate the performance of the clustering algorithm for a growing number of entries in the tensor.

6

Conclusion

Two methods for co-clustering tensor data have been presented, namely the sCP-S and sCP-HC. The two methods are based on either clustering of sign patterns (sCP-S) or using agglomerative hierarchical clustering (sCP-HC), on the sparse CP components. The methods are similar, so with the increased flexibility of the sCP-HC taken into consideration, only the sCP-HC was evaluated more thoroughly. To assess the sCP-HC co-clustering performance in the following simulations, two different scoring functions, i.e. the mean square residue (MSR) and the mean sum of squares (MSQ), were used. Through a series of simulations, the sCP-HC was demonstrated capable of finding several types of additive co-clusters. However, it was observed to yield a non-exhaustive listing of all the possible co-cluster merging alternatives, due to the greedy co-cluster merging approach.

After this simulation, the sCP-HC was evaluated in regards to the stability of co-cluster detection under perturbations of the input data. This was done by comparing the co-clusterings, with and without the addition of noise to the data, and using the adjusted Rand index to determine the consistency of the obtained co-clusters. It was found that the sCP-HC can yield quite consistent co-clusterings, even in the presence of considerable noise. Certain layers could tolerate noise better than others, an observation as to which no conclusive explanation could be found. After the stability test, the sCP-HC was applied on a real genomic data set. Several interesting types of co-clusters could be found in the data, when using the MSR as a scoring function. Additionally, the property of the sCP-HC that each layer can yield complementary co-clusters was observed. However, the MSQ scoring function resulted in less interesting co-clusters than the MSR. Overall, it is concluded that the sCP-HC is a useful as a tool for exploratory data analysis.

Bibliography

[1] D. Reinsel, J. Gantz, and J. Rydning, "The Digitization of the World From Edge to Core," IDC, Tech. Rep., 2018.

[2] C. S. Greene, J. Tan, M. Ung, J. H. Moore, and C. Cheng, "Big data bioinformatics," pp. 1896–1900, 2014.

[3] M. Shirer and J. Rydning, "IDC's Global DataSphere Forecast Shows Continued Steady Growth in the Creation and Consumption of Data," 2020. [Online]. Available: https://www.idc.com/getdoc.jsp?containerId=prUS46286020&fbclid=IwAR1X6A1YsUIrNLQuXd8aeXiQN2TyKjUeomjKm68yEiuo_wstCEssU_DQF4M

[4] C. F. Beckmann and S. M. Smith, "Tensorial extensions of independent component analysis for multisubject FMRI analysis," *NeuroImage*, vol. 25, no. 1, pp. 294 – 311, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1053811904006378

[5] Y. Luo, F. Wang, and P. Szolovits, "Tensor factorization toward precision medicine," *Briefings in Bioinformatics*, vol. 18, no. 3, pp. 511–514, 3 2016. [Online]. Available: https://doi.org/10.1093/bib/bbw026

[6] Acar E., Çamtepe S.A., Krishnamoorthy M.S., and Yener B., "Modeling and multiway analysis of chatroom tensors," in *International Conference on Intelligence and Security Informatics*. Springer, 2005, pp. 256–268.

[7] B. Chen, A. P. Petropulu, and L. De Lathauwer, "Blind Identification of Convolutive MIMO Systems with 3 Sources and 2 Sensors," *EURASIP Journal on Advances in Signal Processing*, vol. 2002, no. 5, p. 432606, 2002. [Online]. Available: https://doi.org/10.1155/S1110865702000823

[8] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009.

[9] I. Oseledets, "Tensor-Train Decomposition," *SIAM J. Scientific Computing*, vol. 33, pp. 2295–2317, 7 2011.

[10] H. Zhao, Z. Wei, and H. Yan, "Multimodal Co-clustering Analysis of Big Data Based on Matrix and Tensor Decomposition," in *Multimodal Analytics for Next-Generation Big Data Technologies and Applications*. Springer International Publishing, 2019, ch. 5, pp. 95–124.

[11] M. Lee, H. Shen, J. Z. Huang, and J. S. Marron, "Biclustering via Sparse Singular Value Decomposition," *Biometrics*, vol. 66, no. 4, pp. 1087–1095, 12 2010.

[12] B. Pontes, R. Giráldez, and J. S. Aguilar-Ruiz, "Biclustering on expression data: A review," pp. 163–180, 10 2015.

[13] S. Van Aelst, X. Wang, R. H. Zamar, and R. Zhu, "Linear grouping using orthogonal regression," *Computational Statistics and Data Analysis*, vol. 50, no. 5, pp. 1287–1312, 3 2006.

[14] E. E. Papalexakis, N. D. Sidiropoulos, and R. Bro, "From K-means to higher-way co-clustering: multilinear decomposition with sparse latent factors," *IEEE transactions on signal processing*, vol. 61, no. 2, pp. 493–506, 2012.

[15] S. Jegelka, S. Sra, and A. Banerjee, "Approximation Algorithms for Tensor Clustering," in *International Conference on Algorithmic Learning Theory*. Springer, 2009, pp. 368–383.

[16] P. A. Forero, P. A. Baxley, and M. Capella, "Co-clustering of high-order data via regularized Tucker decompositions," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 3442–3446.

[17] W. H. Yang, D. Q. Dai, and H. Yan, "Finding correlated biclusters from gene expression data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 4, pp. 568–584, 2011.

[18] G. I. Allen Baylor, "Sparse Higher-Order Principal Components Analysis," Tech. Rep., 2012.

[19] T. G. Kolda, "SANDIA REPORT Multilinear operators for higher-order decompositions," Tech. Rep. [Online]. Available: http://www.doe.gov/bridge

[20] L. De Lathauwer, B. De Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1253–1278, 2000.

[21] S. Roman, S. Axler, and F. Gehring, *Advanced linear algebra*, 2nd ed. Springer, 2005.

[22] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, no. 3, pp. 211–218, 1936.

[23] L. Mirsky, "Symmetric gauge functions and unitarily invariant norms," *The quarterly journal of mathematics*, vol. 11, no. 1, pp. 50–59, 1960.

[24] D. P. Berrar, W. Dubitzky, and M. Granzow, *A practical approach to microarray data analysis*, 1st ed. Springer, 2003.

[25] F. L. Hitchcock, "The expression of a tensor or a polyadic as a sum of products," *Journal of Mathematics and Physics*, vol. 6, no. 1-4, pp. 164–189, 1927.

[26] J. D. Carroll and J.-J. Chang, “ANALYSIS OF INDIVIDUAL DIFFERENCES IN MULTIDIMEN-SIONAL SCALING VIA AN N-WAY GENERALIZATION OF "ECKART-YOUNG" DECOMPOSITION,” Tech. Rep., 1970.

[27] R. A. Harshman, “Foundations of the PARAFAC procedure: Models and conditions for an" explanatory" multimodal factor analysis,” *UCLA Working Papers in Phonetics*, vol. 16, pp. 1–84, 1970.

[28] J. B. Kruskal, “Three-Way Arrays: Rank and Uniqueness of Trilinear Decompositions, with Application to Arithmetic Complexity and Statistics,” *Linear algebra and its applications*, vol. 18, no. 2, pp. 95–138, 1977.

[29] S. Rabanser, O. Shchur, and S. Günnemann, “Introduction to Tensor Decompositions and their Applications in Machine Learning,” 11 2017. [Online]. Available: http://arxiv.org/abs/1711.10781

[30] J. B. Kruskal, “Rank, decomposition, and uniqueness for 3-way and N-way arrays,” *Multiway data analysis*, pp. 7–18, 1989.

[31] A. Bhar, M. Haubrock, A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, and E. Wingender, “δ-TRIMAX: Extracting Triclusters and Analysing Coregulation in Time Series Gene Expression Data,” in *International Workshop on Algorithms in Bioinformatics*. Springer, 2012, pp. 165–177.

[32] R. Henriques and S. C. Madeira, “Triclustering algorithms for three-dimensional data analysis: a comprehensive survey,” *ACM Computing Surveys (CSUR)*, vol. 51, no. 5, pp. 1–43, 2018.

[33] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[34] P. Cichosz, *Data mining algorithms: explained using R*. John Wiley & Sons, 2014.

[35] A. C. Rencher, *Methods of Multivariate Analysis*, 2nd ed. John Wiley & Sons, 2003.

[36] G. M. Downs and J. M. Barnard, “Clustering Methods and Their Uses in Computational Chemistry,” *Reviews in computational chemistry*, vol. 18, pp. 1–40, 2002.

[37] Z. Li, “5.43 - In Vitro Micro-Tissue and -Organ Models for Toxicity Testing,” in *Comprehensive Biotechnology*, 2nd ed., M. Moo-Young, Ed. Burlington: Academic Press, 2011, p. 553. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780080885049005031

[38] M. I. Love, W. Huber, and S. Anders, “Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2,” *Genome biology*, vol. 15, no. 12, p. 550, 2014.

[39] I. G. Costa, F. D. A. T. De Carvalho, and M. C. P. De Souto, "Comparative analysis of clustering methods for gene expression time course data," *Genetics and Molecular Biology*, vol. 27, no. 4, pp. 623–631, 2004. [Online]. Available: www.sbg.org.br

[40] K. Y. Yeung and W. L. Ruzzo, "An empirical study on principal component analysis for clustering gene expression data," *Bioinformatics*, vol. 17, no. 9, pp. 763–774, 2001. [Online]. Available: http://www.cs.washington.edu/homes/kayee/pca

[41] G. W. Milligan and M. C. Cooper, "A study of the comparability of external criteria for hierarchical cluster analysis," *Multivariate behavioral research*, vol. 21, no. 4, pp. 441–458, 1986.

[42] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[43] J. Kossaifi, Y. Panagakis, A. Anandkumar, and M. Pantic, "TensorLy: Tensor Learning in Python," *CoRR*, 2018.

[44] J. Kossaifi, "Tensor Unfolding." [Online]. Available: http://jeankossaifi.com/blog/unfolding.html

[45] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[46] M. Waskom, O. Botvinnik, D. O'Kane, P. Hobson, S. Lukauskas, and D. C. Gemperline, "mwaskom/seaborn: v0.8.1 (September 2017)," 2017.

A

Types of Co-cluster Simulation Figures

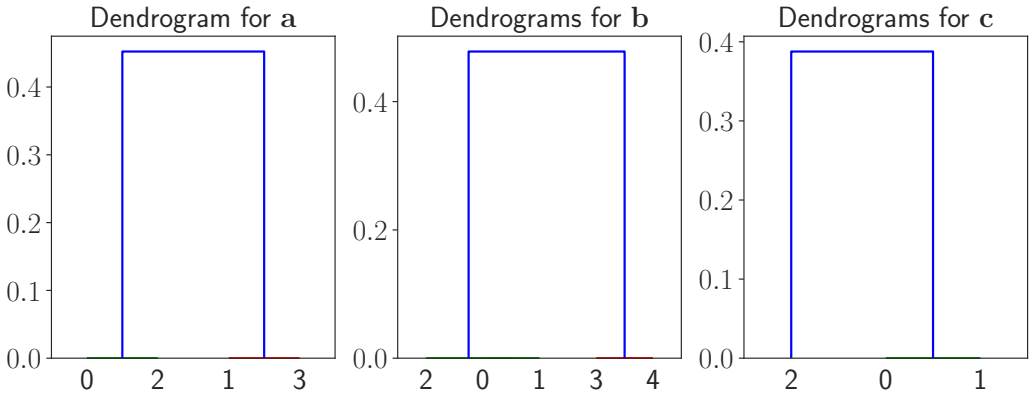


Figure A.1: *Dendrograms obtained from agglomerative hierarchical clustering on the sparse CP vectors* **a**, **b** *and* **c**. *The data set contains eight constant co-cluster.*

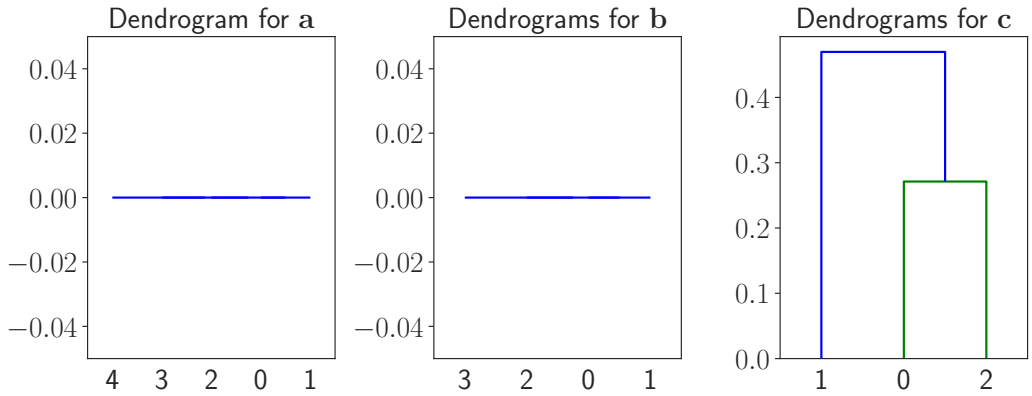


Figure A.2: *Dendrograms obtained from agglomerative hierarchical clustering on the sparse CP vectors* **a**, **b** *and* **c**. *The data set contains a co-cluster of constant values on the frontal slices.*

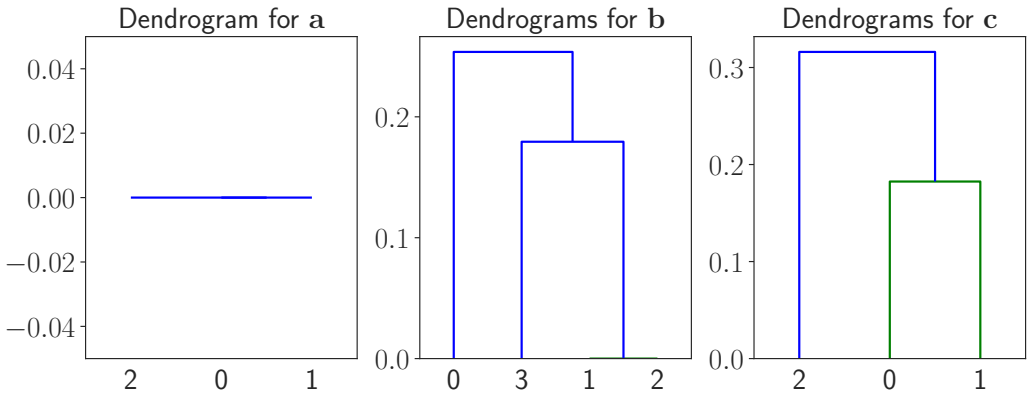


Figure A.3: *Dendrograms obtained from agglomerative hierarchical clustering on the sparse CP vectors* **a**, **b** *and* **c**. *The data set contains one constant mode-1 co-cluster.*

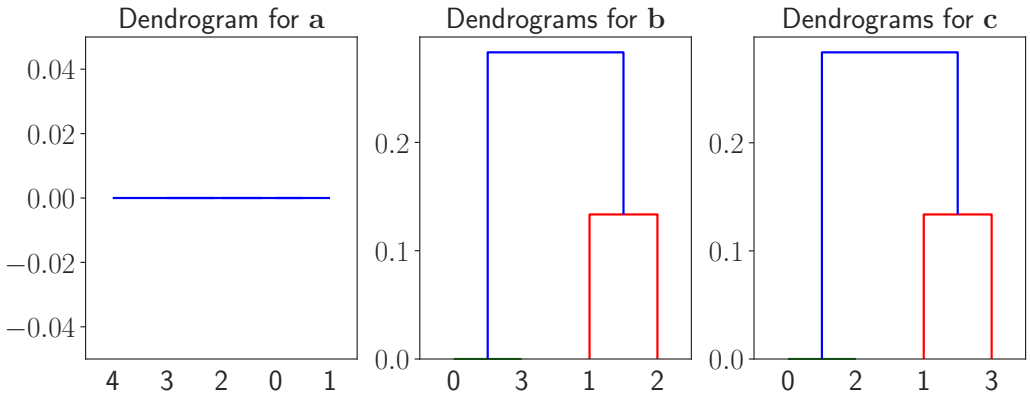


Figure A.4: *Dendrograms obtained from agglomerative hierarchical clustering on the sparse CP vectors* **a**, **b** *and* **c**. *The data set contains one co-cluster of coherent values on the rows, columns and tubes.*

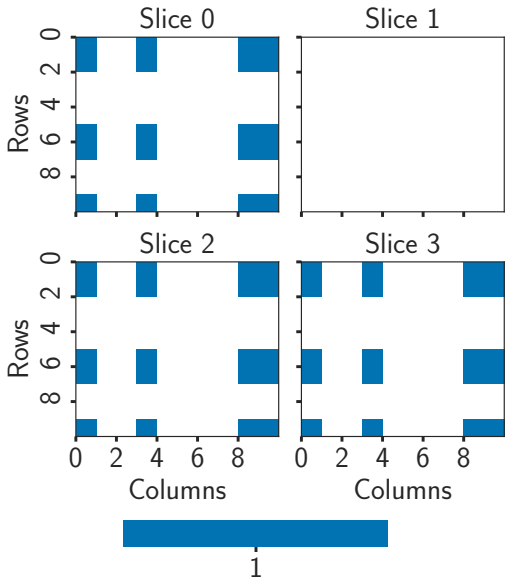


Figure A.5: *Slicewise illustration of co-clusters after co-cluster merging using MSR, for a data set consisting of a co-cluster of constant frontal slices. Two co-cluster mergings was performed, resulting in a single co-cluster. The data set before post-processing is seen in Figure 4.3b.*

B

Stability Test Figures

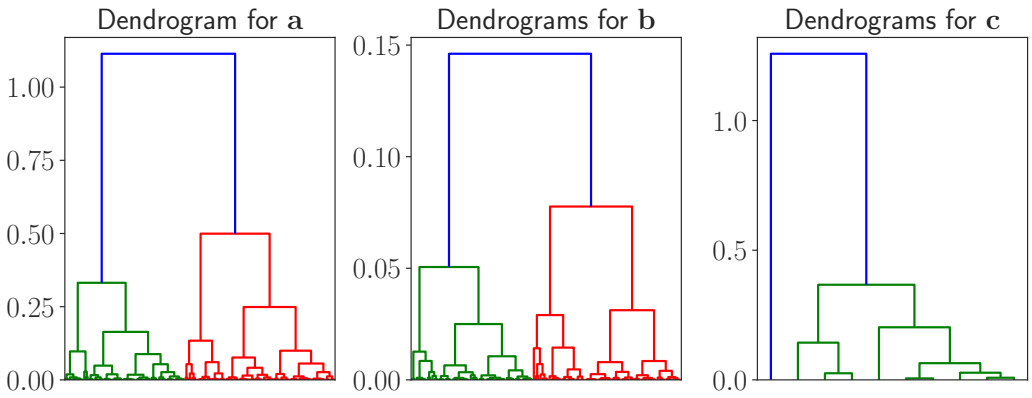


Figure B.1: *Dendrograms obtained from agglomerative hierarchical clustering on the sparse CP vectors* **a**, **b** *and* **c**. *The data set has no added noise, and corresponds to the second sparse CP layer.*

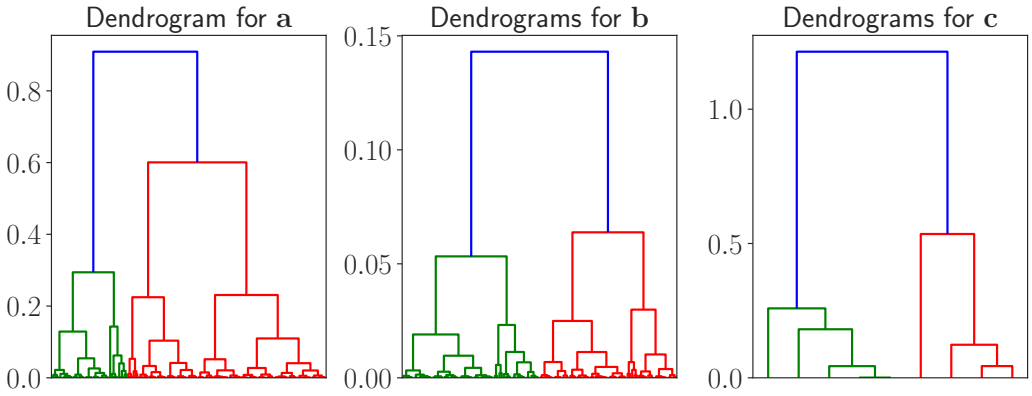


Figure B.2: *Dendrograms obtained from agglomerative hierarchical clustering on the sparse CP vectors* **a**, **b** *and* **c**. *The data set has no added noise, and corresponds to the third sparse CP layer.*

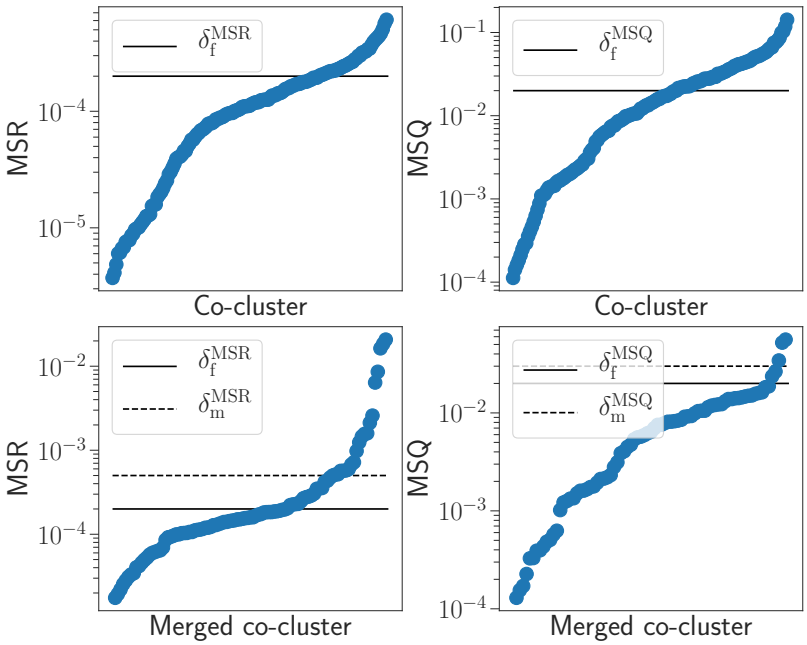


Figure B.3: *Sorted MSR and MSQ for found co-clusters, and merged co-clusters. The upper plots are used for selection of filtering thresholds δ_f^{MSR} and δ_f^{MSQ}, and lower plots are used for selection of merging thresholds δ_m^{MSR} and δ_m^{MSQ}. The data set has no added noise, and corresponds to the second sparse CP layer.*

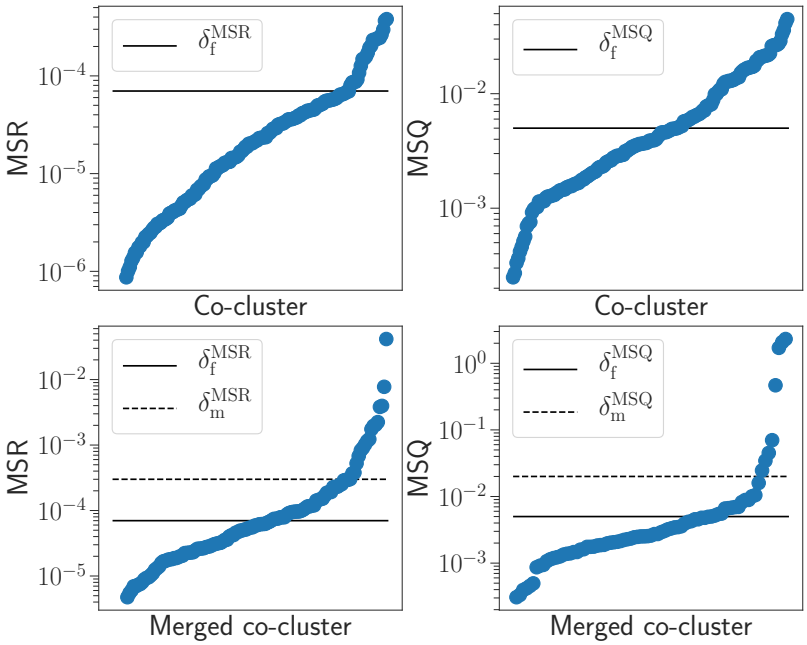
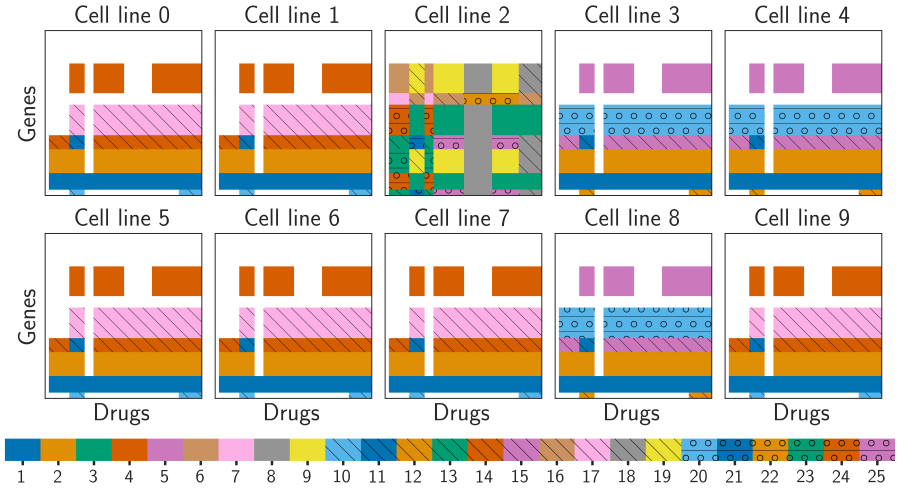
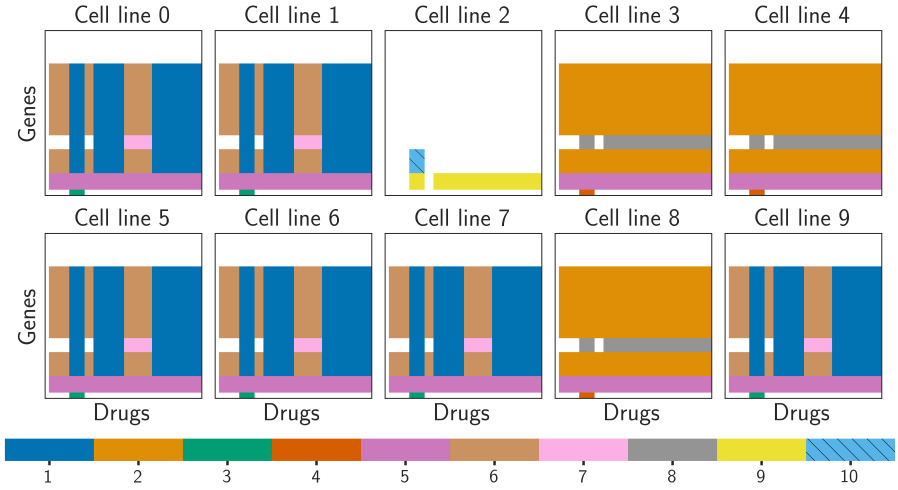


Figure B.4: *Sorted MSR and MSQ for found co-clusters, and merged co-clusters. The upper plots are used for selection of filtering thresholds δ_f^{MSR} and δ_f^{MSQ}, and lower plots are used for selection of merging thresholds δ_m^{MSR} and δ_m^{MSQ}. The data set has no added noise, and corresponds to the third sparse CP layer.*

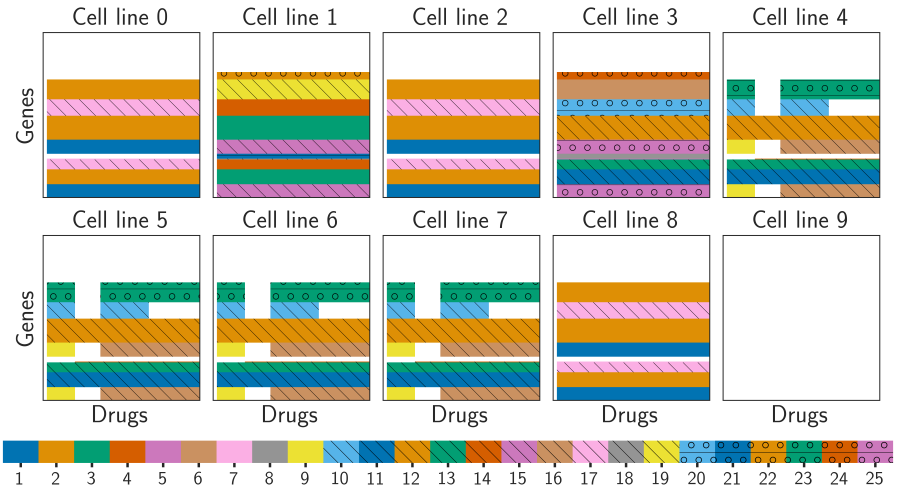


(a) *Final co-clusters using MSR.*

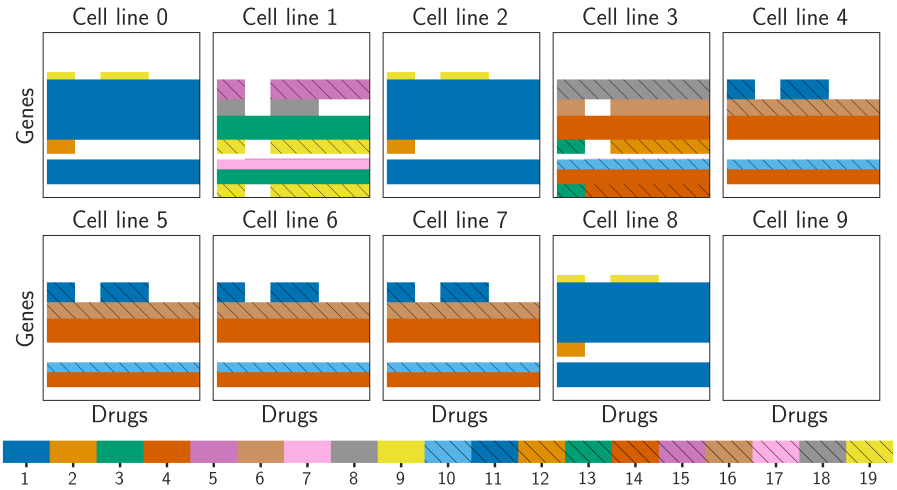


(b) *Final co-clusters using MSQ.*

Figure B.5: *Final co-clusters using MSR (a), and using MSQ (b), on the second layer of the data set with no noise.*

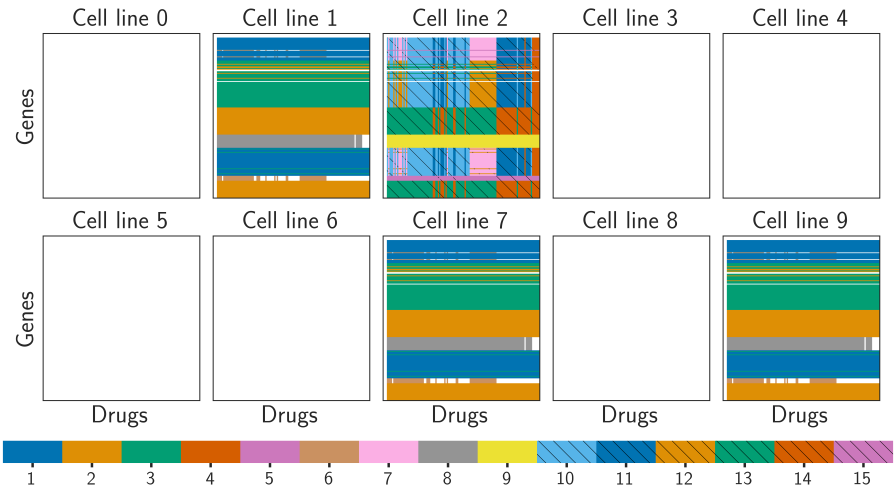


(a) *Final co-clusters using MSR.*

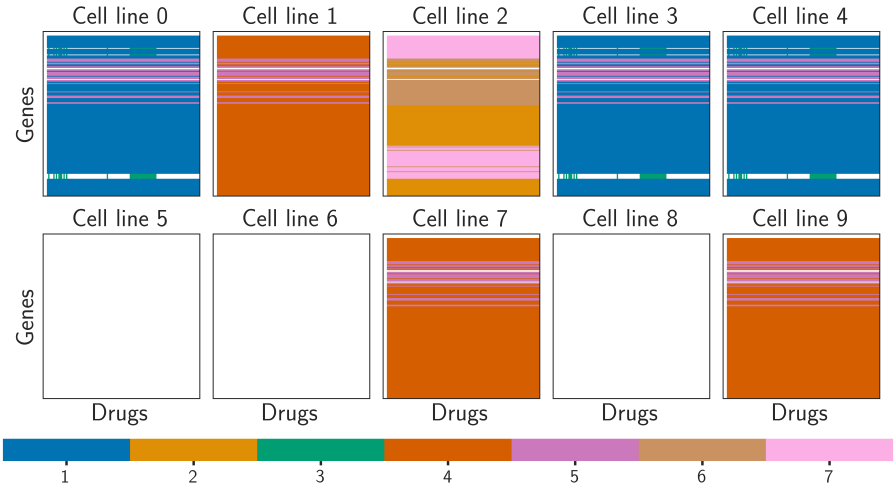


(b) *Final co-clusters using MSQ.*

Figure B.6: *Final co-clusters using MSR (a), and using MSQ (b), on the third layer of the data set with no noise. Cell line 9 is set to zero in the sparse CP decomposition.*



(a) *Final co-clusters using MSR.*



(b) *Final co-clusters using MSQ.*

Figure B.7: *Final co-clusters using MSR (a), and using MSQ (b), on the first layer of the data set with $\gamma = 3$. Cell line 5 and 6 are set to zero in the sparse CP decomposition. The adjusted Rand index compared to the run with no added noise is 0.016 for (a), and 0.64 for (b).*

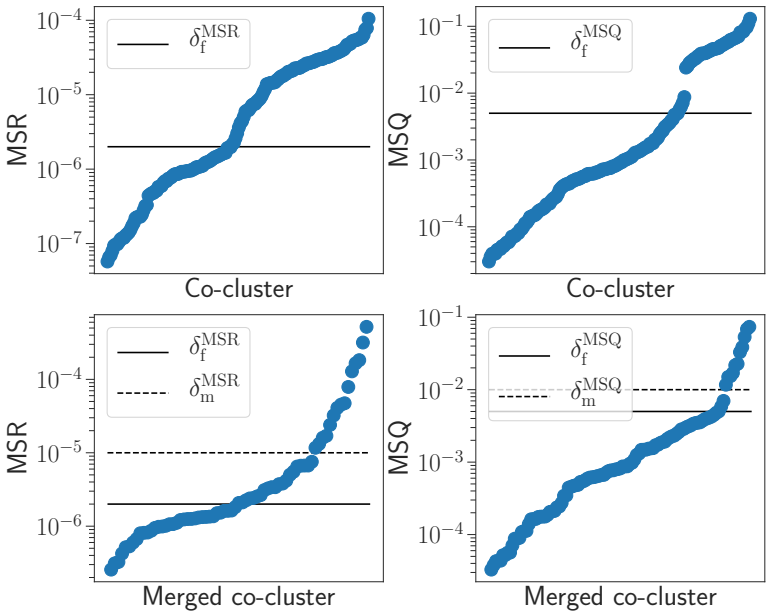


Figure B.8: *Sorted MSR and MSQ for found co-clusters, and merged co-clusters. The data set has added noise of $\gamma = 3$, and the co-clustering is performed on the first sparse CP layer. The thresholds δ_f^{MSR} and δ_f^{MSQ} are selected from the simulation with no added noise.*

C

MSR and MSQ Plots From Application on Genomic Data Set

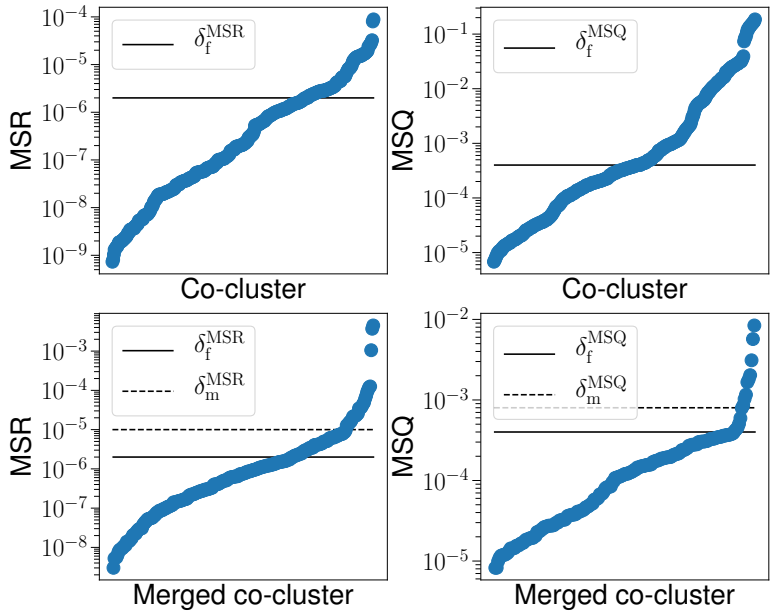


Figure C.1: *Sorted MSR and MSQ for found co-clusters, and merged co-clusters. The upper plots are used for selection of filtering thresholds δ_f^{MSR} and δ_f^{MSQ}, and lower plots are used for selection of merging thresholds δ_m^{MSR} and δ_m^{MSQ}. The data set corresponds to the first sparse CP layer.*

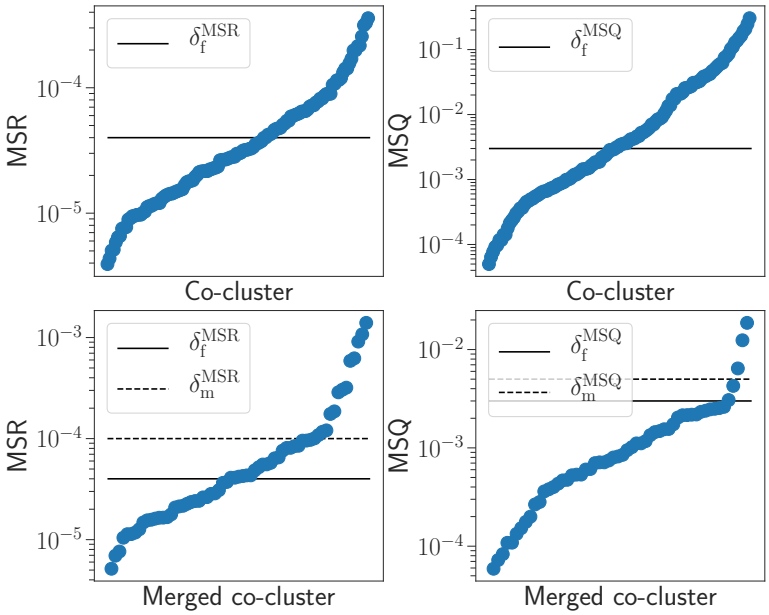


Figure C.2: *Sorted MSR and MSQ for found co-clusters, and merged co-clusters. The upper plots are used for selection of filtering thresholds δ_f^{MSR} and δ_f^{MSQ}, and lower plots are used for selection of merging thresholds δ_m^{MSR} and δ_m^{MSQ}. The data set corresponds to the second sparse CP layer.*

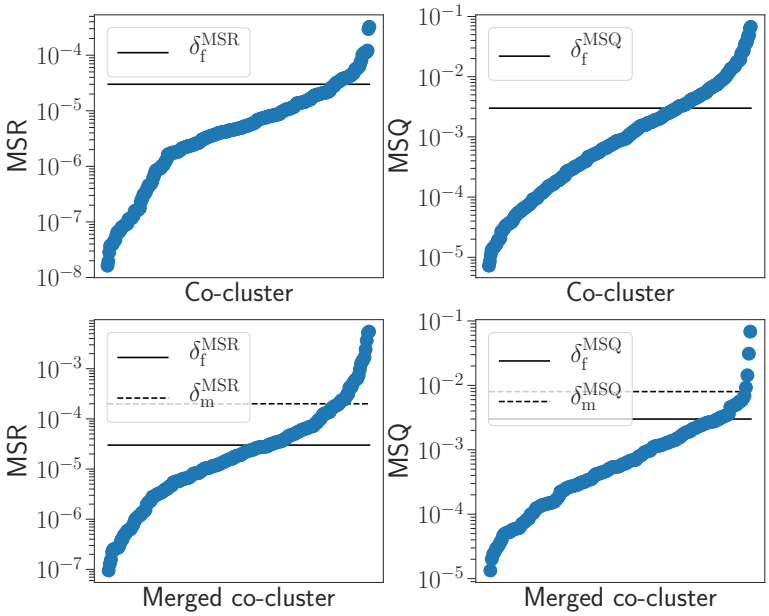


Figure C.3: *Sorted MSR and MSQ for found co-clusters, and merged co-clusters. The upper plots are used for selection of filtering thresholds δ_f^{MSR} and δ_f^{MSQ}, and lower plots are used for selection of merging thresholds δ_m^{MSR} and δ_m^{MSQ}. The data set corresponds to the third sparse CP layer.*