KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Rank Matrix Factorisation and its Applications

Thanh Le Van

Supervisor:
Prof. dr. Luc De Raedt
Co-supervisors:
Prof. dr. Kathleen Marchal
Prof. dr. Siegfried Nijssen

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Engineering Science: Computer Science

January 2017

Rank Matrix Factorisation and its Applications

Thanh LE VAN

Examination committee:
Prof. dr. Paul Van Houtte, chair
Prof. dr. Luc De Raedt, supervisor
Prof. dr. Kathleen Marchal, co-supervisor
Prof. dr. Siegfried Nijssen, co-supervisor
Prof. dr. ir. Jan Ramon
Prof. dr. ir. Johan Suykens

Dr. Anna Carolina Fierro
(Biogazelle NV)
Prof. dr. ir. Tijl De Bie
(Ghent University)
Dr. Matthijs van Leeuwen
(Leiden University)
Prof. dr. Pierre Schaus
(Université catholique de Louvain)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Engineering Science: Computer Science

January 2017

Abstract

Rank data, in which each row is a complete or partial ranking of available items (columns), is ubiquitous. It can be used to represent, for instance, preferences of users, the levels of gene expression, and the outcomes of sports events. While rank data has been analysed in the data mining literature, mining patterns in such data has so far not received much attention. To alleviate this state of affairs, in this thesis we study pattern set mining in rank data, i.e., the discovery of a small set of patterns that can describe the structure of rank data, and its applications in data mining and bioinformatics.

First, we propose a general framework based on matrix factorisation for mining different types of patterns in rank data. Rather than relying on the traditional linear algebra for matrix factorisation, we employ semiring theory, which results in a more elegant way of aggregating rankings. Subsequently, we introduce two instantiations of the framework: Sparse RMF and ranked tiling. We introduce Sparse RMF to mine a set of sparse rank vectors that can be used to summarise given rank matrices succinctly and show the main categories of rankings. We introduce ranked tiling to discover a set of data regions in a rank matrix which have high ranks. Such data regions are interesting as they can show local associations between subsets of the rows and subsets of the columns of the given matrices. Finally, we propose to use ranked tiling to formally define the concept of driver pathways, which is the molecular mechanism driving tumorigenesis. Given the discovered driver pathways, we can find cancer subtypes, which are groups of tumour samples having a unique combination of driver pathways.

Beknopte samenvatting

Gerangschikte data, waarin elke rij een complete of partiële ranking voorstelt van beschikbare items (kolommen), is alomtegenwoordig. Men kan er bijvoorbeeld preferenties van eindgebruikers, de verschillende niveau's van genexpressie, of de resultaten van sportevenementen mee voorstellen. Desondanks het feit dat gerangschikte data reeds worden geanalyseerd in de bestaande data mining literatuur, is er tot op heden weinig aandacht voor pattern mining binnen de context van gerangschikte data. Deze thesis vult deze schaarste in. Ze bestudeert het zoeken naar patronen in gerangschikte data, met een nadruk op het zoeken naar een kleine verzameling patronen. Verder worden enkele toepassingen van deze technologie in data mining en bioinformatica onderzocht.

Vooreerst introduceren we een een algemeen raamwerk voor het identificeren van verschillende types van patronen in gerangschikte data, gebaseerd op matrix factorisatie. In plaats van terug te vallen op de traditionele lineaire algebra voor matrix factorisatie, beroepen we ons op semiring theorie. Dit resulteert in een meer elegante manier om rankings te aggregeren. Vervolgens breiden we het framework uit met 2 instanties: Sparse RMF, en ranked tiling. We gebruiken Sparse RMF om een set van sparse rank vectors te identificeren die in staat zijn om gerangschikte matrices op een beknopte wijze samen te vatten. Daarnaast beschrijven we de voornaamste categorieën van rankings. We wenden ranked tiling aan voor de identificatie van een set van data regio's met een hoge rang in een gerangschikte matrix. Deze regio's zijn interessant gezien het feit dat ze lokale associaties tussen subsets van rijen en kolommen van de gegeven matrices kunnen weergeven. Tot slot introduceren we het gebruik van ranked tiling om driver pathways te definiëren. Dit zijn de moleculaire mechanismen onderliggend aan tumorigenesis. Gegeven de ontdekte driver pathways, zijn we in staat om verschillende kanker subtypes aan te duiden. Een kanker subtype is een verzameling van tumor stalen gekenmerkt door een unieke combinatie van driver pathways.

Acknowledgements

My PhD research has been a long journey, which I could not have accomplished without the help of many people. I would like to say a big THANK YOU to all of you.

First of all, I would like to thank my supervisors, Luc De Raedt, Siegfried Nijssen and Kathleen Marchal, for providing fundings and guiding me to do research during the past years. Without their expertise, vision, patience and support, I could not obtain this thesis text as it is today.

I am very grateful to Luc for offering me the opportunity to pursue a PhD research in his group when I was in Vietnam and for cultivating me in the truly friendly, supportive, honest and high-grade research environment that he has been maintaining. I would like to especially thank Siegfried who tremendously helped me to develop many research skills such as mathematical modelling. I am very grateful to have both Luc and Siegfried as computer science supervisors: Luc taught me to think critically about research problems, with the why, what and how questions, and to be concise on defining problems, while Siegfried taught me to be bold, to critically analyse algorithms to understand the intuition of their behaviours and to be dedicated to our research questions. I am also very grateful to Kathleen, my bioinformatics supervisor. I came to Leuven to pursue a PhD research with almost zero biological knowledge. After years of working with Kathleen, I now increasingly have the feeling that I can ask biological questions by myself.

I would like to thank all members of my jury, Jan Ramon, Johan Suykens, Anna Carolina Fierro, Tijl De Bie, Matthijs van Leeuwen, Pierre Schaus, for their interesting and useful feedback, which helped me to improve my thesis. Special thanks go to Matthijs and Carolina for the nice collaboration and help. It was a great pleasure to share the office with Matthijs for a number of years, where we could have a lot of coffee breaks and useful discussions. Together with Siegfried, Matthijs also trained me how to give presentations, how to critically

evaluate the performance of algorithms, how to write research papers and many other skills for data miners. Many thanks for the nights you stayed up late with me and Siegfried to finish papers on time! Carolina has patiently explained me many biological concepts, helped me to verify results and discussed with me many bioinformatics papers, all of which laid the foundation for the cancer research presented in this thesis when she left Kathleen's group. I also would like to thank Paul Van Houtte for chairing the committee.

I had the pleasure to share offices with many nice colleagues such as Tias, Vladimir (Vova), Matthijs, Siegfried, Albrecht, Vincent, Laura, Bogdan, Parisa and Martijn. Especially thanks to Tias and Vova for their help and discussions which were definitely useful for me to overcome research and life challenges. Many thanks to Vincent for translating the thesis abstract to Dutch. I also would like to thank Bernd, Angelika, Davide, Ingo who helped me to play with Problog when I was doing a pre-doc in the group. I have enjoyed ICON meetings with Anton, Tias, Sergey, Behrouz, Benjamin, Vova, Matthijs, Siegfried, Vincent and Samuel for the past years. It was also really fun to spend the 2013 summer school with Antoine, Sergey, Jan, Vova, Irma in Spain. Many thanks to other DTAI collegues such as Irma, Leander, Kurt, Jessa, Sebastijan, Francesco, Dries, Toon, Tim, Jonas, Christop, Juan, Jérôme, Alex and many others for Alma lunches, tea breaks, and discussions.

I also had the pleasure to collaborate and discuss with many bioinformatics colleagues in Kathleen's group: Carolina, Jimmy, Lieven, Dries, Sergio, Yan, Lore, Mostafa. Many thanks to all of you!

I also would like to thank Bettina Berendt who offered me a chance to do the teaching tasks. I enjoyed the course that I co-taught with Bettina for the Master students of Digital Humanities. Bettina showed me how to engage students in discussions and how to listen to students to get their ideas and then further develop these ideas into useful cases for the class.

I would like to thank the OscaR team for developing the elegant OscaR solver in Scala that I extensively used in my PhD research.

I would like to thank the Vietnamese Student Association in Leuven for organising many social activities, such as our traditional New Year Eve, barbecues and sports, for our family to join. Many thanks to the family of Van-Duc who helped me to set up my life when I first came to Leuven, to the family of Tam-Giang for countless number of sports activities, to Nga-Viet, Hai-Trung, Nga-Chum, Dat, Hong Trang, Thanh, Quoc, Tuan Bui, Quynh, Phuong Dong, Tung and many others.

I would like to thank the teachers in Sint Lambertus school in Heverlee, Juf Ria, and other teachers in Logopedisch, who helped Mai, my daughter, to learn

Dutch and to integrate with the society here.

I would like to thank the Groot Begijnhof village for offering our family a chance to stay in a beautiful place with multiple contract extensions.

I would like to thank Prof. Le Thi Hoai An for inviting me to visit her lab at University of Lorraine in Metz in 2015. I would like to thank Prof. Trinh Xuan Hoai and Dr. Pham Quang Dung for hosting my visit to Hanoi University and Hanoi University of Technology in 2016.

I would not have come to Leuven to do PhD if I had not met Prof. Peter Haddawy, my former advisor at AIT (Thailand) during 2005–2007. I would like to thank Peter for encouraging me to do research and convincing me research is an interesting career.

I am very grateful to the eye doctors in UZ Leuven hospital, who treated my eye inflammation in 2013 and got my vision back. A million thanks to the doctors! I also would like to thank other doctors in UZ Leuven, who took care of my family's health.

Finally, I would like to thank my family, my parents, my brother for their unconditional and continuous support and encouragement. I would like to thank my wife for understanding my interest in doing research and for supporting me to pursue the PhD research.

Thanh Le Van
Leuven, December 2016

Abbreviations

BMF	Boolean Matrix Factorisation
CP	Constraint Programming
cpRMT	CP-based approach to Rank Matrix Tiling
IP	Integer Programming
mRMT	Max-product Semiring Rank Matrix Tiling
NMF	Non-Negative Matrix Factorisation
PCA	Principal Component Analysis
RMF	Rank Matrix Factorisation
RMT	Rank Matrix Tiling
Sparse mRMF	Sparse Max-product Semiring Rank Matrix Factorisation
Sparse pRMF	Sparse Plus-product Semiring Rank Matrix Factorisation
Sparse RMF	Sparse Rank Matrix Factorisation
SRF	Subtyping Through Ranked Factors
sRMF	Semiring Rank Matrix Factorisation
SVD	Singular Value Decomposition

Contents

Chapter 1

Introduction

This thesis presents a framework for pattern set mining in rank data and its applications in data mining and bioinformatics. We first introduce rank data and pattern set mining. Then, we introduce motivations for a bioinformatics problem that we address using our framework. Finally, we provide an overview of the contributions of the thesis and its general structure.

1.1 Rank data

Data mining is the discovery of knowledge from data [113]. The discovered knowledge can be a predictive model, which can be used to predict the outcome of unseen instances, or a descriptive model, which can be used to describe what is in the data. The main research theme of this thesis is to discover descriptive data models. One typical example of this line of research is gene expression analysis. In this problem, we are given a data matrix, in which rows are genes, columns are samples and values are the gene expression values of the samples. Given this data, we would like to identify a subset of the genes which are consistently active (over-expressed) or inactive (under-expressed) in a subset of the samples. The data region identified by the subset of the genes and the subset of the samples is a descriptive model, which tells us the specific genes of the group of the samples.

The type of data often determines the tools and techniques that can be used for knowledge discovery. Much research focuses on data mining with Boolean data, numeric data, graph data and sequence data. In this thesis, however, we

study methods to discover descriptive models from *rank data*. In this type of data, each row is a complete or partial ranking of the columns. That is, for complete rankings, all of the columns are compared, and the most preferred column will be assigned the highest value, which is the number of the column; the least preferred column will be assigned the lowest value, which is 1. For partial rankings, only a subset of the columns are compared. The uncompared columns will be assigned 0. Note that the way the most/least preferred column is assigned a value is a choice of this thesis; it might be different in other work.

To illustrate rank data, let us consider the SUSHI dataset [54], in which customers are asked to rank 10 different types of sushis. The data can be represented as a matrix as follows:

$$\begin{pmatrix} 10 & 7 & 8 & 9 & 1 & 5 & 2 & 3 & 4 & 6 \\ 9 & 7 & 10 & 8 & 3 & 1 & 2 & 5 & 4 & 6 \\ 7 & 9 & 8 & 10 & 3 & 5 & 2 & 1 & 4 & 6 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

In this matrix, the first customer (row) likes the first sushi the most and likes the fifth sushi the least. This data consists of complete rankings.

There are two main motivations to study data mining in rank data:

- Rank data is ubiquitous in many applications. In social sciences, rank data has been used to represent users' preferences over their favorite countries [69], presidency candidates [26, 10] or products [54]. In biological sciences, rank data has been used to represent the levels of gene expression [7, 69, 67]. In sport analytics, rank data has been used to rank sport teams/participants [92, 24, 60].
- Rank data is a useful concept to abstract numeric data. In many cases, especially when data has rows at different scales, transforming the data to rankings may result in a simpler representation [69, 68, 67].

Rank data analysis has been studied in machine learning with *learning to rank* [75], in artificial intelligence with *preference learning* [34], in social science with *rank aggregation* [73]. Surprisingly, so far descriptive data mining in rank data has not gained much attention. Therefore, in this research, we develop data mining tools to support knowledge discovery in this type of data.

1.2 Pattern set mining

In this thesis, we only consider descriptive data mining models, as we would like to gain insight in the regularities of the data, which can be used to describe a concerned concept, for example, a disease mechanism.

Most approaches to descriptive data mining can be classified as either *local models* or *global models*. Local models, also called *patterns*, produce descriptions for only a particular region in the data. In contrast to local models, global models result in succinct summarisation of the entire data.

One well-known instance of pattern mining is *frequent itemset mining*, which was first introduced by Agrawal et al. (1993) [3]. In their work, they studied transaction databases, in which each transaction consists of items purchased by a customer. The pattern in their study is a set of items, also called an *itemset*, which are frequently bought together. Later, Mannila and Toivonen (1997) [79] developed a framework that generalises the frequent itemset mining problem for the discovery of such local models. The framework is formally stated as follows. Given a language $\mathcal{L}$, the task is to identify all patterns in this language that fulfil a given constraint Φ on a database $\mathcal{D}$:

$$\{p \in \mathcal{L} \mid \Phi(p, \mathcal{D})\}$$

One of the biggest problems of local patterns such as frequent itemsets is the exponential number of such patterns, which makes it hard to interpret them. Hence, another approach that has been developed to overcome this challenge is to find a small set of local patterns, which together can summarise the data succinctly. In the data mining community, this line of research is often called *pattern set mining* [86]. Formally, the task is to identify a subset $\mathcal{S}$ out of all patterns in the pattern language $\mathcal{L}$, such that every pattern $X \in \mathcal{S}$ fulfils the given constraints Φ on a database $\mathcal{D}$, while $\mathcal{S}$ is optimal with respect to a scoring function Ψ on sets of patterns [86]:

$$\underset{\mathcal{S} \subseteq \mathcal{F}}{\operatorname{argmax}} \Psi(\mathcal{S}, \mathcal{D})$$

with

$$\mathcal{F} = \{p \in \mathcal{L} \mid \Phi(p, \mathcal{D})\}$$

In this thesis, we will study pattern set mining for rank data as it will avoid exponential number of patterns. Moreover, we will see that such rank pattern sets are generally interesting and useful.

1.3 Bioinformatics

Most of the work in this thesis originates from the need to model a bioinformatics problem called *cancer subtyping*. In this problem, we would like to discover biologically meaningful groups of tumour samples called *cancer subtypes* as well as the molecular mechanisms that drive their cancer development. To see why this problem is worthy of investigation and how the rank pattern set mining framework can help, we will first discuss the context of the problem.

DNA, which stands for deoxyribonucleic acid, encodes programs/instructions that tell cells what to do. When those programs contain mistakes, their outputs, including mRNA, miRNA, proteins and many others, can malfunction. Recent advances in genetics technology can detect the changes of the programs at a high resolution (at the nucleotide level) and collect many of the outputs which correspond to the changed programs. By analysing such data, it is conjectured that we can find changed programs and malfunctioning outputs that might cause diseases. Indeed, if we could successfully analyse this data, we would not only have a better understanding of the mechanisms of the diseases but also a more precise way of prognosis and treatment for patients.

In practice, there have been reports maintaining that molecular information provides a better prognosis and treatment than conventional methods. For example, Emily Sohn [106] reported a case study that a patient had not known her disease and how to treat her illness for 40 years until she had her genome sequenced. The result of the sequencing revealed that she had a mutated gene, from which doctors knew how to treat her effectively.

In cancer research, projects such as TCGA [116], ICGC [51] and the 100K Genome Project [33] have been collecting many different molecular data types, ranging from genomic variants in the genome to molecular phenotypes downstream such as mRNA and other proteins, to realise the data analysis idea discussed above. At the time of writing this thesis, the TCGA project has sequenced the genome for more than 10000 tumour samples with 33 types of cancer [70]; the ICGC project has done the same job for more than 25000 samples [51]. In general, for each cancer type, we have a few hundreds of samples for which we have multiple types of information. From a computer science perspective, we have multiple matrices, each of which has the same set of columns. Each matrix can have a different data type depending on the corresponding intrinsic molecular data and/or measurement techniques. Analysing such heterogeneous data to discover the altered molecules driving the tumorigenesis is challenging. This is mainly because:

1. Each cancer type can have many subgroups/subtypes with different

molecular mechanisms and hence different ways of treatment. For example, in breast cancer, patients are typically divided into four subtypes, called Basal, Her2, LumA and LumB, each of which has a different way of treatment [119, 12].

2. There are *passenger genes*, which do not cause diseases when they are changed, and there are *driver genes*, which cause diseases when they are changed [126].

3. The data consists of many matrices at different scales due to the characteristics of the molecular data and/or the measurement techniques. For example, data related to genomic variances is typically Boolean while data related to other molecular phenotypes such as mRNA, miRNA and protein are usually numeric.

4. The data is high-dimensional: the number of features is usually much larger than the number of samples. If we only consider protein-coding genes, the number of features is approximately 20000 genes according to the Human Genome Project [47].

The first two challenges obviously show that discovering meaningful cancer subtypes and its underlying molecular mechanisms is an important problem. It is key for precise prognosis and treatment. Furthermore, in this thesis, we maintain that the two challenges should be ideally solved simultaneously. That is, discovering driver genes driving tumorigenesis and subtyping tumour samples are confounded problems. How subtypes are defined depends on the driver genes used to group samples in subtypes. Conversely, the subtypes define which driver genes are relevant for a certain sample grouping.

With regard to the third challenge, our discovery is that rankings are a good abstraction to remove the scale differences of the data. Finally, our rank pattern set framework helps us to overcome the last challenge by focusing on important features in a global way.

1.4 Contributions

As rank data is ubiquitous in many applications, discovering regularities hidden in this type of data is of interest. *The main question is whether we can discover interesting patterns in rank data?* This thesis offers a positive answer to this question. The main contributions of this thesis are the introduction of a framework for rank pattern set mining using principles of matrix factorisation

as well as the introduction of new types of rank patterns which can be used in real life applications.

In this thesis, we focus on the following three research questions:

- **Q1** How can we model rank pattern set mining?
- **Q2** Which types of rank patterns are potentially interesting and useful?
- **Q3** How useful are rank patterns in real life applications?

The main contributions and findings with respect to **Q1**, *How can we model rank pattern set mining*, are the following:

- We contribute a matrix factorisation-based framework to mine rank pattern sets. As we will show, the framework can be used to model different types of rank patterns.
- The key insight of using matrix factorisation for rank pattern set mining is the fact that the linear algebra traditionally used in matrix products is not always appropriate for aggregating rankings. We therefore contribute the idea of using semiring theory to generalise the matrix product and propose to use other semirings, for example, the max-product semiring, to handle rank data.

With regard to **Q2**, *which types of rank patterns are potentially interesting and useful*, our main contributions are the introduction of two new data mining problems, each of which corresponds to one new rank pattern:

- We introduce the *Sparse Rank Matrix Factorisation* (Sparse RMF) problem, which studies how to discover a small set of sparse rank vectors that can be used to succinctly summarise a given rank matrix.
- We introduce the *ranked tiling* problem, which studies how to discover a small set of data rectangles in a rank matrix that have high ranks and together cover the matrix as much as possible. The ranked tiling pattern shows local associations among subsets of the columns and subsets of the rows of the matrix.
- We demonstrate how to study the two different data mining problems using the single framework that we propose. We also contribute algorithmic ideas of how to solve the two problems in parallel.

With respect to **Q3**, *how useful are rank patterns in real life applications*, our contributions are the following:

- We use rank-based transformation to integrate three main different data types, including Boolean mutation data, numeric gene expression and prior knowledge encoded as biological networks, for cancer subtyping. To the best knowledge of the author, at the time of writing this thesis, we were the first to integrate these data types to study the cancer subtyping problem.
- We contribute a definition for cancer subtype, i.e., a group of tumour samples having the same unique combination of driver pathways.
- We introduce a rank pattern set mining problem in a multi-view setting using the ranked tiling pattern to formally define the concept of driver pathways.

1.5 Structure of the thesis

Chapter 2 presents the general framework for rank pattern set mining based on matrix factorisation. First, we will discuss our findings that the linear algebra traditionally used in matrix products is not always appropriate for aggregating rankings. Then, we show to how to generalise the matrix product using the semiring theory to handle rank data. We discuss related work, which makes clear there is a need for new methods.

This chapter is based on the following paper:

> Le Van, T., Nijssen, S., van Leeuwen, M., and De Raedt, L. Semiring rank matrix factorisation. *IEEE Transactions on Knowledge and Data Engineering, Under revision.*

Chapter 3 introduces the Sparse RMF problem. This problem aims at discovering k sparse rank vectors, i.e., having many 0s, which can be used to succinctly summarise a given rank matrix. We will study how to formalise the Sparse RMF problem using two different semirings, i.e., the plus-product semiring and the max-product semiring, to see the effect of the semiring on the modelling.

This chapter is based on the following two papers:

> Le Van, T., Nijssen, S., van Leeuwen, M., and De Raedt, L. Semiring rank matrix factorisation. *IEEE Transactions on Knowledge and Data Engineering, Under revision.*

> Le Van, T., van Leeuwen, M., Nijssen, S., and De Raedt, L. Rank Matrix Factorisation. In *Proc. of the 19th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-15)*, pp. 734–746. DOI: 10.1007/978-3-319-18038-0_57.

Chapter 4 introduces the ranked tiling problem. It essentially aims at discovering a set of data rectangles in a data matrix having high ranks with respect to a user-defined threshold. We will first study how to discover one maximal data rectangle that has high ranks. Then, we will study how to find a set of such maximal data rectangles. In each case, we will show two different approaches to modelling and solving. One approach is based on Constraint Programming and the other is based on the rank matrix factorisation framework.

This chapter is based on the following two papers:

> Le Van, T., Nijssen, S., van Leeuwen, M., and De Raedt, L. Semiring rank matrix factorisation. *IEEE Transactions on Knowledge and Data Engineering, Under revision.*
>
> Le Van, T., van Leeuwen, M., Nijssen, S., Fierro, A. C., Marchal, K., and De Raedt, L. Ranked tiling. In *Proc. of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD-14) (2)* (2014), pp. 98–113. DOI: 10.1007/978-3-662-44851-9_7.

Chapter 5 studies an application of the ranked tiling pattern and the rank matrix factorisation framework in the field of cancer research. We will study how to use molecular data, including mutation data, gene expression data, and biological networks (prior knowledge), to simultaneously cluster tumour samples into biologically meaningful groups called cancer subtypes and identify their specific features.

This chapter is based on the following paper:

- Le Van, T., van Leeuwen, M., Fierro, A. C., De Maeyer, D., Van den Eynden, J., Verbeke, L., De Raedt, L., Marchal, K., and Nijssen, S. Simultaneous discovery of cancer subtypes and subtype features by molecular data integration. *Bioinformatics 32* (2016), i445– i454. DOI: 10.1093/bioinformatics/btw434.

In the concluding chapter, we summarise the thesis and discuss opportunities for future work.

Chapter 2

Semiring Rank Matrix Factorisation

Rank data, in which each row is a complete or partial ranking of available items (columns), is ubiquitous. It can be used to represent, for instance, preferences of users [54, 26, 10, 69], the levels of gene expression [7, 69, 67], and the outcomes of sports events [92, 24, 60]. While rank data has been analysed in data mining, mining patterns in such data has so far not received much attention. To alleviate this state of affairs, in this chapter[1] we introduce a generic framework for rank pattern set mining. It is based on a semiring matrix factorisation framework. Rather than using the traditional linear algebra for matrix factorisation, we employ semiring theory. In the following chapters, we will show how to use the semiring rank matrix factorisation framework for a number of data mining tasks in different domains, including bioinformatics.

2.1 Introduction

We develop a generic framework for the unsupervised discovery of regularities (patterns) in *rank data*. In this type of data, each row (transaction) is a complete or a partial ranking of the available columns (items). While rank data is ubiquitous, only few data mining methods have been developed for rank data analysis. Exceptions include the work by Ben-Dor et al. [7], who proposed a

[1] Based on the journal paper "Semiring rank matrix factorisation" [66], which is in turn based on the conference paper [68].

probabilistic model to discover a fix-sized order-preserving rectangle; the work by Calders et al. [11] who studied rank correlation models to mine frequent patterns in the presence of rank data; the work by Ukkonen et al. [123] who studied the problem of discovering a set of items that are ranked either distinctively high or low in a cluster when compared to an aggregate representation of the entire data set; and the work by Henzgen et al. [46], who proposed an algorithm to enumerate frequent order-preserving items. Each of these works aimed at a single type of rank pattern and they did not aim at a general framework for different types of rank pattern set mining, i.e., a small, non-redundant, and interesting set of patterns globally describing the structure of the data [127].

Matrix factorisation has been an appealing data mining method for pattern set mining. It has been used in many fields such as data mining [105, 32], recommender systems [64] and bioinformatics [9]. Depending on the constraints on the data or the patterns users are interested in, one applies different forms of matrix factorisation. For example, if the given data has non-negative value constraints, non-negative matrix factorisation [71] can be employed; if the data has Boolean data constraints, Boolean matrix factorisation [82] can be used; if users are interested in sparse features, sparse dictionary learning [78] can be considered. Though matrix factorisation has been extensively studied, it can not be directly applicable to rank data due to the fact that the linear algebra used in the traditional matrix factorisation methods does not provide a way to aggregate/sum rankings over items (see Section 2.2).

Another class of methods that have been developed to find patterns in numerical data are biclustering methods [77], which are particularly popular in bioinformatics; however, biclustering algorithms for the rank data settings studied in this thesis do not currently exist either.

The rest of the chapter is organised as followed. In Section 2.2, we introduce a generic Semiring Rank Matrix Factorisation framework named *sRMF* for mining sets of patterns in rank data. In Section 2.3, we discuss at a high level how to use the sRMF framework to mine a set of rank patterns. Finally, we discuss related work in Section 2.4.

2.2 Semiring rank matrix factorisation (sRMF)

In this section, we first illustrate the rank pattern set mining problem. Next, we explain the reason why traditional matrix factorisation based on the linear algebra cannot be directly used for mining rank data. Then, we introduce the semiring rank matrix factorisation framework.

Definition 2.1 (Rank matrix)**.** *An $m \times n$ matrix $\boldsymbol{M}$ is a rank matrix iff $\boldsymbol{M}_{r,c} \in \sigma$, for all $1 \leq r \leq m$ and $1 \leq c \leq n$, where $\sigma = \{1, 2, ..., n\}$.*

In our setting, columns are items or products that need to be ranked; rows are rankings of items. Matrix cell $\mathbf{M}_{r,c}$ indicates that column c is ranked $\mathbf{M}_{r,c}$th for row r. In this matrix, multiple items may have the same rank. Such cases are named *ties*, which can be represented, for instance, by the minimum rank of the items.

In this thesis, we only consider rank matrices that consists of full rankings. Rank matrices containing *unknown* rankings are left for future work.

It is important to note that a rank matrix can be an abstraction of a numeric matrix. Obviously, there are two ways to abstract a given numeric matrix to obtain a rank matrix as defined in Definition 2.1. The first one is to perform a complete ranking of the columns per row. The second one is to first rank the rows per column and then transpose the resulting matrix. The question is when we should rank per row and when we should rank per column? We suggest that we should rank per row when the dataset consists of *incomparable rows*, i.e., values in each row are in different scales. Datasets with incomparable rows can occur when the rows are measured by different technologies. For example, we might have a molecular dataset consisting of mRNA and miRNA for the same set of samples, in which mRNA is measured by microarray and miRNA is measured by RNA-Seq. Conversely, we suggest to rank per column when the data contains incomparable columns. For instance, we might have a weather dataset, in which the columns are winds, temperature, humidity, and the rows are weather instances recorded at different time points or places. Obviously, in this case values in each column are incomparable. Hence, we should rank per column rather than per row. Finally, when the values of the entries in the matrix are comparable, we can perform the ranking procedure in either of the directions. In the case that we rank per row and the number of the columns is much smaller than the number of rows, we will have a higher chance to see a hidden data regularity, i.e., the data regularity appears in a large number of rows.

Many different types of patterns can exist in rank matrices. We will first discuss the intuitions behind two such pattern types.

Example 2.1 (Consistent Ranks)**.** *Consider the following rank matrix:*

$$\boldsymbol{M} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 3 & 5 & 6 & 4 & 1 \\ 2 & 3 & 5 & 6 & 1 & 4 \end{pmatrix}$$

In red and blue we indicated parts of the matrix in which the rank is consistent for a subset of rows and columns of the matrix: for instance, in the first two rows, the ranks of the items are identical. Red and blue here highlight patterns in the matrix.

Example 2.2 (High Ranks)**.** *Consider the following rank matrix:*

$$\boldsymbol{M} = \begin{pmatrix} 1 & 2 & 5 & 4 & 6 & 3 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 3 & 2 & 4 & 5 & 6 \\ 2 & 3 & 5 & 6 & 4 & 1 \\ 2 & 3 & 5 & 6 & 1 & 4 \end{pmatrix}$$

In red and blue we indicated subsets of columns and rows in which the highest ranks of the matrix occur. These subsets of rows and columns point towards patterns in the data; while the rank within these patterns may be consistent, this is not necessarily the case, as illustrated by the red pattern.

The aim of this thesis is to present a generic framework that is expressive and flexible enough to model and discover small sets of these different types of rank patterns. Our main observation is that finding such small sets of patterns can be formalised as a rank factorisation problem.

Definition 2.2 (Rank matrix factorisation (RMF))**.** *Given a rank matrix* $\boldsymbol{M} \in \sigma^{m \times n}$ *and an integer* k*, find a matrix* $\boldsymbol{C}^* \in \{0,1\}^{m \times k}$ *and a matrix* $\boldsymbol{F}^* \in \sigma_p^{k \times n}$ *such that:*

$$(\boldsymbol{C}^*, \boldsymbol{F}^*) \equiv \operatorname*{argmax}_{\boldsymbol{C},\boldsymbol{F}} \; f(\boldsymbol{M}, \boldsymbol{C} \odot \boldsymbol{F}), \tag{2.1}$$

where

- $f(,)$ *is a scoring function that measures the similarity between matrices;*
- $\odot$ *is an operator that creates a data matrix based on two factor matrices;*
- $\sigma_p \subseteq \sigma \cup \{0\}$ *is a set of permissible values. 0s denote unconcerned rankings;*
- *possibly additional constraints for specifying the patterns users are interested in.*

Intuitively, in matrix $\mathbf{F}$ the rows $\mathbf{F}_{i,:}$ indicate *partial rankings*, i.e., rank vectors that have 0*s*. Columns $\mathbf{C}_{:,i}$ of matrix $\mathbf{C}$ indicate in which rows the corresponding partial ranking appears. The following example illustrates this intuition for Example 2.1.

Example 2.3 (Rank matrix factorisation)**.** *The patterns for Example 2.1 can be represented as follows using two matrices* $\boldsymbol{C}$ *and* $\boldsymbol{F}$*:*

$$\boldsymbol{C} = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} \qquad \boldsymbol{F} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 3 & 5 & 6 & 0 & 0 \end{pmatrix}$$

This factorisation summarises matrix $\boldsymbol{M}$ *with two rank vectors: one is the full rank vector* $u = (1, 2, 3, 4, 5)$*, which appears in row 1 and row 2 of the matrix, and the other is the partial rank vector* $v = (2, 3, 5, 6, 0, 0)$*, which appears in the last two rows.*

A first important choice that needs to be made in this framework concerns the choice for the operator $\odot$. An obvious choice for this operator may be to use the traditional matrix product. However, this choice can cause problems.

Example 2.4 (Overlapping rank profiles)**.**

$$\begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 2 & 5 & 6 & 0 & 0 \\ 0 & 0 & 4 & 6 & 1 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 5 & 6 & 0 & 0 \\ 1 & 2 & 9 & 12 & 1 & 2 \\ 1 & 2 & 9 & 12 & 1 & 2 \\ 0 & 0 & 4 & 6 & 1 & 2 \\ 0 & 0 & 4 & 6 & 1 & 2 \end{pmatrix}$$

The factorisation in this example says that the two partial rank profiles are both present in row 2 & 3. Using the normal matrix product, the combined rankings for both row 2 and 3 become $v = (1, 2, 9, 12, 1, 2)$*. This is an invalid rank vector as it violates the definition of a rank matrix (Definition 2.1), which requires values in each row to belong to* σ*.*

For this reason, we require a different choice for the $\odot$ operator. In this thesis, we will consider operators that are based on *semirings* [39] to ensure that the output of a matrix product remains within the range of valid ranks.

Definition 2.3 (Semiring)**.** *A semiring* $(\sigma, \oplus, \otimes)$ *is a set* σ *equipped with two binary operations* $\oplus$ *and* $\otimes$ *satisfying the following properties:*

- $\oplus$ *is commutative:* $a \oplus b = b \oplus a$*;*
- $\otimes$ *and* $\oplus$ *are associative:* $a \otimes (b \otimes c) = (a \otimes b) \otimes c$*,* $a \oplus (b \oplus c) = (a \oplus b) \oplus c$*;*
- σ *has identity elements for* $\oplus$ *and* $\otimes$*, indicated with* 0 *and* 1*, such that* $a \otimes 1 = a$*,* $1 \otimes a = a$*, and* $a \oplus 0 = a$*;*

- $\otimes$ *left and right distributes over* $\oplus$: $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$, $(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)$;
- *the 0 element annihilates for all elements in* σ: $0 \otimes a = a \otimes 0 = 0$.

Semirings can be used to combine two matrices by generalising the matrix product.

Definition 2.4 (Matrix product based on semirings)**.** *The matrix product for two matrices* $\boldsymbol{C}$ *and* $\boldsymbol{F}$ *based on a semiring* $(\sigma, \oplus, \otimes)$ *is defined as follows:*

$$(\boldsymbol{C} \odot \boldsymbol{F})_{r,c} = \oplus_i (\boldsymbol{C}_{r,i} \otimes \boldsymbol{F}_{i,c}).$$

The traditional matrix product is the matrix product based on the plus-product semiring $(\mathbb{R}, +, \times)$ semiring, which is formally defined in the following definition.

Definition 2.5 (Plus-product semiring)**.** *The plus-product semiring is the semiring* $(\sigma, \oplus, \otimes)$ *in which* $a \oplus b = a + b$ *and* $a \otimes b = a \times b$.

As shown earlier, the plus-product semiring can cause problems as in the resulting matrix the resulting values may not be valid.

Hence, we require a different semiring. In this thesis, we will mainly focus on the *max-product* semiring, even though other choices are also possible, such as the *min-product* semiring.

Definition 2.6 (Max-product semiring)**.** *The max-product semiring is the semiring* $(\sigma, \oplus, \otimes)$ *in which* $a \oplus b = \max(a, b)$ *and* $a \otimes b = a \times b$.

Example 2.5 (Max-product semiring)**.**

$$\begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} \odot \begin{pmatrix} 1 & 2 & 5 & 6 & 0 & 0 \\ 0 & 0 & 4 & 6 & 1 & 2 \end{pmatrix} \equiv \begin{pmatrix} 1 & 2 & 5 & 6 & 0 & 0 \\ 1 & 2 & 5 & 6 & 1 & 2 \\ 1 & 2 & 5 & 6 & 1 & 2 \\ 0 & 0 & 4 & 6 & 1 & 2 \\ 0 & 0 & 4 & 6 & 1 & 2 \end{pmatrix}$$

Using the max-product semiring, we can combine the two factorised matrices in Example 2.4 into a single matrix; the two partial rank profiles are aggregated for user 2 and 3 by taking the maximum (green values).

Note that the max-product semiring chooses the highest rank in case two ranks overlap. Hence, the matrix product based on the max-product semiring

is best interpreted as an *optimistic rank aggregation*. That is, when a user (row) has multiple ways of ranking items (multiple rank profiles), the consensus way of ranking the items of that user is represented by an aggregated rank vector, which is obtained by taking the maximum rank values for each item. For example, in Example 2.5, User 2 has two ways of ranking the items: $f_1 = (1, 2, 5, 6, 0, 0)$ and $f_2 = (0, 0, 4, 6, 1, 2)$. The aggregated rank vector for User 2 is $u = (max(f_{11}, f_{21}), ..., max(f_{16}, f_{26}) = (1, 2, 5, 6, 1, 2)$. With a min-product semiring, the lowest value would be chosen, and hence the matrix product based on that semiring can be interpreted as a *pessimistic rank aggregation*.

Definition 2.7 (Semiring rank matrix factorisation - sRMF)**.** *Given a rank matrix $\boldsymbol{M} \in \sigma^{m \times n}$ and an integer k, find a matrix $\boldsymbol{C}^* \in \{0, 1\}^{m \times k}$ and a matrix $\boldsymbol{F}^* \in \sigma_p^{k \times n}$ such that:*

$$(\boldsymbol{C}^*, \boldsymbol{F}^*) \equiv \underset{\boldsymbol{C}, \boldsymbol{F}}{\operatorname{argmax}} \; f(\boldsymbol{M}, \boldsymbol{C} \odot \boldsymbol{F}), \tag{2.2}$$

where

- $f(,)$ *is a scoring function that measures the similarity between matrices;*
- $\odot$ *is a matrix product based on semirings. The default semiring for sRMF in this thesis is the max-product semiring unless explicitly stated;*
- $\sigma_p \subseteq \sigma \cup \{0\}$ *is a set of permissible values. 0s denote unconcerned rankings;*
- *possibly additional constraints for specifying the patterns users are interested in.*

Another important choice in the semiring rank matrix framework is the choice of the scoring function f. In this article we will limit our attention to *additive* scoring functions.

Definition 2.8 (Additive scoring function)**.** *Given two matrices $\boldsymbol{M}$ and $\boldsymbol{R}$, a scoring function $f(\boldsymbol{M}, \boldsymbol{R})$ is additive if we can write the scoring function as follows:*

$$f(\boldsymbol{M}, \boldsymbol{R}) = \sum_{r=1}^{m} \sum_{c=1}^{n} \delta(\boldsymbol{M}_{r,c}, \boldsymbol{R}_{r,c}),$$

where $\delta : \sigma \times \sigma \to \mathbb{R}$ scores the difference between values $\boldsymbol{M}_{r,c}$ and $\boldsymbol{R}_{r,c}$.

The main arguments in favour of these choices are that they are conceptually easy and that they enable more efficient algorithms.

2.3 Rank pattern set mining using sRMF

The sRMF framework provides a principled way to model and discover small sets of different types of rank patterns. To use the framework to mine a particular rank pattern set users are interested in, we follow the following steps:

- Choose a suitable semiring for the matrix product of the factorisation. Note that the proposed semiring for rank matrix factorisation in this thesis is in most cases the max-product semiring;
- Define the permissible set values for $\sigma_p \subseteq \sigma \cup \{0\}$ to define the type of the pattern users want to discover;
- Design the scoring function in Equation (2.2) to measure the quality of the factorisation.

In the next chapters, we will demonstrate how to use the sRMF framework to mine three different types of rank patterns, namely Sparse RMF, ranked tilings and ranked tilings in the multi-view setting [111, 134, 93, 130, 84].

Sparse RMF aims at discovering a set of partial rankings, i.e., rank vectors with many zeros, which repeatedly occur in the data. Such set of rankings can be interpreted as local patterns. In addition, they can be used to succinctly summarise the given rank matrix and show main categories of rankings. An example of this type of pattern is shown in Example 2.1 on page 11.

Ranked tiling aims at discovering a set of regions called ranked tiles that have high ranks with respect to a user-defined threshold. Ranked tiling is interesting as they show local associations among subsets of the rows and subsets of the columns of a given matrix. Ranked tiling is also an important concept for the cancer subtyping problem that we study in this thesis. An example of ranked tiling is shown in Example 2.2 on page 12.

Ranked tiling in the multi-view setting. The last type of rank pattern we study in this thesis is a set of ranked tiles that are related across multiple matrices. Figure 2.1 shows an example of this setting. This problem stems from an application in bioinformatics, in which we want to discover cancer subtypes and subtype features from molecular data (Chapter 5).

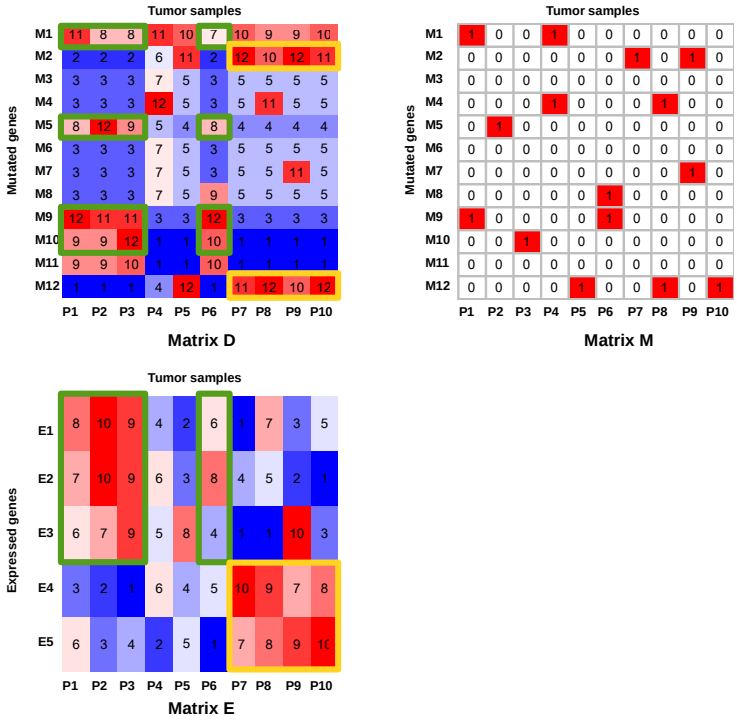


Figure 2.1: Example of ranked tiling across multiple matrices. In this case, matrix $\mathbf{D}, \mathbf{M}$ and $\mathbf{E}$ contain molecular information of the same set of patients $\{P_1, \ldots, P_{10}\}$. In addition, matrix $\mathbf{D}$ and matrix $\mathbf{M}$ have the same set of features $\{M_1, \ldots, M_{12}\}$, which can be different from those of matrix $\mathbf{E}$. Each column of matrix $\mathbf{D}$ is the rankings of the importance of the genes to a disease w.r.t. some scoring function. Matrix $\mathbf{M}$ contains Boolean mutation information of the considered genes. Each row of matrix $\mathbf{E}$ is the rankings of gene expression of the patients. Given this data, we can discover two groups of patients, called cancer subtypes, and their specific features (genes). The two subtypes are painted in green and repsectively in yellow. Note that each subtype consists of two ranked tiles, one in matrix $\mathbf{D}$ and the other in matrix $\mathbf{E}$. The two tiles are coupled by the same subset of the columns and are required to have high ranks. In addition, each of the corresponding selected rows in matrix $\mathbf{M}$ should have value 1 at least once in the selected columns of the tiles.

2.4 Related work

This section discusses prior work related to the research problems presented in this thesis. The discussion in this section is more general, and some of the more specific related works are discussed later in appropriate places.

2.4.1 Rank pattern mining

Ben-Door et al. [7] introduced the *order-preserving submatrices* (OPSMs) problem, which aimed at discovering a data rectangle having a consistent ordering of a subset of the columns. As this problem concerns the order of the items, two rank vectors $u = (1, 2, 3)$ and $v = (4, 5, 6)$ are the same. Indeed, this problem is related to the Sparse RMF problem that we introduce in this thesis. To illustrate the difference between the two problems, consider the following rank matrix:

$$M = \begin{pmatrix} 1 & 2 & 3 & 5 & 4 \\ 3 & 4 & 5 & 1 & 2 \\ 1 & 2 & 3 & 4 & 5 \\ 3 & 1 & 2 & 5 & 4 \end{pmatrix}$$

OPSM returns a data rectangle indicated by the rows $R = \{1, 2, 3\}$ and the columns $C = \{1, 2, 3\}$, while Sparse RMF returns a smaller rectangle, which is indicated by the rows $R^{'} = \{1, 3\}$ and the columns $C^{'} = \{1, 2, 3\}$. In other words, in Sparse RMF, the absolute rank values matter, while in OPSM the order of the rank values is considered.

Henzgen et al. [46] studied the frequent subranking problem, which they introduced to discover subrankings of a subset of items supported by a minimum number of transactions (rows). There are two fundamental differences between their work and our proposed patterns. First, the pattern they studied is actually similar to the one in the OPSM problem [7] that we discuss above. That is the order of the items is important rather than the rank values as we study in the Sparse RMF problem. Second, their work is in the direction of frequent pattern mining [2] while our work is a type of pattern set mining [86]. Frequent pattern mining methods often result in exponential number of patterns, which make it hard to interprete, while pattern set mining methods result in a small set of patterns.

Calders et al. [11] studied the problem of discovering rank-correlated itemsets, which tend to have consistent ranks. Indeed, the pattern they studied is similar

to Sparse RMF. However, their scoring functions were derived from statistics [59] and they follow the frequent pattern mining method [2]. In Sparse RMF, we propose a different scoring function (see Chapter 3) and we follow the direction of pattern set mining [86].

Ukkonen et al. [123] studied the problem of discovering *outlying items*, i.e., items that are ranked either distinctively high or low in a cluster when compared to an aggregate representation of the entire data set. This problem is related to the ranked tiling problem that we introduce in this thesis. However, we have different ways of measuring how high the ranks should be: their work uses a randomisation algorithm to determine how surprising a pattern is; our work uses a scoring function based on an optimisation problem. The other difference is the algorithm proposed by Ukkonen et al. [123] could not find overlapping patterns in term of rows while our method can. In general, Ukkonen et al. [123] did not aim to do tiling rank matrices, i.e., discovering local associations between subsets of rows and subsets of columns, which is quite different from our work. Finally, they can handle partial rankings while we have not considered this type of data in our studies.

2.4.2 Semiring-based matrix factorisation

Our rank matrix factorisation framework uses semirings [39] to define the matrix product to factorise rank matrices. Karaev et al. [57, 56] also used this idea to factorise non-negative real-valued matrices. However, Karaev et al. [57, 56] aimed at a factorisation of which the reconstructed matrix is as close to the original matrix as possible. In other words, given a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ and positive integer k, they try to find two matrices $\mathbf{B} \in \mathbb{R}^{m \times k}$ and $\mathbf{C} \in \mathbb{R}^{k \times n}$ such that the following function is minimised:

$$\sum_{i,j} |\mathbf{A}_{i,j} - (\mathbf{B} \odot_m \mathbf{C})_{i,j}|^p, \tag{2.3}$$

where $\odot_m$ is the matrix product based on the max-product semiring[2] and p can be either 1 [57] or 2 [56].

There are two main differences between our sRMF framework and the works by Karaev et al. [57, 56]. First, our sRMF framework is open for data miners to define the scoring function and to constrain the values of the factorised matrices. In other words, sRMF provides a principled way, i.e., through the algebra and the format of the factorisation, to discover different types of pattern while their

[2]Karaev et al. [57, 56] used different terminologies, namely *max-times algebra* and *subtropical algebra*, to mention the max-product semiring used in this thesis.

works focus on a specific one. Second, our framework typically does not aim at such a factorisation that the reconstructed matrix resembles the original matrix as long as the factorisation can capture the structure of the data. In contrast, their works do. Consequently, our results are typically *sparse*, i.e., the factorised matrices contain many 0s, and can be interpreted as *local patterns* while theirs are usually *dense*, i.e., many non-zeros in the factorised matrices, and lack such an interpretation.

In the literature, there has been work on using other semirings for matrix factorisation. De Schutter et al. [99] studied the Max-plus semiring $(\mathbb{R}, \oplus, \otimes)$, in which $a \oplus b = max(a, b)$ and $a \otimes b = a + b$, for Singular-Value Decomposition. Their work applied to numeric matrices, which are quite different from the discrete setting that we consider for rank matrices in this thesis. More importantly, the Max-plus semiring is not applicable to the semantics of the rank matrix factorisation framework that we propose. In our work, we factorise a given rank matrix $\mathbf{M}$ into a product of two smaller matrices, i.e., $\mathbf{M} \sim \mathbf{C} \odot \mathbf{F}$, where $\mathbf{C}$ is a Boolean matrix and $\mathbf{F}$ is typically a sparse rank matrix. The $0s$ and $1s$ in a particular row (user) of matrix $\mathbf{C}$ indicate which rank profiles in matrix $\mathbf{F}$ can be applied to that row. This can be only obtained using multiplication, i.e., $a \otimes b = ab$. If we use the Max-plus semiring, in which $a \otimes b = a + b$, it will not have the effect of selecting rank profiles as we want.

2.4.3 Boolean matrix factorisation

Boolean matrix factorisation (BMF) studies how to decompose Boolean matrices. Many methods have been proposed to approximate Boolean matrices, to name just a few, exact methods described in Kim's textbook [61], probabilistic methods such as topic models [100] and logistic PCA [98]. Probably, the work by Miettinen et al. [82] is the one that is most relevant to the work done in this thesis. It works as follows. Given a m-by-n Boolean matrix $\mathbf{C}$ and an integer k, find an m-by-k Boolean matrix $\mathbf{B}$ and a k-by-n Boolean matrix $\mathbf{X}$ that minimises:

$$\sum_{i=1}^{m} \sum_{i=1}^{n} |\mathbf{C}_{i,j} - (\mathbf{B} \circ \mathbf{X})_{i,j}|, \tag{2.4}$$

where $\circ$ is the Boolean matrix product.

Note that the Boolean matrix product in Equation (2.4) is the matrix product based on the semiring $(\{0, 1\}, \bigvee, \bigwedge)$. Therefore, it is a special case of our sRMF framework, where we set $\sigma_p = \{0, 1\}$ and the matrix product is based on the

max-product semiring. In addition, Miettinen et al. [82] showed that BMF is NP-complete.

2.4.4 Real-valued matrix factorisation

Matrix factorisation methods are popular for numerical data analysis [41] and data mining [105, 32, 17, 94]. Some widely-used matrix factorisation methods for descriptive data mining include the Singular Value Decomposition (SVD) [35], the Non-Negative Matrix Factorisation (NMF) [88, 71].

Singular Value Decomposition (SVD). The SVD of a matrix $\mathbf{A}$ with m rows and n columns is

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \tag{2.5}$$

with the following properties [94]:

1. $\mathbf{U}$ is a $m \times r$ *column-orthonormal matrix*; that is, each of its columns is a unit vector and the dot product of any two columns is 0.
2. $\mathbf{V}$ is an $n \times r$ column-orthonormal matrix.
3. $\mathbf{\Sigma}$ is a diagonal matrix; that is, all elements not on the main diagonal are 0. The elements of $\mathbf{\Sigma}$ are called the *singular values* of $\mathbf{A}$.

SVD is useful when there are a small number of concepts that connect the rows and columns of the original matrix [94]. Typically, $\mathbf{V}$ represents concepts, $\mathbf{\Sigma}$ represents the strength of the concepts and $\mathbf{U}$ connects rows to the concepts. SVD has been used for, e.g., network analysis [30], document analysis [23] and gene expression studies [129].

Non-Negative Matrix Factorisation (NMF). The NMF of a non-negative matrix $\mathbf{A}$ with m rows and n columns is

$$\mathbf{A} = \mathbf{U}\mathbf{V} \tag{2.6}$$

with additional constraints that both $\mathbf{U}$ and $\mathbf{V}$ contain non-negative values. Typically, $\mathbf{V}$ represents prominent profiles of the columns and $\mathbf{U}$ represents weights for the linear combination the profiles. NMF is interesting because the non-negativity constraints make the representation purely additive [50], which often results in sparse representation. In addition, it is shown that NMF is equivalent to k-means clustering [27], Probabilistic Latent Semantic Analysis [36, 28]. In practice, NMF is used in a large number of applications: text mining [135], community detection [131], gene expression analysis [9], multi-omics data integration [132, 128, 137], to name just a few.

In general, the matrix factorisation methods discussed above and many others such the Principal Component Analysis (PCA) [90] use the plus-product semiring to calculate the product of the factorisation. However, that semiring is not suitable for the rank data studied in this thesis (see Section 2.2).

2.4.5 Bi-clustering and tiling

Bi-clustering methods [77] aims at clustering simultaneously both dimensions of a matrix. A bi-cluster is a data rectangle identified by a subset of the column indexes and a subset of the row indexes. In addition, the data in the bi-cluster should possess some desired data regularity, for example, constant value [77]. Hence, bi-clustering is related to our work as sRMF factorises rank matrices to identify (sparse) rank profiles occuring in a number of rows, of which goal is also to discover local associations between subsets of the columns and subsets of the rows. However, up to best knowledge of the author, bi-clustering methods for the rank data settings studied in this thesis do not currently exist.

Tiling transaction databases (Boolean matrices) [37] is another relevant research theme. A tile is a data rectangle in the matrix full of 1s and a tiling is a small set of such tiles such that it covers the data matrix as much as possible. By constraining $\sigma_p = \{0, 1\}$, tiling can become an instantiation of sRMF.

To the best knowledge of the author, the bi-clustering problem was first introduced by Cheng and Church (2000) for analysing numerical gene expression values [15]. Since then, much research has been developed for both numeric data and Boolean data with different approaches, to name just a few, constraint programming [65], pattern mining [101, 45], information theory [25, 6, 63], probabilistic graphical models [38], parametric Bayesian inference [104, 42, 102], non-parametric Bayesian inference [58, 136, 133].

2.4.6 Rank data analysis

Rank aggregation [73] studies the problem of finding a single rank vector that has the least discrepancy with all of the rankings over the items provided by the users in a database. This problem appears in domains such as aggregating user preferences [95, 53] and combining search results [31]. The main difference to sRMF is that different types of rank patterns can be discovered while rank aggregation only aims at finding one type of pattern, which is a (dense) representation that has the least discrepancy w.r.t. the given rank data.

In the rank aggregation setting, it is assumed that all users are identical or there is only one group of users. However, there has been research looking at

the setting where there could be multiple homogeneous groups of users hidden in the data. A number of methods have been proposed to discover such groups together with their consistent rank profiles. A non-exhaustive list of work in this direction include [85, 10, 5, 29, 16, 1]. Similar to the rank aggregation setting discussed above, such methods look for complete rankings of all items, which are typically dense. In constrast, our sRMF framework aims to find multiple types of rank patterns, which are typically sparse. Besides that, these methods only aim at one type of rank pattern while our work aims at a framework for mining different rank patterns.

2.5 Conclusions

Rank data is ubiquitous and useful. To discover regularities hidden in this type of data, new data mining methods are needed.

We developed a generic semiring rank matrix factorisation framework for mining sets of patterns. We proposed to use a max-product semiring defined on permissible rank values of the data to calculate the matrix product of the two factorised matrices. To mine a specific type of data regularity, we proposed to use the two factorised matrices to define the patterns of interest by constraining the values of these matrices as well as an appropriate scoring function to measure the quality of the factorisation. In the next chapters, we will demonstrate how to apply the proposed framework on real world problems in different domains, including bioinformatics.

Chapter 3

Sparse RMF

This chapter[1] presents the first instantiation of the sRMF framework to mine rank pattern sets in rank data. We study the *Sparse RMF* problem, which we introduced to find a set of (partial) rankings repeatedly occurring in the data. Such sets of partial rankings can be used to succinctly summarise the given rank matrix and show the main categories of the rankings.

We present two approaches to modelling Sparse RMF. The first approach uses the plus-product semiring, i.e., the traditional linear algebra for matrix factorisation, and the second approach uses the max-product semiring. The two approaches not only show the chronological development of the research, i.e., the former was first introduced in [68] and the latter was developed after that in [66], but also illustrate the effects of the semiring to the modelling. Experimental results confirm that the second approach leads to improved solution qualities but needs more time to compute.

3.1 Introduction

Given a rank matrix, the aim of Sparse RMF is to discover a small set of rankings that repeatedly occur in the data. To be tolerant with noise, the rankings are not required to be complete; they can be a partial ranking, i.e., a rank vector with many zeros. Discovering such sets of rankings is essential as

[1] Based on the journal paper "Semiring rank matrix factorisation" [66] and the conference paper [68].

they can be used to succinctly summarise the given rank matrix and show the main categories of the rankings.

Example 3.1 (Sparse RMF example)**.**

$$\boldsymbol{M} = \begin{pmatrix} 1 & 2 & 5 & 4 & 6 & 3 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 3 & 2 & 4 & 5 & 6 \\ 2 & 3 & 5 & 6 & 4 & 1 \\ 2 & 3 & 5 & 6 & 1 & 4 \end{pmatrix} \boldsymbol{C} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \boldsymbol{F} = \begin{pmatrix} 1 & 2 & 0 & 4 & 0 & 0 \\ 1 & 0 & 0 & 4 & 5 & 6 \\ 2 & 3 & 5 & 6 & 0 & 0 \end{pmatrix}$$

To illustrate the Sparse RMF problem, let's consider the illustration shown in Example 3.1. It shows a rank matrix that is approximated by the product of two smaller matrices. Rank matrix $\mathbf{M}$ consists of five rows and six columns. Assuming no ties and complete rankings, each row contains each of the numbers 1 to 6 exactly once. Sparse RMF factorises matrix $\mathbf{M}$ into the product of a binary 5×3 matrix named $\mathbf{C}$ and a 3×6 rank matrix named $\mathbf{F}$. Each row of matrix $\mathbf{F}$ is a rank vector with many zeros called *a sparse rank profile* and can be interpreted as a *local pattern*. The sparse rank profiles help us to discover recurrent structures in the matrix and to not focus on any noise that may be present.

The rest of this chapter discusses two approaches to modelling and solving this problem as well as experiments on both synthetic and real data. In Section 3.2, we show the first approach, which uses the plus-product semiring, i.e., the traditional linear algebra, in combination with additional constraints. In Section 3.3, we present the second approach, which uses the max-product semiring for factorising rank matrices. Experiments on synthetic and real data are presented Section 3.7 and Section 3.8 respectively.

3.2 Sparse plus-product semiring rank matrix factorisation (Sparse pRMF)

In this section, we choose the traditional algebra, i.e., the plus-product semiring (Definition 2.5), to calculate the matrix product of the factorisation. This choice leads to a specific instantiation for Sparse RMF named *Sparse Plus-product Semiring Rank Matrix Factorisation* (Sparse pRMF).

Using the plus-product semiring, the reconstructed matrix is calculated by the following formula:

$$(\mathbf{C} \odot_p \mathbf{F})_{r,c} = \oplus_i (\mathbf{C}_{r,i} \otimes \mathbf{F}_{i,c}) \tag{3.1}$$

$$= \sum_i \mathbf{C}_{r,i} \times \mathbf{F}_{i,c} \tag{3.2}$$

where $\odot_p$ denotes the matrix product using the plus-product semiring.

An important drawback of the plus-product semiring is that the product $\mathbf{C} \odot_p \mathbf{F}$ computed by that semiring is not necessarily a rank matrix (see Example 3.2). To address this problem, we restrict the set of acceptable matrices to those that satisfy the following constraints:

$$(\mathbf{C} \odot_p \mathbf{F})_{r,c} \leq n, \forall r = 1, \ldots, m, \forall c = 1, \ldots, n. \tag{3.3}$$

where m, n are the number of the rows and the columns of the given rank matrix respectively.

To illustrate the effect of the selected semiring to the acceptable factorisations, let us examine the following two examples. Example 3.2 shows a factorisation which produces an invalid rank matrix due to the plus-product semiring used in the matrix product. Example 3.3 shows one possible way to fix this problem by adjusting the overlapping structure of the two rank profiles.

Example 3.2 (Invalid solution for Sparse pRMF)**.**

$$\begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} \odot_p \begin{pmatrix} 1 & 2 & 5 & 6 & 0 & 0 \\ 0 & 0 & 4 & 6 & 1 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 5 & 6 & 0 & 0 \\ 1 & 2 & 9 & 12 & 1 & 2 \\ 1 & 2 & 12 & 12 & 1 & 2 \\ 0 & 0 & 4 & 6 & 1 & 2 \\ 0 & 0 & 4 & 6 & 1 & 2 \end{pmatrix}$$

This factorisation is an invalid solution for Sparse pRMF as the reconstructed matrix contains ranks (green values) that are not in the permissible rank values $\sigma = \{0, 1, \ldots, 6\}$.

Example 3.3 (Valid solution for Sparse pRMF)**.**

$$\begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} \odot_p \begin{pmatrix} 1 & 2 & 5 & 6 & 0 & 0 \\ 0 & 0 & 4 & 6 & 1 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 5 & 6 & 0 & 0 \\ 1 & 2 & 5 & 6 & 0 & 0 \\ 1 & 2 & 5 & 6 & 0 & 0 \\ 0 & 0 & 4 & 6 & 1 & 2 \\ 0 & 0 & 4 & 6 & 1 & 2 \end{pmatrix}$$

We have defined a way to compute the reconstructed matrix by using the plus-product semiring. Next, we have to define the scoring function $f(,)$ in

Equation (2.2), which is the sum of additive scoring fuctions δ (Equation 2.8), to score the similarity between the reconstructed matrix and the original rank matrix. To support the aim of mining sparse rank profiles in rank matrices, the scoring function δ between elements needs to be designed in such a way that it: 1) rewards patterns that have a high coverage (ideally, the whole data would be covered), 2) penalises patterns that make a large error within the cover of the factorisation.

We achieve this by defining the scoring function δ between elements as follows:

$$\delta(a,b) = \begin{cases} 0 & \text{if } b = 0; \\ \alpha - |a-b| & \text{otherwise.} \end{cases} \tag{3.4}$$

If $b > 0$, for a given cell in the reconstructed matrix $\mathbf{R} = \mathbf{C} \odot \mathbf{F}$, we call this cell *covered*. The term α defines how much reward is given for covering a cell in the data; the larger α is, the larger the patterns will be. The reward is lowered by penalising for errors; for errors higher than α the term $\delta(a,b)$ will be negative. Hence, setting α low enough will ensure that we will not cover the complete data.

The error term $|a-b|$ is related to the *Footrule distance*, which is a well-known distance for comparing rankings.

Definition 3.1 (Footrule distance)**.** *Given two rank vectors,* $\boldsymbol{u} = (u_1, \ldots, u_n)$ *and* $\boldsymbol{v} = (v_1, \ldots, v_n)$*, the Footrule distance is defined as* $\sum_{i=1}^{n} |u_i - v_i|$.

The α parameter balances errors against coverage. Indeed, an alternative way of writing our scoring function is as follows:

$$f(\mathbf{M}, \mathbf{R}) = \alpha \cdot \mathit{coverage}(\mathbf{R}) - \mathit{error}(\mathbf{M}, \mathbf{R}),$$

where

- $\mathbf{R} = \mathbf{C} \odot \mathbf{F}$;
- $\mathit{coverage}(\mathbf{R}) = \sum_{\mathbf{R}_{r,c}>0} 1$;
- $\mathit{error}(\mathbf{M}, \mathbf{R}) = \sum_{\mathbf{R}_{r,c}>0} |\mathbf{M}_{r,c} - \mathbf{R}_{r,c}|$.

Many other scoring functions could also be used to measure the disagreement between rows, for instance, Kendall's tau; see [80, 4] for a survey. We choose the Footrule as it can be calculated relatively efficiently.

Note that we do not take into account the error for cells that are not covered; this reflects our interest in discovering *local patterns* that not necessarily characterise the complete data.

Plugging the scoring function in Equation (3.4), the constraints in Equation (3.3) and the plus-product semiring in the sRMF framework in Definition 2.7, we can summarise the problem instance of Sparse RMF using the plus-product semiring (Sparse pRMF) as follows.

Problem 3.1 (Sparse pRMF)**.** *Given a rank matrix $\boldsymbol{M} \in \sigma^{m \times n}$, Sparse pRMF is the semiring rank matrix factorisation problem obtained by using*

- *the plus-product semiring;*
- *the set of permissible values $\sigma_p = \sigma \cup \{0\}$;*
- *the additive scoring function based on formula (3.4);*
- *constraints that $(\boldsymbol{C} \odot_p \boldsymbol{F})_{ij} \leq n, \forall i = 1, \ldots, m, \forall j = 1, \ldots, n$.*

To illustrate the effects of using the plus-product semiring to mine Sparse rank profiles, we apply Sparse pRMF to the rank matrix created by combining the two overlapping rank profiles in Example 2.5. The optimal result is shown in Example 3.4, which could not detect the overlapping structure in the data.

Example 3.4 (Sparse pRMF example)**.**

$$\boldsymbol{M} = \begin{pmatrix} 1 & 2 & 5 & 6 & 0 & 0 \\ 1 & 2 & 5 & 6 & 1 & 2 \\ 1 & 2 & 5 & 6 & 1 & 2 \\ 0 & 0 & 4 & 6 & 1 & 2 \\ 0 & 0 & 4 & 6 & 1 & 2 \end{pmatrix}$$

$$\simeq \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} \odot_p \begin{pmatrix} 1 & 2 & 5 & 6 & 0 & 0 \\ 0 & 0 & 4 & 6 & 1 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 5 & 6 & 0 & 0 \\ 1 & 2 & 5 & 6 & 0 & 0 \\ 1 & 2 & 5 & 6 & 0 & 0 \\ 0 & 0 & 4 & 6 & 1 & 2 \\ 0 & 0 & 4 & 6 & 1 & 2 \end{pmatrix}$$

Sparse pRMF approximates matrix $\boldsymbol{M}$ by the product of two smaller matrices with $k = 2$. If $\alpha = 1$, then $cover(\boldsymbol{C} \odot_p \boldsymbol{F}) = 20$, $error(\boldsymbol{M}, \boldsymbol{C} \odot_p \boldsymbol{F}) = 0$, and the optimal score $f(\boldsymbol{M}, \boldsymbol{C} \odot_p \boldsymbol{F}) = 20$. Note that the optimal solution by Sparse pRMF does not have overlapping rank profiles.

3.3 Sparse max-product semiring rank matrix factorisation (Sparse mRMF)

In this section, we choose the max-product semiring to calculate the matrix product of the factorisation. This choice leads to a specific instantiation for Sparse RMF named *Sparse Max-product Semiring Rank Matrix Factorisation* (Sparse mRMF).

Using the max-product semiring, the reconstructed matrix is calculated by the following formula:

$$(\mathbf{C} \odot_m \mathbf{F})_{r,c} = \oplus_i (\mathbf{C}_{r,i} \otimes \mathbf{F}_{i,c}) \tag{3.5}$$

$$= \max_i \{\mathbf{C}_{r,i} \times \mathbf{F}_{i,c}\} \tag{3.6}$$

where $\odot_m$ denotes the matrix product using the max-product semiring.

Therefore, the resulting matrix product using the max-product semiring always consists of permissible rank values. Consequently, Sparse mRMF does not need additional constraints specified by Equation (3.3) as Sparse pRMF does.

Plugging this semiring and the scoring function in Equation (3.4) in the sRMF framework in Definition (2.7), we can summarise the problem instance of Sparse RMF using the max-product semiring (Sparse mRMF) as follows.

Problem 3.2 (Sparse mRMF)**.** *Sparse mRMF is the rank matrix factorisation problem obtained by using*

- *the max-product semiring;*
- *the set of permissible values* $\sigma_p = \sigma \cup \{0\}$*;*
- *the additive scoring function based on formula (3.4).*

To demonstrate the advantages of using the max-product semiring for rank matrix factorisation, we apply Sparse mRMF to the same rank matrix shown in Example 3.2. The result presented in Example 3.5 shows that in this case Sparse mRMF not only recovers the overlapping structure, but also obtains higher coverage than Sparse pRMF while having the same error.

Example 3.5 (Sparse mRMF example)**.**

$$\boldsymbol{M} = \begin{pmatrix} 1 & 2 & 5 & 6 & 0 & 0 \\ 1 & 2 & 5 & 6 & 1 & 2 \\ 1 & 2 & 5 & 6 & 1 & 2 \\ 0 & 0 & 4 & 6 & 1 & 2 \\ 0 & 0 & 4 & 6 & 1 & 2 \end{pmatrix}$$

$$\simeq \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} \odot_m \begin{pmatrix} 1 & 2 & 5 & 6 & 0 & 0 \\ 0 & 0 & 4 & 6 & 1 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 5 & 6 & 0 & 0 \\ 1 & 2 & 5 & 6 & 1 & 2 \\ 1 & 2 & 5 & 6 & 1 & 2 \\ 0 & 0 & 4 & 6 & 1 & 2 \\ 0 & 0 & 4 & 6 & 1 & 2 \end{pmatrix}$$

Sparse mRMF approximates the matrix $\boldsymbol{M}$ *shown in Example 3.4 where Sparse pRMF is applied with the same value for* k *and* α*. The optimal solution produced by Sparse mRMF has* $cover(\boldsymbol{C} \odot_m \boldsymbol{F}) = 24$*,* $error(\boldsymbol{M}, \boldsymbol{C} \odot_m \boldsymbol{F}) = 0$*, and the optimal score* $f(\boldsymbol{M}, \boldsymbol{C} \odot_m \boldsymbol{F}) = 24$*. Note that Sparse mRMF could discover the overlapping structure of the data while Sparse pRMF could not.*

3.4 Sparse mRMF factorisation is not unique

The following example illustrates that Sparse mRMF may not have a unique factorisation.

Example 3.6 (Uniqueness of the Sparse mRMF factorisation)**.** *Given the following rank matrix* M*, the threshold* $\alpha = 1$ *and* $k = 2$*, Sparse mRMF can return two ways of factorising the matrix, each of which has the same coverage* $= 14$ *and error* $= 0$*.*

$$M = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 1 & 2 & 3 & 5 & 6 \\ 2 & 3 & 4 & 1 & 5 & 6 \end{pmatrix}$$

Factorisation 1:

$$C_1 = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} F_1 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 0 & 0 & 0 & 0 & 5 & 6 \end{pmatrix}$$

Factorisation 2:

$$C_2 = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} F_2 = \begin{pmatrix} 1 & 2 & 3 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 6 \end{pmatrix}$$

3.5 Solving Sparse pRMF using IP

The optimisation model for Sparse mRMF is presented in Problem 3.2. Now we re-write that optimisation model in such a form that we can easily derive an equivalent model, which can be put into generic Integer Programming (IP) solvers. The restatement also provides an easy comparison between Sparse pRMF and Sparse mRMF.

Theorem 3.1 (Optimisation model for Sparse pRMF)**.** *Given a rank matrix $\boldsymbol{M}$, an integer k and a threshold α, Sparse pRMF is equivalent to the following optimisation problem.*

$$(\boldsymbol{C}^*, \boldsymbol{F}^*) \equiv \operatorname*{argmax}_{\boldsymbol{C},\boldsymbol{F}} \sum_{r,c} (\alpha \boldsymbol{A}_{r,c} - |\boldsymbol{A}_{r,c}\boldsymbol{M}_{r,c} - \sum_i \boldsymbol{C}_{r,i}\boldsymbol{F}_{i,c}|) \tag{3.7}$$

subject to

$$\boldsymbol{C} \in \{0,1\}^{m \times k} \tag{3.8}$$

$$\boldsymbol{F} \in \sigma^{k \times n} \tag{3.9}$$

$$\sum_i \boldsymbol{C}_{r,i}\boldsymbol{F}_{i,c} \leq n \tag{3.10}$$

where matrix $\boldsymbol{A}$ denotes the region that is covered by the factorisation, i.e., $\boldsymbol{A}_{i,j} = 1$ iff $(\boldsymbol{C} \times \boldsymbol{F})_{i,j} > 0$.

Proof. Let $(\mathbf{C}^*, \mathbf{F}^*)$ be the optimal solution for Sparse pRMF. Using definitions (2.7) and (2.8), and the additional constraints in Equations (3.28), we have:

$$(\mathbf{C}^*, \mathbf{F}^*) \equiv \underset{\mathbf{C},\mathbf{F}}{\operatorname{argmax}} \sum_{r,c} \delta(\mathbf{M}_{r,c}, (\mathbf{C} \odot_p \mathbf{F})_{r,c}) \tag{3.11}$$

subject to

$$\mathbf{C} \in \{0,1\}^{m \times k} \tag{3.12}$$

$$\mathbf{F} \in \sigma^{k \times n} \tag{3.13}$$

$$(\mathbf{C} \odot_p \mathbf{F})_{r,c} \leq n \tag{3.14}$$

As we use the plus-product semiring, the matrix product of the factorisation is calculated by using Equation (3.2). Hence, the above model is equivalent to the following:

$$(\mathbf{C}^*, \mathbf{F}^*) \equiv \underset{\mathbf{C},\mathbf{F}}{\operatorname{argmax}} \sum_{r,c} \delta(\mathbf{M}_{r,c}, \sum_i \mathbf{C}_{r,i}\mathbf{F}_{i,c}) \tag{3.15}$$

subject to

$$\mathbf{C} \in \{0,1\}^{m \times k} \tag{3.16}$$

$$\mathbf{F} \in \sigma^{k \times n} \tag{3.17}$$

$$\sum_i \mathbf{C}_{r,i}\mathbf{F}_{i,c} \leq n, \forall r, \forall c \tag{3.18}$$

Replace the $\delta(,)$ function by Equation (3.4) and let $\mathbf{A}$ be the covered matrix, the above model is equivalent to the following:

$$(\mathbf{C}^*, \mathbf{F}^*) \equiv \underset{\mathbf{C},\mathbf{F}}{\operatorname{argmax}} \sum_{r,c} (\alpha \mathbf{A}_{r,c} - |\mathbf{A}_{r,c}\mathbf{M}_{r,c} - \sum_i \mathbf{C}_{r,i}\mathbf{F}_{i,c}|) \tag{3.19}$$

subject to

$$\mathbf{C} \in \{0,1\}^{m \times k} \tag{3.20}$$

$$\mathbf{F} \in \sigma^{k \times n} \tag{3.21}$$

$$\sum_i \mathbf{C}_{r,i}\mathbf{F}_{i,c} \leq n, \forall r, \forall c \tag{3.22}$$

which concludes the proof. □

To solve the optimisation model for Sparse pRMF stated in Equations (3.7) – (3.10), we denote by $\mathbf{Y}_{i,j}$ the value of $|\mathbf{M}_{i,j} - \sum_{t=1}^{k} \mathbf{C}_{i,t}\mathbf{F}_{t,j}|, i = 1, \ldots, m,\ j = 1, \ldots, n$. Then, the model can be re-written as follows.

$$\arg\max_{C,F,Y} \sum_{i=1}^{m}\sum_{j=1}^{n} \alpha\mathbf{A}_{i,j} - \mathbf{Y}_{i,j} \tag{3.23}$$

subject to

$$\mathbf{A}_{r,c}\mathbf{M}_{i,j} - \sum_{t=1}^{k} \mathbf{C}_{i,t}\mathbf{F}_{t,j} \leq \mathbf{Y}_{i,j} \tag{3.24}$$

$$-\mathbf{A}_{r,c}\mathbf{M}_{i,j} + \sum_{t=1}^{k} \mathbf{C}_{i,t}\mathbf{F}_{t,j} \leq \mathbf{Y}_{i,j} \tag{3.25}$$

$$\mathbf{A}_{i,j} \leq \sum_{t=1}^{k} \mathbf{C}_{i,t}\mathbf{F}_{t,j} \tag{3.26}$$

$$n\mathbf{A}_{i,j} \geq \sum_{t=1}^{k} \mathbf{C}_{i,t}\mathbf{F}_{t,i} \tag{3.27}$$

$$\sum_{t=1}^{k} \mathbf{C}_{i,t}\mathbf{F}_{t,j} \leq n \tag{3.28}$$

$$\mathbf{C}_{i,t} \in \{0, 1\} \tag{3.29}$$

$$\mathbf{F}_{t,j} \in \sigma \tag{3.30}$$

Here, $1 \leq i \leq m$, $1 \leq j \leq n$, $1 \leq t \leq k$. $\mathbf{M}$ is the given data matrix; variables $\mathbf{A}$, $\mathbf{C}$, $\mathbf{F}$, $\mathbf{Y}$ are to be found.

Inequalities (3.24) and (3.25) are introduced to remove the absolute operator of the summations in Equation (3.7). Inequalities (3.26) and (3.27) are used to ensure matrix $\mathbf{A}$ to be the cover matrix (Theorem 3.1). Inequality (3.28) ensures that the reconstructed matrix is a rank matrix.

Note that the newly introduced optimisation problem in (3.23) - (3.30) is an *Integer Linear Programming (ILP)* problem if either $\mathbf{C}$ or $\mathbf{F}$ is known. This makes it possible to apply an EM-style algorithm as shown in Algorithm 1, in which the matrix $\mathbf{F}$ is optimised given matrix $\mathbf{C}$, and matrix $\mathbf{C}$ is optimised

given matrix **F**, and we repeat the iterative optimisation till the optimal score cannot be improved any more.

Algorithm 1 *Sparse pRMF algorithm*

Require: Rank matrix **M**, integer k, threshold α
Ensure: Factorisation **C**, **F**
1: Initialise **C** using K-means algorithm
2: **while** not converged **do**
3: **F** $\leftarrow$ Optimise (3.23) - (3.30) given **C**
4: **C** $\leftarrow$ Optimise (3.23) - (3.30) given **F**
5: **end while**

To avoid local maxima, we need to initialise the iterative process in a reasonable way, i.e., smarter than random. The solution we choose is to initialise the matrix **C** using the well-known K-means algorithm. To compute the similarities of rank vectors in K-means, we use the Footrule scoring function. The K-means algorithm clusters the rows in k groups, which can be used to initialise the k columns of **C**. Note that this results in initially disjoint patterns, in terms of their covers, but the iterative optimisation approach may introduce overlap.

Implementation. We implemented the algorithm in OscaR [87], which is an open source Scala toolkit for solving Operations Research problems. OscaR supports a modelling language for ILP. We configured OscaR to use Gurobi[2] as the back-end solver. Source code can be downloaded from the following address: `https://github.com/rankmatrixfactorisation/Sparse_pRMF`.

3.6 Solving Sparse mRMF using IP

To solve the Sparse mRMF problem, we use the same solving strategy that we apply for Sparse pRMF (see Section 3.5). That is, we will first re-write the optimisation model for Sparse mRMF, which is defined in Problem 3.2, in such a form that it can be put into generic Integer Programming (IP) solvers.

Theorem 3.2 (Optimisation model for Sparse mRMF)**.** *Given a rank matrix* ***M****, an integer* k *and a threshold* α*, Sparse mRMF is equivalent to the following*

[2] `http://www.gurobi.com/`

optimisation problem.

$$(\boldsymbol{C}^*, \boldsymbol{F}^*) \equiv \underset{\boldsymbol{C},\boldsymbol{F}}{\operatorname{argmax}} \sum_{r,c} (\alpha \boldsymbol{A}_{r,c} - |\boldsymbol{A}_{r,c}\boldsymbol{M}_{r,c} - \max_i \{\boldsymbol{C}_{r,i}\boldsymbol{F}_{i,c}\}|) \tag{3.31}$$

subject to

$$\boldsymbol{C} \in \{0,1\}^{m \times k} \tag{3.32}$$

$$\boldsymbol{F} \in \sigma^{k \times n} \tag{3.33}$$

where matrix $\boldsymbol{A}$ *denotes the region that is covered by the factorisation, i.e.,* $\boldsymbol{A}_{i,j} = 1$ *iff* $(\boldsymbol{C} \times \boldsymbol{F})_{i,j} > 0$.

Proof. Theorem (3.2) can be proven in a similar way used for Theorem (3.1). □

The optimisation model for Sparse mRMF stated in Equations (3.31) – (3.33) can be re-written as follows.

$$\text{maximize} \sum_i \sum_j (\alpha \mathbf{A}_{i,j} - \mathbf{Y}_{i,j}) \tag{3.34}$$

subject to

$$0 \leq \mathbf{C}_{i,t} \leq 1 \tag{3.35}$$

$$0 \leq \mathbf{F}_{i,t} \leq n \tag{3.36}$$

$$\mathbf{R}_{i,j} \geq \mathbf{C}_{i,t}\mathbf{F}_{t,j} \tag{3.37}$$

$$\mathbf{R}_{i,j} \leq \mathbf{C}_{i,t}\mathbf{F}_{t,j} + (1 - \mathbf{B}_{i,j,t})n \tag{3.38}$$

$$\mathbf{B}_{i,j,t} \in \{0,1\} \tag{3.39}$$

$$\sum_t \mathbf{B}_{i,j,t} = 1 \tag{3.40}$$

$$\mathbf{A}_{i,j} \in \{0,1\} \tag{3.41}$$

$$n\mathbf{A}_{i,j} \geq R_{i,j} \tag{3.42}$$

$$\mathbf{A}_{i,j} \leq R_{i,j} \tag{3.43}$$

$$\mathbf{M}_{i,j}\mathbf{A}_{i,j} - \mathbf{R}_{i,j} \leq \mathbf{Y}_{i,j} \tag{3.44}$$

$$-\mathbf{M}_{i,j}\mathbf{A}_{i,j} + \mathbf{R}_{i,j} \leq \mathbf{Y}_{i,j} \tag{3.45}$$

Here, $1 \leq i \leq m$, $1 \leq j \leq n$, $1 \leq t \leq k$. $\mathbf{M}$ is the given data matrix; variables $\mathbf{R}$, $\mathbf{C}$, $\mathbf{F}$, $\mathbf{B}$, $\mathbf{A}$ and $\mathbf{Y}$ are to be found.

The correctness of this model follows from the following arguments:

- variables $\mathbf{R}_{i,j}$ encode the result of $\mathbf{C} \odot \mathbf{F}$: formula (3.37) ensures that $\mathbf{R}_{i,j} \geq \max_t \mathbf{C}_{i,t}\mathbf{F}_{t,j}$; formulas (3.38)-(3.40) ensure that $\mathbf{R}_{i,j} \leq \mathbf{C}_{i,t}\mathbf{F}_{t,j}$ for one choice for t (indicated by $\mathbf{B}_{i,j,t}$), which in combination with formula (3.37) means that the maximum is chosen;
- formulas (3.41)-(3.43) ensure that variables $\mathbf{A}_{i,j}$ encode those cells that are covered by the factorisation; i.e., $\mathbf{A}_{i,j} = 1$ iff $\mathbf{R}_{i,j} > 0$;
- formulas (3.34), (3.44) and (3.45) encode the additive scoring function based on formula (3.4); formula (3.44) and (3.45) ensure that $\mathbf{Y}_{i,j} \geq |\mathbf{M}_{i,j}\mathbf{A}_{i,j} - \mathbf{R}_{i,j}| = |\mathbf{M}_{i,j}\mathbf{A}_{i,j} - \mathbf{R}_{i,j}\mathbf{A}_{i,j}| = |\mathbf{M}_{i,j} - \mathbf{R}_{i,j}|\mathbf{A}_{i,j}$; as we look for maximal solutions in formula (3.34), which can only be obtained when $\mathbf{Y}_{i,j}$ is minimal, we can conclude that $\mathbf{Y}_{i,j} = |\mathbf{M}_{i,j} - \mathbf{R}_{i,j}|\mathbf{A}_{i,j}$. The optimisation criterion for one cell (i, j) can then be written as $\alpha\mathbf{A}_{i,j} - |\mathbf{M}_{i,j} - \mathbf{R}_{i,j}|\mathbf{A}_{i,j} = (\alpha - |\mathbf{M}_{i,j} - \mathbf{R}_{i,j}|)\mathbf{A}_{i,j}$, which corresponds to equation (3.4).

Note that this modeling approach can trivially be modified to solve variations of the Sparse mRMF problem; e.g., to deal with a min-product semiring we only need to modify equations (3.37)–(3.40).

Similar to the technical problem of solving Sparse pRMF (Section 3.5), the model for Sparse mRMF above is not a linear model if we would need to search for both $\mathbf{F}$ and $\mathbf{C}$. However, if we assume that one of these is fixed, the model is linear, and consequently, in each iteration of our algorithm we can use Integer Linear Programming (ILP) solvers.

Efficient parallel search. In the solution discussed above, we repeatedly solve integer linear programs to determine $\mathbf{C}$ and $\mathbf{F}$. These integer linear programs are still hard to solve and involve finding $m \times k$ and $k \times n$ assignments, respectively, for the matrices $\mathbf{C}$ and $\mathbf{F}$.

We use the following properties to make solving these integer linear programs more efficient:

- the reconstructed matrix $\mathbf{R}$ is calculated based on a matrix product over a semiring;
- the scoring function is additive.

The consequence of these properties is that for a given $\mathbf{C}$, optimal values for all columns of $\mathbf{F}$ can be determined independently of each other; similarly, for a given $\mathbf{F}$, the rows of $\mathbf{C}$ can be determined independently of each other.

Consider the case of determining $\mathbf{C}$ given $\mathbf{F}$, i.e., determining the optimal occurrences of given rank patterns for each row. Given that our scoring function is additive, the error score for one row in the reconstructed matrix $\mathbf{R}$ will not affect the error made for another row. Furthermore, given the use of products based on semirings, a row in the reconstructed matrix $\mathbf{R}$ is only determined by the corresponding row in the matrix $\mathbf{C}$ and the complete rank matrix $\mathbf{F}$.

We exploit this property by running the solver for each row of $\mathbf{C}$ independently to determine the optimal solution for that row only.

A further consequence of the independence of rows is that we can determine row assignments in parallel to each other. Consequently, we can distribute the optimisation problem over multiple cores.

Implementation. To implement our system, we also relied on the OscaR system [87]. OscaR supports a modelling language for ILP. We configured OscaR to use the Gurobi[3] IP solver as the back-end solver. A benefit of OscaR/Scala is that it has built-in support for exploiting multiple cores to solve independent optimisation problems in parallel. The implementation is available at the following address: `https://github.com/rankmatrixfactorisation/Sparse_mRMF`.

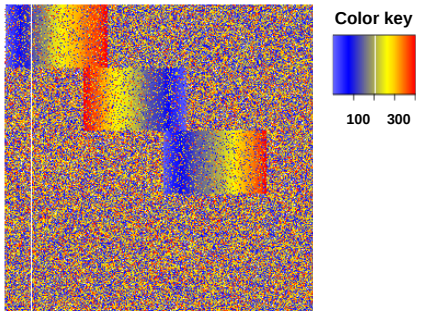
3.7 Experiments with synthetic data

In this section, we set up a number of experiments on synthetic datasets to evaluate Sparse mRMF, the generic approach for Sparse RMF. The results will show: 1) the accuracy of recovering order-preserving patterns; 2) the robustness of the algorithm with respect to noise; 3) the sensitivity of the algorithm with respect to the parameter thresholds.

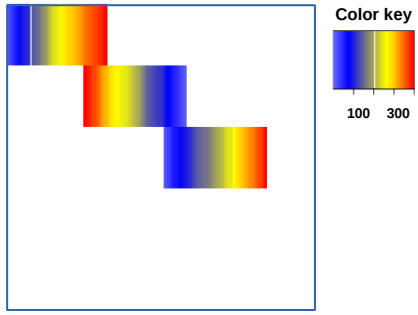
Data generation. As we would like to evaluate the capability of Sparse mRMF to recover consistent rankings of a subset of columns in a subset of rows, we generate ranked data with implanted orders.

For each dataset, we first implant three ranked patterns and then generate its background information. Patterns are created by generating a reference rank profile and repeating this profile for a number of rows. Each reference rank

[3] http://www.gurobi.com/



(a) Rank data (1000 × 400)



(b) Sparse mRMF

Figure 3.1: Recovering three implanted rank profiles from the synthetic data with implanted orders. Figure 3.1a shows the data ($w = 20$). Figure 3.1b is obtained by Sparse mRMF with $k = 3, \alpha = 20\%$.

profile is generated by uniformly sampling l integer numbers from the range $[1 \dots n]$, where l is the number of columns in the pattern. Noise is simulated by swapping w number of column pairs for each row in the pattern. After the patterns are implanted, each row is completed by a random permutation of the values in $[1 \dots n]$ not in its reference rank profile (i.e., the set difference).

We generate four 1000 rows × 400 columns datasets, one for each $w \in \{0, 10, 20, 30\}$. In each dataset, we implant three overlapping order-preserving rank patterns, each of which spans 200 rows and 130 columns. Figure 4.8a shows the dataset generated with $w = 20$.

Accuracy of the recovered tiles by Sparse mRMF. We ran Sparse mRMF on the four simulated datasets with $k = 3$ and varying values for the threshold α, i.e., $\alpha \in \{10\%, 20\%, 30\%\}$. For each combination of threshold and dataset, we ran the algorithm 10 times and used the result that had the highest score. Similar to the previous experiments, we also used *precision* and *recall* to evaluate recovery accuracy.

We now formally define precision and recall. Let $\mathcal{S}_t = \{B_1 = (R_1, C_1), \dots, B_k = (R_k, C_k)\}$ be the k implanted rank patterns, each of which is specified by a tuple (R_i, C_i) to indicate the rows and the columns of the data region where it occurs. Let $\mathcal{S}_a = \{A_1 = (R_1, C_1), \dots, A_k = (R_k, C_k)\}$ be the k rank patterns discovered by the algorithm. The covered region of the implanted patterns is defined as follows:

$$U = \{(x, y) | \exists B_i = (R_i, C_i) \in \mathcal{S}_t : x \in R_i, y \in C_i\} \tag{3.46}$$

The covered region of the discovered patterns is defined in a similar way.

$$V = \{(x, y) | \exists A_i = (R_i, C_i) \in \mathcal{S}_a : x \in R_i, y \in C_i\} \tag{3.47}$$

Then, precision and recall are calculated by the following equations.

$$\text{Precision} = \frac{|V \cap U|}{|V|} \tag{3.48}$$

$$\text{Recall} = \frac{|V \cap U|}{|U|} \tag{3.49}$$

Having defined precision and recall, we can now measure the performance of Sparse mRMF. Figure 3.2a shows that Sparse mRMF succeeds in recovering the three simulated patterns with high precision and recall when $\alpha = 20\%$. As expected, Sparse mRMF obtains high recall and low precision when the threshold is increased (to 30%); Figure 3.2b shows that both coverage and error increase steeply. When the threshold is (too) low, i.e., $\alpha = 10\%$ in this case, the implanted patterns cannot be recovered. Hence, for this case, Sparse mRMF has low values for both precision and recall.

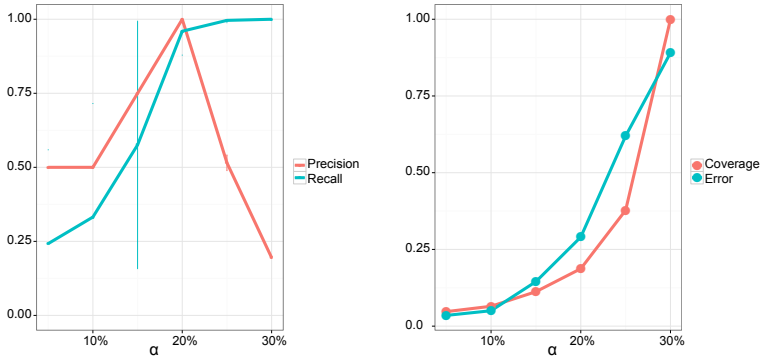
We did not run Sparse pRMF, the specific approach for Sparse RMF, on the synthetic datasets as these datasets do not overlap in the row dimension. Hence, Sparse mRMF is general enough to show capability of the two proposed approaches in discovering the structure we are interested in. We postpone the comparison of the Sparse pRMF and Sparse mRMF in terms of recovering overlapping rank profiles until real world case studies in the next section, where we have realistic settings. We also did not compare with BMF [82] and the traditional tiling method [37] as converting ranked data to Boolean data will loose the rank information of the columns, which is essential for this type of pattern.

3.8 Real world case studies

In this section we report on two real world case studies concerning the European Song Festival, and sushi consumption.

3.8.1 European Song Festival dataset

The Eurovision Song Contest (ESC) has been held annually since 1956. Each participating country gives voting scores, which are a combination of televoting



(a) Precision and recall (b) Coverage and error

Figure 3.2: Evaluation of precision and recall scores for Sparse mRMF on the synthetic data with implanted orders.

and jury voting, to competing countries. Scores are in the range of $1 \ldots 8$, 10, and 12. Each country awards 12 points to its most favourite country, 10 points to its second favourite, and $8 \ldots 1$ to the third...tenth favourites respectively. The data can be represented by a matrix in which rows correspond to voting countries, columns correspond to competing countries, and cell values to the scores.

We collected voting data for the final rounds of the period 2010 – 2013. We filtered out countries participating in fewer than 3 years. The data is aggregated by calculating average scores that voting countries award to competing countries during the study period. After the pre-processing step, the final data consists of 44 rows and 37 columns, corresponding to 44 voting countries and 37 competing countries. The pre-processed data is transformed to ranked data using minimum ranks in case of ties.

Running experiments. We applied both Sparse pRMF and Sparse mRMF on the ESC dataset with varying threshold values and then analysed the obtained coverage and error per covered cells to empirically select the best thresholds.

With Sparse mRMF, we varied the threshold α in the range $\{5\%, 10\%, \ldots, 30\%\}$, for each of which we factorised the rank matrix with different values for k, i.e., $k \in \{5, \ldots, 12\}$. Figure 3.3a shows the average coverage and error scores for the different α values. It can be seen that, for $\alpha = 5\%$, Sparse mRMF made almost no mistakes (error $= 0.04$) but coverage is low (19%). When $\alpha = 30\%$, on the other hand, Sparse mRMF covers almost the entire matrix ($> 90\%$) while having an average cell-based error of 5, which might be acceptable as the

range of the rank score in this dataset is $[1 \dots 37]$. Hence, we would choose a threshold value based on coverage and/or error depending on the background knowledge and preferences of the data miner. Here we choose $\alpha = 10\%$ as the corresponding error is low and the coverage is substantially higher than that for $\alpha = 5\%$. Given α, we next have to decide an appropriate value for k. For this we examine Figure 3.3b, which shows the coverage for $\alpha = 10\%$ and varying k. We choose $k = 10$ as coverage appears to be stable for higher k.

Voting patterns. The heatmap of the reconstructed matrix obtained by Sparse mRMF is shown in Figure 3.4b. Compared to the original rank matrix in Figure 3.4a, the reconstructed heatmap shows that Sparse mRMF strongly *sparsified* the original rank matrix. Note that it also shows the rank profiles produced by Sparse mRMF contain both high and low ranks.

Table 3.1 illustrates the benefits of using the max-product-based formalisation for rank matrix factorisation. It shows that compared to the specific approach (Sparse pRMF), which uses the plus-product semiring (linear algebra) and constraints to avoid the 'overranking' problem, the general approach (Sparse mRMF) using max-product semiring to aggregate the rank profiles attains higher coverage (32% compared to 30%) and a lower error (1.12 compared to 1.59). Sparse mRMF also enjoys a substantial increase of the overlap among the rank profiles in their covered rows. However, it takes more time. Note that the wall-clock runtimes shown in this table were measured without using parallel computation.

To show how the rank profiles produced by the two methods can provide insight into the data, we visualise two rank profiles of Sparse mRMF in Figure 3.5. They show the typical voting behaviour of Western European countries towards Nordic countries (Figures 3.5b), and that of Eastern European countries toward some other countries (Figures 3.5a). For example, countries in Eastern Europe tend to give higher scores to Russia and Nordic countries than to other countries. In general, the discovered patterns confirm that countries tend to give high scores to their neighbours, which confirms common knowledge about the European Song Contest.

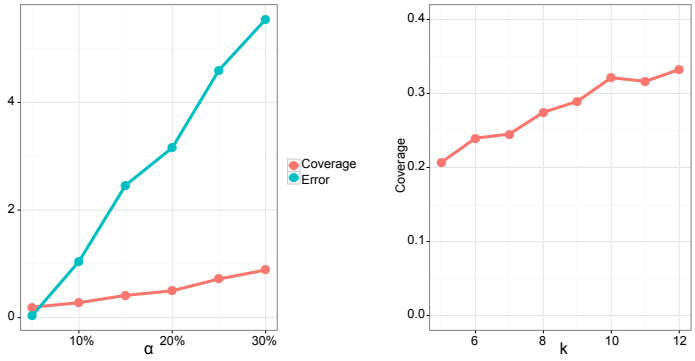
3.8.2 The Sushi dataset

The Sushi dataset, collected by Kamishima [54], contains preferences of 5000 people over ten different sushi types.

We ran both Sparse pRMF and Sparse mRMT on this dataset. With Sparse sRMF, we chose the same threshold values ($\alpha = 20\%$ and $k = 8$) as we used in our previous work [68] for a fair comparison.

Table 3.1: Performance statistics of Sparse pRMF and Sparse mRMF on the European Song Festival dataset. Overlap is the percentage of the covered rows involving in more than 1 rank profiles. 0s/pattern is the average percentage of 0s in rank profiles.

Algorithm	Coverage	Avg. error	0s/pattern	Overlap	Time
Sparse pRMF	30%	1.59	59.7%	2%	3s
Sparse mRMF ($\alpha = 10\%, k = 10$)	32%	1.12	52.4%	30%	69.2s

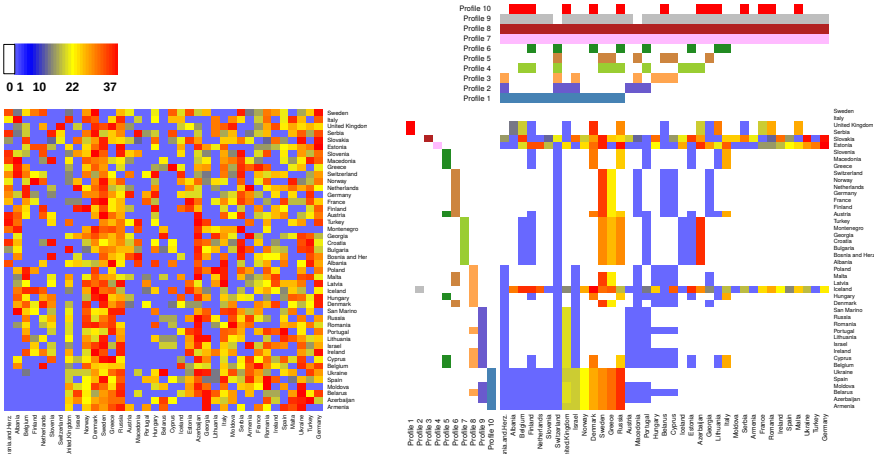


(a) Sparse mRMF (b) Sparse mRMF

Figure 3.3: Parameter tuning for Sparse mRMF on the European Song Festival dataset.

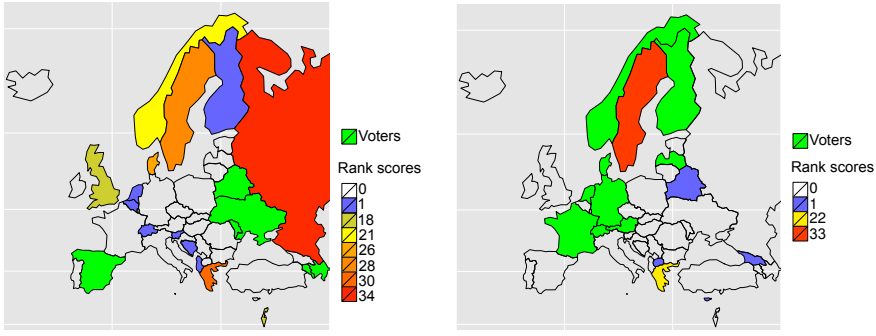
Table 3.2 shows that the rank profiles found using Sparse mRMF have much larger overlaps and are sparser than the ones found using Sparse pRMF, while the average error per covered cell is smaller. The only drawback of Sparse mRMF are the longer runtimes that come with these improved results.

Figure 3.6 shows the eight rank profiles found by Sparse mRMF. In general, the customers have clear preferences over the ten sushi types. For example, the first rank profile F1 in Figure 3.6 depicts that there is a group of customers preferring light sushi types, such as maguro, ebi, and tekka maki, to oily and seasoning sushi like uni and anago. Interestingly, the eighth rank profile F8 maintains that there exists a group of customers who have a completely opposite taste when they prefer anago sushi to ebi sushi.



(a) Rank data (b) The reconstructed matrix by Sparse mRMF

Figure 3.4: EU Song Festival results show how Sparse RMF focuses on specific structure and hence sparsify the data. Figure 3.4a and Figure 3.4b have the same color key.



(a) Tile 1 (Sparse mRMF) (b) Tile 6 (Sparse mRMF)

Figure 3.5: Rank patterns discovered on the ESC dataset by Sparse mRMF. The rank profiles, which depict the obtained voting scores of competitors, are painted in red; the corrresponding rows (voting countries), which show the places where these rank profiles appear, are painted in green.

3.9 Related work

Sparse RMF is in favor of factorisations that can capture patterns users are interested in. This means that Sparse RMF does not aim to completely recover the original matrix as long as the detected patterns can capture the main

Table 3.2: Performance statistics of Sparse pRMF and Sparse mRMF on the Sushi dataset [54]. Overlap and 0s/pattern have the same meaning as those in Table 3.1.

Algorithm	Coverage	Avg. error	0s/pattern	Overlap	Time/run
Sparse pRMF ($\alpha = 20\%, k = 8$)	78.2%	1.31	13.8%	0%	53mi
Sparse mRMF ($\alpha = 20\%, k = 8$)	77.2%	1.22	41.3%	75.2%	2h32mi

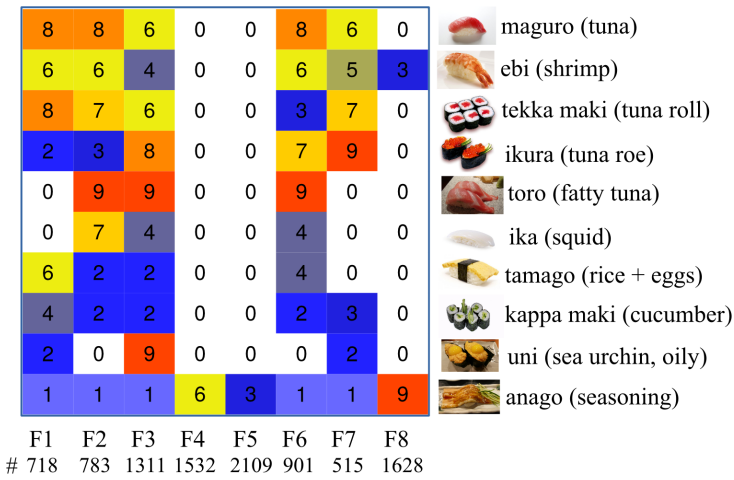


Figure 3.6: Sushi preference patterns found by Sparse mRMF. The bottom row below the heatmaps, indicated by #, shows the number of users having the corresponding rank profile.

structure in the matrix. In this regard, Sparse RMF is related to *sparse dictionary learning* [118, 14, 78]. However, the strategy to define the optimisation functions to find the sparse encodings is different. In sparse dictionary learning [78], the distance function is defined to minimise the error between the original matrix and the reconstructed matrix for all data points and encourages to cover less by using regularisation. In Sparse pRMF and Sparse mRMF, the scoring functions are defined in such a way that they can encourage to cover more data while having lower error in the covered region rather than in the whole data matrix.

Method	Semiring	Data	Comment
NMF	Plus-product	Rank data	Cannot discover consistent subsets of rankings
sRMF	Max-product	Rank data	Discover consistent subsets of rankings
sRMF	Plus-product	Rank data	Results have less overlappings than the one using sRMF with the max-product semiring
sRMF	Min-product	Rank data	More or less the same as using sRMF with the max-product semiring

Table 3.3: Comparison of methods in combination with different semirings for solving the Sparse RMF problem.

3.10 Discussion

Table 3.3 summarises the effects of applying different methods in combination with different semirings to solve the Sparse RMF problem.

First, Non-negative Matrix Factorisation (NMF) [71] is not applicable to the Sparse RMF problem that we study in this thesis. This is mainly because NMF seeks a factorisation such that the reconstructed matrix is similar to the original matrix as much as possible. The reconstructed matrix is often dense as it takes the noise outside the patterns into account. Consequently, local patterns cannot be detected.

Second, the sRMF framework that we introduce in this thesis has a different behaviour under different semirings. That is, when we use the sRMF framework with the plus-product semiring, i.e., Sparse pRMF, the results will have less overlapping parts than the case in which we use the framework with the max-product semiring, i.e., Sparse mRMF. However, the result should not be much different between the case that we apply sRMF with the max-product semiring and the case with the min-product semiring. This is because when we have a consistent subset of ranks in a number of rows, no matter whether we take min or max, the aggregated result should be the same.

3.11 Conclusions

We study the Sparse RMF problem, which aims at discovering a set of k sparse rank profiles in rank matrices. Such sparse profiles can be used to succinctly summarise the given rank matrix and show main categories of rankings.

We apply the sRMF framework to model Sparse RMF with two different semirings, each of which leads to a problem instance for Sparse RMF. Modelling the problem using this framework illustrates the expressiveness and flexibility of the proposed framework. Experiments on both synthetic datasets and real world problems shows that the framework is capable of discovering sparse rank profiles as well as obtaining high quality solutions.

Chapter 4

Ranked Tiling

This chapter[1] presents the second instantiation of the sRMF framework to mine rank pattern sets in rank data. That is, we study how to use the framework to do *tiling* in rank matrices to discover *ranked tiles*, which are data regions having high ranks. Such ranked tiles are interesting as they can show local associations between subsets of the rows and subsets of the columns of the given matrices.

We first study the problem of finding one maximal ranked tile. Then, we study the ranked tiling problem, i.e., find k maximal ranked tiles that together cover the given rank matrix as much as possible. For each of the problem, we present two different approaches to modelling and solving. The first approach proposed in [69], Constraint Programming (CP) is used. The second approach developed later in [66] uses the sRMF framework and results in improved solution quality. We present experimental results on three real-life datasets, which demonstrate the interestingness as well as the practical applicability of the ranked tiling pattern. One case study involves a heterogeneous dataset concerning the discovery of specific molecular features for different subtypes of breast cancer patients. An analysis of the tiles by a domain expert shows that our approach can lead to the discovery of novel insights.

[1] Based on the journal paper "Semiring rank matrix factorisation" [66] and the conference paper "Ranked tiling" [69]

4.1 Introduction

The problem of tiling was introduced by Geerts et al. [37]. It is a popular pattern mining technique that searches for a set of tiles (that is, a *tiling*) in a 0/1 matrix. Such matrices often represent transactional data, where each transaction specifies the presence or absence of a set of items in the transaction. A tile is then a subset of the rows and columns of the matrix, for which the corresponding submatrix contains all 1s. Tilings are interesting as they provide groupings of both the rows and the columns that may give new insights in the data.

In this chapter, we extend tiling towards a setting which is not binary. That is, we study the problem of *ranked tiling*, in which each transaction in the data is a ranking of all available items. In comparison with Sparse RMF studied in Chapter 3, in ranked tiling we do not require that the ranks of the items are (approximately) the same between different rows included in the tile; we only require that sufficiently high ranks are included in a tile. In other words, we only care of whether the ranks are high or low.

One real-life example that we shall use is concerned with the discovery of biomarkers to group cancer patients into *subtypes*. Finding a set of biomarkers that characterise different cancer subtypes is clearly important. Thanks to advances in genome sequencing and high throughput technologies, a lot of data is becoming available about patients. However, different types of data may be obtained with different technologies, for example, data concerning mRNA, miRNA, copy number variations, or proteins. This means that we are given a number of data matrices, each of which corresponds to one data type or experiment, and each of which is measured on a different scale. Hence, the values of the rows of the matrices are incomparable. Still we would expect to find a tiling in which the same set of patients is shared across the different data matrices. We shall show that by using a ranked version of the data, which makes the rows comparable, and ranked tiling, this is feasible.

To illustrate the problem of ranked tiling, let us consider the toy example in Figure 4.1. It depicts a rank matrix containing five rows and ten columns. Assuming no ties, each row contains each of the numbers one to ten exactly once. In this paper, we assume that a desirable high rank is indicated by a high number, i.e., in this case the highest possible rank is ten. Now, we are intuitively interested in rectangular areas in the matrix that have relatively high values, as these correspond to columns and rows which are highly ranked. In this particular example, the maximal ranked tile that we would like to find consists of five rows and three columns, i.e., the area defined by $\{R1, R2, R3, R5\}$ and $\{C1, C2, C3\}$.

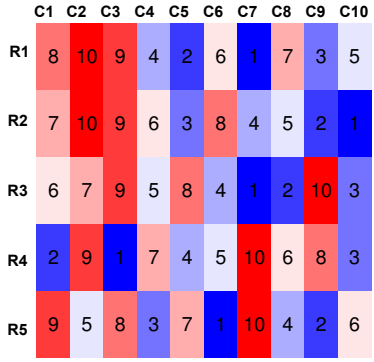


Figure 4.1: Example rank matrix, with maximal ranked tile $B = (\{R1, R2, R3, R5\}, \{C1, C2, C3\})$.

4.2 One maximal ranked tile mining

In this section, we study the problem of finding one maximal ranked tile. We first introduce an optimisation model to formally define a maximal ranked tile. Then, we show how we can solve the optimisation problem by using CP and the sRMF framework.

4.2.1 One maximal ranked tile

Definition 4.1 (Maximal ranked tile)**.** *Given a rank matrix $\boldsymbol{M} \in \sigma^{m \times n}$ and a threshold θ, find the ranked tile $B = (R^*, C^*)$, with $R^* \subseteq \mathcal{R}$ and $C^* \subseteq \mathcal{C}$, such that:*

$$B = (R^*, C^*) = \operatorname*{argmax}_{R,C} \sum_{r \in R, c \in C} (\boldsymbol{M}_{r,c} - \theta). \tag{4.1}$$

where θ is an absolute-valued threshold.

Example 4.1. *Going back to the example rank matrix in Figure 4.1 and choosing $\theta = 5$, the maximal ranked tile is defined by $R = \{1, 2, 3, 5\}$ and $C = \{1, 2, 3\}$. The score obtained by this tile is 37, and no more columns or rows can be added without decreasing the score. This result matches the desired outcome that we described in the introduction.*

In practice, we often use a relative instead of an absolute threshold. We denote such a threshold as a percentage, i.e., $\theta = a\%$ implies $\theta = a\% \times n$.

The optimisation objective in Equation 4.1 rewards cells in a tile having values higher than θ, and vice versa for cells having lower values. Since we look for tiles that maximise this score, this threshold θ plays an important role. That is, higher values for θ result in smaller tiles with larger ranks. This implies that the threshold can be used to influence both 1) the size of the mined tiles, and 2) the extent to which the ranks deviate from the mean rank. An alternative interpretation is that the threshold can be used by the analyst to express her prior belief about how high the ranks should be to make a tile interesting.

4.2.2 Maximal ranked tile mining using CP

In this section, we present a CP-based technique to solve the problem of finding one maximal ranked tile introduced in the previous section. That is, we introduce a constraint-based model equivalent to Equation 4.1, but add two constraints to make solving more efficient without affecting the results.

The technique we use to solve the optimisation problem is constraint programming (CP). We follow the approach that was originally introduced by De Raedt et al. [22] and further developed by Guns et al. [44]. The idea is to formalise the problem as *a constraint satisfaction problem* and then use existing solvers to find solutions. There are a number of advantages to this solving paradigm. First, it is a declarative approach, meaning that the data analyst can focus on modelling the problem rather than on complex procedural implementations. Second, CP is very flexible.

Constraint-based model

To speed up the search process, we add two redundant constraints to the optimisation problem of Equation 4.1. That is, we require that the average values in rows and columns in the selected submatrix $\mathbf{M}_{R,C}$ are higher than the threshold θ. The resulting constraint-based model is as follows:

$$\text{argmax}_{R,C} \sum_{r\in R, c\in C} (\mathbf{M}_{r,c} - \theta) \tag{4.2}$$

subject to

$$\forall r \in \mathcal{R} : r \in R \leftrightarrow \quad \frac{\sum_{c\in C} \mathbf{M}_{r,c}}{|C|} \geq \theta \tag{4.3}$$

$$\forall c \in \mathcal{C} : c \in C \leftrightarrow \quad \frac{\sum_{r\in R} \mathbf{M}_{r,c}}{|R|} \geq \theta \tag{4.4}$$

Theorem 4.1 (Model equivalence)**.** *The constraint-based optimisation model in Equations (4.2) –(4.4) is equivalent to the optimisation model in Equation (4.1).*

Proof. Let us assume that (R, C) is the optimum solution found without additional constraints. Without loss of generality we can assume that (R, C) is maximal in both rows and columns, i.e., there is no row nor column that can be added to obtain the same score or a better score.

Then this optimal solution must also satisfy the constraint:

$$\forall r \in \mathcal{R} : r \in R \leftrightarrow \quad \frac{\sum_{c \in C} \mathbf{M}_{r,c}}{|C|} \geq \theta \leftrightarrow (\sum_{c \in C} \mathbf{M}_{r,c} - \theta) \geq 0. \tag{4.5}$$

Indeed, assume that $r \in R$ while (R, C) is optimal, then $(\sum_{c \in C} \mathbf{M}_{r,c} - \theta) \geq 0$ must hold, as otherwise the score $\sum_{c \in C, r \in R} (\mathbf{M}_{r,c} - \theta)$ could be improved by removing r from R while keeping C fixed. Conversely, if $r \notin R$ while (R, C) is optimal, it can only be the case that $(\sum_{c \in C} \mathbf{M}_{r,c} - \theta) < 0$, as otherwise the score of the tile could be improved by adding r to R while keeping C fixed. □

Problem formalisation using CP

The formalisation in Equations (4.2) – (4.4) is defined over set variables. Unfortunately, in earlier work it was shown [22] that set variables do not lead to good performance in some CP systems and a reformalisation in terms of boolean variables can be desirable.

We therefore introduce two Boolean decision vectors: $T = (T_1, T_2, ..., T_m)$, with $T_r \in \{0, 1\}$, for rows and $I = (I_1, I_2, ..., I_n)$, with $I_c \in \{0, 1\}$, for columns. An assignment to the Boolean vectors T and I corresponds to an indication of rows and columns belonging to a tile. Given this, we have the following theorem.

Theorem 4.2 (Highly ranked rows constraint)**.** *If the following constraint is satisfied:*

$$\forall r \in \mathcal{R} : T_r = 1 \leftrightarrow \sum_{c \in \mathcal{C}} (\boldsymbol{M}_{r,c} - \theta) * I_c \geq 0 \tag{4.6}$$

then rows that satisfy the inequality of Equation (4.3) are identified by:

$$\{r \in \mathcal{R} \mid T_r = 1\} \tag{4.7}$$

Proof.

$$\forall r \in \mathcal{R} : T_r = 1 \leftrightarrow \sum_{c \in \mathcal{C}} (\mathbf{M}_{r,c} - \theta) * I_c \geq 0 \leftrightarrow \sum_{c \in \mathcal{C}} \mathbf{M}_{r,c} * I_c \geq \theta * \sum_{c \in \mathcal{C}} I_c$$

$$\leftrightarrow \frac{\sum_{c \in \mathcal{C}} \mathbf{M}_{r,c} * I_c}{\sum_{c \in \mathcal{C}} I_c} \geq \theta \quad \leftrightarrow \frac{\sum_{c \in C} \mathbf{M}_{r,c}}{|C|} \geq \theta$$

Here we assume that $\sum_{c \in \mathcal{C}} I_c \geq 1$. Overall, $T_r = 1 \leftrightarrow r \in R$, which concludes the proof.

□

A similar property can be obtained for the column constraint. This leads to a CP model which is equivalent to the constraint-based model in Equations (4.2)–(4.4):

$$\underset{T,I}{\operatorname{argmax}} \sum_{r \in \mathcal{R}} T_r * (\sum_{c \in \mathcal{C}} (\mathbf{M}_{r,c} - \theta) * I_c) \tag{4.8}$$

subject to

$$\forall r \in \mathcal{R} : T_r = 1 \leftrightarrow \sum_{c \in \mathcal{C}} (\mathbf{M}_{r,c} - \theta) * I_c \geq 0 \tag{4.9}$$

$$\forall c \in \mathcal{C} : I_c = 1 \leftrightarrow \sum_{r \in \mathcal{R}} (\mathbf{M}_{r,c} - \theta) * T_r \geq 0 \tag{4.10}$$

The constraints in this formalisation are similar to those used to mine frequent itemsets in [22]. The main difference is that we also have an objective function.

Mining maximal ranked tiles using CP

Mining a single ranked tile is equivalent to finding an assignment to vectors T, I such that T and I satisfy constraints (4.9) – (4.10) and maximise objective function (4.8). We solve this constrained optimisation problem using constraint programming.

Solving a problem using CP is done in two phases: 1) modelling, and 2) solving. Equations (4.8) – (4.10) can be written down as a model in any CP solver. As we will see, however, the problem of ranked tiling is not easy to solve, and finding exact solutions is difficult. To allow for finding approximate solutions, we choose the OscaR solver [87], which is an open source CP solver written

in Scala. A distinguishing feature is that it provides good support for both exhaustive and local search methods, which we will use for our approach.

When the CP solver is asked to perform exhaustive search, it essentially builds a search tree. The key idea here is that it uses the constraints to remove inconsistent values while searching. This removal of inconsistent values is called propagation and can reduce the search space significantly.

Variable and value ordering heuristic. The order in which variables are considered for branching, as well as the order in which values are assigned to the variables, determine the shape of the search tree and the effectiveness of constraint propagation. We use the following heuristic. We order column variables $I_c,\ c = 1 \ldots n$, by their total sum scores, $\sum_{r \in \mathcal{R}} (\mathbf{M}_{r,c} - \theta)$, in ascending order. That is, variables that have lower scores will be branched on first, and the value zero will be assigned to a variable before the value one. Using this heuristic, CP has a higher chance to add variables having high scores to the solution when backtracking. Consequently, CP needs less backtracks to find a first valid assignment as variables having higher scores have higher probability of satisfying constraint (4.10).

Large neighbourhood search. Equation (4.9) shows that vector T can be completely determined given a complete assignment to I. Hence, the size of the search space is $O(2^n)$, where n is the number of columns. Even taking into account propagation, this search space is in practice often still too large to be traversed completely and hence we use a form of local search to speed up the search.

Large Neighbourhood Search (LNS) is a hybridization of local search and exhaustive search in CP. Local search refers to the idea that one solution can be transformed into another by changing the assignment of a number of variables. While traditional local search methods only change a limited number of variables (e.g., one variable at a time), LNS selects a relatively large subset of the variables in a problem (e.g., a random subset of half of the variables) and performs complete search over these variables while fixing the remaining variables. Two main questions involved with LNS include a) which variables should be selected to search over, and b) how to search on these variables? In our implementation, we use stochastic variable selection and an exhaustive search approach. The stochastic variable selection uniformly selects half of the column variables to search over.

4.2.3 Maximal ranked tile mining using sRMF

The maximal ranked tile mining problem can also be studied from the matrix factorisation perspective. Example 4.2 illustrates how the maximal ranked tile shown in Example 4.1 can be found by using the sRMF framework.

Example 4.2 (Maximal ranked tile mining using sRMF)**.**

$$\boldsymbol{M} = \begin{pmatrix} 8 & 10 & 9 & 4 & 2 & 6 & 1 & 7 & 3 & 5 \\ 7 & 10 & 9 & 6 & 3 & 8 & 4 & 5 & 2 & 1 \\ 6 & 7 & 9 & 5 & 8 & 4 & 1 & 2 & 10 & 3 \\ 2 & 9 & 1 & 7 & 4 & 5 & 10 & 6 & 8 & 3 \\ 9 & 5 & 8 & 3 & 7 & 1 & 10 & 4 & 2 & 6 \end{pmatrix} \boldsymbol{C} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \boldsymbol{F} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

This example shows a rank matrix $\boldsymbol{M}$ *in which the maximal ranked tile (red color) with a rank higher than 5 are indicated by means of two Boolean matrices; the matrix* $\boldsymbol{C}$ *identifies the rows included in the tiles; the matrix* $\boldsymbol{F}$ *indicates the columns.*

To formalise the problem of maximal ranked tile mining as a rank matrix factorisation problem, our first choice is to limit the set of permissible values to $\{0, 1\}$; as a result, we only characterise the columns included in the tiles.

Next, we define the scoring function δ, defined in Definition (2.8), as follows:

$$\delta(a, b) = \begin{cases} 0 & \text{if } b = 0; \\ a - \theta & \text{otherwise.} \end{cases} \tag{4.11}$$

Again, in this scoring function we only look at those cells of the rank matrix covered by the tiles. Here, however, we give a higher score for a covered cell if its rank is higher; if the rank is too low, the contribution of the cell may be negative, which will discourage covering too many cells with low ranks.

Note that the effect of the parameter θ is the opposite of the parameter α in Sparse sRMF: the higher we choose the value θ, the smaller will be the tiles that will be found; the higher we choose the value α, the larger the factors we will find.

In summary, the maximal ranked tile mining problem can be defined as follows.

Definition 4.2 (Maximal ranked tile)**.** *The maximal ranked tile mining problem is the rank matrix factorisation problem obtained using*

- *the max-product semiring;*
- *the set of permissible values* $\sigma_p = \{0, 1\}$*;*

- *the additive scoring function based on formula (4.11);*
- $k = 1$.

Definition 4.2 provides another way to define the maximal ranked tile based on the matrix factorisation principle. We will postpone how to solve the new optimisation problem until the next section when we generalise it to the setting in which k is greater than 1.

4.3 Ranked tiling

Till now we study how to find a single ranked tile, but we are of course interested in finding a set of such tiles. In other words, we would like to discover a ranked tiling. Below is an example.

Example 4.3 (Ranked tiling example)**.**

$$\begin{pmatrix} 1 & 2 & 5 & 4 & 6 & 3 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 3 & 2 & 4 & 5 & 6 \\ 2 & 3 & 5 & 6 & 4 & 1 \\ 2 & 3 & 5 & 6 & 1 & 4 \end{pmatrix}$$

When the threshold θ is set to 3, the matrix above has a ranked tiling consisting of two ranked tiles: one is painted blue and the other is painted red. The part which is painted green is the overlapping region of the two ranked tiles.

Next, we will study two approaches to tiling rank matrices: one uses CP and the other uses the sRMF framework.

4.3.1 Ranked tiling using CP (cpRMT)

Directly solving the ranked tiling problem with $k > 1$ using CP is inefficient. Note that the problem is NP-complete (see Section 4.3.3 on page 59). Hence, we propose a greedy CP-based approach to tiling rank matrices. That is, the first tile is found by solving the optimisation problem in Equations (4.8)–(4.10). Next, we remove that tile by setting all cells in the matrix that are covered to the lowest rank. Then, we search in the resulting matrix for the second tile. This process is repeated until the number of desired tiles is found. The proposed greedy approach above is named *cpRMT*, which stands for a *CP-based approach to Rank Matrix Tiling.*

Example 4.4 illustrates how cpRMT works. Note that the ranked tiling found by cpRMT in this example does not have any overlapping region compared to the one shown in Example 4.3. This is due to the greedy strategy of cpRMT.

Example 4.4 (cpRMT example)**.**

$$\boldsymbol{M} = \begin{pmatrix} 1 & 2 & 5 & 4 & 6 & 3 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 3 & 2 & 4 & 5 & 6 \\ 2 & 3 & 5 & 6 & 4 & 1 \\ 2 & 3 & 5 & 6 & 1 & 4 \end{pmatrix} \quad \boldsymbol{M}_1 = \begin{pmatrix} 1 & 2 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 & 0 & 0 \\ 1 & 3 & 0 & 0 & 0 & 0 \\ 2 & 3 & 5 & 6 & 4 & 1 \\ 2 & 3 & 5 & 6 & 1 & 4 \end{pmatrix}$$

This example illustrates how cpRMT tiles the given matrix with $k = 2$. First, the maximal ranked tile (red color) is discovered. Next, the cells in that region are set to 0. Finally, the second maximal ranked tile (blue color) is returned. The total coverage of the discovered ranked tiling is 18.

Implementation We implemented cpRMT in OscaR [87]. The implementation is available at the following address: `https://github.com/rankmatrixfactorisation/cpRMT`.

4.3.2 Ranked tiling using sRMF (mRMT)

In this section, we study how to use the max-product semiring in combination with the sRMF framework for discovering ranked tilings. This results in a problem instance of sRMF called *Max-product Semiring Rank Matrix Tiling* or *mRMT*. It is formally defined as followed.

Definition 4.3 (mRMT problem)**.** *The mRMT problem is the rank matrix factorisation problem obtained using*

- *the max-product semiring;*
- *the set of permissible values $\sigma_p = \{0, 1\}$;*
- *the additive scoring function based on formula (4.11).*

Compared to the maximal ranked tile mining problem defined in Definition 4.2, mRMT does not have the constraint that requires k to be equal to 1. That is, mRMT finds a ranked tiling consisting of $k \geq 1$ ranked tiles.

Example 4.5 shows the result obtained by mRMT using the same rank matrix in Example 4.4. Compared to the result produced by cpRMT in Example 4.4, the one by mRMT has higher coverage (19 compared to 18) and contains two overlapping ranked tiles.

Example 4.5 (mRMT example)**.**

$$\boldsymbol{M} = \begin{pmatrix} 1 & 2 & 5 & 4 & 6 & 3 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 3 & 2 & 4 & 5 & 6 \\ 2 & 3 & 5 & 6 & 4 & 1 \\ 2 & 3 & 5 & 6 & 1 & 4 \end{pmatrix} \boldsymbol{C} = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} \boldsymbol{F} = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

This example shows the result when we apply mRMT to Example 4.3 with $k = 2$ and $\theta = 3$. The overlapping region of the two discovered ranked tiles is painted green. The total coverage of the discovered ranked tiling is 19.

4.3.3 Computational complexity of the mRMT problem

We will prove mRMT is NP-complete by reduction, i.e., showing that the BMF [82] problem, which is known NP-complete, can be reduced to the mRMT problem.

Problem 4.1 (BMF problem [82])**.** *Given a m-by-n Boolean matrix $\boldsymbol{M}$ and an integer k, find an m-by-k Boolean matrix $\boldsymbol{B}$ and a k-by-n Boolean matrix $\boldsymbol{X}$ that minimises:*

$$\sum_{r=1}^{m} \sum_{c=1}^{n} |\boldsymbol{M}_{r,c} - (\boldsymbol{B} \circ \boldsymbol{X})_{r,c}|, \tag{4.12}$$

where $\circ$ is the Boolean matrix product.

Consider the case that $k = 1$. Let $R = \{r|\mathbf{B}_{r,1} = 1\}$ and $C = \{c|\mathbf{X}_{1,c} = 1\}$, the BMF problem becomes the problem of discovering a noisy tile whose rows are indexed by R and columns are indexed by C. Therefore, Equation 4.12 is equivalent to the following:

$$argmin_{R,C} \sum_{(r,c)\in(R,C)} |\mathbf{M}_{r,c} - 1| + \sum_{(r,c)\notin(R,C)} |\mathbf{M}_{r,c}| \tag{4.13}$$

$$\leftrightarrow argmin_{R,C} \sum_{(r,c)\in(R,C)} [\mathbf{M}_{r,c} = 0] + \sum_{(r,c)\notin(R,C)} [\mathbf{M}_{r,c}] \tag{4.14}$$

where $[f] = 1$ if $f = true$; otherwise 0.

Equation (4.14) implies that BMF aims at finding a tile which minimises the number of 0 within the tile and minimises the number of 1 outside the tile. This problem is known to be NP-complete [82].

Now we show that BMF can be reduced to mRMT.

Theorem 4.3. *BMF can be reduced to mRMT.*

Proof. We consider a rank matrix M, which has only two rank values: 0 for "dislike" and 1 for "like". In other words, M is a Boolean matrix. Using the threshold $\theta = 0.5$ and $k = 1$, the mRMT problem, defined in Problem 4.3, is equivalent to the following optimisation problem:

$$\operatorname*{argmax}_{R,C} \sum_{(r,c)\in(R,C)} (M_{r,c} - 0.5) \tag{4.15}$$

$$\leftrightarrow \quad \operatorname*{argmax}_{R,C} \sum_{(r,c)\in(R,C)} 0.5([M_{r,c} = 1] - [M_{r,c} = 0]) \tag{4.16}$$

$$\leftrightarrow \quad \operatorname*{argmin}_{R,C} \Big(\sum_{(r,c)\in(R,C)} [M_{r,c} = 0] \Big) - \Big(\sum_{(r,c)\in(R,C)} [M_{r,c} = 1] \Big) \tag{4.17}$$

$$\leftrightarrow \quad \operatorname*{argmin}_{R,C} \Big(\sum_{(r,c)\in(R,C)} [M_{r,c} = 0] \Big) + \Big(\sum_{(r,c)\notin(R,C)} [M_{r,c} = 1] \Big) \tag{4.18}$$

Equation (4.18) is the same as Equation (4.14), which shows that BMF can be reduced to mRMT. □

4.3.4 mRMT factorisation is not unique

mRMT factorisation is not unique, which can be seen using the following example.

Example 4.6 (Uniqueness of the mRMT factorisation)**.** *Given the following rank matrix M, the threshold $\theta = 3$ and $k = 2$, mRMF can return two ways of factorising the matrix, each of which has the same scoring value, i.e., 12. Hence mRMT factorisation is not unique.*

$$M = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 4 & 3 & 5 \\ 3 & 2 & 1 & 4 & 5 \\ 1 & 3 & 2 & 4 & 5 \end{pmatrix}$$

Factorisation 1:

$$C_1 = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} F_1 = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Factorisation 2:

$$C_2 = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} F_2 = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

4.3.5 Solving mRMT using Integer Programming

The following theorem establishes an equivalent model for the mRMT problem defined in Definition 4.3.

Theorem 4.4 (IP model for mRMT)**.** *Solutions to the following optimisation model are solutions to the mRMT problem:*

$$\textit{maximize} \sum_r \sum_c (\boldsymbol{M}_{r,c} - \theta) \boldsymbol{A}_{r,c} \tag{4.19}$$

subject to

$$\boldsymbol{A} \in \{0,1\}^{m \times n} \tag{4.20}$$

$$\boldsymbol{C} \in \{0,1\}^{m \times k} \tag{4.21}$$

$$\boldsymbol{F} \in \{0,1\}^{k \times n} \tag{4.22}$$

$$\forall r = 1 \ldots m, \forall c = 1 \ldots n : \boldsymbol{A}_{r,c} \leq \sum_{t=1}^{k} \boldsymbol{C}_{r,t} \boldsymbol{F}_{t,c} \tag{4.23}$$

$$\forall r = 1 \ldots m, \forall c = 1 \ldots n : n\boldsymbol{A}_{r,c} \geq \sum_{t=1}^{k} \boldsymbol{C}_{r,t} \boldsymbol{F}_{t,c} \tag{4.24}$$

Here, $\boldsymbol{M}$ is the given data matrix; variables $\boldsymbol{A}$, $\boldsymbol{C}$ and $\boldsymbol{F}$ are to be found.

Proof. Let $(\mathbf{C}^*, \mathbf{F}^*)$ be the optimal solution for mRMT. Using the definition of the mRMT probem (Definition 4.3), which constrains the set of permissible values $\sigma_p = \{0, 1\}$, we have:

$$(\mathbf{C}^*, \mathbf{F}^*) \equiv \operatorname*{argmax}_{\mathbf{C},\mathbf{F}} \sum_{r,c} \delta(\mathbf{M}_{r,c}, (\mathbf{C} \odot_m \mathbf{F})_{r,c}) \tag{4.25}$$

subject to

$$\mathbf{C} \in \{0,1\}^{m \times k} \tag{4.26}$$

$$\mathbf{F} \in \{0,1\}^{k \times n} \tag{4.27}$$

Replace the $\delta(,)$ function by Equation (4.11), we obtain the following equivalent model:

$$(\mathbf{C}^*, \mathbf{F}^*) \equiv \operatorname*{argmax}_{\mathbf{C},\mathbf{F}} \sum_{r,c} \mathbf{A}_{r,c}(\mathbf{M}_{r,c} - \theta(\mathbf{C} \odot_m \mathbf{F})_{r,c}) \tag{4.28}$$

subject to

$$\mathbf{C} \in \{0,1\}^{m \times k} \tag{4.29}$$

$$\mathbf{F} \in \{0,1\}^{k \times n} \tag{4.30}$$

where $\mathbf{A}$ is the cover matrix, i.e., $\mathbf{A}_{r,c} = 1 \leftrightarrow (\mathbf{C} \odot_m \mathbf{F})_{r,c} > 0$; otherwise $\mathbf{A}_{r,c} = 0$. In other words, $\mathbf{A} \in \{0,1\}^{m \times n}$.

As $\mathbf{A}$ is the cover matrix, the optimisation model can be re-written as follows:

$$(\mathbf{C}^*, \mathbf{F}^*) \equiv \operatorname*{argmax}_{\mathbf{C},\mathbf{F}} \sum_{r,c} \mathbf{A}_{r,c}(\mathbf{M}_{r,c} - \theta(\mathbf{C} \odot_m \mathbf{F})_{r,c}) \tag{4.31}$$

subject to

$$\mathbf{C} \in \{0,1\}^{m \times k} \tag{4.32}$$

$$\mathbf{F} \in \{0,1\}^{k \times n} \tag{4.33}$$

$$\mathbf{A} \in \{0,1\}^{m \times n} \tag{4.34}$$

$$\mathbf{A}_{r,c} \leq \sum_t \mathbf{C}_{r,t}\mathbf{F}_{t,c} \tag{4.35}$$

$$n\mathbf{A}_{r,c} \geq \sum_t \mathbf{C}_{r,t}\mathbf{F}_{t,c} \tag{4.36}$$

Note that $\odot_m$ is the matrix product based on the max-product semiring and $\mathbf{C}$ and $\mathbf{F}$ are Boolean matrices. Hence, the maximum value of $(\mathbf{C} \odot_m \mathbf{F})_{r,c}$ is 1. This results in a simpler model:

$$(\mathbf{C}^*, \mathbf{F}^*) \equiv \sum_{r,c} (\mathbf{M}_{r,c} - \theta)\mathbf{A}_{r,c} \tag{4.37}$$

subject to

$$\mathbf{C} \in \{0,1\}^{m \times k} \tag{4.38}$$

$$\mathbf{F} \in \{0,1\}^{k \times n} \tag{4.39}$$

$$\mathbf{A} \in \{0,1\}^{m \times n} \tag{4.40}$$

$$\mathbf{A}_{r,c} \leq \sum_t \mathbf{C}_{r,t}\mathbf{F}_{t,c} \tag{4.41}$$

$$n\mathbf{A}_{r,c} \geq \sum_t \mathbf{C}_{r,t}\mathbf{F}_{t,c} \tag{4.42}$$

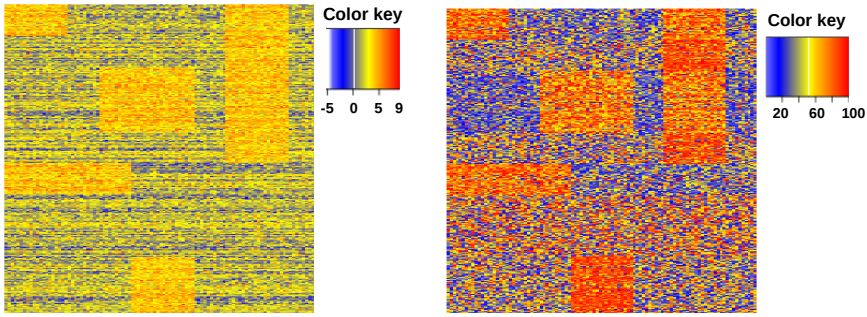
which concludes the proof. □

Compared to Sparse mRMF (Section 3.3), this model can be solved more efficiently as it contains a much smaller number of variables.

To solve the mRMT problem defined in Equations (4.19) –(4.24), we apply the same EM-style algorithm that we use for Sparse pRMF and Sparse mRMF. That is, we first initialise matrix $\mathbf{C}$ using $k-means$ clustering algorithm. Then, matrix $\mathbf{F}$ is optimised given matrix $\mathbf{C}$, and matrix $\mathbf{C}$ is optimised given matrix $\mathbf{F}$, and we repeat the iterative optimisation untill the optimal score cannot be improved any more.

Similarly to Sparse mRMF, in mRMT, each row of matrix $\mathbf{C}$ can be determined independently when matrix $\mathbf{F}$ is given; each column of matrix $\mathbf{F}$ can also determined independently when matrix $\mathbf{C}$ is given. Consequently, we can develop a parallel implementation to solve the optimisation problem for mRMT by distributing it over multiple cores.

Implementation To implement our system, we also relied on the OscaR system [87] as we did for Sparse pRMF and Sparse mRMF. The implementation is available at the following address: `https://github.com/rankmatrixfactorisation/mRMT`.



(a) Numerical data (1000×100) (b) Rank matrix (1000×100)

Figure 4.2: Recovering five implanted ranked tiles from the first set of synthetic data. Figure 4.6a shows the part of the matrix covered by mRMT ($k = 5, \theta = 60\%$). Figure 4.6b shows the reconstructed matrix obtained by Sparse mRMF ($k = 5, \alpha = 20\%$).

4.4 Experiments with synthetic data

We experiment on two sets of synthetic datasets to 1) evaluate the algorithms and 2) compare the patterns that the proposed algorithms for ranked tiling and Sparse RMF can discover. The first set of data is synthetic data with implanted ranked tiles, i.e., the absolute ranks rather than the orders of the columns of the implanted tiles matter. It is mainly used to demonstrate the capabilities of the ranked tiling algorithms, including its suitability for data having incomparable rows. We will use the second set of data to study the behaviours of the proposed algorithms on the data with implanted order-preserving patterns.

4.4.1 Synthetic data with implanted ranked tiles

For the first set of experiments we use synthetic data with incomparable rows, i.e., rows having different scales, to show that the ranked tiling algorithms (cpRMT and mRMT) find the relevant patterns in such data while bi-clustering methods do not. Since bi-clustering methods work on numeric data, we use a simple generative model to generate continuous data. This numeric data is then transformed to a rank matrix to apply mRMT. For bi-clustering, we choose the constant-row setting, as there are many bi-clustering algorithms designed for this type of pattern and it is conceptually close to mRMT.

Data generation [69]. To generate synthetic datasets, we first generate background data, and then implant a number of constant-row bi-clusters with higher average values.

The values within each row are sampled, with a certain probability, from one of two distributions: one that represents background noise and one that is likely to interfere with the implanted patterns. First, for each row r, we uniformly sample μ_r^1, μ_r^2 from two ranges:

$$\mu_r^1 \sim U(0,3), \forall r \in \mathcal{R} \tag{4.43}$$

$$\mu_r^2 \sim U(3,5), \forall r \in \mathcal{R} \tag{4.44}$$

Second, for every cell in a row, indicated by row r and column c, we sample a latent binary variable $X_{r,c}$ from a Bernoulli distribution $Bin(p, 1-p)$, given some p. Depending on the value of this latent variable, the data is sampled from either the low-average or high-average distribution:

$$\mathcal{D}_{r,c} \sim \begin{cases} N(\mu_r^1, 1) & \text{if } X_{r,c} = 1 \\ N(\mu_r^2, 1) & \text{otherwise} \end{cases} \tag{4.45}$$

To plant a constant-row bi-cluster in a submatrix $\mathcal{D}_{R,C}$, specified by R and C, we use the following two equations:

$$\forall r \in R, \quad \mu_r \sim U(3,5) \tag{4.46}$$

$$\forall r \in R, \quad \mathcal{D}_{r,c} \sim N(\mu_r, 1) \tag{4.47}$$

Equation 4.46 is used to sample a mean for every row in a bi-cluster. This mean is uniformly sampled from the range $[3 \dots 5]$, which is higher than the sampling range used for the background ($[0 \dots 3]$).

Using this procedure we generated seven 1000 rows $\times$ 100 columns datasets, one for each $p \in \{0.10, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40\}$. We implanted five ranked tiles in each dataset. Figure 4.2a depicts the numerical dataset for $p = 10\%$, Figure 4.2b depicts its corresponding rank matrix.

Performance gain by adding redundant constraints for cpRMT. To evaluate how the redundant constraints affect the time needed for search, we compare the runtime of the optimisation model in Equation 4.1, which does not have any constraints, to the runtime of the constraint-based model shown in Equations (4.8)–(4.10). To make the comparison fair, we perform exhaustive search in both cases.

To this aim, we generate a number of datasets with varying sizes using the procedure described above. All datasets have the same number of columns, i.e. 20, and a varying number of rows: {100, 200, 300, 400, 500, 600}, with $p = 0.1$. They have the same two non-overlapping tiles: one is 10×5 and the other is 30×10. We execute the mentioned models on these datasets to find

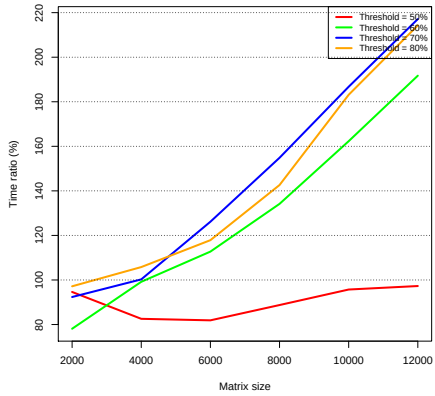


Figure 4.3: cpRMT reduces the solving time by adding the two redundant constraints. The y-axis indicates the ratio between the runtime needed with and without the constraints, and is computed for varying threshold θ values and matrix sizes.

the maximal ranked tile. All experiments are executed single threaded on a desktop computer (Intel i7-2600 CPU @ 3.40GHz, 16GB RAM).

Figure 4.3 shows the ratio between the time needed to solve the problem with and without the extra constraints, for varying θ thresholds. We can see that adding the extra constraints reduces the search time when $\theta > 50\%$. In particular for larger datasets and values of the threshold, the model with the added constraints always outperforms the optimisation-only model. This demonstrates that adding the constraints results in better propagation and hence more efficient search. The absolute time needed to find the optimal solution on these datasets ranges from 1ms to 4h 37m 26s.

Accuracy of the tiles found by cpRMT. We now evaluate the ability of the algorithm to recover the implanted ranked tiles. We do this by measuring *recall* and *precision*, using the implanted tiles as ground truth. Overall performance is quantified by the *F1* measure, which is the average of the two scores.

We varied the threshold θ; for each value of θ and each dataset, we performed ranked tiling five times, each time mining $k = 5$ tiles. Then, we calculated average precision, recall and F1 score over these five runs. Figure 4.4 summarises the results obtained with and without using the variable ordering heuristic (see Section 4.2.2). It can be seen that the heuristic contributes to improved performance. When the threshold θ is around 60%, the algorithm achieves

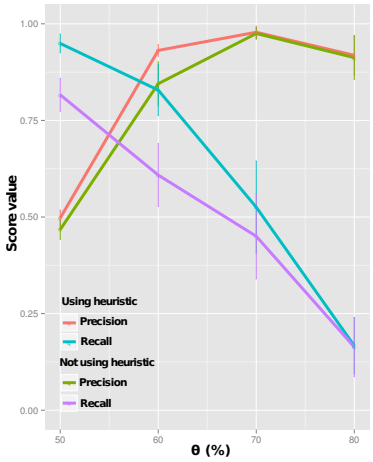


Figure 4.4: Sensitivity analysis for cpRMT on the data with implanted tiles.

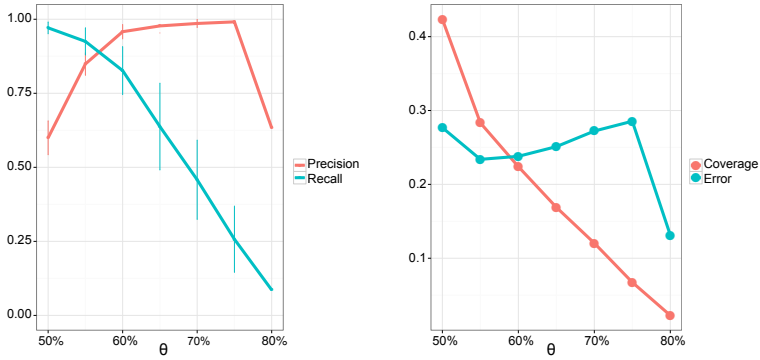
high prediction accuracy (average F1 = 86%). At lower thresholds, it has low precision, while higher thresholds result in lower recall. This completely matches our expectation, since higher thresholds result in smaller tiles with higher values.

Accuracy of the tiles found by mRMT. We also evaluate mRMT by measuring the precision and recall scores reflecting how it can recover the implanted tiles as we do for cpRMT.

We varied the threshold θ and ran mRMT to factorise the rank matrix ($k = 5$) 10 times for each combination of θ and dataset. Then, the result that had the highest score was used to calculate the average precision, recall and F1 score over these combinations. Figure 4.5a summarises the results. When θ is around 60%, the algorithm achieves high accuracy (average F1 = 88%). At lower thresholds precision is low, while higher thresholds result in lower recall. This matches our expectation, as higher thresholds result in smaller tiles with higher values; this is also shown in Figure 4.5b, where the higher thresholds result in lower coverage (thus lower recall).

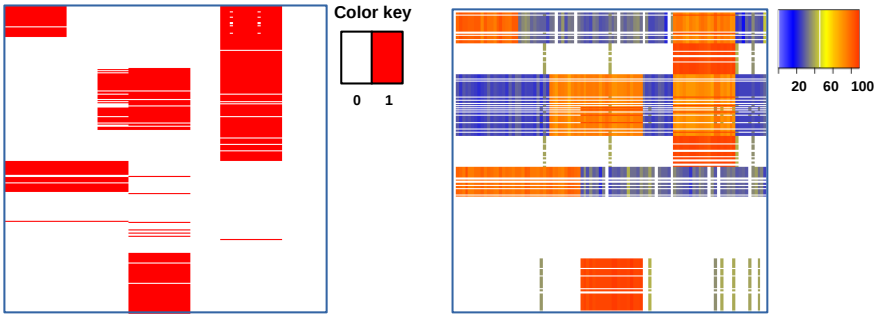
Table 4.1 shows that the results obtained by mRMT are comparable to those obtained by cpRMT. This demonstrates that mRMT, the approach developed later, behaves properly.

Comparison to Sparse mRMF. To contrast the ranked tiling and Sparse RMF patterns that both mRMT and Sparse mRMF can discover, we also ran Sparse mRMF on the generated datasets. Figure 4.6b shows the reconstructed



(a) Precision and recall (b) Coverage and error

Figure 4.5: Sensitivity analysis for mRMT on the synthetic data with implanted tiles.



(a) mRMT (b) Sparse mRMF

Figure 4.6: Recovering five implanted ranked tiles from the first set of synthetic data. Figure 4.6a shows the part of the matrix covered by mRMT ($k = 5, \theta = 60\%$). Figure 4.6b shows the reconstructed matrix obtained by Sparse mRMF ($k = 5, \alpha = 20\%$).

matrix produced by Sparse mRMF ($k = 5, \alpha = 20\%$) on the rank matrix generated with $p = 0.2$. It can be seen that the result produced by Sparse mRMF includes regions having low rank values (indicated in blue) instead of only focusing on the regions having high ranks (in red). As a result, its recall and precision are relatively low, which can also be seen in Figure 4.7. This confirms that the two algorithms discover different types of rank patterns.

Comparison to bi-clustering. In the next experiment, we compare our approach to several bi-clustering algorithms. SAMBA [114] was designed for

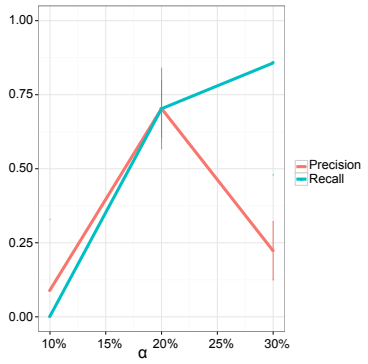


Figure 4.7: Sensitivity analysis for cpRMT on the synthetic data with implanted tiles.

coherent evolution bi-clusters, in which there is coherence of the signs of values, i.e., up or down. The other methods discover coherent-valued bi-clusters, of which a constant-row bi-clusters are a special case. CC [15], Spectral [62], and Plaid [122] are available in the R biclust[2] package. FABIA[3] [48] and SAMBA[4] were downloaded from their respective websites. ISA [52] is from the R isa2 package[5].

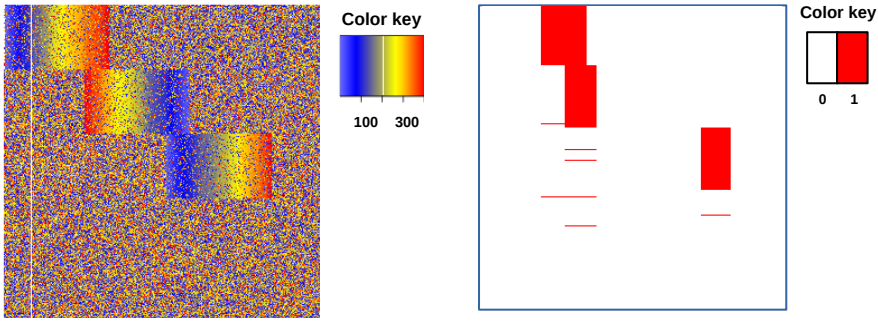
Since large noise levels make the recovery task hard for any algorithm, we use one of the previously generated datasets with average noise level, i.e., $p = 0.20$. We ran all algorithms on this dataset and took the first five tiles/bi-clusters they produced. For most of the benchmarked algorithms, we used their default parameter values. For CoreNode, we used $msr = 1.0$ and $overlap = 0.5$, as preliminary experiments showed that this combination produced the best result. For ISA, we applied its built-in normalisation method before running the algorithm.

The results in Table 4.1 show that our algorithm achieves much higher precision and recall than the bi-clustering methods, which were run on the original data. This indicates that when the rows in a numerical matrix are incomparable, converting the data to a ranked matrix and applying mRMT is a better solution than applying bi-clustering.

[2] `http://cran.r-project.org/web/packages/biclust/`
[3] `http://www.bioinf.jku.at/software/fabia/fabia.html`
[4] `http://acgt.cs.tau.ac.il/expander/`
[5] `http://cran.r-project.org/web/packages/isa2/`

Table 4.1: Comparison of cpRMT, mRMT, Sparse mRMF, and bi-clustering methods. Precision, recall and F1 quantify how accurately the methods recover the 5 implanted tiles. $k = 5$.

Algorithm	Data type	Pattern	Precision	Recall	F1
mRMT	Ranks	Ranked tile	**95%**	81%	**88%**
cpRMT	Ranks	Ranked tile	88%	**83%**	86%
Sparse mRMF	Ranks	Sparse rank profile	70%	70%	70%
CoreNode [121]	Numerical	Coherent values bicluster	43%	72%	58%
FABIA [48]	Numerical	Coherent values bicluster	40%	24%	32%
Plaid [122]	Numerical	Coherent values bicluster	90%	6%	48%
SAMBA [114]	Numerical	Coherent evolution bicluster	67%	3%	35%
ISA [52]	Numerical	Coherent values bicluster	64%	44%	54%
CC [15]	Numerical	Coherent values bicluster	35%	22%	29%
Spectral [62]	Numerical	Coherent values bicluster	-	-	-



(a) Rank data (1000×400) (b) mRMT

Figure 4.8: Recovering three implanted rank profiles from the synthetic data with implanted orders. Figure 4.8a shows the data ($w = 20$). Figure 4.8b is obtained by mRMT with $k = 3, \theta = 60\%$.

4.4.2 Synthetic data with implanted orders

In the second set of experiments, we evaluate the capability of mRMT to recover consistent rankings of a subset of columns in a subset of rows. Hence, for this we use the ranked data with implanted orders that we described in Section 3.7.

Accuracy of the recovered tiles by mRMT. Figure 4.8b displays the regions covered by the best result produced by mRMT (on the synthetic data shown in Figure 4.8a). It can be seen that these regions contain the high ranks of the implanted patterns. In other words, mRMT only partially recovers the

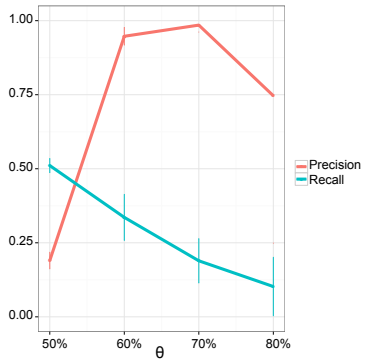


Figure 4.9: Sensitivity analysis for mRMT on the synthetic data with implanted orders.

reference rank profiles, which explains the low recall in Figure 4.9. This is in contrast to the result obtained by Sparse mRMF (Figure 3.1b on page 39), where the implanted rank profiles could be fully recovered. Hence, this experiment again confirms our expectations that Sparse mRMF and mRMT were designed for different types of rank patterns.

Comparision to tiling. We did not compare mRMT/cpRMT with tiling [37] in the experiments with the synthetic datasets discussed above. In general, we can binarise a rank matrix and transform it into a Boolean matrix using the user-defined threshold value θ, i.e., cell values that are greater than θ become 1; otherwise 0. However, the tiling model proposed in [37] is *not fault-tolerant* with noise. That is, it requires each tile to be full of 1s. In constrast, our ranked tiling is *tolerant* with noise. In other words, a rank tile can have cells of which values are less than θ as long as the average values of the rows/columns of the tile are greater than θ.

4.5 Real world case studies

In this section we report on three real world case studies concerning the European Song Festival, breast cancer subtypes, and sushi consumption.

4.5.1 European Song Festival dataset

We used the European Song Contest (ESC) dataset that was collected for the period 2010–2013 and described in Section 3.8.1.

We applied mRMT on the ESC dataset with varying threshold values and then analysed the obtained coverage and error per covered cells to empirically select the best thresholds.

We use the same parameter selection procedure as for Sparse mRMF (see Section 3.8.1. Figure 4.10a depicts average coverage and error scores obtained by mRMT with varying θ values. We choose $\theta = 80\%$ as both coverage and error decrease slowly beyond that point. Similarly, we choose $k = 10$ based on Figure 4.10b.

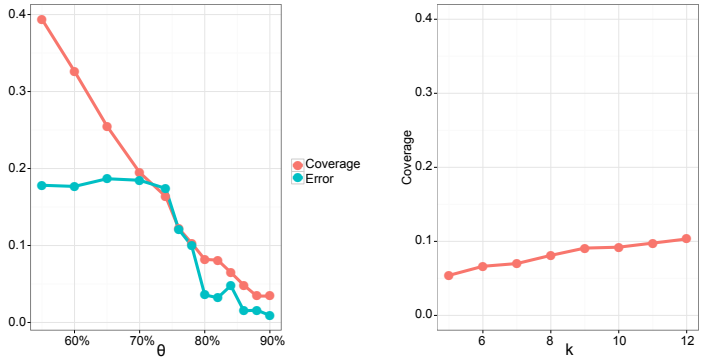
The heatmap of the covered matrix by mRMT are shown in Figures 4.11b. Compared to the original rank matrix in Figure 4.11a, the heatmap shows that the method strongly *sparsified* the original rank matrix. Compared to the rank profiles produced by Sparse mRMF (Figure 3.5), which contains both high and low ranks, the ones produced by mRMT only indicate the places where high ranks appear, as expected.

We visualise two rank profiles produced by mRMT in Figure 4.12. They show the typical voting behaviour of Western European countries towards Nordic countries (Figure 4.12b), and that of Eastern European countries toward some other countries (Figure 4.12a). For example, countries in Eastern Europe tend to give higher scores to Russia and Nordic countries than to other countries.

We do not present a comparison for mRMT and cpRMT in this case study as we can see from the experiments in Section 4.4.1 that the two methods typically have similar behaviours on small datasets. In the next sub-section, where we will study a much larger dataset, we will compare mRMT with cpRMT.

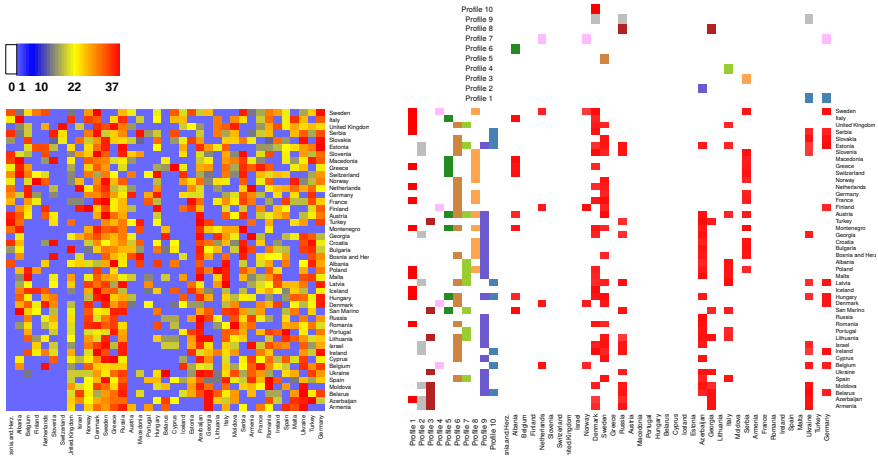
4.5.2 Discovering breast cancer subtypes

Breast cancer is known to be a heterogeneous disease that can be categorised in clinical and molecular subtypes [117]. Assignment of patients to such subtypes is crucial to give adapted treatments to patients. The increasing availability of tumor related molecular data provides the opportunity to look at the molecular mechanisms that drive carcinogenesis at different angles. Computational models have been proposed to integrate multiple data types and discover cancer subtypes [84, 130], but these integrative subtyping methods do not explicitly extract the subtype-specific features. The goal of this case study is to demonstrate that:



(a) mRMT (b) mRMT

Figure 4.10: Parameter tuning for mRMT on the European Song Festival dataset.

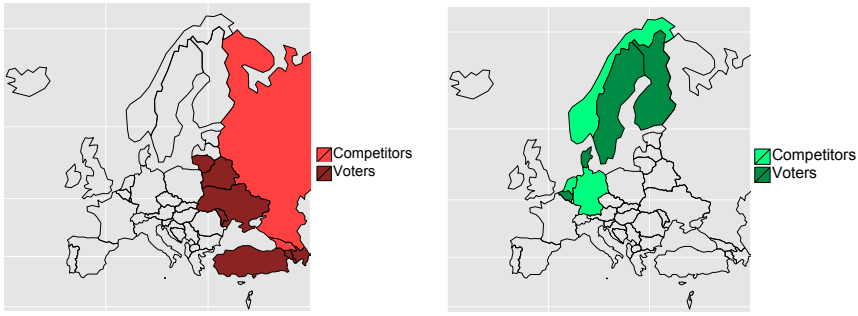


(a) Rank data (b) The part of the data covered by mRMT

Figure 4.11: EU Song Festival results show how mRMT focuses on specific structure and hence sparsifies the data. In Figure 4.11b, *"red"* is 1 and *"white"* is 0.

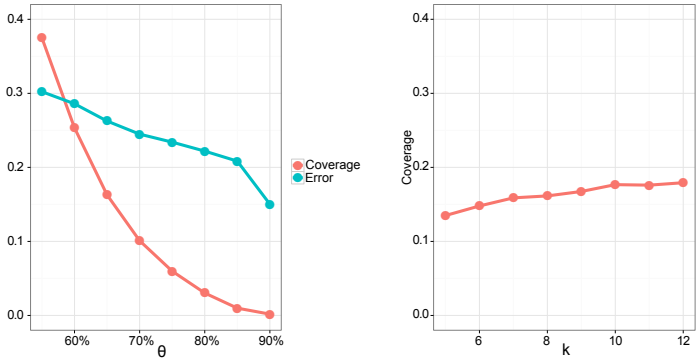
1) we can integrate multiple data types that are inherently incomparable but can be compared when transformed to ranked data; 2) we can simultaneously discover breast cancer subtypes and their subtype-specific features.

The case study we present here concerns a simplified setting of the one presented in the next chapter. We here only consider a single, integrated ranked matrix and focus on evaluating mRMT.



(a) Tile 8 (mRMT) (b) Tile 7 (mRMT)

Figure 4.12: Rank patterns discovered on the ESC dataset by mRMT. Voting countries are painted in dark colors and competing countries are painted in light colors.



(a) (b) $\theta = 65\%$

Figure 4.13: Parameter tuning on breast cancer dataset.

Data pre-processing. We use the well-studied TCGA breast cancer dataset [117], which provides the following four data types for the same set of samples (patients): mRNA measured by microarray technology, microRNA measured by RNA-Seq, proteins, and copy number variations (CNV).

We first selected all the tumour samples that have measurements at the four molecular levels, which resulted in 363 samples. Second, we filtered mRNA and microRNAs as in our previous study [69]. That is, we selected genes based on their differential expression relative to normal (non-tumour) samples. For each gene, a normal distribution was fitted using the normal expression samples, and z-scores were calculated for the tumour samples. We evaluated the 5- and

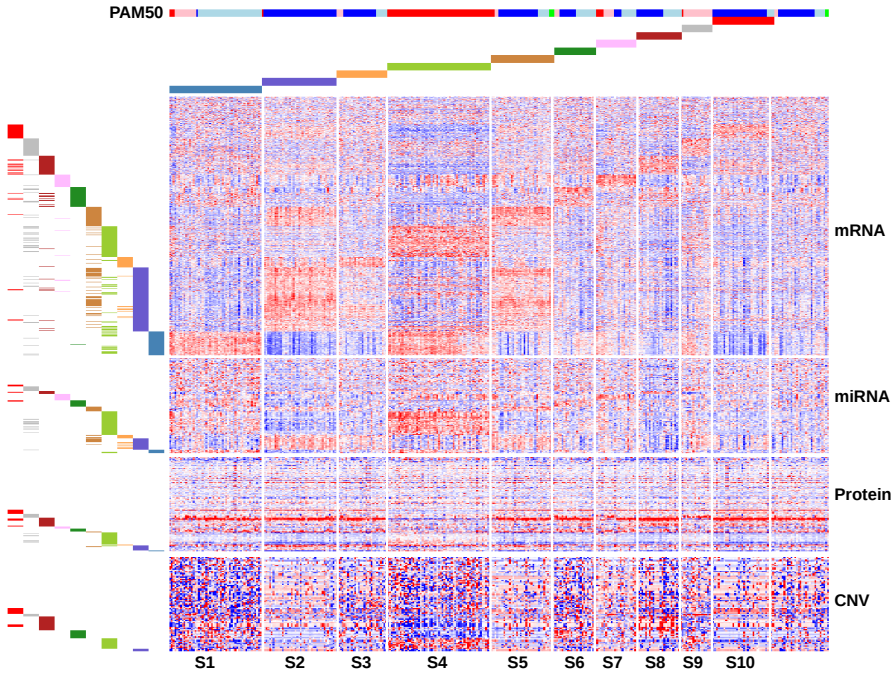


Figure 4.14: sRMT on heterogeneous breast cancer data. The rows correspond to mRNA, miRNA, protein and CNV levels, the columns correspond to breast cancer samples. High expression numeric values are represented by red, low numeric expression by blue. The left and upper color bars indicate the tiles. PAM50 subtypes are indicated at the top (LuminalA=blue, LuminalB=light blue, Basal=red and HER2=pink).

95-percentile of the tumour samples. Genes in which the p-value for these percentiles was below 0.001 and their log-fold change relative to the mean normal expression was at least 2.5 were selected. The filtering step resulted in 1761 mRNAs out of 17814 mRNAs and 138 microRNAs out of 1222 microRNAs. Third, we used all the protein data (131 proteins), which were post-processed by the UCSC genome browser [40]. Finally, copy number regions (82 in total) were identified with GISTIC tool [81], of which the analysis result was provided together with the TCGA paper [117]. Finally, each data level was converted to ranks and combined into a single rank matrix consisting of 2112 rows and 363 columns.

Running experiments. As it is our aim to discover cancer subtypes consisting of a number of tumor samples having consistently similar expression patterns,

Sparse mRMF is not suitable for this type of application. Hence, for this case study we restrict ourselves to mRMT.

We ran the parallel implementation of the mRMT method on the TCGA breast cancer dataset with varying values for the threshold θ and k, i.e., $\theta \in \{55\%, 60\%, \ldots, 90\%\}$ and $k \in \{5, \ldots, 16\}$. For each combination of the two parameters, we ran the algorithm 100 times and took the best result. Figure 4.13a shows the *coverage* and *error* scores obtained by the algorithm with varying θ, from which we can infer that there is no clear cut to choose θ in this case. In general, the higher the threshold value, the lower the error and the coverage. To trade off the coverage and the error, we chose $\theta = 65\%$. Given the selected θ, we next had to decide the value for k. We plotted the coverage score w.r.t k (Figure 4.13b) and then decided to stop at $k = 10$ as we found the coverage score to increase only very slowly after that point.

With $\theta = 65\%$ and $k = 10$, the average running time for one run is 432s on a desktop computer (Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz, 8 threads, 16GB RAM). With the chosen parameter values, the algorithm produces 10 overlapping ranked tiles. Though the overlap structure can be useful to study the similarity among the discovered subtypes, for practical reasons we decided to choose a simple interpretation in which each sample is assigned to a single subtype. With this aim, we developed a post-processing step in which each sample belonging to multiple subtypes according to the mRMT method is assigned to the subtype giving the highest rank score in Equation (4.1). Figure 5.4 shows the result obtained using this procedure.

Subtype analysis. First, we observe that most discovered subtypes comprise all four types of features. The exceptions are subtypes S1, S3, S5 and S7, which have mRNA, miRNA and protein features but lack CNVs. Further, Figure 5.4 shows that the selected features of the identified subtypes are typically "*red*", i.e., highly over-expressed, as one would expect when using mRMT.

Next, we test to what extent the discovered subtypes agree with known clinical information. In particular, we compare to the PAM50 annotation [89], which classifies breast cancer patients into four subtypes, Luminal A, Luminal B, Basal and Her2, using the expression of 50 mRNAs. Figure 4.15 shows that our approach does not only match the PAM50 classification to a large extent, it also further refines known subtypes. For example, our approach recovered the Basal subtype in subtype S4; recovered the Her2 subtype in subtype S9 and sub-divided the Lumial A subtype into four smaller groups, namely, subtype S2, S3, S5 and S10.

Comparison with cpRMT. We next compare mRMT with cpRMT. For a fair comparison, we ran cpRMT 100 times, repeating large neighbourhood

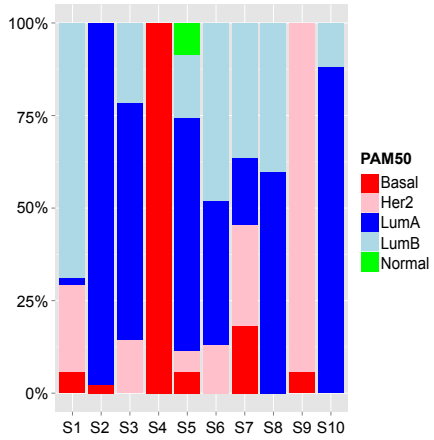


Figure 4.15: Percentages of PAM50 samples in the subtypes discovered in the breast cancer data.

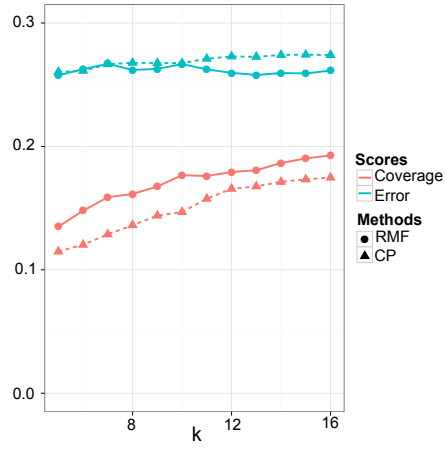


Figure 4.16: Comparing mRMT with cpRMT on the breast cancer dataset. In both cases, θ was set to 65%.

search (LNS) as many times as repeating mRMT procedure. Figure 4.16 shows that mRMT obtains higher coverage and lower error scores than cpRMT. This matches with our expectation as mRMT implementation employs a *global optimisation* procedure, whereas cpRMT uses a *greedy approach*, i.e., it finds one 'maximal' ranked tile, removes it, and proceeds to find the next one.

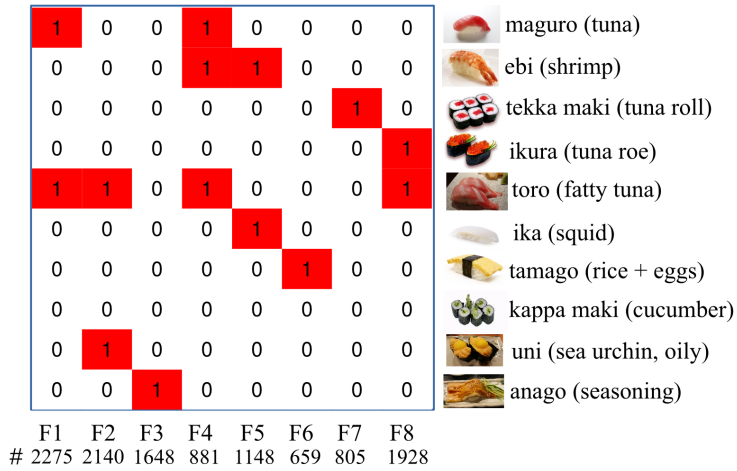


Figure 4.17: Sushi preference patterns found by mRMT. The bottom row below the heatmap, indicated by #, shows the number of users having the corresponding rank profile.

Overall, we conclude that mRMT can identify cancer subtypes and their features from several data types by searching for patterns in ranked data.

4.5.3 The Sushi dataset

We used the Sushi dataset [54] described in Section 3.8.2 to mainly demonstrate the patterns produced by mRMT and Sparse mRMF.

We ran mRMT on this dataset $\theta = 65\%$ to select subjectively high rank values and $k = 8$ as in Sparse mRMF (Section 3.8.2). Figure 4.17 shows the eight rank profiles found by mRMT. In general, we also observe similar rank patterns (but only in the high ranks) as we do for Sparse mRMF (Figure 3.6). For example, we also see there is a group of customers highly preferring light sushi types, such as ebi, and ika, as shown in the rank profile F5. In constrast, the third rank profile F3 maintains that there exists a group of customers who have completely an opposite taste, preferring anago, which is a type of seasoning sushi.

4.6 Discussion

We discuss possible effects when we apply different methods in combination with different semirings and data types for solving the ranked tiling problem.

Method	Semiring	Data	Comments
NMF	Plus-product	Numeric data	Cannot directly discover ranked tiles. By using a post-processing step to threshold on both matrices in the factorisation, ranked tiles can be discovered with datasets having comparable rows; might not be possible with datasets having incomparable rows.
NMF	Plus-product	Rank data	Cannot directly discover ranked tiles. By thresholding on both matrices in the factorisation, ranked tiles can be discovered; might not work with overlapping tiles.
sRMF	Max-product	Rank data	Designed to discover overlapping ranked tiles.
sRMF	Min-product	Rank data	Only discover the overlapping parts of ranked tiles.
sRMF	Plus-product	Rank data	The overlapping parts will be penalised; Discover ranked tiles with less overlapping.

Table 4.2: Comparision of methods in combination with different semirings and data types for solving the ranked tiling problem.

Table 4.2 summarises the comparison.

First, we discuss whether NMF [71] can discover ranked tiles using numeric data and its corresponding rank data version. In principle, NMF cannot directly discover ranked tiles in both numeric data and rank data for the same reason we discussed in Section 3.10 on page 46. However, in some cases, we can apply a post-processing step, which thresholds the values of both factorised matrices in the factorisation, to discover ranked tiles. With numeric data, finding a right threshold value will be certainly challenging when datasets have incomparable rows as the data are multimodal. With rank data, finding a right threshold value is easier. Yet, NMF might not be able to fully deal with overlapping ranked tiles as NMF uses the plus-product semiring (see Sparse pRMF and Spare mRMF in Chapter 3).

Second, our sRMF framework is quite sensitive to the type of semiring used in

this case. For example, using the max-product semiring, the sRMF framework can discover overlapping ranked tiles as expected. However, if we use the min-product semiring, we just obtain the overlapping parts of the ranked tiles.

4.7 Conclusions

In this chapter, we study ranked tiling, which is a set of data regions that have high ranks. Experiments on real data show that this type of pattern is useful and can lead to the discovery novel insights in heterogeneous data.

We study two different approaches to modelling and discovering ranked tilings: cpRMT based on constraint programming and mRMT based on the max-product semiring rank matrix factorisation. mRMT is more scalable and accurate than cpRMT, particularly in large datasets, as mRMT has a parallel implementation and employs a global optimisation procedure.

In the next chapter, we study another setting for ranked tiling, in which the ranked tiles are constrained on multiple views (matrices). This setting helps us to integrate multiple molecular data, including mutation data, gene expression data and biological networks, to simultaneously discover cancer subtypes and subtype specific features.

Chapter 5

Simultaneous Discovery of Cancer Subtypes and Subtype Features using sRMF

This chapter[1] presents the third instantiation of the sRMF framework to mine rank pattern sets in rank data. At a high level, we study the ranked tiling problem in a multi-view setting [111, 134, 130, 84, 93], in which two ranked tiles residing in two different rank matrices are coupled through the same subset of the columns. This problem originates from a study on the development of an integrative model that could use both genotype information, such as mutation data, and molecular phenotype data, such as gene expression data, to cluster tumour samples into biologically meaningful groups called *cancer subtypes*.

Molecular subtypes are defined as groups of samples that have a similar molecular mechanism at the origin of the carcinogenesis. The molecular mechanisms are reflected by subtype-specific mutational and expression features. Data-driven subtyping is a complex problem as subtyping and identifying the molecular mechanisms that drive carcinogenesis are confounded problems. Many current integrative subtyping methods use global mutational and/or expression tumour profiles to group tumour samples in subtypes but do not explicitly extract the subtype-specific features. We therefore present a method that solves both tasks of subtyping and identification of subtype-specific features simultaneously.

[1] Based on the journal paper "Simultaneous discovery of cancer subtype and subtype features by molecular data integration" [67]

5.1 Introduction

As cancer is a heterogeneous disease, subtyping cancer is key to an improved and more personalised prognosis and treatment. With cancer genomes, transcriptomes, and epigenomes becoming increasingly available, one of the major challenges in cancer research is to use these molecular data to define clinically or biologically meaningful subtypes.

Successful seminal research on cancer subtyping aimed at grouping patients based on similarities in their molecular profiles (gene expression) or extracting expression derived features to optimally classify patients according to clinically relevant phenotypes [91, 107, 83, 120].

With the availability of NGS data, charting cancer genomes, transcriptomes and even epigenomes offers the opportunity to refine subtyping by taking into account not only the molecular phenotypes (expression) but also likely driver events (mutations, CNVs, methylations) that are at the origin of the tumorigenesis [126].

Several efforts have been taken to integrate these different molecular data in order to extract relevant subtypes, for instance [138] and [19] relied mainly on combining copy number and expression data to define subtypes, whereas the more generic models of [84], [130] and [109] use next to expression and CNV also mutation and methylation data.

The problem of these early approaches, which aim at clustering samples based on shared CNV and mutational profiles, is that they overlook one of the major properties of tumorigenesis: its clonality. By directly using copy number alterations to discriminate between samples they ignore the fact that CNVs are prevalent in cancerous cells and that many CNVs are passenger events, not involved in driving the phenotype ($> 70\%$) [139, 97]. Using passenger events to group patients might blur the true sample grouping in the data as driving events are rare compared to passenger events.

The same goes for the sample grouping based on shared somatic mutational profiles. Doing this implicitly assumes that true driving somatic mutations are frequent across tumour samples, which is because of the clonality of the carcinogenesis not necessarily true. Because they evolve independently, tumors can trigger the same driver pathways through mutations in different genes. By focusing only on frequent alterations, rare events that are very characteristic for a subtype are ignored. In addition, if similarities between tumour samples are scored using the raw mutation data, results are mainly driven by the dense data, such as copy number and gene expression with a negligible contribution from the mutation data.

The most advanced state-of-the-art integrative methods for cancer analysis do take into account the clonal properties of cancer by searching for mutational consistency at pathway level rather than at the individual gene level. They do so by exploiting the connectivity of mutations occurring across different tumour samples on an interaction network. An interaction network here consists of a comprehensive compilation of all molecular interaction information, available on an organism of interest; the network is represented as a graph in which the nodes correspond to genes and the edges to interactions between the genes. Mutations that are recurrently affecting sets of genes that are closely connected on the interaction network are identified as drivers [72, 125, 21]. [49] successfully applied this strategy to use mutation data for subtyping.

Here we introduce a novel analysis framework that combines CNVs and mutation data with an expression phenotype to identify subtypes while considering mutational consistency at a pathway level. Because identifying subtypes and defining the molecular mechanisms (driver pathways) that drive cancer are confounded (a subtype depends on the molecular mechanism but the molecular mechanisms that one can identify also depends on how patients are grouped), our method performs the two tasks simultaneously.

Our method includes two main steps:

1. Find a set of *driver pathways*, which are assumed to be the molecular origins of the disease. Each driver pathway is a set of mutated genes that are closely connected in a biological network and that are recurrently mutated in a number of samples. In addition, the driver pathway is correlated with the abberant expression of a subset of the expressed genes in those samples. We formally define the driver pathway concept using two ranked tiles, one for the mutated genes of the driver pathway and the other for the abberantly expressed genes, that are coupled through a subset of the tumour samples. For the sake of simple reference, we call the two ranked tiles *rank factor*. The task of finding k driver pathways or rank factors is formulated using the sRMF framework.

2. Derive cancer subtypes from the discovered driver pathways. Each subtype is a group of samples that have a unique combination of driver pathways.

We extensively tested the performance of our method on simulated data. Comparing our method with other state-of-the-arts on the well-studied TCGA breast cancer dataset shows how our method is able to grasp the most prominent signatures in the data that are also retrieved by other methods, but also how it is able to capture subtle differences that are missed by methods that compare samples based on global profiles of similarities.

5.2 The SRF algorithm

In the following we develop an instance of the sRMF framework that can combine three different data types, i.e., transcription data, mutation data, and prior knowledge encoded in a biological interaction network, to discover cancer subtypes. This model links the genomic variants of the subtypes in the genome, i.e., the different causes of cancer [126], with the phenotypes observed in the transcription data.

Combining these data types is particularly challenging because they are in different measurement scales: mutation data is Boolean, transcription data such as microarray and RNASeq data is typically numeric, and biological networks are discrete structures. To be able to jointly analyse this heterogeneous data, we transform them to ranked data. That is, we turn mutation data and the biological network into a rank matrix, in which each column (patient) contains rankings of the potential contributions to the disease of mutated genes w.r.t. a network model. This transformation allows us to search for pathway level consistency across the tumour samples. In this way, not only genes that are mutated will receive high relevance scores, but also genes that are close to the mutated genes in the network. Identifying groups of patients with a consistent mutation profile in this transformed matrix allows searching for mutational consistency at the pathway level (Hofree et al., 2013) and accounts for the clonality of carcinogenesis. Similarly, we turn the transcription data into a second rank matrix. Transforming the expression and mutation matrix to rank matrices is key to removing the scale differences.

A subtype is subsequently defined as a set of tumour samples that share a similar molecular origin of their disease, i.e., a driver pathway where the driver mutations occur. The effect of a mutated driver pathway is assumed to be reflected in the expression phenotype, consistently down- or upregulated compared to the reference, of a subset of the genes downstream in those samples. Hence, selected genes in the expression data and selected mutations in the mutation data of the samples in a subtype can be different.

Detecting a subtype is formalised as a multi-view ranked tiling problem in which one wants to search for a subset of patients that share both a similar set of driver mutations and a subset of consistently differentially expressed genes; given that the clonal phenotypes in cancer are affected in the same driver pathways, this assumption is reasonable. This clustering problem is solved by applying the sRMF framework (Chapter 2) to jointly factorise the ranked mutation and expression matrices into a number of *ranked factors*, each of which consists of two ranked tiles (Chapter 4). Conceptually, each resulting factor consists of a subset of samples associated to a subset of expression and mutation

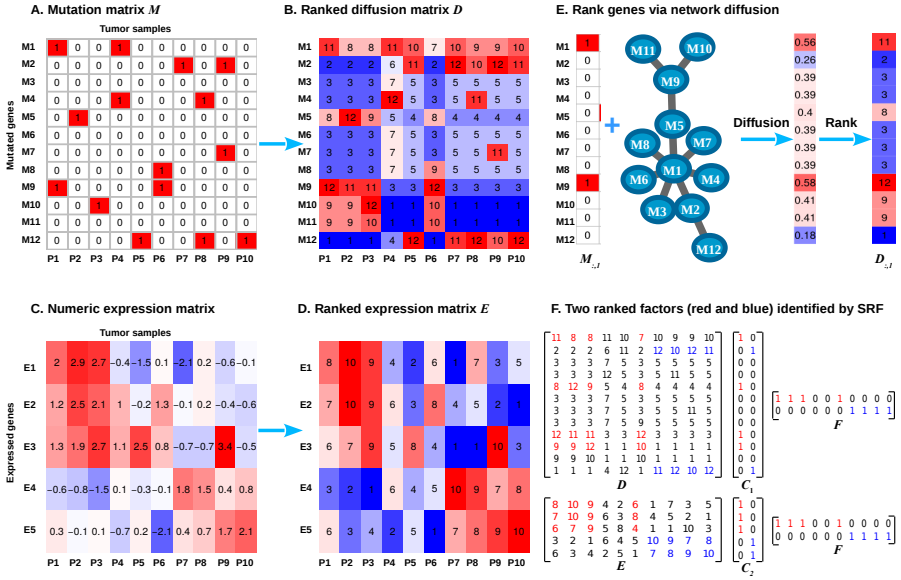


Figure 5.1: SRF illustration. A) Boolean mutation matrix; B) Ranked diffusion matrix derived from the mutation matrix using the network diffusion model shown in Figure E and the parameter $\alpha = 0.7$; C) Numeric expression matrix; D) Ranked expression matrix, obtained by ranking column values in each row; E) Illustration how to derive a ranked diffusion vector for tumour sample $P1$ using his/her mutation profile and a given interaction network. F) Two ranked factors, represented by $\mathbf{C}_1$, $\mathbf{C}_2$ and $\mathbf{F}$, identified by SRF in matrices $\mathbf{D}$ and $\mathbf{E}$.

genes (expression and mutational features) for which the selected samples have respectively highly ranked expression values and highly ranked relevance scores. Expression and mutational features can, but do not have to overlap. Whereas a factor represents a group of patients together with their characteristic features, a subtype is defined as a group of patients covered by a unique combination of factors. Subtypes can thus mutually overlap in their characteristic expression and/or mutational features. This overlap in features between subtypes reflects the fact that subtypes are rarely distinct but rather represent a continuum of possible alterations. The subtyping algorithm is dubbed SRF, for *Subtyping with Ranked Factors*.

5.2.1 Transforming input datasets into rank matrices

The first step is to transform the original data into rank matrices.

Transforming transcription data Given a gene expression matrix $\mathbf{A} \in \mathbb{R}^{l \times n}$, where l is the number of expression genes and n is the number of tumour samples, its corresponding rank matrix is obtained by sorting each row's values from low to high and assigning ranks accordingly (with the largest rank being assigned to the highest value). That is, all samples are ranked for each gene. The resulting *ranked expression matrix* $\mathbf{E}$ has the same size as $\mathbf{A}$ and the values in each row are a subset of $\sigma_1 = \{1, \ldots, n\}$ (ties all get the lowest rank). Figure 5.1C shows an example of an expression matrix $\mathbf{A}$ and Figure 5.1D shows its transformed rank matrix $\mathbf{E}$.

Since this transformation would allow our algorithm to only find over-expressed genes (genes with an average high rank within a subset of the samples), we duplicate each row and also assign ranks in the reverse order. This will allow to find under-expressed genes as well. For example, given an expression vector $g = (-2.0, -3.0, 2.0, 3.0)$, we obtain both rank vectors $r_o = (2, 1, 3, 4)$ (assigning high ranks to over-expressed genes) and $r_u = (3, 4, 2, 1)$ (assigning high ranks to under-expressed genes). As a consequence, the resulting rank matrix has twice as many rows as the original matrix. For ease of exposition we will consider matrix $\mathbf{E}$ to have the same size as $\mathbf{A}$, but the algorithm trivially works on the duplicated matrix and we will use this extended version in the experiments.

Transforming mutation data and interaction network To transform the Boolean mutation matrix into a rank matrix, we first map each patient's mutation profile to the given interaction network [124, 49] and apply a network diffusion model. The obtained diffusion values are then transformed to ranks as before, so that higher ranks indicate that a gene is relatively "close" to a mutated gene (for a particular patient). Figure 5.1e illustrates this procedure, which we next describe in more detail.

Let $\mathbf{M} \in \{0, 1\}^{m \times n}$ be the mutation matrix, where m is the number of mutation genes and n is the number of patients (as before), and let $G = (V, E)$ be the interaction network. For simplicity, we assume that each mutation gene corresponds to exactly one $v \in V$ and vice versa, hence $|V| = m$. Note that, in practice, we can always create such a mutation matrix by filtering mutated genes that are not in the network or adding more genes with all $0s$. Further, let $Y^{(i)} = M_{:,i} = (y_1, \ldots, y_m)^T$ be the mutation profile of patient $i, i = 1 \ldots n$. Note that $y_j = 1, j = 1 \ldots m$, iff patient i carries the mutation in row j and is 0 otherwise. The corresponding diffusion vector $F_t^{(i)}$ is defined by the following formula:

$$F_t^{(i)} = \alpha \mathbf{W} F_{t-1}^{(i)} + (1 - \alpha) Y^{(i)}, \tag{5.1}$$

where $\mathbf{W}$ is a stochastic matrix obtained by multiplying the adjacency matrix of the interaction network by a diagonal matrix with the inverse of its row (or column) sums on the diagonal. Parameter α is used for tuning. Diffusion vector

$F_t^{(i)}$ is determined by iteratively solving Equation 5.1 t times, starting with $F_0^{(i)} = Y^{(i)}$, until convergence.

Applying Equation (5.1) to the n columns of matrix $\mathbf{M}$ results in a diffusion matrix $\mathbf{B} \in \mathbb{R}^{m \times n}$, where $\mathbf{B}_{:,i} = F_t^{(i)}$. Finally, ranking the rows for each column we obtain the *ranked diffusion matrix* $\mathbf{D} \in \sigma_2^{m \times n}, \sigma_2 = \{1, \ldots, m\}$, which we use as input for the next step of the analysis.

5.2.2 Mining k ranked factors using sRMF

The matrix factorisation model that we introduce aims to jointly *factorise* the two transformed rank matrices $\mathbf{D}$ and $\mathbf{E}$ into a set of k ranked factors, where k is an integer given by the user. One *factor* consists of a set of mutation genes, a set of expressed genes and a set of related samples. To provide some intuition for our approach, we first present an optimisation model for a single ranked factor. We then generalise this to k ranked factors using rank matrix factorisation.

Mining a single ranked factor As mentioned above, a *ranked factor* represents a group of samples that are consistently over/under-expressed in a subset of expression genes and that share the same affected genes in the ranked diffusion matrix.

Let $\mathcal{P} = \{1, \ldots, n\}$, $\mathcal{M} = \{1, \ldots, m\}$ and $\mathcal{E} = \{1, \ldots, l\}$ be index sets for tumour samples, mutation genes and expression genes respectively. A ranked factor is represented by a tuple (P, G^M, G^E), where $P \subseteq \mathcal{P}$, $G^M \subseteq \mathcal{M}$ and $G^E \subseteq \mathcal{E}$. Inspired by the ranked tiling pattern introduced in Chapter 4, a ranked factor is obtained by optimising:

$$\begin{aligned} \text{argmax}_{P,G^M,G^E} \quad & \textstyle\sum_{m \in G^M, p \in P} (\mathbf{D}_{m,p} - \theta_1) + \\ & \beta \textstyle\sum_{e \in G^E, p \in P} (\mathbf{E}_{e,p} - \theta_2) \end{aligned} \tag{5.2}$$

subject to

$$\forall m \in \mathcal{M} : m \in G^M \quad \rightarrow \quad \textstyle\sum_{p \in P} \mathbf{M}_{m,p} \geq \mu, \tag{5.3}$$

where θ_1 and θ_2 are user-defined thresholds that control how high ranks in $\mathbf{D}$ and $\mathbf{E}$ respectively should be to be included in the solution. We sometimes indicate these thresholds using relative values, i.e., $\theta_1 = a\%$; in this case, the absolute threshold is $\theta_1 = a\% * n$. β is a user-defined threshold to balance the contributions from the values in the two matrices. μ indicates the number of

patients in which a mutation should be present in order to be included in the factor.

The objective in Equation 5.2 selects those rows (mutation and expression genes) and columns (samples) that together maximise the total sum of the values in the corresponding cells in the matrices, adjusted by θ_1 and θ_2. That is, cells that are lower than the thresholds are penalised and those that have higher values are rewarded. Equation 5.3 ensures that each gene that is selected from the ranked diffusion matrix is mutated in at least one of the samples present in the selection P. That is, genes that receive a high rank because they are in the network neighbourhood of genes mutated in the sample but are never mutated themselves will not be selected.

For example, given the matrices in Figure 5.1A, 5.1B, and 5.1D, and parameters $\theta_1 = 7, \theta_2 = 5, \beta = 1$, solving the objective results in $P = \{P1, P2, P3, P6\}$, $G^E = \{E1, E2, E3\}$, $G^M = \{M1, M5, M9, M10\}$. It is clear from the input matrices that this solution corresponds to an area with relatively high ranks. No more samples or genes can be added to the solution without decreasing the score. Note that mutated gene $M11$ was not selected despite having a high rank as no samples in the group carry a mutation for this highly ranked gene.

Mining k single ranked factors using sRMF The optimisation problem in Equations (5.2) – (5.3) finds two ranked tiles, one in the ranked diffusion matrix $\mathbf{D}$ and the other in the ranked expression matrix $\mathbf{E}$, which have the same set of the columns. Hence, it is a joint rank matrix tiling for the two matrices with the mutation constraints in Equation (5.3) and $k = 1$. When we want to find k ranked factors (the SRF problem), we just simply remove the constraint $k = 1$.

Let's define the following two scoring functions:

$$\delta_1(a, b) = \begin{cases} 0 & \text{if } b = 0; \\ a - \theta_1 & \text{otherwise.} \end{cases} \tag{5.4}$$

$$\delta_2(a, b) = \begin{cases} 0 & \text{if } b = 0; \\ \beta(a - \theta_2) & \text{otherwise.} \end{cases} \tag{5.5}$$

Then, the SRF problem can be formally defined as follows.

Definition 5.1 (SRF problem)**.** *The SRF problem is the joint rank matrix factorisation for the rank matrices $\boldsymbol{D}$ and $\boldsymbol{E}$ using*

- *the max-product semiring;*
- *the set of permissible values $\sigma_p = \{0, 1\}$;*

- *the additive scoring function for the rank matrix factorisation for matrix* $\boldsymbol{D}$ *based on formula (5.4);*
- *the additive scoring function for the rank matrix factorisation for matrix* $\boldsymbol{E}$ *based on formula (5.5);*
- *constraints that the two rank matrix factorisations share the same matrix* $\boldsymbol{F}$*;*
- *constraints that* $\mu \boldsymbol{C}_1 \leq \boldsymbol{M}\boldsymbol{F}^T$*, where* $\boldsymbol{C}_1$ *is the factorised matrix of the factorisation for matrix* $\boldsymbol{D}$*.*

To solve the SRF problem, we follow the algorithm for Sparse mRMF. That is, the SRF algorithm follows an iterative EM-style scheme, in which first $\mathbf{C}_1$ and $\mathbf{C}_2$ are optimised given $\mathbf{F}$, and then $\mathbf{F}$ is optimised given $\mathbf{C}_1$ and $\mathbf{C}_2$. We repeat this iterative scheme until the optimisation score cannot be improved any further. When either $\mathbf{C}_1$ and $\mathbf{C}_2$ or $\mathbf{F}$ is known, it can be shown that the SRF problem defined in Definition (5.1) is an integer linear programming (ILP) problem. Each such optimisation problem can be solved optimally. To avoid local maxima, we initialise the algorithm with a matrix $\mathbf{F}$ obtained by performing hierarchical clustering to cluster the columns into k groups.

We can develop a *parallel* implementation for SRF, which makes it scalable to large datasets as we did for mRMT (Section 4.3.5). This is because each row of $\mathbf{C}_1$ and $\mathbf{C}_2$ can be optimised independently given $\mathbf{F}$. Further, given $\mathbf{C}_1$ and $\mathbf{C}_2$, each column of $\mathbf{F}$ can be optimised independently if we relax the inequality in Equation (5.3), which puts a constraint on the columns and hence makes them dependent. However, if we require the iterative process to terminate after the step optimising $\mathbf{C}_1$ and $\mathbf{C}_2$ given matrix $\mathbf{F}$, we still obtain a very good approximation upon convergence of the algorithm.

Implementation. We implemented SRF in OscaR [87] and used Gurobi as the back-end solver. The implementation is available at the following address: `https://github.com/rankmatrixfactorisation/SRF`.

5.2.3 Deriving cancer subtypes from ranked factors

Ranked factors model groups of tumour samples that are homogeneous in gene expression as well as in mutations. Hence, if we obtain k *non-overlapping ranked factors*, i.e., factors that cover fully disjoint sets of samples, each factor found is considered to represent a unique subtype.

If the factors *overlap* in the sample dimension, however, we consider each group of samples that is covered by a unique combination of ranked factors to form

a subtype. The reason for this is that each combination of ranked factors represents a different combination of expression and mutation profiles. In this case, the mutation and expression gene sets of a subtype are formed by the union of the mutation and expressed genes (respectively) of all factors in the combination. Section 5.3 shows examples of this concept. In practice, we prune subtypes covering fewer samples than a user-defined threshold, to avoid the discovery of small 'subtypes' that are most likely artefacts of noise in the data.

5.3 Results

We report results on both synthetic datasets and the well-studied TCGA breast cancer dataset.

5.3.1 Results on simulated datasets

To test the performance of the method in recovering known subtypes, we generated datasets in which each subtype was defined as a set of tumour samples carrying a number of driver genes and a concomitant set of consistently over- and/or under-expressed genes of which the expression phenotype is assumed to be triggered by the driver mutations. The data contained 4 subtypes that occasionally shared genes mutated in the same driver pathways or genes displaying the same consistent expression. We imposed the rule that whenever two subtypes share genes mutated in the same driver pathway(s), they should share a set of consistently expressed genes. Figures 5.2A and 5.2B show an example dataset.

Driver genes were modeled to display mutational consistency at the pathway level across tumour samples belonging to the same subtype by selecting the drivers of those patients from a pre-selected set of genes that are *closely connected* in a real protein-protein interaction network and therefore assumed to belong to the same driver pathway. We varied the size of the driver pathways as well as the mutational recurrency of the driver genes for the samples within the subtypes to generate datasets (see Section 5.4).

For each simulated dataset, we ran our algorithm with varying parameter settings. We used two parameters to specify the preferred ranges of the ranks from the two input matrices, and another to balance the contributions of the two matrices. For each parameter setting, we used SRF to search for $k = 5$ subtypes, where the 5th subtype serves as the collection of tumour samples that have no clear subtype assignment. We initialised the algorithm with five

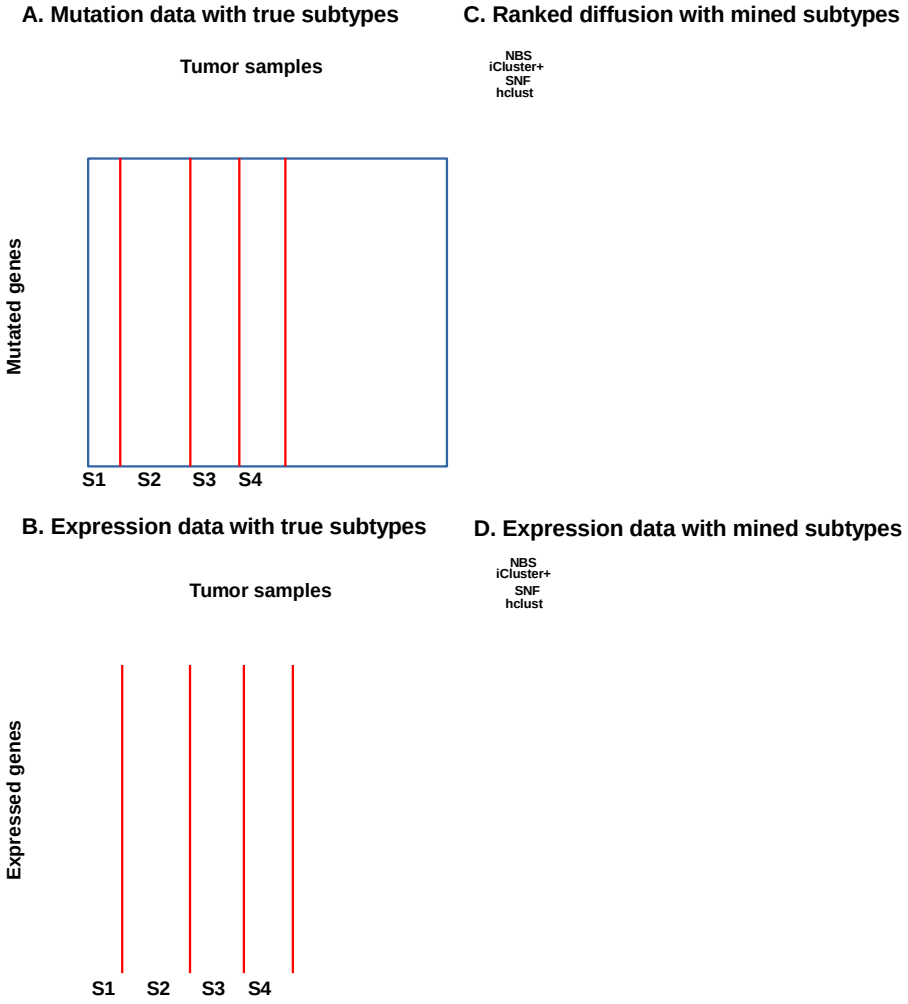


Figure 5.2: Evaluation on simulated datasets. Panel A–B: Example data with ground truth. The heatmaps show mutation and numeric expression data of a representative simulated dataset, with a 10% mutational recurrency (meaning that a gene is mutated in at least 10% of the samples in a given subtype) and pathway size of 40. The four ground truth subtypes are marked by the horizontal and vertical coloured bars above and to the left of the heatmaps. Panel C – D: Results on the data shown in panels A and B. Results obtained by NBS [49], iCluster+ [84], SNF [130] and the hierarchical clustering algorithm (hclust), which we used to initialize our model, are shown in the coloured bars above the heatmaps. The results obtained with SRF are indicated by the four coloured horizontal and vertical bars, just above and to left of the heatmaps; each bar indicates the patients (horizontally) and genes (vertically) selected by a ranked factor.

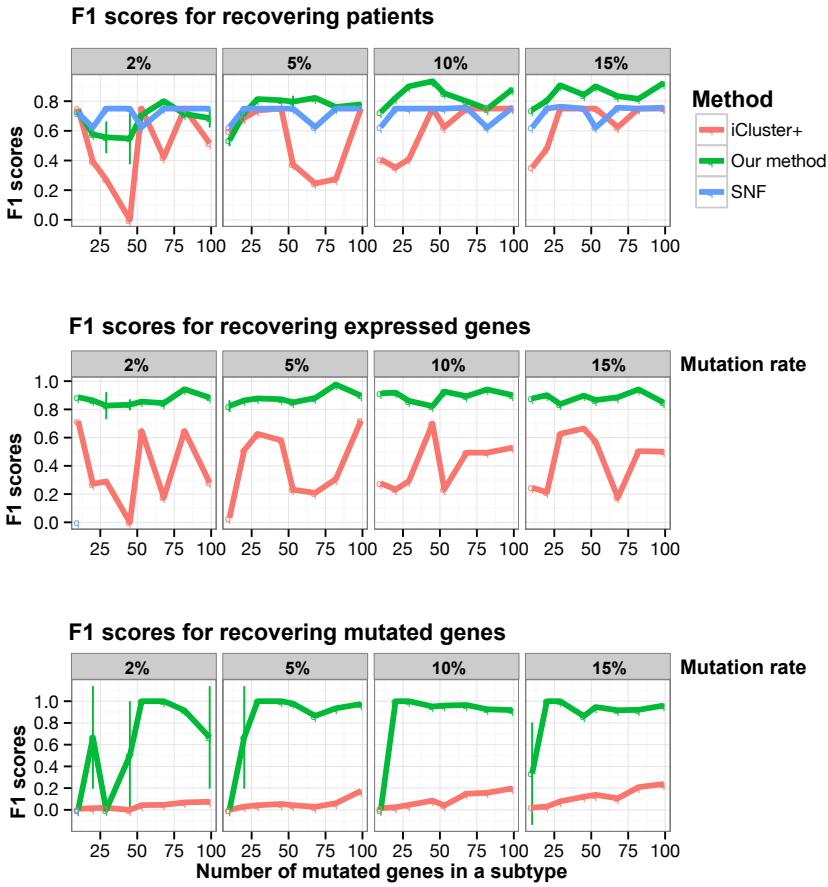


Figure 5.3: F1 score comparison. The three plots denote F1 scores for 1) patient recovery (top), 2) expression gene recovery (middle), and 3) mutation gene recovery (bottom) for iCluster+, NBS, SRF, and SNF, for simulated datasets of varying driver pathway sizes and mutational recurrencies. Note that NBS does not work with expression data and we were unable to recover the mutated genes due to a lack of documentation.

sample groups obtained by a hierarchical clustering of the tumors using the ranked expression data.

After factorising the rank matrices, resulting subtypes containing less than 4% of the total number of samples were pruned (see Section 5.4).

Accuracy of the identified subtypes We evaluated the performance of our algorithm in recovering the known subtypes as well as their characteristic expression and mutational features. For this we used the F1 score, which assesses the trade-off between correctly and comprehensively distinguishing between samples, expression genes, and mutational features that truly belong to the subtypes from those that do not.

To optimise the parameter settings, we calculated F1 scores for different parameter settings and chose the one that resulted in the highest average score (see Section 5.4). Then, we used that parameter setting to evaluate the performance of the algorithm on all the simulated datasets.

Figure 5.3 shows the F1 scores obtained for different driver pathway sizes and mutational recurrencies. We can observe that the F1 score of recovering tumour samples of the simulated subtypes is high and largely independent of the sizes of the driver pathways and the mutational recurrencies. This demonstrates the added value of integrating the expression data. Further, the F1 scores of recovering mutation and expression genes relevant to the subtypes are generally high. As expected, the higher the mutational recurrency, the larger the number of mutated genes that can be recovered.

Comparison to related work To show that our method performs at least as well as state-of-the-art subtyping methods, we compared the results obtained by our method to those obtained with iCluster+ [84], NBS [49] and SNF [130], applied on the same simulated data. Both iCluster+ [84] and SNF [130] identify subtypes by jointly clustering the expression and untransformed mutation data, while NBS [49] exploits mutational information but does not use expression data.

Results obtained by iCluster+, NBS, SNF and SRF are summarised in Figure 5.2E. Our method obtained higher F1 scores than its competitors for both recovering expression and mutation genes. This is because our model couples genes, including mutation and expression genes, and patients to define subtypes and thus explicitly identifies subtype-specific genes. For iCluster+ and SNF, this is not the case and the selected expression or mutation genes are thus always the same, irrespective of the subtype.

As a representative example, we illustrate in more detail the results produced by the different methods on the simulated dataset with a 10% mutational

recurrency and a pathway size of 40 (Figure 5.2C - 5.2D). The figures show that compared to the other methods, our method can discover overlap between very similar subtypes in both the patient and gene dimension. In addition, our model is shown to be tolerant to noise: the fifth ranked factor found by SRF remained empty, revealing that no 'noisy' patients and genes were incorrectly marked as belonging to a subtype.

5.3.2 Results on the TCGA breast cancer data

To test SRF in a real-world setting, we applied it to the well-studied TCGA breast cancer dataset. We ran the method as outlined in Section 5.4. As we were mostly interested in identifying subtype-specific features, we chose stringent parameters to only identify subtypes with representative profiles in terms of expression and mutations. Factorising the dataset into $k = 8$ factors resulted in 13 subtypes. The number of identified subtypes is higher than the number of factors because subtypes are defined as combinations of factors (see Section 5.2.3). The results are visualised in Figure 5.4.

To validate our subtypes, we tested 1) to what extent the discovered subtypes corresponded to the PAM50 classification, and 2) to what extent SRF could further refine it. Figures 5.4 and 5.6 show that most subtypes are enriched in samples with the same PAM50 label [89] as shown in Figure 5.4A. All samples of the same PAM50 class rarely end up in a single subtype. The Basal subtype, for example, is divided into two major subgroups: S10, S12; LumA is divided into S3, S4, S5, S6, S8, S13; LumB into S1, S2, S5, S9; Her2 into S11 and S7. So our approach does not only match the PAM50 classification to a large extent, it also further refines known subtypes.

This high-resolution subtype refinement is a characteristic property of the method's intrinsic feature selection. Rather than using global profiles to group samples, the methods actively searches for combinations of feature sets (factors) that characterise samples using rather stringent criteria. As a result differences between expression and mutational profiles are marginal for some subtypes (e.g., for LumA-related subtypes S3, S4, S8, S13, and for LumB-related subtypes S7 and S9). Retrospectively, it might have been possible to merge these subgroups. However, in case of subtypes S10 and S12, carrying samples with the same Basal label, the subtype-specific mutational and gene expression profiles are quite distinct for the selected feature sets, corresponding to the brown and pink bars in Figure 5.4.

Next to the subtypes that have rather homogeneous PAM50 labels assigned to their samples, subtypes S1, S2, S5 and S9 contain a mixture of LumA and LumB samples, and S11 contains a mixture of Her2 and LumB samples. Although

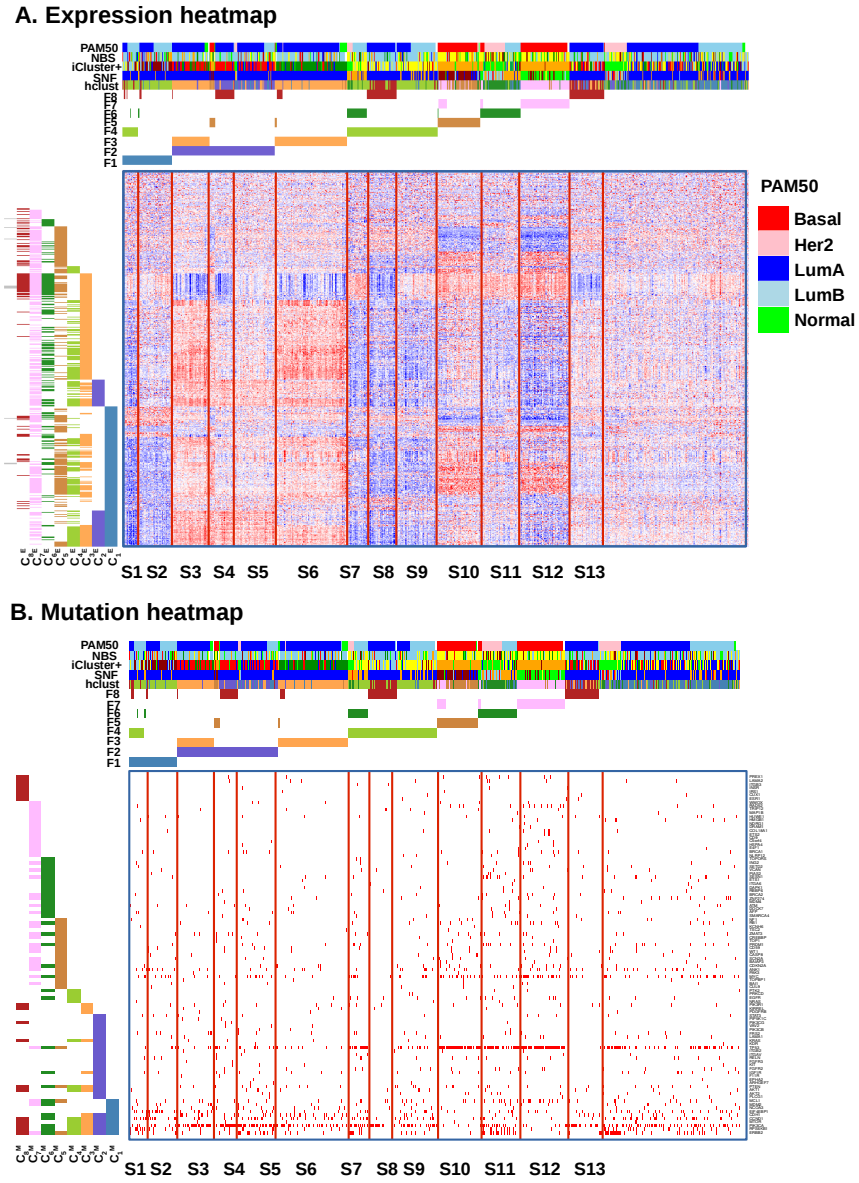


Figure 5.4: Results of applying SRF ($k = 8$) on the TCGA breast cancer dataset, which resulted in the 13 subtypes denoted by S1, ..., S13. For heatmaps in Panels A, and B: red implies over-expressed, white neutral, blue under-expressed. Panel A: Expression data. The gene and tumour sample sets corresponding to the eight ranked factors are marked by the vertical and horizontal colour bars. Each subtype is a unique combination of ranked factors in the tumour sample dimension. Panel B: Mutation data. Samples are ordered as in panel A. Only mutated genes that belong to any of the factors are displayed. On panels A and B, the top bar indicates the PAM50 annotation of the samples together with the subtyping results of iCluster+, NBS, SNF, and hierarchical clustering.

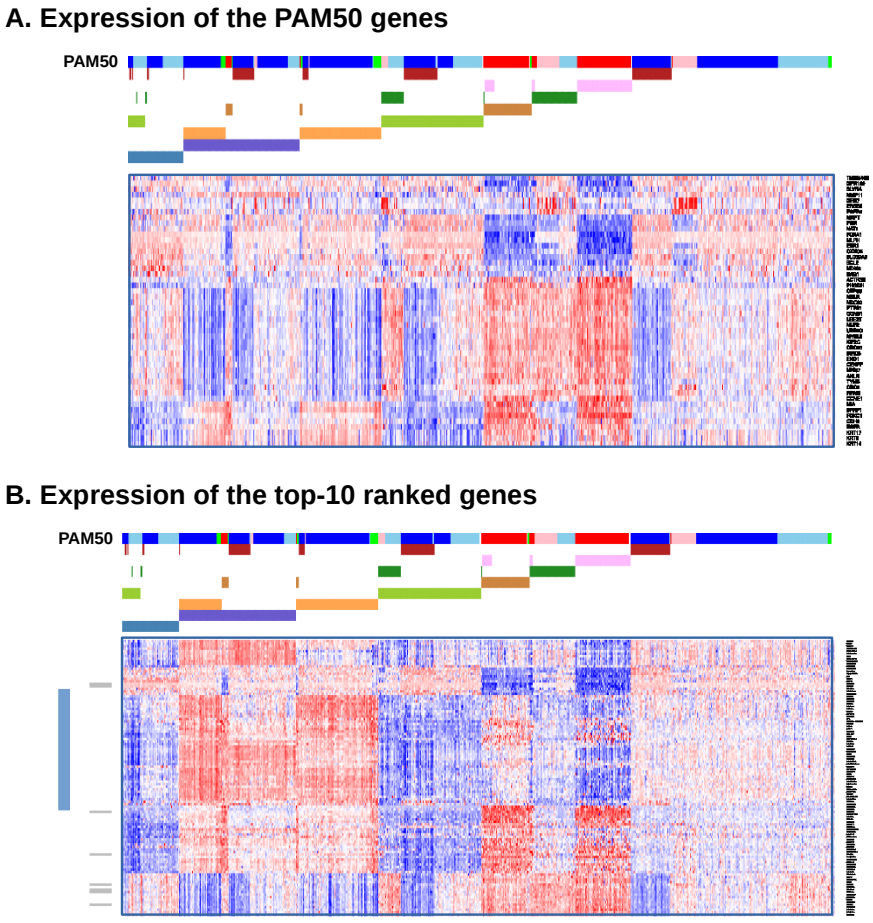


Figure 5.5: Panel A: Expression heatmap of the PAM50 genes. The samples are ordered as in Figure 5.4A. Red implies over-expressed, white neutral, blue under-expressed. Panel B: Expression heatmap of the top-10 genes per subtype, i.e., the ten genes having the highest average ranked scores per subtype.

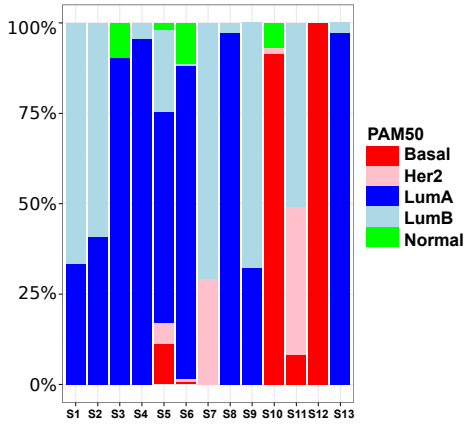


Figure 5.6: Distribution of PAM50 samples in the identified subtypes

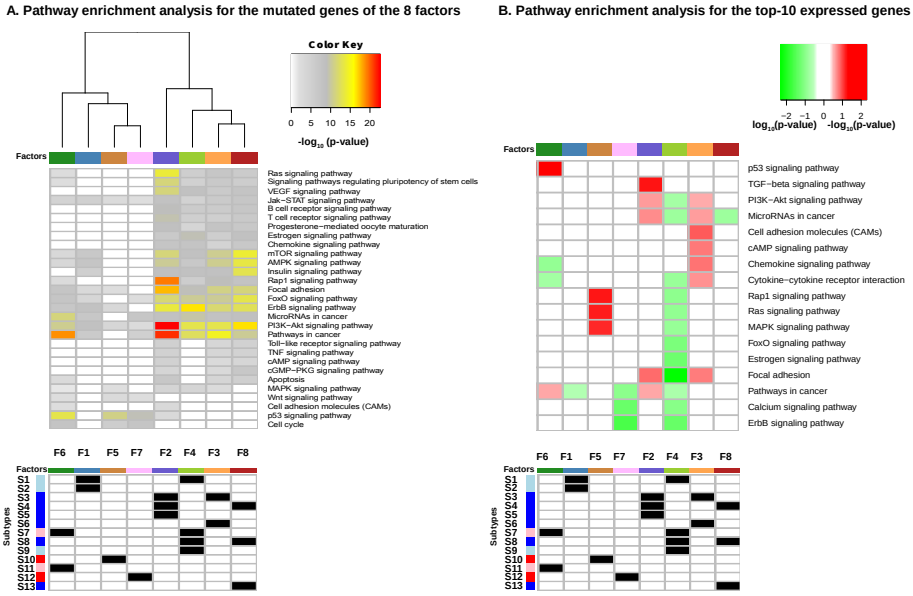


Figure 5.7: Cancer pathway enrichment analysis. Panel A: KEGG pathway [55] enrichment analysis for the selected mutated genes of the eight factors; resulting $-log_{10}$ p-values are shown for cancer related pathways that were found to be significantly enriched in at least one of the factors. Panel B: KEGG pathway enrichment analysis for the top-10 ranked expressed genes of the eight factors. log_{10} and $-log_{10}$ p-values are shown for pathways having under- and over-expressed genes respectively.

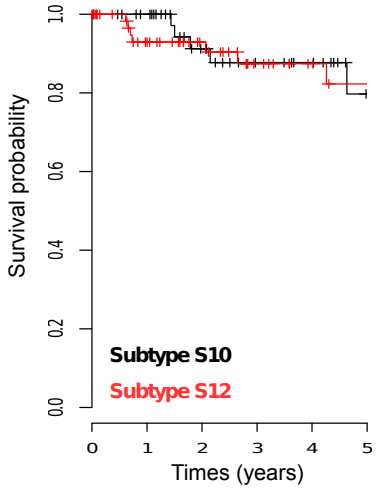


Figure 5.8: Kaplan-Meier plot for the two Basal-related subtypes S10 and S12. (Not statistically significant because of the low mortality rate.)

some inconsistency between the mere expression-based PAM50 classification and subtyping protocols based on the integration of expression and genomic information is to be expected [19], a closer inspection of the expression and mutational profiles of the subtypes with mixed PAM50 class membership shows why our method does not distinguish between, e.g., the selected Her2 and some LumB samples. That is, the selected LumB samples of subtype S11 contain clear Her2-related features that distinguish them from other LumB samples, such as an increased ERBB2 amplification and a more pronounced over-expression of a characteristic subset of genes.

Our approach towards identifying subtypes together with their features can only be meaningful if the selected features are biologically relevant. To assess this, we first tested to what extent the expression features used to build the PAM50 classifier are amongst our selected features. From the 50 PAM50 features, 49 were present amongst the features selected by our method after pre-processing (see Section 5.4). The ranked factors found by SRF used 2221 features in total, including 48 out of 50 PAM50 features. To select a smaller representative feature set, the 10 genes with the highest average ranked score per subtype were selected, resulting in 110 instead of 2221 features (Figure 5.5B). Those 110 features contained 8 out of the 48 remaining features of the PAM50 classifier. SRF selects all high-ranking features, hence the selected feature sets are more redundant than those used by PAM50, which were designed for classification. If our approach is to coincide with PAM50, we expect each group of features

to be covered by a few PAM genes. Except for one subtype this is indeed the case. The exception, indicated with the blue row bar in Figure 5.5B, does not have a corresponding PAM50 feature. Remarkably this is the feature set that has the most distinct difference in expression between the two subtypes with the same PAM50 Basal label (S10 and S12). Figure 5.5A shows how indeed no differences can be observed between Basal subtypes S10 and S12 using the PAM50 features, whereas the subdivision is clear using the expression-based features selected by our method and shown in Figure 5.5B. Figure 5.9 shows how the subtype subdivision of the Basal-like subtypes and the selection of the corresponding expression features is also driven by the simultaneous selection of the mutation-based features: S10 and S12 show clearly distinctive mutational profiles with different mutational frequencies of, e.g., *CD9, DRAM1* and *E4F1*. Interestingly, survival analysis of these two Basal-like subtypes, despite not being significant due to the low mortality rate, shows that subtype S12 tends to be more aggressive than S10 during the first two years (Figure 5.8).

To assess whether the selected feature sets belong to known driver pathways in breast cancer, we did per factor a KEGG pathway enrichment analysis on respectively the selected mutational and expression features. Figures 5.7A and 5.7B display the enrichment levels for a representative set of cancer related pathways that were found to be enriched. They also indicate how each subtype is a composition of different factors and how the factors overlap in genes and thus also in enriched pathways. For instance, the genes with a characteristic expression profile in factor 5 (representative for lumA and B) and factor 4 (representative for basal S10) are enriched in rap1, ras1 and mapK signaling, but with an anticorrelated expression profile for the Luminal subtypes versus the Basal one. Figure 5.7A shows how, as expected, [12, 119, 125], the Basal-related (S10, S12) and Her2-related subtypes (S11) are highly enriched in *p53 signaling* and *cell cycle* whereas other subtypes are not. In addition, the Luminal subtypes (LumA and LumB; S2, S5, S6, S9 and S13) are enriched with cancer pathways known to be specific for this group: *PI3K-Akt signaling pathway* [12, 125], *Estrogen signaling pathway* [74], *AMPK signaling pathway* [125].

Comparison with state-of-the-art methods To test to what extent our method agrees with state-of-the-art subtyping methods, we also ran iCluster+ [84], NBS [49] and SNF [130] on the same dataset. Parameter settings for each of these methods was optimised as explained in Section 5.4. Figure 5.4 illustrates how our results compare with those of the other tools in terms of matching the PAM50 subtyping. SNF and iCluster+, the two integrative methods that do not use mutational consistency at the pathway level, do not perform well for some subtypes. For example, SNF could not discern the heterogeneity of the Luminal samples, which has been known to be the most heterogeneous breast cancer subtype [12, 19], when it clustered all LumA samples and a large number of

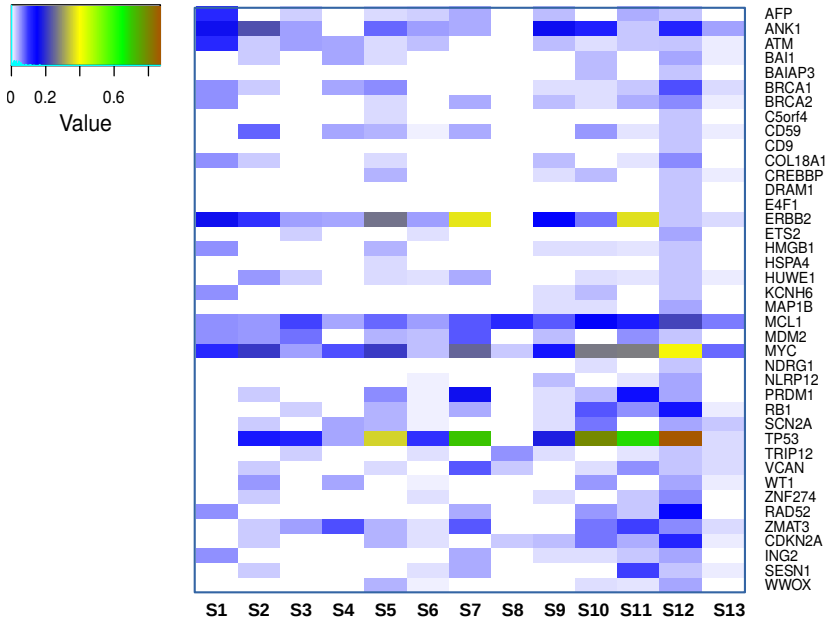


Figure 5.9: Comparing the mutation frequencies among the identified subtypes using the mutation gene set selected by subtype S12.

LumB samples into one group; iCluster+ could not subdivide the Basal subtype. NBS, which could use mutational consistency at the pathway level but could not integrate with expression data, could not distinguish the LumA from the LumB samples. In contrast, our method can integrate expression data and mutational data at the pathway level and hence can capture subtle differences that might be missed otherwise.

Comparison with hierarchical clustering We performed a hierarchical clustering of the tumour samples into clusters, of which the result is annotated by the *hclust* column bar in Figure 5.4A. This clustering is also used to initialise the matrix $\mathbf{F}$ of the two factorisations defined in Problem 5.1. In contrast to hierarchical clustering, our SRF algorithm identifies clusters (subtypes) that are highly overlapping and removes noisy samples.

5.4 Materials and methods

Simulated data Mutational data was generated by first selecting driver pathways for each of the simulated subtypes. Driver pathways were selected from the densely connected sub-networks obtained by applying the *InfoMap* algorithm [96] implemented in the *igraph* [18] R package on the STRING network [112] post-processed by [49]. The selected driver pathway sizes varied from $\{20, \ldots, 100\}$ genes and each such gene was assigned a mutational recurrency between 2% and 15%. Passenger mutations were simulated by sampling, for each patient, from a Bernoulli distribution with $p = 0.005$ (the average mutational recurrency we observed in the TCGA breast cancer data). The total number of passengers was chosen such that the total number of mutation genes, including both drivers and passengers, was 8000, which was approximately equal to the number of mutation genes used in the TCGA breast cancer data. Each simulated mutation matrix consists of 8000 genes $\times$ 350 patients.

Expression data was simulated as previously described by [68]. That is, first background information was generated by sampling from a mixture of three Gaussians, of which means were uniformly sampled from three different ranges, namely, [-5,3), [-3,3] and (3,5]. Then, over-expressed and under-expressed modules were implanted. Values within over-expressed and under-expressed modules were sampled from a Gaussian, with mean uniformly sampled from (3,5] and [-5,-3) respectively. Per set of driver mutations and thus per subtype, we ensured that the simulated dataset consisted of at least one set of genes that was consistently differentially expressed across the samples in the subtype.

To simulate noise in the expression data, we simulated 100 small expression modules that could be seen as the result of some confounding factors such as sex and tissue type. The number of rows and columns of these confounding modules were sampled from a normal distribution, whose mean was equal to 25% of the medium-sized pattern and standard deviation was equal to 40% of the mean. Each simulated expression matrix consists of 4000 genes $\times$ 350 patients.

TCGA breast cancer dataset Breast cancer somatic mutation, copy number alteration (CNA), expression (RNA-Seq v2), and clinical data were downloaded from the TCGA data portal. Mutational data were converted to a Boolean mutation matrix. CNA data were analysed using Gistic 2.0 [81] with default settings. This data was then binarised by considering how genes are classified by Gistic: as either deleted or amplified. This information was added to the mutation matrix. We restricted our analysis to mutations and CNVs in genes that also appear in the STRING network (12232 vertices) prepared by [49]. Expression genes were selected based on their differential expression relative

to normal (non-tumour) samples: for each gene a normal distribution was fitted using the normal expression samples and z-scores were calculated for the tumour samples. We then evaluated the 5th and 95th percentiles of the tumour samples. Genes were selected if 1) the p-values for these percentiles were below 0.001 and 2) their log-fold change relative to the mean normal expression was at least 2.5. After the filtering steps mentioned above, the final mutation matrix consisted of 8604 genes $\times$ 719 patients, and the final expression matrix of 2472 genes $\times$ 719 patients.

Diffusion threshold selection Our SRF algorithm transforms the Boolean mutation matrix into a rank matrix by applying a diffusion model [124], [49] for each mutation profile of the tumour samples on a protein–protein interaction network. Intuitively, the diffusion model links mutated genes to its neighbours with respect to the interaction network structure. The strength of these links is governed by a tuning parameter called α. Hofree et. al., [49] found the optimal value of α is network dependent. In the case of the STRING [112] network, they also discovered that the optimal value for α is 0.7. As we used the STRING network, which was post-processed by Hofree et. al., [49], we set the α threshold to 0.7 for all of the experiments presented in this paper.

SRF parameter selection Our algorithm has five parameters: θ_1 and θ_2 specify the minimal threshold on respectively the diffusion and expression ranks for a gene/sample to be included in a ranked factor; β specifies the importance of the mutation relative to the expression data; k specifies the number of ranked factors and μ specifies the number of patients in which a mutation should be present in order to be included in a factor. μ was set to 2 for all the experiments in this paper.

For each simulated dataset, we performed a parameter sweep using combinations of the following parameter settings: $\theta_1 \in \{82\%, 85\%, 90\%\}, \theta_2 \in \{65\%, 70\%\}$, and $\beta \in \{2, 18, 35, 50\}$. As we knew the ground truth for these datasets, we evaluated F1 scores and chose the parameter setting that resulted in the mean highest average score over all simulated datasets. This resulted in the following choice for the parameters: $\theta_1 = 82\%, \theta_2 = 65\%, \beta = 18$. Note that with this choice of β, the mutation component is implicitly given two times the weight of the expression component in the optimisation problem, as $(max(\mathbf{D})/(\beta * max(\mathbf{E})) = 1.94$, where $max(\mathbf{D}) = 12232$ and $max(\mathbf{E}) = 350$.

Given that our artificial data has similar properties as the TCGA data, for the TCGA data we used the same parameter settings for θ_2 and β as for the artificial data, i.e., $\theta_2 = 65\%$ and $\beta = 18$; also in an earlier study [69], on expression data only, it was demonstrated that $\theta_2 = 65\%$ is a good choice. We considered alternative settings for θ_1, with $\theta_1 \in \{70\%, 72\%, \ldots, 90\%, 92\%\}$. Our motivation is that we wished to end up with a number of mutations smaller

than 40, which is the number of cancer genes of this disease found by [110]. Although the impact of the θ_1 parameter is small, we decided to use a parameter setting of $\theta_1 = 86\%$ to reduce the set of mutated genes. For k we considered values in the range $k \in \{5, \ldots, 14\}$. We observed that for $k > 8$ the results only change slowly and hence stopped at $k = 8$.

To validate whether our choice is reasonable for the TCGA dataset or not, we ran SRF with a selected number of combinations of the parameter thresholds, i.e., varying one parameter while fixing the others to the selected values described above. Then, we evaluated two scores: *coverage* and *error*. The coverage score is the percentage of the region in the reconstructed matrix of the factorisation that has non-zero values. The error score is the average number of cells in the covered region whose value is less than the user-defined threshold.

Figure 5.10 shows the behaviour of the algorithm when we varied the value of β (the relative importance threshold of mutation to expression) and fixed the other parameters to the selected values mentioned above ($\theta_1 = 86\%, \theta_2 = 65\%, k = 8$). The figure confirms that $\beta = 18$ was a good choice as from that point the mutation coverage levelled off and became reasonably small. At the same time, the mutation error per cell slowly increased since $\beta = 18$.

Figure 5.11 illustrates the performance of the algorithm when we varied θ_1 (the rank threshold for ranked diffusion) and fixed the other parameters to the selected values. The figure confirms that $\theta_1 = 86\%$ is a good choice as the mutation coverage (and hence the number of mutations per subtype) becomes reasonably small from that point on. At the same time, the expression coverage starts to increase while the expression error per cells decreases. In other words, high quality expression data were added into the solutions from $\theta_1 = 86\%$ on.

Figure 5.12 shows the behaviour of the algorithm when we vary θ_2 (the rank threshold for ranked expression) and fix the other parameters to the selected values. The figure shows $\theta_2 = 70\%$ could be a good choice as both mutation error and expression error decreases at that point. However, the mutation coverage increases and hence the number of mutations per subtype also increases. Because we wished to end up with a number of mutations smaller than 40, $\theta_2 = 65\%$ was chosen.

Figure 5.13 shows that the mutation coverage becomes stable from $k = 8$ on, while expression coverage slowly increases. Hence, we could stop at $k = 8$.

Parameter selection for the benchmarked methods With iCluster+ [84], we used the model selection algorithm provided by the software to obtain the optimal parameters. With SNF [130], we varied the α parameter in the range of $\{0.3, 0.4, 0.5, 0.6, 0.7, 0.8\}$ and chose the one that resulted in the highest average F1 score on the simulated data. With NBS [49], we used the default parameter

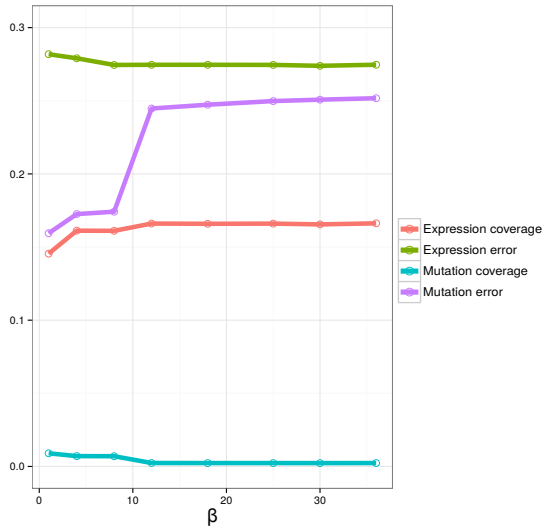


Figure 5.10: Varying the value of β while fixing $\theta_1 = 86\%, \theta_2 = 65\%, k = 8$

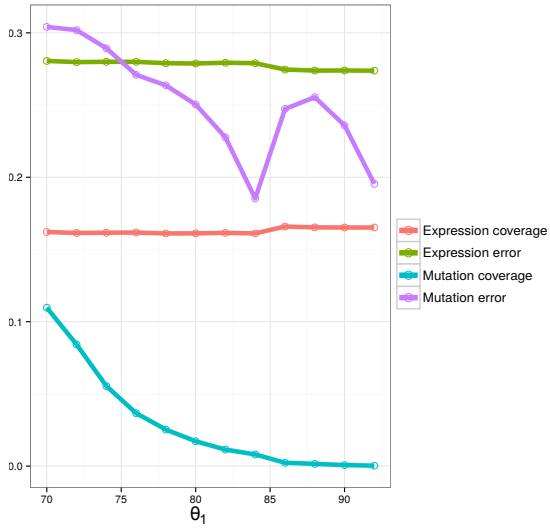


Figure 5.11: Varying the value of θ_1 while fixing $\theta_2 = 65\%, \beta = 18, k = 8$

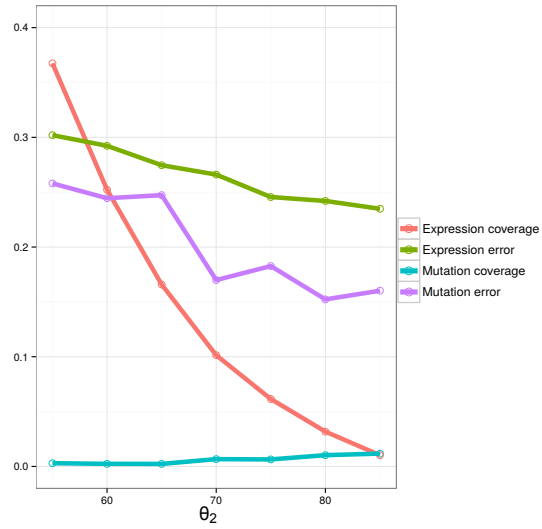


Figure 5.12: Varying the value of θ_2 while fixing $\beta = 18, \theta_1 = 86\%, k = 8$

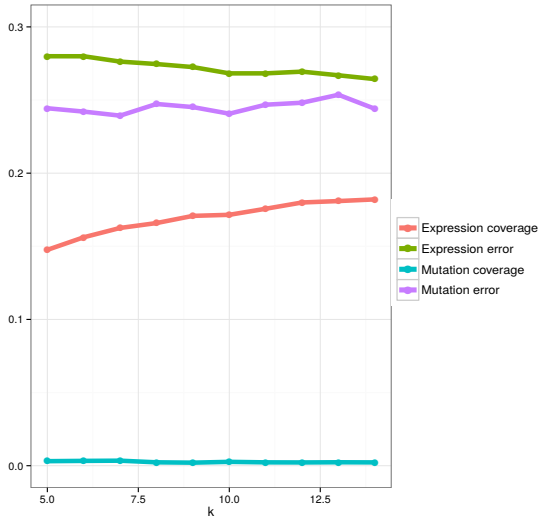


Figure 5.13: Varying k while fixing $\beta = 18, \theta_1 = 86\%, \theta_2 = 65\%$

settings. Note that the two scores were calculated for both rank matrices.

Gene feature selection To compare SRF to other methods concerning the recovery of subtype-specific genes in the simulated datasets, we used gene feature selection. With our method, it was straightforward to extract the mutation genes and expression genes representative of the individual subtypes, as described in Section 5.2. With iCluster+ [84], we used a quantile cut-off of $p = 0.75$ to select the important genes according to the model. With SNF [130], we first ordered the genes by *Normalized Mutual Information* using the SNF software. We then selected the top-n expression and the top-m mutation genes, where n and m are the total number of true expression and mutation genes of the simulated subtypes respectively. With NBS [49], we could in theory obtain subtype-specific mutation genes, but were not able to recover them given the lack of documentation. It is important to note that with both iCluster+ and SNF, all identified subtypes have the same set of mutation and expression genes.

Hierarchical clustering SRF requires an initialised matrix $\mathbf{F}$ to start from, which was obtained through hierarchical clustering: we used the *hclust* package in R to cluster the columns of the ranked expression matrix into k groups (with Euclidean distance).

Pruning small subtypes To be more tolerant towards noise, derived subtypes that contain less samples than a predefined threshold (less than 4% of the total number of samples for the simulated datasets) were pruned. Samples of the pruned subtypes were re-assigned to the remaining subtype that results in the highest score for the function in Equation 5.2.

Survival and pathway enrichment analysis Survival analysis was performed using the R *survival* package. We used time to follow, time to event and the subtype information produced by our algorithm to calculate the survival probability. Pathway enrichment analysis was done using the ClueGo plugin [8] in Cytoscape [103].

5.5 Discussion

Previous integrative models, such as iCluster+ [84] and SNF [130], used between-sample similarities from the sample's global expression/mutational profiles to derive subtypes. However, molecular subtypes are defined by the molecular mechanisms that drive carcinogenesis. How subtypes are defined thus depends on the features used to group samples in subtypes. Conversely, the subtypes define which features are relevant for a certain sample grouping. Hence, subtyping and

feature identification are confounded problems that ideally should be solved simultaneously.

In this work we therefore developed SRF, an approach that does so. To this end we approached the subtyping problem by decomposing patient–mutation and patient–expression data into ranked factors. A factor here represents a set of samples for which a set of genes display mutational consistency and a (possibly overlapping) second set of genes display expression consistency. A factor thus is an expressed and mutational feature set shared by a group of samples, and can be viewed as a *bi-cluster* [77] in respectively the expression and mutation data that are coupled in the patient dimension. We developed a *global model* in the form of matrix factorisation to identify these bi-clusters.

Subtypes are then defined as each patient set that is covered by a unique combination of ranked factors. As a result subtypes can overlap in the factors that characterise them, reflecting the fact that subtypes are never mechanistically completely different, but share common representative features/driver mechanisms.

Compared to state-of-the-art methods, our method is most related to [49] 1) as it uses a network model to account for pathway level parallelism between independently evolved tumour samples and 2) because it extracts features explicitly. However, it is different from [49] by integrating both mutational and expression data.

Compared to related methods, samples without clear-cut signals will not be assigned to any subtypes. This prevents samples (noisy or heterogeneous samples) from blurring the molecular characteristics that are representative for a subtype. However, if desired we could assign samples to the closest subtype identified by our method.

In this work, we did not consider metabolomics data, as was done in the work by [115]. The type of data described in [115] can only be generated for cell lines and not for a tumour biopsy. It is not available in the context of TCGA or ICGC.

5.6 Conclusions

In this chapter, we study the ranked tiling problem in a multi-view setting, in which we would like to find sets of two ranked tiles that reside in two different rank matrices and that are coupled through the same subset of the columns. This setting stems from a real bioinformatics problem called cancer subtyping, which aims at integrating different molecular data types, including mutation

data, gene expression data and prior knowledge encoded in biological networks, to discover groups of tumour samples that have the same molecular origins of their cancer development.

We approach the cancer subtyping problem by 1) defining the concept of driver pathways using the ranked tiling pattern; 2) derive cancer subtypes from the driver pathways using the rule that a cancer subtype is a group of samples that have a unique combination of driver pathways. Our approach leads to simultaneous discovery of cancer subtypes and subtype specific features.

We formalise the task of discovering k driver pathways by defining a joint rank matrix factoristation problem, which jointly factorises the ranked version of the mutation data and the ranked version of the expression data. We extensively tested the performance of our method on simulated data. Testing and comparing our method with other state-of-the-art subtyping methods on the well studied TCGA breast cancer dataset shows how our method is able to grasp the most prominent signatures in the data. In addition, however, our method is also able to capture subtle differences that are missed by methods that compare samples based on global profiles of similarities.

Chapter 6

Conclusions and Future Work

This chapter summarises the discoveries and the contributions of the thesis, and discusses opportunities for future works.

6.1 Summary and conclusions

In this thesis, we studied rank data. We find that rank data naturally occurs in many applications and it is a useful abstraction of numeric data. In many cases, particularly when data consists of incomparable rows and the relative information is more important than the absolute values, for example, high/low gene expression values, transforming numeric data into rank data may result in a more informative representation.

We addressed the pattern set mining problem in rank data, i.e., the problem of discovering a small set of rank patterns that together globally describe the structure of the data. To support the discovery of such sets of patterns, we developed a matrix factorisation framework using semiring theory. We successfully applied the framework to model and solve two new data mining problems, namely, ranked tiling and Sparse RMF, which we introduced while conducting research in this thesis. We also demonstrated that rank patterns are generally interesting and useful in practice. For example, we were able to use ranked tiling to link genotype information, such as mutations and copy number variations, with molecular phenotype downstream, such as mRNA expression, for the discovery of cancer subtypes and subtype features.

Now we review the three research questions that we raised in the introduction chapter and the answers that we contributed.

Q1 *How can we model rank pattern set mining?*

We contribute a semiring rank matrix factorisation framework for pattern set mining in rank data. The novelty of the framework lies in the definition of the matrix product using the semiring theory rather than the linear algebra. This makes the framework handle rank data more easily (with fewer constraints) than conventional methods using linear algebra. This also leads to improved solution qualities, for instance, higher overlaps.

The framework is open to discover different types of patterns in rank data. To mine a specific pattern, we proposed to use the two factorised matrices to define the patterns of interest by constraining the values of these matrices as well as an appropriate scoring function to measure the quality of the factorisation.

The framework aims at local patterns. Hence, the reconstructed matrix of the factorisation might deviate from the original matrix as long as the patterns produced by the factorisation can capture the main structures of the data.

Q2 *Which types of rank patterns are potentially interesting and useful?*

We contribute the introductions of two new rank patterns, namely, Sparse RMF and ranked tiling. Sparse RMF studies how to discover a small set of sparse rank vectors that can be used to succinctly summarise a given rank matrix. Ranked tiling studies how to discover a small set of data rectangles in a rank matrix that have high ranks and together cover the matrix as much as possible. Ranked tiling reveals local associations between subsets of the rows and subsets of the columns. We also demonstrate how we can model and solve the two data mining problems using the single proposed framework.

The experiments maintain that the newly introduced patterns appear in many domains, for example, social science (the European Song Festival dataset), artificial intelligence (the SUSHI dataset), cancer biology (the TCGA breast cancer dataset). In addition, the discovered patterns from the datasets are generally intersesting and are capable of giving insights about the structure of the data, which results in useful knowledge.

Q3 *How useful are rank patterns in real life applications?*

We apply the framework to discover Sparse RMF and ranked tiling patterns in real-life datasets. In general, the discovered patterns reveal interesting categories of rankings in the data. For example, we discover interesting ways of rankings sushis in the SUSHI dataset [54], which illustrates the existence of different groups of customers. Such type of knowledge may be useful for

business purposes.

Importantly, we successfully apply the ranked tiling pattern to study the cancer subtyping problem. Ranked tiling is used to formally define the concept of mutated driver pathways, from which we can derive cancer subtypes. Applying our model in the TCGA breast cancer dataset, we could discover specific mutation and expression features of breast cancer subtypes, which may shed a light into the molecular origins of the disease subtypes.

6.2 Future work

This thesis concerns multidisciplinary research between computer science and bioinformatics. The cancer biology problem inspired us to advance data mining algorithms. In return, advances in data mining modelling and solving techniques helped us to solve the problem that matters. Obviously, this loop helps to improve both fields. Our future work will continue this philosophy.

Exploring other types of rank patterns and their applications. Up till now, we have used the sRMF framework to study two types of rank patterns, namely, Sparse RMF and ranked tiling. In the future, we will further explore other types of rank patterns that are potentially interesting, as well as verify the broad applicability of the framework. We believe new patterns can appear when we extend the current framework to new settings, for instance, supervised pattern mining, network-based pattern mining, temporal pattern mining and relational pattern mining. To have a concrete example, let's imagine that we have a heterogeneous dataset consisting of different types of molecular information for a number of tumour samples, which now have a label, for example, drug response or drug resistance. The problem is to discover molecular information which can be used to discriminate the two groups of the samples. This problem belongs to the class of supervised pattern mining. In this setting, the ranked tiling pattern is not suitable any more. However, we can extend the ranked tiling work to have a new pattern that has the discriminative feature required for this type of problem.

Other solving strategies. In this thesis, we only consider discrete optimisation models, in which decision variables are constrained to have discrete values. Solving discrete optimisation problems is in general harder than solving convex optimisation problems. Hence, it is worthy of considering convex relaxations for the optimisation models studied in this thesis and possible consequences of the relaxation. For example, applying the relaxation for Sparse mRMF might require us to re-consider the regularisation using the coverage

(Equation 3.4 on page 28) as the coverage matrix now easily becomes dense and might not have the regularisation effect.

Other interesting directions might include a scalable algorithm that can directly solve the optimisation problems for Sparse RMF and mRMT without using the EM-style iterating method.

Continuous values for rank data. In this thesis, we only consider rank data of which values are discrete. However, rank data can also be represented by numeric values. These two types of representations have been considered in statistics. For example, Mallows model [80] uses discrete values to represent rankings while Random Utility Theory [108] and Placket-Luce models [76, 92, 43, 13] consider numeric values. These two directions are both of equal importance.

Dealing with unknown rankings. In this thesis, we assume that we have complete rankings. However, in practice, we can encounter datasets that do not have complete rankings. For example, in rating datasets, users often do not provide full rankings for all of the items. How to improve the current framework to have a probabilistic model that can deal with unknown rankings is an interesting direction for future work.

Subjective interestingness for rank pattern set mining. In this thesis, we focus on discovering rank patterns that can cover the data as much as possible. Though the discovered patterns can reveal hidden data regularities, it may be the case that the patterns are not much surprising with respect to users' knowledge. There has been research that models users' subjective interestingness for pattern mining such as the work by De Bie [20]. How to incorporate users' subjective interestingness in the rank factorisation framework to mine rank patterns is also an interesting direction for future work.

Developing model selections. In practice, we see that it is quite hard to select the right parameters for a given data. Sometimes, we still have to use prior knowledge of the data miner to select appropriate values for the parameters. Inspired from previous work on parameter-free data mining algorithms, for example the work in [127], the question is whether we can develop model selections for sRMF instances, such as Sparse RMF and ranked tiling?

Data mining and machine learning for precision oncology. One of the prominent features of our proposed model for cancer subtyping compared to the others is the capability of extracting subtype specific features, including mutational features and expression features. This is essential for identifying core driver features (pathways) that can be used to describe cancer subtypes and to predict clinical phenotypes. However there are still many open questions that have not been explored:

- What are the core driver pathways that can be used to described all possible cancer subtypes in different types of human tissues?
- Are there any relations among the core driver pathways?
- Given the discovered core driver pathways, can we build (probabilistic) machine learning programs to predict clinical phenotypes accurately?
- How do the core driver pathways change when tumours evolve and develop drug resistance?

To conclude, the rank matrix factorisation framework has proven useful and can be extended in various interesting directions.

Bibliography

[1] Agarwal, S. On ranking and choice models. In *Proc. of the 25th International Joint Conference on Artificial Intelligence (IJCAI-16)*, pp. 4050–4053.

[2] Aggarwal, C. C., and Han, J., Eds. *Frequent Pattern Mining*. Springer, 2014.

[3] Agrawal, R., Imieliński, T., and Swami, A. Mining association rules between sets of items in large databases. In *Proc. of the 1993 ACM SIGMOD International Conference on Management of Data (SIGMOD-93)*, pp. 207–216.

[4] Alvo, M., and Yu, P. L. *Statistical methods for ranking data*. Springer, 2014.

[5] Awasthi, P., Blum, A., Sheffet, O., and Vijayaraghavan, A. Learning mixtures of ranking models. In *Advances in Neural Information Processing Systems (NIPS-14)*, pp. 2609–2617.

[6] Banerjee, A., Dhillon, I., Ghosh, J., Merugu, S., and Modha, D. S. A generalized maximum entropy approach to Bregman co-clustering and matrix approximation. *Journal of Machine Learning Research 8* (2007), 1919–1986.

[7] Ben-Dor, A., Chor, B., Karp, R., and Yakhini, Z. Discovering local structure in gene expression data: the order-preserving submatrix problem. *Journal of Computational Biology 10*, 3-4 (2003), 373–384. DOI: 10.1089/10665270360688075.

[8] Bindea, G., Mlecnik, B., Hackl, H., Charoentong, P., et al. ClueGO: a Cytoscape plug-in to decipher functionally grouped gene ontology and pathway annotation networks. *Bioinformatics 25*, 8 (2009), 1091–1093. DOI: 10.1093/bioinformatics/btp101.

[9] Brunet, J. P., Tamayo, P., Golub, T. R., and Mesirov, J. P. Metagenes and molecular pattern discovery using matrix factorization. *Proc. of the National Academy of Sciences (PNAS) 101* (2004), 4164–4169. DOI: 10.1073/pnas.0308531101.

[10] Busse, L. M., Orbanz, P., and Buhmann, J. M. Cluster analysis of heterogeneous rank data. In *Proc. of the 24th International Conference on Machine Learning (ICML-07)*, pp. 113–120. DOI: 10.1145/1273496.1273511.

[11] Calders, T., Goethals, B., and Jaroszewicz, S. Mining rank-correlated sets of numerical attributes. In *Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-06)*, pp. 96–105. DOI: 10.1145/1150402.1150417.

[12] Cancer Genome Atlas Network. Comprehensive molecular portraits of human breast tumours. *Nature 490*, 7418 (2012), 61–70. DOI: 10.1038/nature11412.

[13] Caron, F., and Teh, Y. Bayesian nonparametric models for ranked data. In *Advances in Neural Information Processing (NIPS-12)*, pp. 1520–1528.

[14] Chen, S. S., Donoho, D. L., and Saunders, M. A. Atomic decomposition by basis pursuit. *SIAM Review. 43*, 1 (2001), 129–159. DOI: 10.1137/S003614450037906X.

[15] Cheng, Y., and Church, G. M. Biclustering of expression data. *Proc. of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB-00) 8* (2000), 93–103.

[16] Chierichetti, F., Dasgupta, A., Kumar, R., and Lattanzi, S. On learning mixture models for permutations. In *Proc. of the 2015 Conference on Innovations in Theoretical Computer Science (ITCS-15)*, pp. 85–92. DOI: 10.1145/2688073.2688111.

[17] Cichocki, A., Zdunek, R., Phan, A. H., and Amari, S.-i. *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation.* Wiley Publishing, 2009.

[18] Csardi, G., and Nepusz, T. The igraph software package for complex network research. *International Journal of Complex Systems* (2006), 1695.

[19] Curtis, C., Shah, S. P., Chin, S.-F., Turashvili, G., et al. The genomic and transcriptomic architecture of 2,000 breast tumours reveals novel subgroups. *Nature 486*, 7403 (2012), 346–352. DOI: 10.1038/nature10983.

[20] De Bie, T. Maximum entropy models and subjective interestingness: an application to tiles in binary databases. *Data Mining and Knowledge Discovery 23*, 3 (2011), 407–446. DOI: 10.1007/s10618-010-0209-3.

[21] De Maeyer, D., Weytjens, B., De Raedt, L., and Marchal, K. Network-based analysis of eQTL data to prioritize driver mutations. *Genome Biology and Evolution* (2016). DOI: 10.1093/gbe/evw010.

[22] De Raedt, L., Guns, T., and Nijssen, S. Constraint programming for itemset mining. In *Proc. of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-08)*, pp. 204–212. DOI: 10.1145/1401890.1401919.

[23] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. Indexing by latent semantic analysis. *Journal of the American Society for Information Science 41*, 6 (1990), 391–407.

[24] Deng, K., Han, S., Li, K. J., and Liu, J. S. Bayesian aggregation of order-based rank data. *Journal of the American Statistical Association 109*, 507 (2014), 1023–1039. DOI: 10.1080/01621459.2013.878660.

[25] Dhillon, I. S., Mallela, S., and Modha, D. S. Information-theoretic co-clustering. In *Proc. of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-03)*, pp. 89–98. DOI: 10.1145/956750.956764.

[26] Diaconis, P. *Group representations in probability and statistics*. Lecture notes-monograph series. Institute of Mathematical Statistics, 1988.

[27] Ding, C., He, X., and Simon, H. D. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proc. of the 2005 SIAM International Conference on Data Mining (SDM-05)*, pp. 606–610. DOI: 10.1137/1.9781611972757.70.

[28] Ding, C., Li, T., and Peng, W. On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing. *Computational Statistics and Data Analysis 52*, 8 (2008), 3913–3927. DOI: 10.1016/j.csda.2008.01.011.

[29] Ding, W., Ishwar, P., and Saligrama, V. A topic modeling approach to ranking. In *Proc. of the 18th International Conference on Artificial Intelligence and Statistics (AISTATS-15)*.

[30] Drineas, P., Frieze, A., Kannan, R., Vempala, S., and Vinay, V. Clustering large graphs via the Singular Value Decomposition. *Machine Learning 56*, 1-3 (2004), 9–33. DOI: 10.1023/B:MACH.0000033113.59016.96.

[31] Dwork, C., Kumar, R., Naor, M., and Sivakumar, D. Rank aggregation methods for the Web. In *Proc. of the 10th International Conference on World Wide Web (WWW-01)*, pp. 613–622. DOI: 10.1145/371920.372165.

[32] Eldén, L. *Matrix Methods in Data Mining and Pattern Recognition*. Society for Industrial and Applied Mathematics, 2007.

[33] England, G. *The 100,000 Genomes Project*, 2016 (accessed September 20, 2016). `https://www.genomicsengland.co.uk/the-100000-genomes-project/`.

[34] Frünkranz, J., and Hüllermeier, E. *Preference Learning*. Springer, 2010.

[35] G. Golub, W. K. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics: Series B, Numerical Analysis 2*, 2 (1965), 205–224.

[36] Gaussier, E., and Goutte, C. Relation between PLSA and NMF and implications. In *Proc. of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-05)*, pp. 601–602. DOI: 10.1145/1076034.1076148.

[37] Geerts, F., Goethals, B., and Mielikäinen, T. Tiling databases. In *Proc. of the 7th International Conference on Discovery Science (DS-04)*, pp. 278–289. DOI: 10.1007/978-3-540-30214-8_22.

[38] Givoni, I., Cheung, V., and Frey, B. Matrix tile analysis. In *Proc. of the 22rd Conference on Uncertainty in Artificial Intelligence (UAI-06)*.

[39] Golan, J. S. *Semirings and Their Applications*. Kluwer, Dordrecht, 1999.

[40] Goldman, M., Craft, B., Swatloski, T., Ellrott, K., Cline, M., et al. The UCSC cancer genomics browser: update 2013. *Nucleic Acids Research 41*, D1 (2013), D949–D954. DOI: 10.1093/nar/gks1008.

[41] Golub, G. H., and Van Loan, C. F. *Matrix Computations*. Johns Hopkins University Press, 1996.

[42] Gu, J., and Liu, J. S. Bayesian biclustering of gene expression data. *BMC Genomics 9 Suppl 1* (2008). DOI: 10.1186/1471-2164-9-S1-S4.

[43] Guiver, J., and Snelson, E. Bayesian inference for Plackett-Luce ranking models. In *Proc. of the 26th Annual International Conference on Machine Learning (ICML-09)*, pp. 377–384.

[44] Guns, T., Nijssen, S., and De Raedt, L. Itemset mining: a constraint programming perspective. *Artificial Intelligence 175*, 12 (2011), 1951 – 1983. DOI: 10.1016/j.artint.2011.05.002.

[45] Henriques, R., and Madeira, S. C. BicPAM: pattern-based biclustering for biomedical data analysis. *Algorithms for Molecular Biology 9*, 1 (2014), 1–30. DOI: 10.1186/s13015-014-0027-z.

[46] Henzgen, S., and Hüllermeier, E. Mining rank data. In *Proc. of the 17th International Conference on Discovery Science (DS-14)*, pp. 123–134.

[47] HGP. *The Human Geome Project*, 2016 (accessed September 20, 2016). `https://www.genome.gov/12011238/an-overview-of-the-human-genome-project/`.

[48] Hochreiter, S., Bodenhofer, U., Heusel, M., Mayr, A., et al. FABIA: factor analysis for bicluster acquisition. *Bioinformatics 26*, 12 (2010), 1520–7. DOI: 10.1093/bioinformatics/btq227.

[49] Hofree, M., Shen, J. P., Carter, H., Gross, A., and Ideker, T. Network-based stratification of tumor mutations. *Nature Methods 10*, 11 (2013), 1108–15. DOI: 10.1038/nmeth.2651.

[50] Hoyer, P. O. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research 5* (2004), 1457–1469.

[51] ICGC. *The International Cancer Genome Consortium*, 2016 (accessed September 20, 2016). `http://icgc.org/`.

[52] Ihmels, J., Friedlander, G., Bergmann, S., Sarig, O., Ziv, Y., and Barkai, N. Revealing modular organization in the yeast transcriptional network. *Nature Genetics 31*, 4 (2002), 370–7. DOI: 10.1038/ng941.

[53] Kaci, S. *Working with Preferences: Less Is More*. Springer, 2011.

[54] Kamishima, T. Nantonac collaborative filtering: recommendation based on order responses. In *Proc. of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-03)*, pp. 583–588. DOI: 10.1145/956750.956823.

[55] Kanehisa, M., and Goto, S. KEGG: Kyoto Encyclopedia of genes and genomes. *Nucleic Acids Research 28*, 1 (2000), 27–30. DOI: 10.1093/nar/28.1.27.

[56] Karaev, S., and Miettinen, P. Cancer: another algorithm for subtropical matrix factorization. In *Proc. of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD-16)*.

[57] Karaev, S., and Miettinen, P. Capricorn : An algorithm for subtropical matrix factorization. In *Proc. of the 2016 SIAM International Conference on Data Mining (SDM-16)*.

[58] Kemp, C., Tenenbaum, J. B., Griffiths, T. L., Yamada, T., and Ueda, N. Learning systems of concepts with an infinite relational model. In *Proc. of the 21st National Conference on Artificial Intelligence (AAAI-06)*, pp. 381–388.

[59] Kendall, M. *Rank Correlation Methods*. Griffin, London, 1948.

[60] Khta, M. U., and Ali, R. M. Probability models on horse-race outcomes. *Journal of Applied Statistics 25*, 2 (1998), 221–229. DOI: 10.1080/02664769823205.

[61] Kim, K. H. *Boolean Matrix Theory and Applications*. Dekker (1982), 1982.

[62] Kluger, Y., Basri, R., Chang, J. T., and Gerstein, M. Spectral biclustering of microarray data: coclustering genes and conditions. *Genome Research 13* (2003), 703–716. DOI: 10.1101/gr.648603.

[63] Kontonasios, K.-N., and De Bie, T. An information-theoretic approach to finding informative noisy tiles in binary databases. In *Proc. of the 2010 SIAM International Conference on Data Mining (SDM-10)*, pp. 153–164. DOI: 10.1137/1.9781611972801.14.

[64] Koren, Y., Bell, R., and Volinsky, C. Matrix factorisation techniques for recommender systems. *Computer 42*, 8 (2009), 30–37. DOI: 10.1109/MC.2009.263.

[65] Le Van, T., Fierro, A. C., Guns, T., van Leeuwen, M., Nijssen, S., De Raedt, L., and Marchal, K. Mining local staircase patterns in noisy data. In *Proc. of the 12th International Conference on Data Mining Workshops (ICDM-12)*, pp. 139–146. DOI: 10.1109/ICDMW.2012.83.

[66] Le Van, T., Nijssen, S., van Leeuwen, M., and De Raedt, L. Semiring rank matrix factorisation. *IEEE Transactions on Knowledge and Data Engineering Under revision*.

[67] Le Van, T., van Leeuwen, M., Fierro, A. C., De Maeyer, D., Van den Eynden, J., Verbeke, L., De Raedt, L., Marchal, K., and Nijssen, S. Simultaneous discovery of cancer subtypes and subtype features by molecular data integration. *Bioinformatics 32* (2016), i445–i454. DOI: 10.1093/bioinformatics/btw434.

[68] Le Van, T., van Leeuwen, M., Nijssen, S., and De Raedt, L. Rank Matrix Factorisation. In *Proc. of the 19th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-15)*, pp. 734–746. DOI: 10.1007/978-3-319-18038-0_57.

[69] Le Van, T., van Leeuwen, M., Nijssen, S., Fierro, A. C., Marchal, K., and De Raedt, L. Ranked tiling. In *Proc. of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD-14) (2)* (2014), pp. 98–113. DOI: 10.1007/978-3-662-44851-9_7.

[70] Ledford, H. End of cancer-genome project prompts rethink. *Nature 517* (2015), 128–9. DOI: 10.1038/517128a.

[71] Lee, D. D., and Seung, H. S. Learning the parts of objects by non-negative matrix factorization. *Nature 401*, 6755 (1999), 788–791. DOI: 10.1038/44565.

[72] Leiserson, M. D. M., Vandin, F., Wu, H.-T., Dobson, J. R., et al. Pan-cancer network analysis identifies combinations of rare somatic mutations across pathways and protein complexes. *Nature Genetics 47*, 2 (2014), 106–114. DOI: 10.1038/ng.3168.

[73] Lin, S. Rank aggregation methods. *Wiley Interdisciplinary Reviews: Computational Statistics 2*, 5 (2010), 555–570. DOI: 10.1002/wics.111.

[74] Lisa, B., Friederike, M., Petra, S., Andreas, S., et al. Intrinsic breast cancer subtypes defined by estrogen receptor signalling - prognostic relevance of progesterone receptor loss. *Modern Pathology 26*, 9 (2013), 1161–1171. DOI: 10.1038/modpathol.2013.60.

[75] Liu, T.-Y. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval 3*, 3 (2009), 225–331. DOI: 10.1561/1500000016.

[76] Luce, R. D. *Individual Choice Behavior: A Theoretical Analysis*. Wiley, 1959.

[77] Madeira, S. C., and Oliveira, A. L. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Transactions on*

Computational Biology and Bioinformatics 1, 1 (2004), 24–45. DOI: 10.1109/TCBB.2004.2.

[78] Mairal, J., Bach, F., Ponce, J., and Sapiro, G. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research 11* (2009), 19–60.

[79] Mannila, H., and Toivonen, H. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery 1*, 3 (1997), 241–258. DOI: 10.1023/A:1009796218281.

[80] Marden, J. I. *Analyzing and Modeling Rank Data*. Chapman & Hall, 1995.

[81] Mermel, C., Schumacher, S., Hill, B., Meyerson, M., et al. GISTIC2.0 facilitates sensitive and confident localization of the targets of focal somatic copy-number alteration in human cancers. *Genome Biology 12*, 4 (2011), R41. DOI: 10.1186/gb-2011-12-4-r41.

[82] Miettinen, P., Mielikainen, T., Gionis, A., Das, G., and Mannila, H. The discrete basis problem. *IEEE Transactions on Knowledge and Data Engineering 20*, 10 (2008), 1348–1362. DOI: 10.1109/TKDE.2008.53.

[83] Mischel, P. S., Shai, R., Shi, T., et al. Identification of molecular subtypes of glioblastoma by gene expression profiling. *Oncogene 22*, 15 (2003), 2361–73. DOI: 10.1038/sj.onc.1206344.

[84] Mo, Q., Wang, S., Seshan, V. E., Olshen, et al. Pattern discovery and cancer gene identification in integrated cancer genomic data. *Proc. of the National Academy of Sciences (PNAS) 110*, 11 (2013), 4245–50. DOI: 10.1073/pnas.1208949110.

[85] Murphy, T. B., and Martin, D. Mixtures of distance-based models for ranking data. *Computational Statistics & Data Analysis 41*, 3–4 (2003), 645 – 655. DOI: 10.1016/S0167-9473(02)00165-2.

[86] Nijssen, S., Vreeken, J., Zimmermann, A., Tatti, N., and Bringmann, B. 2011 ICDM tutorial Mining Sets of Patterns - Next Generation Pattern Mining, 2011. Available from `http://usefulpatterns.org/msop/`.

[87] OscaR Team. OscaR: Scala in OR, 2012. Available from `https://bitbucket.org/oscarlib/oscar`.

[88] Paatero, P., and Tapper, U. Positive matrix factorization: a non-negative factor model with optimal utilization of error estimates

of data values. *Environmetrics 5*, 2 (1994), 111–126. DOI: 10.1002/env.3170050203.

[89] Parker, J. S., Mullins, M., Cheang, M. C. U., Leung, S., et al. Supervised risk predictor of breast cancer based on intrinsic subtypes. *Journal of Clinical Oncology 27*, 8 (2009), 1160–7. DOI: 10.1200/JCO.2008.18.1370.

[90] Pearson, K. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine 2*, 6 (1901), 559–572.

[91] Perou, C. M., Sørlie, T., Eisen, M. B., van de Rijn, M., et al. Molecular portraits of human breast tumours. *Nature 406*, 6797 (2000), 747–52. DOI: 10.1038/35021093.

[92] Placket, R. L. The analysis of permutations. *Journal of the Royal Statistical Society 24* (1975), 193–202.

[93] Pratanwanich, N., Lió, P., and Stegle, O. Warped matrix factorisation for multi-view data integration. In *Proc. of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD-16)*, pp. 789–804. DOI: 10.1007/978-3-319-46227-1_49.

[94] Rajaraman, A., and Ullman, J. D. *Mining of Massive Datasets*. Cambridge University Press, 2011.

[95] Rossi, F., Venable, K. B., and Walsh, T. A short introduction to preferences: Between artificial intelligence and social choice. *Synthesis Lectures on Artificial Intelligence and Machine Learning 5*, 4 (2011), 1–102. DOI: 10.2200/S00372ED1V01Y201107AIM014.

[96] Rosvall, M., and Bergstrom, C. T. Maps of random walks on complex networks reveal community structure. *Proc. of the National Academy of Sciences (PNAS) 105*, 4 (2008), 1118–1123. DOI: 10.1073/pnas.0706851105.

[97] Sanchez-Garcia, F., Villagrasa, P., Matsui, J., Kotliar, D., et al. Integration of genomic data enables selective discovery of breast cancer drivers. *Cell 159*, 6 (2014), 1461–1475. DOI: 10.1016/j.cell.2014.10.048.

[98] Schein, A. I., Saul, L. K., and Ungar, L. H. A generalized linear model for principal component analysis of binary data. In *Proc. of the 9th International Workshop on Artificial Intelligence and Statistics* (2003).

[99] Schutter, B. D., and Moor, B. D. The singular-value decomposition in the extended max algebra. *Linear Algebra and its Applications 250* (1997), 143 – 176.

[100] Seppänen, J. K., Bingham, E., and Mannila, H. A simple algorithm for topic identification in 0–1 data. In *Proc. of the 7th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD-03)*, pp. 423–434. DOI: 10.1007/978-3-540-39804-2_38.

[101] Serin, A., and Vingron, M. DeBi: discovering differentially expressed biclusters using a frequent itemset approach. *Algorithms for Molecular Biology 6*, 1 (2011), 1–12. DOI: 10.1186/1748-7188-6-18.

[102] Shan, H., and Banerjee, A. Bayesian co-clustering. In *Proc. of the 8th IEEE International Conference on Data Mining (ICDM-08)*, pp. 530–539. DOI: 10.1109/ICDM.2008.91.

[103] Shannon, P., Markiel, A., Ozier, O., Baliga, N. S., et al. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Research 13* (2003), 2498–2504. DOI: 10.1101/gr.1239303.

[104] Sheng, Q., Moreau, Y., and De Moor, B. Biclustering microarray data by Gibbs sampling. *Bioinformatics 19*, suppl 2 (2003), ii196–ii205. DOI: 10.1093/bioinformatics/btg1078.

[105] Skillicorn, D. *Understanding Complex Datasets: Data Mining with Matrix Decompositions*. Chapman & Hall/CRC, 2007.

[106] Sohn Emily. Diagnosis: a clear answer. *Nature 537*, 7619 (2016), S64–S65. DOI: 10.1038/537S64a 10.1038/537S64a.

[107] Sørlie, T., Perou, C. M., Tibshirani, R., Aas, T., et al. Gene expression patterns of breast carcinomas distinguish tumor subclasses with clinical implications. *Proc. of the National Academy of Sciences (PNAS) 98*, 19 (2001), 10869–74. DOI: 10.1073/pnas.191367098.

[108] Soufiani, H. A., Parkes, D. C., and Xia, L. Random utility theory for social choice. In *Advances in Neural Information Processing Systems (NIPS-12)*, pp. 126–134.

[109] Speicher, N. K., and Pfeifer, N. Integrating different data types by regularized unsupervised multiple kernel learning with application to cancer subtype discovery. *Bioinformatics 31*, 12 (2015), i268–i275. DOI: 10.1093/bioinformatics/btv244.

[110] Stephens, P. J., Tarpey, P. S., Davies, H., Van Loo, P., et al. The landscape of cancer genes and mutational processes in breast cancer. *Nature 486*, 7403 (2012), 400–4. DOI: 10.1038/nature11017.

[111] Sun, S. A survey of multi-view machine learning. *Neural Computing and Applications 23*, 7 (2013), 2031–2038. DOI: 10.1007/s00521-013-1362-6.

[112] Szklarczyk, D., Franceschini, A., Kuhn, M., Simonovic, M., et al. The STRING database in 2011: functional interaction networks of proteins, globally integrated and scored. *Nucleic Acids Research 39*, suppl 1 (2011), D561–D568. DOI: 10.1093/nar/gkq973.

[113] Tan, P.-N., Steinbach, M., and Kumar, V. *Introduction to Data Mining*. Addison-Wesley, 2005.

[114] Tanay, A., Sharan, R., and Shamir, R. Discovering statistically significant biclusters in gene expression data. *Bioinformatics 18*, Suppl. 1 (2002), S136–S144.

[115] Tardito, S., Oudin, A., Ahmed, S. U., Fack, F., et al. Glutamine synthetase activity fuels nucleotide biosynthesis and supports growth of glutamine-restricted glioblastoma. *Nature Cell Biology 17*, 12 (dec 2015), 1556–1568. DOI: 10.1038/ncb3272.

[116] TCGA. *The Cancer Genome Atlas*, 2016 (accessed September 20, 2016). `http://cancergenome.nih.gov/`.

[117] The Cancer Genome Atlas Network. Comprehensive molecular portraits of human breast tumours. *Nature 490*, 7418 (2012), 61–70. DOI: 10.1038/nature11412.

[118] Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B 58* (1994), 267–288.

[119] Toss, A., and Cristofanilli, M. Molecular characterization and targeted therapeutic approaches in breast cancer. *Breast Cancer Research 17*, 1 (2015), 60. DOI: 10.1186/s13058-015-0560-9.

[120] Tothill, R. W., Tinker, A. V., George, J., Brown, R., et al. Novel molecular subtypes of serous and endometrioid ovarian cancer linked to clinical outcome. *Clinical Cancer Research 14*, 16 (2008), 5198–5208. DOI: 10.1158/1078-0432.CCR-08-0196.

[121] Truong, D. T., Battiti, R., and Brunato, M. Discovering non-redundant overlapping biclusters on gene expression data. In *Proc. of the 13th IEEE International Conference on Data Mining (ICDM-13)*, pp. 747–756. DOI: 10.1109/ICDM.2013.36.

[122] Turner, H., Bailey, T., and Krzanowski, W. Improved biclustering of microarray data demonstrated through systematic performance tests. *Computational Statistics & Data Analysis 48*, 2 (2005), 235–254. DOI: 10.1016/j.csda.2004.02.003.

[123] Ukkonen, A., and Mannila, H. Finding outlying items in sets of partial rankings. In *Proc. of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD-07)*, pp. 265–276.

[124] Vanunu, O., Magger, O., Ruppin, E., Shlomi, T., and Sharan, R. Associating genes and protein complexes with disease via network propagation. *PLoS Computational Biology 6*, 1 (2010), e1000641. DOI: 10.1371/journal.pcbi.1000641.

[125] Verbeke, L. P. C., Van den Eynden, J., Fierro, A. C., Demeester, P., Fostier, J., and Marchal, K. Pathway relevance ranking for tumor samples through network-based data integration. *PLoS ONE 10*, 7 (2015), 1–22. DOI: 10.1371/journal.pone.0133503.

[126] Vogelstein, B., Papadopoulos, N., Velculescu, V. E., Zhou, S., et al. Cancer genome landscapes. *Science 339*, 6127 (2013), 1546–1558. DOI: 10.1126/science.1235122.

[127] Vreeken, J., van Leeuwen, M., and Siebes, A. Krimp: mining itemsets that compress. *Data Mining and Knowledge Discovery 23*, 1 (2011), 169–214. DOI: 10.1007/s10618-010-0202-x.

[128] Žitnik, M., and Zupan, B. Data fusion by matrix factorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence 37*, 1 (2015), 41–53. DOI: 10.1109/TPAMI.2014.2343973.

[129] Wall, M. E., Rechtsteiner, A., and Rocha, L. M. *Singular Value Decomposition and Principal Component Analysis*. Springer, 2003, pp. 91–109. DOI: 10.1007/0-306-47815-3_5.

[130] Wang, B., Mezlini, A. M., Demir, F., Fiume, M., Tu, Z., Brudno, M., Haibe-Kains, B., and Goldenberg, A. Similarity network fusion for aggregating data types on a genomic scale. *Nature Methods 11*, 3 (2014), 333–7. DOI: 10.1038/nmeth.2810.

[131] Wang, F., Li, T., Wang, X., Zhu, S., and Ding, C. Community discovery using nonnegative matrix factorization. *Data Mining and Knowledge Discovery 22*, 3 (2011), 493–521. DOI: 10.1007/s10618-010-0181-y.

[132] Wang, H.-Q., Zheng, C.-H., and Zhao, X.-M. jNMFMA: a joint non-negative matrix factorization meta-analysis of transcriptomics data. *Bioinformatics 31*, 4 (2014), 572–580. DOI: 10.1093/bioinformatics/btu679.

[133] Wang, P., Laskey, K. B., Domeniconi, C., and Jordan, M. I. Nonparametric bayesian co-clustering ensembles. In *Proc. of the 2011 SIAM International Conference on Data Mining (SDM-11)*, pp. 331–342.

[134] Xu, C., Tao, D., and Xu, C. A survey on multi-view learning. *CoRR abs/1304.5634* (2013).

[135] Xu, W., Liu, X., and Gong, Y. Document clustering based on non-negative matrix factorization. In *Proc. of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval (SIGIR-03)*, pp. 267–273.

[136] Xu, Z., Tresp, V., Yu, K., and Kriegel, H.-P. Infinite hidden relational models. In *Proc of the 22nd International Conference on Uncertainity in Artificial Intelligence (UAI-06)*, pp. 544—-551.

[137] Yang, Z., and Michailidis, G. A non-negative matrix factorization method for detecting modules in heterogeneous omics multi-modal data. *Bioinformatics 32* (2015), btv544. DOI: 10.1093/bioinformatics/btv544.

[138] Yuan, Y., Savage, R. S., and Markowetz, F. Patient-specific data fusion defines prognostic cancer subtypes. *PLoS Computational Biology 7*, 10 (2011), e1002227. DOI: 10.1371/journal.pcbi.1002227.

[139] Zack, T. I., Schumacher, S. E., Carter, S. L., Cherniack, et al. Pan-cancer patterns of somatic copy number alteration. *Nature Genetics 45*, 10 (2013), 1134–1140. DOI: 10.1038/ng.2760.

List of publications

Journal articles

- Le Van, T., Nijssen, S., van Leeuwen, M., and De Raedt, L. Semiring rank matrix factorisation. *IEEE Transactions on Knowledge and Data Engineering, Under revision.*

- Le Van, T., van Leeuwen, M., Fierro, A. C., De Maeyer, D., Van den Eynden, J., Verbeke, L., De Raedt, L., Marchal, K., and Nijssen, S. Simultaneous discovery of cancer subtypes and subtype features by molecular data integration. *Bioinformatics 32* (2016), i445– i454. DOI: 10.1093/bioinformatics/btw434.

Peer-reviewed conference and workshop papers

- Le Van, T., van Leeuwen, M., Nijssen, S., and De Raedt, L. Rank Matrix Factorisation. In *Proc. of the 19th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-15)*, pp. 734–746. DOI: 10.1007/978-3-319-18038-0_57.

- Le Van, T., van Leeuwen, M., Nijssen, S., Fierro, A. C., Marchal, K., and De Raedt, L. Ranked tiling. In *Proc. of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD-14) (2)* (2014), pp. 98–113. DOI: 10.1007/978-3-662-44851-9_7.

- Le Van, T., Fierro, A. C., Guns, T., van Leeuwen, M., Nijssen, S., De Raedt, L., and Marchal, K. Mining local staircase patterns in noisy data. In *Proc. of the 12th International Conference on Data Mining Workshops (ICDM-12)*, pp. 139–146. DOI: 10.1109/ICDMW.2012.83.

Meeting abstracts

- Le Van, T., Van den Eynden, J., De Maeyer, D., Verbeke, L., Fierro Gutiérrez, A., van Leeuwen, M., Nijssen, S., De Raedt, L., Marchal, K. (2015). Ranked tiling based approach to discovering patient subtypes. *Benelux Bioinformatics Conference (BBC-15)*. Antwerp - Belgium, 7-8 December 2015. (*accepted for oral presentation*).
- Le Van, T., Fierro Gutiérrez, A., Guns, T., van Leeuwen, M., Nijssen, S., De Raedt, L., Marchal, K. (2013). Bi-clustering gene expression data under constraints. *Benelux Bioinformatics Conference (BBC-13)*. Brussels - Belgium, 9-10 December 2013.

Curriculum Vitae

Thanh Le Van (*Lê Văn Thành* in Vietnamese) was born in a suburb of Sai Gon (HoChiMinh City), Vietnam, in November 11, 1978. He received his Bachelor degree in Computer Science from Ho Chi Minh City University of Technology in 2001. During the period from 2001 to 2010, he worked for VietNam Post and Telecomunication (VNPT) in HoChiMinh City as a software engineer. In 2005, he won a two-year scholarship from VNPT for his graduate study at the Asian Institute of Technology (AIT), Thailand. He obtained his master degree in Information and Communication Technology from AIT in 2007. His master thesis was entitled "Data mining for financial aid optimisation" and was supervised by Prof. Peter Haddawy.

In November 2010, he joined the Machine Learning group, KU Leuven, Belgium, as a pre-doctoral student. In February 2012, he started his PhD research in the same group. His research is about declarative methods for data mining using Constraint Programming and Integer Programming, matrix factorisation for pattern set mining in rank data and its applications in bioinformatics.

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
DTAI LAB
Celestijnenlaan 200A box 2402
B-3001 Heverlee
thanh.levan@cs.kuleuven.be
http://dtai.cs.kuleuven.be