Bayesian matrix factorisation: inference, priors, and data integration

Thomas Alexander Brouwer

Computer Laboratory
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

Homerton College

December 2017

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 60,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Thomas Alexander Brouwer
December 2017

Acknowledgements

In many ways this thesis has been the hardest thing I have done in my life. Fortunately, I have always been surrounded by amazing people that have helped me through it.

First of all I would like to thank my supervisor, Pietro Lió, for giving me the opportunity to do the PhD, providing me with research ideas over the years, and his unequivocal support for everything I did.

Over the years there have been several times when I lost motivation. Somehow, I always ended up meeting inspirational researchers who were able to show me the merits of my efforts. Dr Naruemon Pratanwanich introduced me to Bayesian probabilistic models, and provided me with many interesting discussions. Professor Jes Frellsen has taught me how to do machine learning research. His energy and positivity are an inspiration to me. Professor Samuel Kaski provided me much-needed enthusiasm for the research that I was conducting, and convinced me it was worthwile. I also want to thank his students for the incredibly useful discussions and insights during my visits to his lab, in particular Muhammad Ammad-ud-din, Eemeli Leppäaho, and Iiris Sundin.

I am grateful for the many friends I made in Cambridge, the Homerton MCR, and elsewhere in the world. They got me through the PhD, and Cambridge would not have been the same without them. Being on the Homerton MCR Committee was one of the best decisions I made while in Cambridge. I want to thank Aaron, Alex, Alice, Amber, Camilla, Ezra, Emily, Fryer, Hannah, Harry, Jack, Joe, Joost, Lucy, Luke, Melanie, Sam, Steph, Vera, and many more.

Above all I would like to thank my parents, Hans and Daphne, and my sister, Laura. Whenever I was stressed, or needed to complain, they would be there for me. They have always supported me in whatever I do, and I could not wish for a better family.

Abstract

In recent years the amount of biological data has increased exponentially. Most of these data can be represented as matrices relating two different entity types, such as drug-target interactions (relating drugs to protein targets), gene expression profiles (relating drugs or cell lines to genes), and drug sensitivity values (relating drugs to cell lines). Not only the size of these datasets is increasing, but also the number of different entity types that they relate. Furthermore, not all values in these datasets are typically observed, and some are very sparse.

Matrix factorisation is a popular group of methods that can be used to analyse these matrices. The idea is that each matrix can be decomposed into two or more smaller matrices, such that their product approximates the original one. This factorisation of the data reveals patterns in the matrix, and gives us a lower-dimensional representation. Not only can we use this technique to identify clusters and other biological signals, we can also predict the unobserved entries, allowing us to prune biological experiments.

In this thesis we introduce and explore several Bayesian matrix factorisation models, focusing on how to best use them for predicting these missing values in biological datasets. Our main hypothesis is that matrix factorisation methods, and in particular Bayesian variants, are an extremely powerful paradigm for predicting values in biological datasets, as well as other applications, and especially for sparse and noisy data. We demonstrate the competitiveness of these approaches compared to other state-of-the-art methods, and explore the conditions under which they perform the best.

We consider several aspects of the Bayesian approach to matrix factorisation. Firstly, the effect of inference approaches that are used to find the factorisation on predictive performance. Secondly, we identify different likelihood and Bayesian prior choices that we can use for these models, and explore when they are most appropriate. Finally, we introduce a Bayesian matrix factorisation model that can be used to integrate multiple biological datasets, and hence improve predictions. This model hybridly combines different matrix factorisation models and Bayesian priors. Through these models and experiments we support our hypothesis and provide novel insights into the best ways to use Bayesian matrix factorisation methods for predictive purposes.

Table of contents

List of figures

List of tables

Summary of notation

The following list describes the different notational characters used in this thesis.

Probability distributions

$\mathcal{E}$ Exponential distribution.

$\mathcal{G}$ Gamma distribution.

$\mathcal{IG}$ Inverse Gaussian distribution.

$\mathcal{L}$ Laplace distribution.

Mult Multinomial distribution.

$\mathcal{N}$ Normal or Gaussian distribution.

$\mathcal{NIW}$ Normal-inverse Wishart distribution.

$\mathcal{P}$ Poisson distribution.

$\mathcal{TN}$ Truncated normal distribution.

Chapter 3

$\boldsymbol{R}$ The main matrix we decompose using matrix factorisation. I rows, J columns. R_{ij} denotes the entry in the ith row and jth column of $\boldsymbol{R}$.

Ω Mask set indicating the observed entries in $\boldsymbol{R}$. (i, j) is in Ω if R_{ij} is observed.
Ω_i indicates observed entries for row i. $j \in \Omega_i$ if R_{ij} is observed.
Ω_j indicates observed entries for column j. $i \in \Omega_j$ if R_{ij} is observed.

$\boldsymbol{U}$ Row factor matrix for matrix factorisation. I rows, K columns.

$\boldsymbol{V}$ Column factor matrix for matrix factorisation. J rows, K columns.

τ Noise random variable.

α_τ, β_τ The hyperparameter for the Gamma prior over the noise random variable τ.

$\lambda_U, \lambda_V, \lambda_F, \lambda_S, \lambda_G$ Hyperparameters for the entries in the factor matrices $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{F}, \boldsymbol{S}, \boldsymbol{G}$.

λ_k Automatic relevance determination random variable for the kth factor.
$\boldsymbol{\lambda}$ denotes the vector containing $\lambda_1, .., \lambda_K$.
For matrix tri-factorisation we use λ_k^F for $\boldsymbol{F}$ and λ_l^G for $\boldsymbol{G}$.

α_0, β_0 Hyperparameters for the Gamma prior over λ_k.

Chapter 4

λ Hyperparameter for the entries in the factor matrices $\boldsymbol{U}, \boldsymbol{V}$ for models GGG, GGGU, GEE, GTT, GL_1^2, and GEG.

η Hyperparameter for the entries in the factor matrices $\boldsymbol{U}, \boldsymbol{V}$ for model GLL.

γ Hyperparameter for the entries in the factor matrix $\boldsymbol{U}$ for models GVG, GVnG.

a, b Hyperparameter for the entries in the factor matrices $\boldsymbol{U}, \boldsymbol{V}$ for model PGG.

$\boldsymbol{\mu_0}, \beta_0, \nu_0, \boldsymbol{W_0}$ Hyperparameter for the $\boldsymbol{\mu_U}, \boldsymbol{\Sigma_U}, \boldsymbol{\mu_V}, \boldsymbol{\Sigma_V}$ random variables with normal-inverse Wishart prior for model GGGW.

μ, λ Hyperparameter for the η_{ik}^U, η_{jk}^V random variables with inverse Gaussian prior for model GLLI.

μ_μ, τ_μ, a, b Hyperparameter for the $\mu_{ik}^U, \tau_{ik}^U, \mu_{jk}^V, \tau_{jk}^V$ random variables for model GTTN.

a', b' Hyperparameter for the h_i^U, h_j^V random variables with Gamma prior for model PGGG.

Chapter 5

E_t Entity types, having I_t instances and K_t factors.

$\boldsymbol{F}_t$ Factor matrix for entity type E_t.

$\boldsymbol{R}^n$ Main dataset for hybrid matrix factorisation, relating entity types E_{t_n} and E_{u_n}. I_{t_n} rows and I_{u_n} columns.

$\boldsymbol{D}^l$ Feature dataset for hybrid matrix factorisation, relating entity type E_{t_l} to a number of features. I_{t_l} rows and J_l columns.

$\boldsymbol{C}^m$ Similarity dataset for hybrid matrix factorisation, relating entity type E_{t_m} to itself. I_{t_m} rows columns.

$\boldsymbol{S}^n, \boldsymbol{G}_l, \boldsymbol{S}^m$ Dataset-specific factor matrices for matrix (tri-)factorisation of $\boldsymbol{R}^n, \boldsymbol{D}^l, \boldsymbol{C}^m$, respectively.

τ^n, τ^l, τ^m Noise parameters for datasets $\boldsymbol{R}^n, \boldsymbol{D}^l, \boldsymbol{C}^m$, respectively.

$\alpha^n, \alpha^l, \alpha^m$ Importance value hyperparameters for datasets $\boldsymbol{R}^n, \boldsymbol{D}^l, \boldsymbol{C}^m$, respectively.

$\Omega^n, \Omega^l, \Omega^m$ Mask sets indicating the observed entries in $\boldsymbol{R}^n, \boldsymbol{D}^l, \boldsymbol{C}^m$, respectively.

λ_S^n, λ_S^m Hyperparameters for the entries in the factor matrices $\boldsymbol{S}^n, \boldsymbol{S}^m$, respectively.

$\lambda_k^t, \lambda_k^{t_l}$ Automatic relevance determination random variable for the kth factor of factor matrices $\boldsymbol{F}^t, \boldsymbol{G}^l$, respectively.

U_1^t, U_2^t Sets indicating the datasets $\boldsymbol{R}^n$ that have entity type E_t as their rows or columns, respectively.

V^t Sets indicating the datasets $\boldsymbol{D}^l$ that have entity type E_t as their rows. V_+^t gives those that use nonnegative factors for $\boldsymbol{G}^l$, and V_-^t real-valued.

W^t Sets indicating the datasets $\boldsymbol{C}^m$ that have entity type E_t as their rows and columns.

Chapter 1

Introduction

In recent years the datasets that can be studied and analysed have exploded both in size and in complexity. Especially in bioinformatics, the rise of high-throughput methods gives us a complex landscape of datasets relating genes, drugs, proteins, cells, and many more. Human experts can no longer analyse and fully understand the ever growing amount of data themselves. The solution: computational methods that can assist these experts in their search.

A typical example is the development of drugs. Bringing a new drug to the market takes around 15 years (Dimasi [2001]) and costs nearly a billion dollars (Adams and Brantner [2006]). The efficacy of drug development, if measured as the number of new drugs approved per dollar spent, has significantly declined in recent years (Booth and Zemmel [2004]). In order to address this problem, we should exploit the vast amounts of biological data that have become available in recent years, in the form of -omics data—genomics, transcriptomics, proteomics, epigenomics, and so on. These data can reveal the biological working of the drugs and diseases that we are considering, and hence allow us to enhance the drug development process. Especially in the earlier stages of drug research, where there often is a large amount of possible drugs or diseases, our search can be pruned using this information. The use of computational methods that can efficiently consider different types of data is essential, as the amount of data has grown too large for humans to process.

Importantly, many of these datasets are actually incomplete. Say we are given a matrix relating the effectiveness of a range of drugs on different diseases, as measured by a

biologist in a wet laboratory. For a thousand drugs and a thousand diseases we already have a million potential combinations—which makes it almost infeasible to measure (or observe) them all. As the number of drugs and diseases increases, and therefore the number of potential combinations, this becomes even less feasible. As a result these datasets are often very sparse (few observed entries). Another problem is that the measurements for these values tend to be very noisy, and repeating an experiment can give a different value as a result. Fortunately, we can use machine learning methods to address these challenges. If we measure say ten thousand of the entries, we can use these to predict the other values and use that to prune what experiments to run next. Furthermore, if these predictions are accurate we can try to analyse how the method does this, yielding novel biological insights or biomarkers.

A popular group of methods for predicting these missing values in incomplete datasets is matrix factorisation. The idea is that we can approximate the incomplete matrix of values by the product of two smaller matrices (a so-called low-rank approximation). We try to find the values of these two smaller matrices in such a way that their product closely resembles the observed entries, which also allows us to obtain predictions for unobserved ones. One popular way of doing this is by minimising a cost function, such as the mean squared error (Lee and Seung [2000]). Alternatively, Bayesian methods treat it as a probabilistic problem where we place a prior distribution over a set of random variables (here, the two smaller matrices) and aim to find the distribution over them after observing the (incomplete) dataset (Salakhutdinov and Mnih [2008]). Matrix factorisation is a growing and very promising field of machine learning, and lends itself extremely well for integrating many datasets (Gligorijević and Pržulj [2015]).

In this thesis we focus on the latter approach: Bayesian matrix factorisation. We believe that these methods are much more capable of providing accurate predictions of missing values, especially as the datasets that we study become more sparse, compared to the non-probabilistic approach to matrix factorisation, as well as other state-of-the-art machine learning methods such as random forests and support vector machines.

We support this hypothesis with empirical evidence on several real-world applications, as well as providing thorough and systematic explorations of the different choices we can make for our Bayesian matrix factorisation models, studying how they impact our predictive performances. We focus on relatively small datasets that allow us to try lots of different model choices, yet are commonly used in practice. Although a lot of current research is looking at theoretical properties and guarantees of statistical

and machine learning methods, we are more interested in the practical usage and performance of the methods on real-world datasets. We also briefly consider the effects of the model choices on our ability to analyse the resulting factor matrices, but our focus is predictive performance. We believe that this knowledge can advance the field of Bayesian matrix factorisation by providing novel insights, guiding future researchers in their design and understanding of the models.

1.1 Research questions

Our main hypothesis is that matrix factorisation methods, and in particular Bayesian variants, are an extremely powerful paradigm for predicting missing values in biological datasets, as well as other applications, especially for sparse and noisy data. The goals of this thesis are demonstrating the competitiveness of these approaches compared to other state-of-the-art methods, and exploring the conditions under which they perform the best.

When developing a new Bayesian matrix factorisation model, a researcher has to make several choices. First of all, if we have multiple datasets, what is the best way of jointly factorising them, and what factor matrices should be shared? The next challenge is to choose the most appropriate likelihood and Bayesian priors for the model, which can influence predictive performance (especially for sparse datasets) and our ability to analyse the resulting factor matrices. Finally, a method of inference has to be chosen. This pipeline is shown in Figure 1.1. There are numerous choices for each challenge, each with different advantages and disadvantages, but very few studies have been performed that explore the trade-offs and help researchers make the best choices.

In this thesis we address the above challenges by answering the following research questions, one in each chapter, which also help us confirm our hypothesis that Bayesian matrix factorisation is a very effective method for predicting missing values.

(i) What influence does the inference approach have on predictive performance? In particular, how does it impact convergence speed and the model's robustness to noise and sparsity?

(ii) What are the trade-offs between different likelihood and Bayesian prior choices for the matrix factorisation models?

(iii) How can we best integrate multiple datasets to improve our predictions?

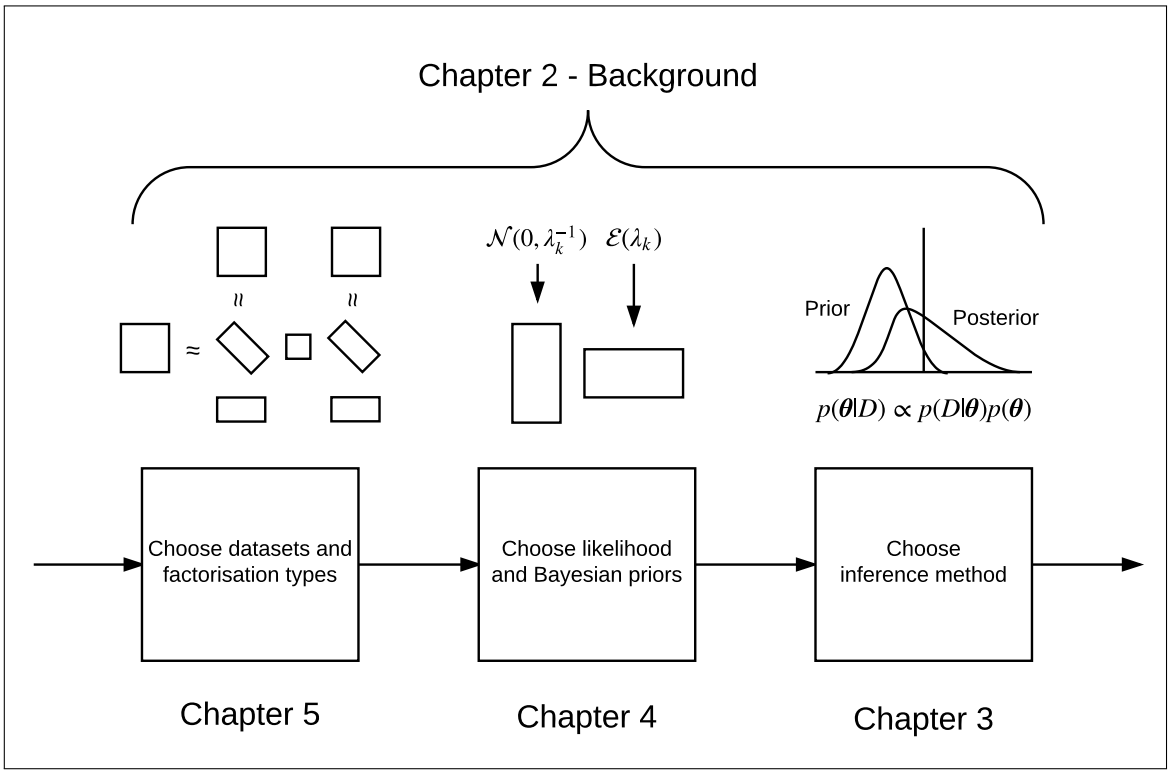


Figure 1.1 Research problems that are addressed in this thesis, split into three fundamental modelling challenges. First of all, given multiple datasets that we wish to predict missing values for, what is the best way of jointly decomposing them? We study this by introducing a general hybrid matrix factorisation model. Secondly, what are the best choices for the likelihood function and Bayesian prior choices for the latent factor matrices? Finally, given the model definition, what are the trade-offs of different inference approaches to solve our problem? We address these questions in order of increasing complexity, first studying inference approaches, then the prior and likelihood choices, and finally introducing a data integration model.

Note that this is the reverse of the design choice order outlined before, and in order of increasing complexity. If we started with studying the best way to jointly decompose multiple datasets, we would not yet know the effects of the inference method on predictive performance. This makes it harder to isolate the effect of one design choice for the next. We therefore address them in reverse order, using knowledge gained in one chapter to make modelling choices in the next.

1.2 Thesis overview

This thesis is structured as follows.

Chapter 2 introduces the theoretical foundations underlying this thesis. In particular, we briefly introduce probabilistic latent variable models and Bayesian inference. Using this knowledge, we review the literature on matrix factorisation methods and their extensions—we focus on nonnegative and Bayesian matrix factorisation; matrix tri-factorisation; multiple matrix factorisation; and tensor decompositions. Finally, we discuss our evaluation methods including nested cross-validation, and introduce the different applications that we consider in this thesis.

Chapter 3 studies the trade-offs of different inference approaches for Bayesian matrix factorisation and tri-factorisation. Starting with a Bayesian nonnegative matrix factorisation model, we extend it to matrix tri-factorisation, and add automatic relevance determination to both models to perform automatic model selection. We consider four different inference methods: non-probabilistic inference, Gibbs sampling, a maximum-a-posteriori approach called iterated conditional modes, and variational Bayesian inference—the last of which is new for these two models. We then present experimental results on synthetic data, as well as four drug sensitivity datasets, studying the convergence and runtime speed, predictive performance in cross-validation, robustness to noise and sparsity, and the effectiveness of automatic relevance determination for model selection.

Chapter 4 provides a study of the effects of different likelihood and Bayesian prior choices for Bayesian matrix factorisation on predictive performance. We review popular approaches and identify four different groups: real-valued, nonnegative, semi-nonnegative, and Poisson matrix factorisation. We consider three different application domains—drug sensitivity predictions, collaborative filtering of movie ratings, and methylation expression profiles—and measure the convergence speed and depth, predictive performance in cross-validation, robustness to sparsity and noise, and factor values on these applications. Using these results we discuss the trade-offs of the different groups and provide novel insights, which can help guide design choices for future models.

Chapter 5 considers the problem of integrating multiple datasets at the same time, by jointly factorising them. We introduce a novel Bayesian model called hybrid matrix factorisation, which hybridly combines multiple matrix factorisation and tri-factorisation, as well as real-valued, nonnegative, and semi-nonnegative factorisations. This model significantly improves in-matrix prediction performances on the drug sensitivity datasets, outperforming state-of-the-art matrix factorisation and machine

learning approaches. For out-of-matrix predictions this model provides excellent performances as well, often outperforming methods like random forests and support vector machines. Finally, we study different parameter choices for our model, including the factorisation types that are used to integrate the datasets.

Chapter 6 summarises the contributions of this thesis, and discusses future research directions.

1.3 Related publications

(a) Thomas Brouwer, Jes Frellsen, and Pietro Lió (2017). Comparative Study of Inference Methods for Bayesian Nonnegative Matrix Factorisation. *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2017).*

(b) Thomas Brouwer and Pietro Lió (2017). Prior and Likelihood Choices for Bayesian Matrix Factorisation on Small Datasets. *Under review.*

(c) Thomas Brouwer and Pietro Lió (2017). Bayesian Hybrid Matrix Factorisation for Data Integration. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS 2017).*

The work presented in publication (a) establishes the content of Chapter 3. An earlier version of this work was presented at the *NIPS 2016 Workshop on Advances in Approximate Bayesian Inference.* The work in Chapter 4 is under review at a conference. Finally, (c) contributes to the content of Chapter 5.

We consciously made an effort to make all code and datasets freely available online for each the publications above—not just for the model implementations, but also the experiments and preprocessing of the datasets. We hope that this will simplify replication and future projects, and contributes to a future of more open research. They can be accessed at at https://github.com/ThomasBrouwer.

Chapter 2

Background

In this thesis we extensively study Bayesian probabilistic models for matrix factorisation. In this chapter, we review the theory behind probabilistic graphical models, Bayesian inference approaches, matrix factorisation methods, and several applications we will consider to validate our methods.

2.1 Probabilistic latent variable models

Latent variable models are a type of stochastic models in which we have a number of variables that we observe, and some that we do not observe—the latent variables. Over each of these variables we define probability distributions that are dependent on a number of other variables in the model, thus defining a generative process that we can use to generate new samples for the collection of random variables. Because we cannot observe the latent variables, we can see our model as a hypothesis of how the data were generated. For example, if we define a latent variable model for a biological problem, our model can try to simulate the biological process that we cannot observe (in other words, the latent process), but is considered to be the cause of what we observe. If we are then given a set of actual observations, we can try to infer what the values or distributions were that most likely generated these data from the specified model. This can give us more precise details of the underlying biology.

Formally we can define probabilistic latent variable models as follows. We have a set of observable variables X, a set of actual observations D (the data), and a set of latent variables $\boldsymbol{\theta}$. For each latent variable $\theta_i \in \boldsymbol{\theta}$ and observable variable $x \in X$, we define a

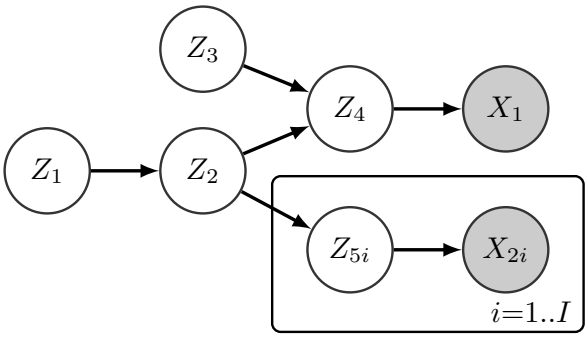


Figure 2.1 Example of a graphical model, detailing the conditional independencies between observed variables X_1, X_2 and latent variables Z_1, Z_2, Z_3, Z_4.

probability distribution that determines the values it can take. These distributions have a number of parameters—for example, a Gaussian distribution has a mean μ and precision τ. We can place distributions over these parameters as well, giving rise to a hierarchical structure of random variables that are dependant on each other. Parameters in a probability distribution that itself do not have a distribution over them (for example, if we did not place a prior over μ) are called *hyperparameters*, and are not random variables.

The dependency structure induced by the hierarchical probability distributions can be exploited to make inference easier, by making a distribution conditionally independent of the other variables given only a few, which we call the *parent nodes*. We can then represent the conditional independencies of the observed and latent variables as a so-called **graphical model**. Each observed variable is represented by a grey node, and each latent one by a white node. A directed arrow $A_1 \to A_2$ indicates that the distribution of A_2 is conditionally dependent on that of A_1; in other words, A_1 is a parent node of A_2. Finally, plates denote repeated variables, each having a subscript to identify it. A parent node of a variable inside the plates is therefore the parent of each repeated variable (for example if $A_1 \to B_i$, where B_i is inside a plate with $i = 1..I$, then $A_1 \to B_1$, $A_1 \to B_2$, and so on).

As an example, in Figure 2.1 we have latent variables $Z_1, Z_2, Z_3, Z_4, Z_{51}, .., Z_{5I}$ and observed variables $X_1, X_{21}, .., X_{2I}$. Note that the flow of the diagram goes in only one direction, allowing us to generate samples from the model: first drawing $Z_1 \sim p(Z_1)$, then $Z_2 \sim p(Z_2|Z_1)$, and so on.

The only parent node of X_1 is Z_4, and the only parent of X_{2n} is Z_{5n} (for each $n = 1..I$), and therefore

$$p(X_1|Z_1, Z_2, Z_3, Z_4, Z_{51}, .., Z_{5I}, X_{21}, .., X_{2I}) = p(X_1|Z_4)$$
$$p(X_{2n}|Z_1, Z_2, Z_3, Z_4, Z_{51}, .., Z_{5I}, X_{21}, .., X_{2(n-1)}, X_{2(n+1)}, .., X_{2I}) = p(X_{2n}|Z_{5n})$$

Note however that variables are not conditionally independent of their children nodes, as observing the value of a child will also yield information about the value of a parent. For example, $p(Z_4|Z_1, Z_2, Z_3) = p(Z_4|Z_2, Z_3)$, but $p(Z_4|Z_1, Z_2, Z_3, X_1)$ is not equal to $p(Z_4|Z_2, Z_3)$. Instead it is

$$p(Z_4|Z_1, Z_2, Z_3, X_1) = \frac{p(X_1|Z_1, Z_2, Z_3, Z_4)p(Z_4|Z_1, Z_2, Z_3)}{p(X_1|Z_1, Z_2, Z_3)} \tag{2.1}$$
$$\propto p(X_1|Z_4)p(Z_4|Z_2, Z_3). \tag{2.2}$$

We used conditional Bayes' theorem for the first step (2.1),

$$p(A|B, C) = \frac{p(B|A, C)p(A|C)}{p(B|C)},$$

and the conditional independence rules specified by the graphical model for the second (2.2). Note that the denominator does not depend on Z_4 and can therefore be left out.

As stated before, we wish to infer the values or distributions of the underlying generative model. In particular, we want to find the values (or distribution over values) of the latent variables $\boldsymbol{\theta}$. There are typically three types of solutions we can find:

- **Maximum likelihood (ML)**. For the ML solution, we simply wish to find the parameter values $\boldsymbol{\theta}$ that are most likely to have generated the data. In other words, finding $\boldsymbol{\theta}_{\text{ML}} = \max_{\boldsymbol{\theta}} p(D|\boldsymbol{\theta})$, where $p(D|\boldsymbol{\theta})$ is the probability of observing data D given the parameters values $\boldsymbol{\theta}$.

- **Maximum a posteriori (MAP)**. The MAP solution also incorporates a prior belief about how likely each of the parameter values is, before even seeing the data. This allows us to discount solutions that are very unlikely, for example containing extremely high values for a parameter θ_i. As a result, inference involves a trade-off between our prior belief of the parameter values, and finding parameters that make the model fit very well to the data. We find $\boldsymbol{\theta}_{\text{MAP}} =$

$\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|D) = \max_{\boldsymbol{\theta}} [p(D|\boldsymbol{\theta})p(\boldsymbol{\theta})]$, where $p(\boldsymbol{\theta})$ is the prior probability of having parameter values $\boldsymbol{\theta}$.

- **Full posterior (fully Bayesian)**. Finally, in the fully Bayesian approach to inference, we wish to find the posterior distribution $p(\boldsymbol{\theta}|D)$ over the parameters $\boldsymbol{\theta}$ given the data D. In contrast to ML and MAP, where we found a single point estimate, we now find a probability distribution, reducing the risk of getting stuck in a local minimum.

We investigate the trade-offs between these different approaches in Chapter 3, where we will see that ML methods are very prone to overfitting (see Section 2.4) to noise and sparsity of a dataset, with MAP methods remedying that to an extent, and the fully Bayesian approaches to inference being the most robust. Bayesian methods are therefore especially useful for biological datasets, where data tend to be noisy, and we have few observed datapoints.

2.2 Approximate Bayesian posterior inference

In Bayesian inference, we are interested in finding the posterior likelihood of the parameters. To do this, we use Bayes' theorem:

$$p(\boldsymbol{\theta}|D) = \frac{p(D|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(D)} = \frac{p(D|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int_{\boldsymbol{\theta}} p(D|\boldsymbol{\theta})p(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}}$$

The integral in the denominator goes over the entire parameter space (all possible values for $\boldsymbol{\theta}$) which is extremely high-dimensional, and as a result is usually intractable to compute. Furthermore, for a lot of models this integral will not reduce to a closed-form distribution (such as a Gaussian), and can therefore not be computed exactly anyways. To address this, we can resort to **approximate Bayesian inference** methods, which seek to approximate the true full posterior with one that we can more easily compute. In this thesis we make use of two of these approaches: Gibbs sampling, and variational Bayesian inference. Many others are also possible, including expectation maximisation, variants of Markov Chain Monte Carlo (Gibbs is one), and expectation propagation.

Both of these approaches require something called **model conjugacy**, meaning that the product of the prior distributions $p(\boldsymbol{\theta})$ and the likelihood $p(D|\boldsymbol{\theta})$ is of the same form of distribution as the prior distribution. Examples of this are a Gaussian likelihood and Gaussian prior, or a Multinomial likelihood and a Dirichlet prior over its distribution

vector. All probability distributions in the exponential family (which includes most common distributions) have a conjugate prior. Inference is also possible without model conjugacy, but often requires significant (and imprecise) approximations and complicated inference algorithms. In this thesis, we will focus on conjugate models to avoid this additional complexity so that we can focus entirely on the research questions, without worrying about potential influence of the approximations.

2.2.1 Gibbs sampling

Recall that our goal is to approximate the true posterior $p(\boldsymbol{\theta}|D)$. We may not be able to compute this distribution directly, but the idea behind Gibbs sampling is to try to sample values from it. These samples can then be used to approximate the posterior, or to estimate the expectation and variance of the parameters.

Gibbs sampling works by sampling new values for each parameter θ_i from its marginal distribution $p(\theta_i|\boldsymbol{\theta}_{-i}, D)$, given the current values of the other parameters $\boldsymbol{\theta}_{-i}$, and the observed data D. If we sample new values in turn for each parameter θ_i from $p(\theta_i|\boldsymbol{\theta}_{-i}, D)$, we will eventually converge to draws from the posterior, which can be used to approximate the posterior $p(\boldsymbol{\theta}|D)$. If our model has conjugacy, this conditional posterior $p(\theta_i|\boldsymbol{\theta}_{-i}, D)$ is of a closed form and we can sample from it. In order to obtain good approximations to the posteriors of the parameters, we have to discard the first n draws because it takes a while to converge (*burn-in*), and since consecutive draws are correlated we only use every ith value (*thinning*).

Intuitively, the idea is that by focusing on one dimension of the posterior parameter space (in other words, one parameter at a time) and sampling a new value from the conditional posterior, we are more likely to sample a value with a higher posterior probability than one with a lower one, and hence converge to high density posterior space by repeatedly doing this. By sampling randomly, we are less likely go get stuck in a bad (low density) posterior space, as there is always a chance that we "jump out" of the bad space and into a better one.

2.2.2 Variational Bayesian inference

Gibbs sampling relies on random sampling behaviour to obtain draws from the true posterior distribution. In contrast, the idea behind variational Bayesian inference (VB) is to introduce an approximation $q(\boldsymbol{\theta})$ to the true posterior (the form of which we can choose), and to make this variational distribution $q(\boldsymbol{\theta})$ as similar to $p(\boldsymbol{\theta}|D)$ as

possible (as measured by the KL-divergence). This new distribution has a number of variational parameters which we can change, and by fine-tuning these we can obtain a distribution that is very close to the true posterior, while being easier to compute. The KL-divergence is defined as

$$\begin{aligned} D_{KL}(q(\boldsymbol{\theta})||p(D|\boldsymbol{\theta})) &= \mathbb{E}_q\left[\log \frac{q(\boldsymbol{\theta})}{p(\boldsymbol{\theta}|D)}\right] \\ &= \mathbb{E}_q\left[\log q(\boldsymbol{\theta})\right] - \mathbb{E}_q\left[\log p(\boldsymbol{\theta}|D)\right] \\ &= \mathbb{E}_q\left[\log q(\boldsymbol{\theta})\right] - \mathbb{E}_q\left[\log p(D, \boldsymbol{\theta})\right] + \log p(D) \\ &= -\mathcal{L} + \log p(D), \end{aligned}$$

where $\mathbb{E}_q\left[f(x)\right]$ denotes the expectation of function $f(x)$, with respect to the distribution $q(x)$ over random variable x. Since $\log p(D)$ is independent of the variational approximation, maximising $\mathcal{L} = \mathbb{E}_q\left[\log p(\boldsymbol{\theta}, D) - \log q(\boldsymbol{\theta})\right]$ (the **evidence lower bound**, ELBO) is equivalent to minimising the KL-divergence. So if we write down the expression for the lower bound, and optimise it with respect to the variational parameters, our distribution q will approximate the true posterior $p(\boldsymbol{\theta}|D)$.

One way to do this is to write down the expression for the ELBO, take the derivative with respect to each variational parameter, set to zero, and solve it to find the optimal update. Alternatively, it can be shown (Beal and Ghahramani [2003]) that the optimal distribution for the ith parameter, $q^*(\theta_i)$, can be expressed as follows (for some constant C), allowing us to more easily find the optimal updates for the variational parameters.

$$\log q^*(\theta_i) = \mathbb{E}_{q(\boldsymbol{\theta}_{-i})}\left[\log p(\boldsymbol{\theta}, D)\right] + C.$$

We now take the expectation with respect to the distribution $q(\boldsymbol{\theta}_{-i})$, which goes over all parameters except the ith one, $\boldsymbol{\theta}_{-i}$. This gives rise to an iterative algorithm: for each parameter θ_i we update its distribution to that of its optimal variational distribution, and then update the expectation and variance with respect to q. We therefore need updates for the variational parameters, and to be able to compute the expectations and variances of the random variables.

We typically assume that the variational distribution $q(\boldsymbol{\theta})$ factorises completely, so that all variables are independent in the approximation of the posterior,

$$q(\boldsymbol{\theta}) = \prod_{\theta_i \in \boldsymbol{\theta}} q(\theta_i).$$

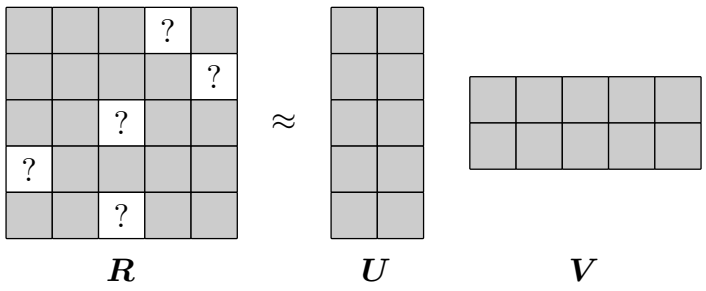


Figure 2.2 Matrix factorisation with $K = 2$ factors. Unobserved entries are indicated by question marks.

This is called the **mean-field assumption**, and it is what makes the variational approximation q easier to compute.

2.3 Matrix factorisation

Matrix factorisation is a group of methods that seek to extract patterns in a given dataset. Say we are given a matrix $\boldsymbol{R} \in \mathbb{R}^{I \times J}$ that related two entity types. We have I row entities (for example users), J columns (for example movies), and each entry R_{ij} gives the relation between row i and column j (such as the rating out of five stars that this user has given to the movie).

Matrix factorisation works on the assumption that we can find a low-level representation of the data, and it seeks to find these low-level patterns through factorisation. In particular, it assumes that each row i has K factor values $U_{i1}, ..., U_{iK}$ that give a low-dimensional representation of the row. Similarly, each column j has K factor values $V_{j1}, ..., V_{jK}$. The value R_{ij} representing the relation between row i and column j then comes from the dot product of these factor values,

$$R_{ij} = \boldsymbol{U}_i \cdot \boldsymbol{V}_j = \sum_{k=1}^{K} U_{ik} V_{jk}.$$

In other words, we decompose the matrix $\boldsymbol{R} \approx \boldsymbol{U}\boldsymbol{V}^T$, where $\boldsymbol{U} \in \mathbb{R}^{I \times K}$, $\boldsymbol{V} \in \mathbb{R}^{J \times K}$. Note that this decomposition is not exact—the datasets we analyse are often very noisy, and therefore we should only seek to approximate them, to prevent overfitting to the noise. There is also a trade-off for the number of factors, or dimensionality, K: if this value is too high we will overfit, and if it is too low we cannot fit enough to the

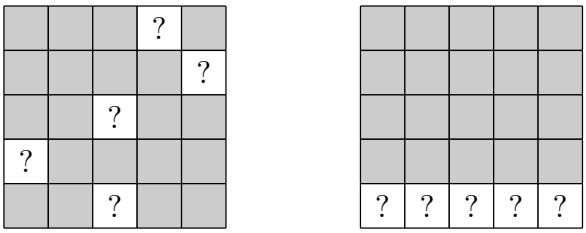


Figure 2.3 Difference between in- and out-of-matrix predictions.

data to find the right patterns. Choosing this value is called model selection, and can for example be done using cross-validation (see Section 2.4).

Not all the entries in the dataset $\boldsymbol{R}$ may be measured, as shown in Figure 2.2. We represent the indices of observed entries by the set

$$\Omega = \{(i, j) \mid R_{ij} \text{ observed}\}.$$

Note that in our example we have at least one other observed entry for each row and column that we can learn from. This is the so-called **in-matrix predictions** setting, and unknown values can be predicted using $\boldsymbol{U}\boldsymbol{V}^T$. The other setting is **out-of-matrix predictions**, where we predict values for entirely unseen rows or columns, such as a new user that has not yet rated any movies. This is illustrated in Figure 2.3. Note that this form of matrix factorisation can only be used for the former type of predictions, and not the latter, as we do not have information about the new row that we wish to predict values for. We will discuss extensions in Section 2.3.6 that allow us to make out-of-matrix predictions.

In the framework of movie ratings introduced above, we can interpret the factor values as capturing a grouping behaviour of users and movies. For example, a factor k might capture in matrix $\boldsymbol{U}$ whether each user likes horror movies, and in matrix $\boldsymbol{V}$ whether this is a horror movie. Therefore, if we then predict the value of an observed entry R_{ij}, and the factor value corresponding to horror movies is high for both the user and movies, we predict a high rating. Note that these factor values are learning in an unsupervised manner, and only by analysing them afterwards can we identify that a factor corresponds to such a grouping of users and movies.

The matrix factorisation problem as defined above uses real-valued factor values. This is sometimes also called **factor analysis**. A related approach is called **independent component analysis**, where we additionally aim to make the rows in the $\boldsymbol{V}$ factor matrix as independent as possible, according to a given similarity measure (such as entropy). There are many popular matrix factorisation models and extensions, which we will review in the following sections.

2.3.1 Nonnegative matrix factorisation

If we assume that the dataset $\boldsymbol{R}$ is nonnegative, and constrain the factor values to be nonnegative, we obtain **nonnegative matrix factorisation** (NMF). This nonnegativity has several advantages: it makes the resulting factor values easier and more intuitive to interpret; it is often inherent to the problem (such as in image analysis); and it can prevent overfitting (as we will see in Chapter 4). This method can also be interpreted as K-means clustering, where the $\boldsymbol{U}$ matrix gives the cluster indicators for the rows, and $\boldsymbol{V}$ gives the cluster centroids (Ding et al. [2005a]).

One of the earliest formulations can be found in Paatero [1997]; Paatero and Tapper [1994], but it was popularised by Lee and Seung [1999, 2000]. The last paper introduced two models based on different cost functions that we seek to minimise in order to find the factorisation, and the majority of papers that followed built on these two models. Lee and Seung [2000] formulated nonnegative matrix factorisation of a matrix $\boldsymbol{R} \in \mathbb{R}_+^{I \times J}$ as follows. We wish to find nonnegative matrices $\boldsymbol{U} \in \mathbb{R}_+^{I \times K}$ and $\boldsymbol{V} \in \mathbb{R}_+^{J \times K}$ such that $\boldsymbol{R} \approx \boldsymbol{U}\boldsymbol{V}$. We do this either by minimising the total square error of predictions, also called the Frobenius norm,

$$||\boldsymbol{R} - \boldsymbol{U}\boldsymbol{V}^T||_F^2 = \sum_{i=1}^{I} \sum_{j=1}^{J} \left(R_{ij} - \boldsymbol{U}_i \boldsymbol{V}_j\right)^2,$$

or by minimising the I-divergence (generalised KL-divergence),

$$D(\boldsymbol{R}||\boldsymbol{U}\boldsymbol{V}^T) = \sum_{i=1}^{I} \sum_{j=1}^{J} \left(R_{ij} \log \frac{R_{ij}}{(\boldsymbol{U}\boldsymbol{V}^T)_{ij}} - R_{ij} + (\boldsymbol{U}\boldsymbol{V}^T)_{ij} \right),$$

subject to the constraints $\boldsymbol{U}, \boldsymbol{V} \geq 0$. The following multiplicative updates can be shown to be correct and to converge using auxiliary functions, for the two cost functions

above (respectively),

$$U_{ik} = U_{ik} \cdot \frac{(\boldsymbol{RV})_{ik}}{(\boldsymbol{UV}^T\boldsymbol{V})_{ik}} \qquad V_{jk} = V_{jk} \cdot \frac{(\boldsymbol{R}^T\boldsymbol{U})_{jk}}{(\boldsymbol{VU}^T\boldsymbol{U})_{jk}}$$
$$U_{ik} = U_{ik} \cdot \frac{\sum_j V_{jk} R_{ij} / (\boldsymbol{UV}^T)_{ij}}{\sum_j V_{jk}} \qquad V_{jk} = V_{jk} \cdot \frac{\sum_i U_{ik} R_{ij} / (\boldsymbol{UV}^T)_{ij}}{\sum_i U_{ik}}$$

They showed that these multiplicative updates are essentially gradient descent updates where the step size is dependent on the current values of $\boldsymbol{U}, \boldsymbol{V}$. Note that for the first cost function it requires the entire matrix $\boldsymbol{R}$ to be observed, whereas for the latter we can modify the cost function to only consider entries in the set Ω (by replacing $\sum_{i=1}^{I} \sum_{j=1}^{J}$ with $\sum_{(i,j)\in\Omega}$) and the updates can be modified as well to reflect this.

These models were extensively applied for clustering and analysis of datasets in image analysis (Kong et al. [2011]; Li et al. [2001]; Shen and Si [2010]), bioinformatics (Brunet et al. [2004]; Kim and Choi [2007]), document analysis (Ding et al. [2006]; Kuang et al. [2012]; Pauca et al. [2004]), and many more fields. The base models are often extended by adding additional penalty terms to the cost function, for example L_1 or L_2 sparsity norms for the factor matrices.

2.3.2 Bayesian matrix factorisation

Probabilistic approaches to matrix factorisation describe the minimisation problem (finding matrices $\boldsymbol{U}, \boldsymbol{V}$) as that of trying to infer the distributions over latent variables $\boldsymbol{U}, \boldsymbol{V}$ after observing the data R. Bayesian approaches extend this by placing prior distributions over $\boldsymbol{U}, \boldsymbol{V}$. As discussed in Section 2.2, we can either try to infer a point estimate of the likelihood $\max_{\{\boldsymbol{U},\boldsymbol{V}\}} p(\boldsymbol{R}|\boldsymbol{U}, \boldsymbol{V})$ (maximum likelihood) or posterior $\max_{\{\boldsymbol{U},\boldsymbol{V}\}} p(\boldsymbol{U}, \boldsymbol{V}|\boldsymbol{R})$ (maximum a posteriori), or find the full posterior distribution $p(\boldsymbol{U}, \boldsymbol{V}|\boldsymbol{R})$.

Bayesian factor analysis papers go back as far as Kaufman and Press [1973] and Mayekawa [1985], with the latter using an expectation-maximisation algorithm to find a maximum-a-posteriori solution. These papers formulated the problem as finding a low-dimensional representation for a number of multi-dimensional observations (corresponding to the rows in matrix factorisation), rather than the decomposition of a matrix directly. Although purely semantic, the former approach makes it easy to forget that the columns of the matrix are also entities of their own right, and worth studying. The matrix factorisation formulation was introduced by probabilistic latent

semantic indexing (PLSI, Hofmann [1999]). Given a number of documents consisting of words, PLSI finds a document-to-topic and topic-to-word distribution matrix for that collection of documents. Inference was performed using EM to obtain a maximum likelihood estimate. Ding et al. [2008] proved that PLSI and NMF were essentially equivalent. This method was later extended as a fully Bayesian model called latent dirichlet allocation (LDA, Blei et al. [2012]) by placing Dirichlet distribution priors over the document-to-topic and topic-to-word distributions, and using using variational Bayes or Gibbs sampling to find an estimate to the full posterior $p(\boldsymbol{U}, \boldsymbol{V}|\boldsymbol{R})$.

Bayesian matrix factorisation approaches use a likelihood distribution to capture noise in the data (usually Poisson for count data, or Gaussian for real-valued data), and place priors over the entries in $\boldsymbol{U}, \boldsymbol{V}$. These priors can induce sparsity or constrain the values to be nonnegative. Inference is often performed using Gibbs sampling or variational Bayesian inference. Note the parallel with the non-probabilistic matrix factorisation approaches: the likelihood acts as a cost function, and the priors are additional penalty terms over the factor matrices.

The basic Bayesian matrix factorisation model that many papers in the literature build on (Gönen [2012]; Salakhutdinov and Mnih [2008]; Virtanen et al. [2011, 2012]) uses a Gaussian likelihood and independent Gaussian priors,

$$R_{ij} \sim \mathcal{N}(R_{ij}|\boldsymbol{U}_i\boldsymbol{V}_j, \tau^{-1}) \qquad \tau \sim \mathcal{G}(\tau|\alpha_\tau, \beta_\tau),$$

where $\mathcal{N}(x|\mu, \tau) = \tau^{\frac{1}{2}}(2\pi)^{-\frac{1}{2}} \exp\left\{-\frac{\tau}{2}(x-\mu)^2\right\}$ is the density of the Gaussian distribution, with precision τ. $\boldsymbol{U}_i, \boldsymbol{V}_j$ denote the ith and jth rows of $\boldsymbol{U}$ and $\boldsymbol{V}$. We place a further Gamma prior over τ, with $\mathcal{G}(\tau|\alpha_\tau, \beta_\tau) = \frac{\beta_\tau{}^{\alpha_\tau}}{\Gamma(\alpha_\tau)} x^{\alpha_\tau - 1} e^{-\beta_\tau x}$, where $\Gamma(x) = \int_0^\infty x^{t-1}e^{-x}dt$ is the gamma function.

$$\boldsymbol{U}_i \sim \mathcal{N}(\boldsymbol{U}_i|\boldsymbol{0}, \lambda^{-1}\mathbf{I}) \qquad \boldsymbol{V}_j \sim \mathcal{N}(\boldsymbol{V}_j|\boldsymbol{0}, \lambda^{-1}\mathbf{I}).$$

Here, $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = |\boldsymbol{\Sigma}|^{-\frac{1}{2}}(2\pi)^{-\frac{K}{2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right\}$ is the density of a K-dimensional multivariate Gaussian distribution, and $\mathbf{I}$ is the identity matrix.

Another popular likelihood choice is the Poisson distribution (Gopalan and Blei [2014]; Gopalan et al. [2015]; Hu et al. [2015]). More complicated likelihood distributions can also be used, such as modelling that the observed values are ordinal (Paquet et al. [2012]).

2.3.3 Bayesian nonnegative matrix factorisation

One particular model of interest, that we will build on in our models and experiments, is Bayesian nonnegative matrix factorisation (BNMF, Schmidt et al. [2009]). This model also uses a Gaussian likelihood, but now constrains $\boldsymbol{U}, \boldsymbol{V}$ to be nonnegative using exponential priors over each individual entry in $\boldsymbol{U}$ and $\boldsymbol{V}$, with rate parameters $\lambda_{ik}^U, \lambda_{jk}^V > 0$.

$$U_{ik} \sim \mathcal{E}(U_{ik}|\lambda_{ik}^U) \qquad V_{jk} \sim \mathcal{E}(V_{jk}|\lambda_{jk}^V),$$

where $\mathcal{E}(x|\lambda) = \lambda \exp\{-\lambda x\}\, u(x)$ is the density of the exponential distribution, and $u(x)$ is the unit step function. For the precision τ we again use a Gamma distribution with shape $\alpha_\tau > 0$ and rate $\beta_\tau > 0$,

$$p(\tau) \sim \mathcal{G}(\tau|\alpha_\tau, \beta_\tau) = \frac{{\beta_\tau}^{\alpha_\tau}}{\Gamma(\alpha_\tau)} x^{\alpha_\tau - 1} e^{-\beta_\tau x}$$

where $\Gamma(x) = \int_0^\infty x^{t-1} e^{-x} dt$ is the gamma function. They introduced a Gibbs sampling algorithm for inference, which gave the following conditional posteriors to sample from.

$$p(U_{ik}|\tau, \boldsymbol{U}_{-ik}, \boldsymbol{V}, \boldsymbol{\lambda}, D) = \mathcal{TN}(U_{ik}|\mu_{ik}^U, \tau_{ik}^U) \qquad p(\tau|\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{\lambda}, D) = \mathcal{G}(\tau|\alpha_\tau^*, \beta_\tau^*)$$
$$p(V_{jk}|\tau, \boldsymbol{U}, \boldsymbol{V}_{-jk}, \boldsymbol{\lambda}, D) = \mathcal{TN}(V_{jk}|\mu_{jk}^V, \tau_{jk}^V),$$

where

$$\mathcal{TN}(x|\mu, \tau) = \begin{cases} \dfrac{\sqrt{\frac{\tau}{2\pi}} \exp\left\{-\frac{\tau}{2}(x-\mu)^2\right\}}{1 - \Phi(-\mu\sqrt{\tau})} & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

is a truncated normal: a normal distribution with zero density below $x = 0$ and renormalised to integrate to one. $\Phi(\cdot)$ is the cumulative distribution function of $\mathcal{N}(0, 1)$. Note that this model is not actually conjugate—the prior is exponential but the conditional posterior is a truncated normal. However, we can sample from the posterior and therefore still perform inference.

The parameter values are given below, with $\Omega_i^1 = \{j \mid (i,j) \in \Omega\}$ and $\Omega_j^2 = \{i \mid (i,j) \in \Omega\}$. The Gibbs sampling algorithm then simply samples a new value for each random variable in turn from these conditional posterior distributions.

$$\alpha_\tau^* = \alpha_\tau + \frac{|\Omega|}{2} \qquad \beta_\tau^* = \beta_\tau + \frac{1}{2} \sum_{(i,j)\in\Omega} (R_{ij} - \boldsymbol{U}_i \boldsymbol{V}_j)^2$$

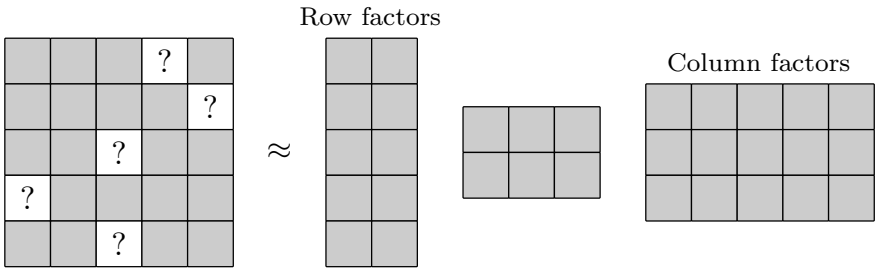


Figure 2.4 Matrix tri-factorisation with $K = 2$ row factors and $L = 3$ column factors. Unobserved entries are indicated by question marks.

$$\tau_{ik}^U = \tau \sum_{j \in \Omega_i^1} V_{jk}^2 \qquad \mu_{ik}^U = \frac{1}{\tau_{ik}^U} \left(-\lambda_{ik}^U + \tau \sum_{j \in \Omega_i^1} \left(R_{ij} - \sum_{k' \neq k} U_{ik'} V_{jk'} \right) V_{jk} \right)$$

$$\tau_{jk}^V = \tau \sum_{i \in \Omega_j^2} U_{ik}^2 \qquad \mu_{jk}^V = \frac{1}{\tau_{jk}^V} \left(-\lambda_{jk}^V + \tau \sum_{i \in \Omega_j^2} \left(R_{ij} - \sum_{k' \neq k} U_{ik'} V_{jk'} \right) U_{ik} \right).$$

2.3.4 Matrix tri-factorisation

Matrix tri-factorisation (MTF) is an extension to matrix factorisation, where instead of decomposing a matrix into two matrices, we decompose it into three matrices, $\boldsymbol{R} \approx \boldsymbol{F}\boldsymbol{S}\boldsymbol{G}^T$, where $\boldsymbol{F} \in \mathbb{R}^{I \times K}$, $\boldsymbol{S} \in \mathbb{R}^{K \times L}$, and $\boldsymbol{G} \in \mathbb{R}^{J \times L}$. In a similar way to how $\boldsymbol{U}$ indicated the clustering for NMF, we now have $\boldsymbol{F}$ indicating the clustering of rows, $\boldsymbol{G}$ the clustering of columns, and $\boldsymbol{S}$ relating the row and column clusters (biclusters). Much like how NMF is equivalent to K-means clustering, this is a form of biclustering (simultaneously clustering rows and columns). If we factorise a single dataset, matrix factorisation and tri-factorisation are not any different for predictive purposes. However, for analysing the factor values it can be useful, as it separates the factor values of $\boldsymbol{V}$ into column clusters in $\boldsymbol{G}$, and bicluster indicators in $\boldsymbol{S}$. Furthermore, when we use this approach for factorising multiple datasets, it also becomes very interesting for predictive purposes.

It is interesting to note that singular value decomposition (SVD, Golub and Reinsch [1970]) is a special form of matrix tri-factorisation. More precisely, we decompose $\boldsymbol{R} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$, with $\boldsymbol{U} \in \mathbb{R}^{I \times I}$, $\boldsymbol{V} \in \mathbb{R}^{J \times J}$, and $\boldsymbol{\Sigma} \in \mathbb{R}^{I \times J}$ is a diagonal matrix containing the singular values of $\boldsymbol{R}$. In contrast, with MTF we allow the off-diagonal entries to be non-zero so that rows and columns can be assigned to different biclusters, and the dimensionalities K, L are often much lower (for SVD, $K = I$ and $L = J$).

Ding et al. [2006] first introduced **nonnegative matrix tri-factorisation** (NMTF) $\boldsymbol{R} \approx \boldsymbol{F}\boldsymbol{S}\boldsymbol{G}^T$, noting that without any constraints this is equivalent to NMF, but with orthogonality constraints on $\boldsymbol{F}$, $\boldsymbol{G}$ this gives a very different solution. Multiplicative updates can again be used for non-probabilistic inference. NMTF has been used for many applications, mainly to extract row and column clusters at the same time, including sentiment analysis (Li [2010]; Li et al. [2009]) and bioinformatics (Hwang et al. [2012]; Liu et al. [2014]; Wang et al. [2013]). No Bayesian models for specifically matrix tri-factorisation were introduced, with only probabilistic (maximum-a-posteriori) ones in Yoo and Choi [2009] and An et al. [2010]. However, MTF is the two-dimensional case of the Tucker decomposition, for which we will see in Section 2.3.8 that several Bayesian models exist.

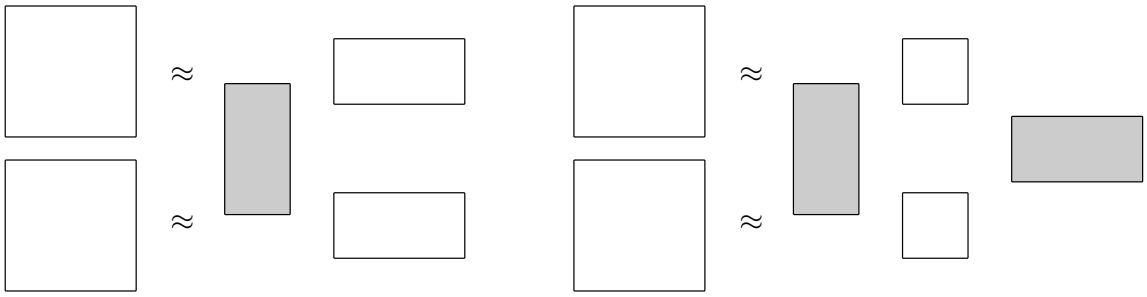
2.3.5 Symmetric matrix factorisation

Symmetric matrix factorisation is a special case of matrix factorisation for which the row entities are the same as the column entities—in other words, $\boldsymbol{R} \in \mathbb{R}^{I \times I}$. Examples are the edge matrix of a graph, or similarity kernels denoting how similar entities are to each other. These datasets are often nonnegative, so nonnegativity constraints are usually placed on the factor matrices. We can either decompose $\boldsymbol{R} \approx \boldsymbol{U}\boldsymbol{U}^T$ with $\boldsymbol{U} \in \mathbb{R}^{I \times K}$, or $\boldsymbol{R} \approx \boldsymbol{F}\boldsymbol{S}\boldsymbol{F}^T$ with $\boldsymbol{F} \in \mathbb{R}^{I \times K}$ and $\boldsymbol{S} \in \mathbb{R}^{K \times K}$. Ding et al. [2005b] argued that the latter decomposition is a better choice: it provides more degrees of freedom to fit to the data, and $\boldsymbol{S}$ gives a good characterisation of the quality of clustering—if clusters are well-separated, the off-diagonal entries in $\boldsymbol{S}$ will be small.

2.3.6 Multiple matrix factorisation

In practical applications we often have a wide range of datasets, relating many different entity types through multiple matrices. Therefore, we should consider how to best integrate these datasets and hence improve our analysis or predictions. Fortunately, matrix factorisation methods offer an elegant way to integrate them. In particular, we usually assume that entities have the same factor values for the different datasets they relate, and we perform a joint matrix factorisation where one or more latent matrices are shared. This allows us to fuse different data sources together.

In the literature this has been addressed under many different names: data fusion by matrix factorisation, joint matrix factorisation, collective matrix factorisation, multiple matrix factorisation, matrix co-factorisation. We will refer to them as **multiple matrix**



Multiple matrix factorisation Multiple matrix tri-factorisation

Figure 2.5 Difference between multiple matrix factorisation and multiple matrix tri-factorisation. The shared factor matrices are highlighted in grey.

factorisation. Models vary in complexity from jointly decomposing two datasets and sharing a single latent matrix, to more general data fusion approaches between any number of datasets.

There are several approaches in the literature. Some models just share the row factor matrix $\boldsymbol{U}$ across multiple datasets (Lee et al. [2012]; Wang et al. [2015a,b]; Yang and Dunson [2015]; Zhang et al. [2005]; Zhang and Yeung [2012]), as shown in Figure 2.5. Bayesian versions are given by Virtanen et al. [2012] and Chatzis [2014]. Some methods do not explicitly share the factor matrix, but instead add a penalisation term based on the similarity between the dataset-specific factor matrices (Liu et al. [2014]; Seichepine et al. [2013]; Zhang et al. [2014]).

Note that these models tend to be very specific: only the row factors can be shared, and therefore if we have any other information about the column entity type, we cannot use it. General matrix factorisation methods address this issue (Bouchard et al. [2013]; Lippert et al. [2008]; Singh and Gordon [2008]), with Bayesian versions given in Khan et al. [2016]; Klami et al. [2014]), by assigning each entity type its own latent matrix, giving a general framework for multiple matrix factorisation. However, these approaches cannot integrate multiple datasets between the same two entity types, since both matrices are shared and the factorisation therefore cannot account for differences between the two datasets. We would require a third, dataset-specific factor matrix to solve this problem.

This is addressed by **multiple matrix tri-factorisation** (shown in Figure 2.5), where we share the row and factor matrices $\boldsymbol{F}, \boldsymbol{G}$ across all factorisations, but we always have a dataset-specific matrix $\boldsymbol{S}$ to account for differences. This is particularly interesting

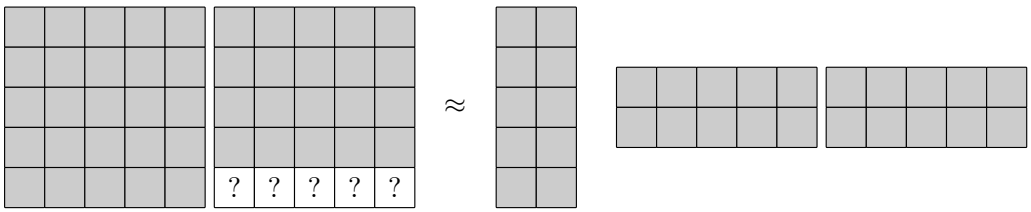


Figure 2.6 Diagram showing how multiple matrix factorisation can be used for out-of-matrix predictions.

for integrating repeated experiments, where different biological labs perform similar experiments between the same two entity types, such as gene expression profiles and methylation levels. Intuitively, this approach assumes that all entities of a specific entity type are clustered in the same way across the datasets, since its factor matrix $\boldsymbol{F}$ is shared. Models for multiple nonnegative matrix tri-factorisation are given by Wang et al. [2008] and Žitnik and Zupan [2015], but they require all given datasets to be fully observed. As a result, missing values inside each matrix need to be imputed. For binary datasets a missing association can easily be imputed as a zero, but for real-valued datasets this is not a viable option. In Chapter 5 we will introduce a Bayesian multiple matrix factorisation and tri-factorisation model that addresses this issue.

Interestingly, the PARAFAC2 model introduced by Harshman [1972] which was formulated in the context of tensors (see Section 2.3.8), is in fact a multiple matrix tri-factorisation model for matrices (rather than tensors) with the same number of columns but different numbers of rows. We jointly decompose $\boldsymbol{R}_k \approx \boldsymbol{F}_k \boldsymbol{S}_k \boldsymbol{G}$, where we share $\boldsymbol{F}_k, \boldsymbol{S}_k$ across the different datasets, and the $\boldsymbol{S}_k$ are diagonal matrices.

These multiple matrix factorisation approaches can help us improve our in-matrix predictions. Importantly, they also allow us to make out-of-matrix predictions, for entirely unobserved rows or columns in a matrix. This is illustrated in Figure 2.6 for a joint factorisation of two matrices, where the row factor matrix is shared. Note that this is effectively the same as concatenating the columns of the two matrices, and doing in-matrix predictions. However, if we have multiple datasets spanning more than two different entity types, concatenating the matrices no longer works, and we need the more general models for multiple matrix factorisation.

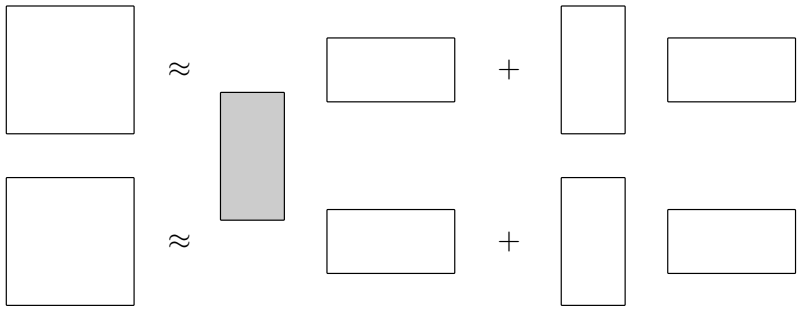


Figure 2.7 An overview of the idea behind Bayesian canonical correlation analysis.

2.3.7 Canonical correlation analysis

Canonical correlation analysis (CCA) is a method for finding the common factors in two datasets. Similarly to multiple matrix factorisation, we share the row factors. The difference is that we also have dataset-specific row factors for both of the datasets. In other words, in matrix factorisation we assume that we can share all the row factors, whereas for CCA we say that some of the variation in the two datasets is not shared. Effectively, we have one shared matrix factorisation, and one dataset-specific one, as shown in Figure 2.7. An interesting Bayesian formulation of canonical correlation analysis is given by Virtanen et al. [2011], where they use a special formulation of the Bayesian automatic relevance determination prior, which we will now introduce.

Automatic relevance determination (ARD) is a Bayesian prior which helps perform automatic model selection, and can be used for matrix factorisation. For the basic Bayesian matrix factorisation model it works by replacing the λ hyperparameter in the Gaussian priors for the factor matrices by one that is shared by all entries in the same column (in other words, shared for each factor). We then place a further Gamma prior over all these λ_k parameters. This changes the priors to

$$\boldsymbol{U_i} \sim \mathcal{N}(\boldsymbol{U_i}|\mathbf{0}, \text{diag}(\boldsymbol{\lambda}^{-1})) \qquad \boldsymbol{V_j} \sim \mathcal{N}(\boldsymbol{V_j}|\mathbf{0}, \text{diag}(\boldsymbol{\lambda}^{-1})) \qquad \lambda_k \sim \mathcal{G}(\lambda_k|\alpha_0, \beta_0),$$

where $\text{diag}(\boldsymbol{\lambda}^{-1})$ is a diagonal matrix with entries $\lambda_1^{-1}, .., \lambda_K^{-1}$ on the diagonal. The Gamma prior for the ARD is conjugate, and Gibbs sampling or variational Bayes can be used for inference. Since this parameter is shared by all entries in the same column, the entire factor k is either activated (if λ_k^t has a low value) or "turned off" (if λ_k^t has a high value), pushing factors that are active for only a few entities further to zero. This is shown in Figure 2.8, where there are three active factors, and one has been turned off. This prior can be used for real-valued matrix factorisation (as in Virtanen et al.

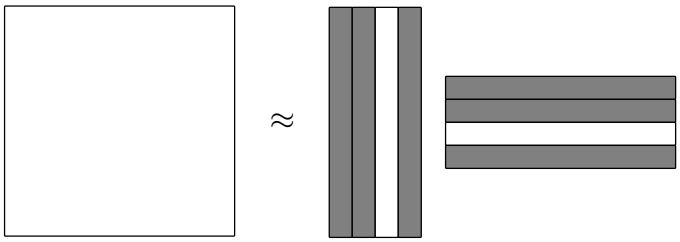


Figure 2.8 An overview of the effect of using automatic relevance determination in matrix factorisation models. The white column in the factor matrices is inactive.

[2011]), as well as nonnegative matrix factorisation (Tan and Févotte [2013]). Instead of having to choose the correct K, we can give an upper bound and the model will automatically try to determine the number of factors to use. A similar approach can be found in Figueiredo and Jain [2002], which incorporates the elimination of unused factors directly into their expectation-maximisation inference algorithm. In contrast, ARD is implemented on a model-level. In Chapter 3 we will validate the effectiveness of the above ARD for the Bayesian nonnegative matrix factorisation model.

The specific version of the ARD prior used in Virtanen et al. [2011] is called a group-wise ARD. Instead of activating or turning off an entire factor, it can be turned on or off for a specific dataset (group) only. This is done by having a separate λ_k^n parameter for each dataset n. As a result, if a factor is active for both groups, it is shared; if it is active for one dataset but turned off for the other it is dataset-specific; and if it is active for neither, it is unused. This formulation allows us to efficiently perform CCA using a special formulation of matrix factorisation. This model is extended to three or more datasets in Virtanen et al. [2012], called Bayesian group factor analysis.

2.3.8 Tensor decomposition

Matrices represent relations between two entity types. Tensors are the higher-dimensional generalisation of matrices, usually relating three or more entity types in as many dimensions. We can perform tensor decompositions very similarly to matrix factorisation.

The **CANDECOMP/PARAFAC** (CP, Harshman [1970]) method decomposes a given tensor $\boldsymbol{R} \in \mathbb{R}^{I \times J \times N}$ into the sum of K rank-1 tensors. This is effectively a generalisation of matrix factorisation to three (rather than two) dimensions, with each dimension getting its own factor matrix: $\boldsymbol{F}^1 \in \mathbb{R}^{I \times K}$, $\boldsymbol{F}^2 \in \mathbb{R}^{J \times K}$, $\boldsymbol{F}^3 \in \mathbb{R}^{N \times K}$. These matrices are often constrained to be orthogonal. Overall, we perform the factorisation $\boldsymbol{R} = \boldsymbol{F}^1 \otimes \boldsymbol{F}^2 \otimes \boldsymbol{F}^3$, where $\otimes$ denotes the matrix outer product. Each individual entry

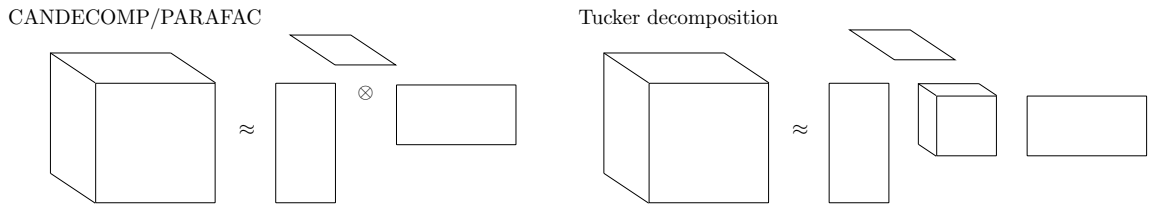


Figure 2.9 An overview of CANDECOMP/PARAFAC (CP, left) and the Tucker decomposition (TD, right).

in $\boldsymbol{R}$ is decomposed as follows:

$$R_{ijn} = \sum_{k=1}^{K} F^1_{ik} \cdot F^2_{jk} \cdot F^3_{nk}.$$

The **Tucker decomposition** (TD, Tucker [1966]) is defined similarly, but in addition to the three factor matrices, we also get a core tensor $\boldsymbol{S} \in \mathbb{R}^{K \times L \times Q}$, and the factor matrices have their own number of latent factors K, L, Q; $\boldsymbol{F}^1 \in \mathbb{R}^{I \times K}$, $\boldsymbol{F}^2 \in \mathbb{R}^{J \times L}$, $\boldsymbol{F}^3 \in \mathbb{R}^{N \times Q}$. We now factorise $\boldsymbol{R} = \boldsymbol{S} \bullet_1 \boldsymbol{F}^1 \bullet_2 \boldsymbol{F}^2 \bullet_3 \boldsymbol{F}^3$, where $\bullet_i$ denotes the matrix dot product using the ith dimension of tensor $\boldsymbol{S}$. Individual entries in $\boldsymbol{R}$ are decomposed as:

$$R_{ijn} = \sum_{k=1}^{K} \sum_{l=1}^{L} \sum_{q=1}^{Q} F^1_{ik} \cdot F^2_{jl} \cdot F^3_{nq} \cdot S_{klq}.$$

An interesting observation is that the tensor decomposition is actually a special case of the Tucker decomposition, where the core tensor is diagonal. Both CP and TD are shown in Figure 2.9.

Papers on CP go back as far as 1970, when it was first introduced by Harshman [1970] with applications in chemometrics. A nonnegative CP model was introduced by Lawson and Hanson [1995], and Welling and Weber [2001] extended the approach by Lee and Seung [1999] to tensors, giving multiplicative updates. There is also a rich literature for probabilistic (Chu and Ghahramani [2009]; Yılmaz and Cemgil [2010]) and Bayesian (Hoff [2013]; Xu et al. [2012]; Zhao et al. [2015]) models for CP and TD models. These often use Gaussian priors over the core tensor and matrices, although a Dirichet prior to enforce sparsity is also possible (Zhe et al. [2015]). Yang and Dunson [2015] give a Bayesian nonnegative approach for binary data.

2.4 Collaborative filtering

In this thesis we are analysing the predictive performance of Bayesian matrix factorisation models for predicting missing values in matrices. This is sometimes also called collaborative filtering, expecially in the context of predicting movie ratings. We test performances under different conditions and explore the best modelling choices. We consider several real-world applications and datasets: drug sensitivity datasets, detailing the effectiveness of cancer drugs on different cell lines; gene expression and methylation values for breast cancer patients; and movie ratings. These applications are briefly introduced in the next sections.

In order to measure how well our models are doing, it is tempting to simply measure how well we can replicate the values in our datasets that we give to the model to learn from. However, this leads to so-called **overfitting**. Real-world datasets always have some amount of random noise in them, since none of the observed values can be measured perfectly. If our models fit exactly to the training data, for example finding a perfect matrix factorisation, this means we must have fitted to the noise in the data, and this tends to give very poor predictions on new datapoints. Hence, we must always measure the predictive performance on datapoints that the model has not yet seen.

To do this, we split the observed entries $\Omega = \{(i,j) \mid R_{ij} \text{ observed}\}$ into a training set Ω_{train} and a test set Ω_{test}. We then give the model only the training datapoints, and obtain predictions for the test datapoints (for matrix factorisation the prediction for a datapoint (i,j) is given by $\boldsymbol{U}_i\boldsymbol{V}_j$). We will use the mean squared error (MSE) to indicate the predictive errror, with a lower value indicating better predictive performance,

$$\text{MSE} = \frac{1}{|\Omega_{\text{test}}|} \sum_{(i,j)\in\Omega_{\text{test}}} (R_{ij} - \boldsymbol{U}_i\boldsymbol{V}_j)^2 \,.$$

There can be a lot of variation in predictive performance if we do this only once, as some entries may be easier to predict than others. To address this, we often use **10-fold cross-validation**: we split the observed entries Ω into ten folds of equal size, use nine folds as training data at a time and the remaining one as test, and do this for all ten possibilities. We can then average across the ten performances. We can do N-fold cross-validation for any N, but it is usually chosen to be 5 or 10. There can still be some variation across repeats of cross-validation, but it goes a long way to addressing the issue.

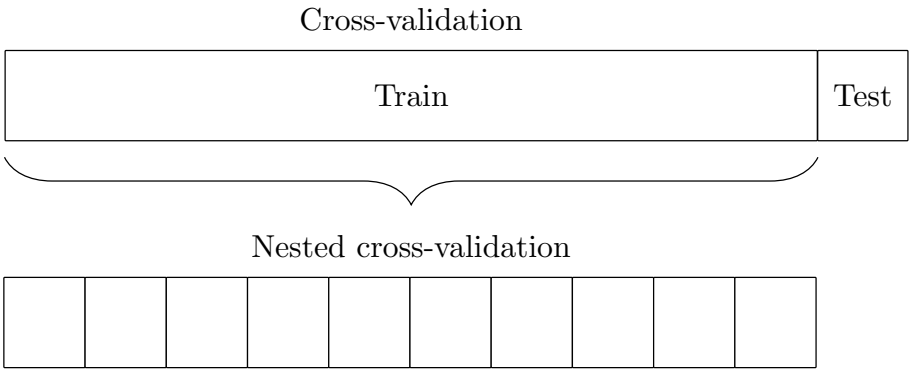


Figure 2.10 Overview of the nested cross-validation procedure. After splitting the data into a train and test set, we split the train set into ten further folds and train a model with each of the parameter values and measure the nested cross-validation performance. After choosing the best parameter values, we then fit a final model to the train data and measure our cross-validation performance on the test data.

Some of our models will have parameters whose values may influence the predictive performance, most importantly the dimensionality K (and L) for matrix (tri-)factorisation. It is tempting to run cross-validation for each of these values and simply pick the best one as the predictive performance, but this can lead to bias: we effectively just rerun the cross-validation until we get a favourable random splitting of train and test data, and use the best performance. Other models with few or no parameter will therefore have a disadvantage, and we cannot trust the final performance.

We should make sure that we choose the value of our parameters without seeing the test data. This can be done using **nested cross-validation** (see Figure 2.10). As before, we split the observed entries into 10 folds, and use nine to train and one to test. However, we now perform an additional cross-validation on these nine folds together, trying each possible value of the parameters that we wish to choose from. We then choose the parameter values with the best cross-validation performance on the training datapoints, and use those to train a model on the training data and measure the final predictive performance of this fold. This method removes any bias towards models with lots of parameters, and reduces subtle overfitting problems in our cross-validation procedure.

2.4.1 Drug sensitivity

Drug sensitivity datasets detail the effectiveness of cancer drugs on different cell lines (cancer types in a tissue). By measuring the cell line activity at different concentrations

of the drug, and fitting a line through these data points, we can summarise the effectiveness of a drug on a cell line. This is usually measured either as IC_{50} or EC_{50} values. IC_{50} indicates the required drug concentration needed to reduce the activity of a given cell line (cancer type in a tissue) by half. We thus measure when an undesired effect has been inhibited by half. With EC_{50} values we measure the maximal (desired) effect a drug can have on a cell line, and then measure the concentration of the drug where we achieve half of this value. In both cases, a lower value is better.

There are four main publicly available drug sensitivity datasets, each detailing IC_{50} or EC_{50} values, where some of the entries are missing. In particular, we consider:

- Genomics of Drug Sensitivity in Cancer (GDSC v5.0, Yang et al. [2013])—giving the natural log of IC_{50} values for 139 drugs across 707 cell lines, with 80% observed entries.

- Cancer Therapeutics Response Portal (CTRP v2, Seashore-Ludlow et al. [2015])—giving EC_{50} values for 545 drugs across 887 cell lines, with 80% observed entries.

- Cancer Cell Line Encyclopedia (CCLE, Barretina et al. [2012])—giving both IC_{50} and EC_{50} values for 24 drugs across 504 cell lines, with 96% and 63% observed entries respectively.

The problem of predicting drug sensitivity values is still an active and ongoing research area. A significant number of papers have explored making predictions using machine learning methods, such as random forests (Menden et al. [2013]), nearest neighbours (Pal et al. [2012]), support vector machines (Dong et al. [2015]), non-probabilistic matrix factorisation (Wang et al. [2017]), kernelised Bayesian matrix factorisation (Ammad-ud din et al. [2014]), and sometimes providing a comparison between different methods (De Niz et al. [2016]; Jang et al. [2014]). Furthermore, a predictive competition (DREAM 7; NCI-DREAM Drug Sensitivity Prediction Challenge, Costello et al. [2014]) was organised in 2012 where a total of 44 teams developed algorithms and pipelines to predict drug sensitivity values.

We use the four drug sensitivity datasets in Chapters 3, 4, and 5.

2.4.2 Gene expression and methylation

The human genome can be seen as a large recipe book, with each gene giving the recipe of one or more proteins, which in turn can catalyse chemical reactions in the body. Diseases are often studied by measuring the change in what genes are being copied (transcribed) and translated into proteins, as this can give an indication of what is going wrong. Gene expression profiles measure how often each gene is being copied, and differential gene expression profiles give the relative change, usually compared to a healthy patient or cell line. We can measure this for different drugs, diseases, or patients. Hence, gene expression profiles are usually given as a matrix relating genes to one of these three entity types.

Methylation values indicate the number of methyl groups that bind to different regions of the genes. There is the promoter-region of the gene (this region initiates transcription of the gene), and the gene body (which encodes the protein). Promoter-region methylation gives the amount of methylation in the former part of the gene, and gene body methylation in the latter. These data can be given as a dataset in the same way that gene expression can. The Cancer Genome Atlas (TCGA, Koboldt et al. [2012]) gives promoter-region methylation, gene body methylation, and gene expression profiles for 254 healthy and tumour tissues, across 13966 genes. It is sometimes worth focusing on a subset of the genes, for example those that are known to be important for cancer—so-called cancer driver genes—for which the IntOGen database (Gonzalez-Perez et al. [2013]) is a good resource.

We use the gene expression and methylation datasets in Chapters 4 and 5.

2.4.3 Movie ratings

The classic collaborative filtering problem is that of predicting the rating that a user will give to an unseen movie, based on the other movies they have rated and other users' ratings of the movie. The Netflix Prize was a famous competition for movie recommendations, with a grand prize of $\$1,000,000$ for the winners. Compared to the drug sensitivity datasets, which have hundreds of rows and columns, movie ratings datasets tend to be much larger in size, relating thousands to tens of thousands of users and movies, with roughly 100 million observed ratings. The values are also integer ratings of 1 to 5 stars, rather than real-valued. The Netflix dataset is publicly available (Bennett and Lanning [2007]). Similarly, the MovieLens datasets (Harper and Konstan

[2015]) provide movie ratings, varying in size from $100,000$ observed entries across thousands of entities (MovieLens 100K), to millions of entries (MovieLens 1M, 10M, 20M).

These datasets have become very popular for benchmarking matrix factorisation methods; see for example Bouchard et al. [2013]; Hu et al. [2015]; Jing et al. [2015]; Kim and Choi [2014]; Lakshminarayanan et al. [2011]; Salakhutdinov and Mnih [2008].

We use the MovieLens 100K and 1M datasets in Chapter 4.

Chapter 3

Effects of inference methods in Bayesian nonnegative matrix factorisation

The inference methods used to find a solution of a Bayesian matrix factorisation model can have a great impact on the predictive performance of this model. For example, if a method fits extremely well to the data, there is a good chance that the method *overfits* to the noise in the data. As a result, our predictive performances become extremely poor—as well as the factor analysis we could perform on the resulting matrices. A key question that arises is: what exactly are the trade-offs between different matrix factorisation inference approaches? In particular, which perform better in terms of speed of convergence, predictive performance, and robustness to noise and sparsity?

In this chapter we answer these questions by performing a thorough empirical study to explore these trade-offs between non-probabilistic and Bayesian inference approaches, for both matrix factorisation and tri-factorisation. We consider the popular non-probabilistic matrix factorisation model from Lee and Seung [2000], and the Bayesian nonnegative matrix factorisation model from Schmidt et al. [2009]. The latter uses exponential priors to enforce nonnegativity, giving Gibbs sampling algorithms for inference. They also introduced a maximum-a-posteriori algorithm called iterated conditional modes (ICM). Both of these approaches rely on a sampling procedure to eventually converge to draws of the desired distribution—in this case the posterior of the matrices. This means that we need to inspect the values of the draws to determine when our method has converged (burn-in), and then take additional draws to estimate the posteriors.

We also introduce a fourth inference technique for the Bayesian nonnegative models, based on variational Bayesian inference (VB), where instead of relying on random draws we obtain deterministic convergence to a solution. Some papers (for instance Salimans et al. [2015]) assert that variational inference gives faster but less accurate inference than sampling methods like Gibbs. One study investigating this for latent dirichlet allocation can be found in Asuncion et al. [2009], and an arXiv paper explores the effects on the sampling quality for Bayesian nonnegative matrix factorisation (Masood et al. [2016]). However, this chapter is the first study giving a thorough empirical study of the effects on predictive performance for matrix factorisation.

Furthermore, we extend the Bayesian nonnegative matrix factorisation model to matrix tri-factorisation. For this model the variational Bayesian inference has a nontrivial element due to the extra dependencies created by the extra matrix. We also add automatic relevance determination (ARD) to both models, which could eliminate the need for model selection.

We perform extensive experiments on both artificial and real-world datasets to explore the trade-offs between speed of inference, and robustness to sparsity and noise for predicting missing values. We show that Gibbs sampling is the most robust, while VB and ICM give significant run-time speedups but sacrifice some robustness, and that non-probabilistic inference tends to be fast but not robust. Finally, we show that ARD is an effective way of performing automatic model selection, and increases the robustness of matrix factorisation models if they are given the wrong dimensionality—even though it does not improve robustness to noise and sparsity.

Although we study a specific Bayesian nonnegative matrix factorisation and tri-factorisation model, we believe that many of our findings and insights apply to the broad range of other matrix factorisation and tri-factorisation methods, as well as tensor and Tucker decomposition methods—their three-dimensional extensions. In Chapter 4 we provide a thorough exploration of the trade-offs between different prior and likelihood choices.

There are also many other inference techniques possible—to name a few: expectation propagation, Hamiltonian Monte Carlo, moment matching—but we believe the four that we have chosen are representative of the wide range of possible inference approaches: a non-probabilistic (maximum likelihood) approach; a maximum a posteriori inference; a fully Bayesian MCMC method; and a deterministic fully Bayesian inference approach.

In summary, the contributions of this chapter are as follows:

- Extending the Bayesian nonnegative matrix factorisation model to tri-factorisation, and deriving Gibbs sampling and iterated conditional modes algorithms for this model.
- Introducing a new variational Bayesian inference algorithm for both of these models.
- Providing extensive experiments comparing four different inference approaches for the models in terms of speed of convergence, predictive performance, and robustness to noise and sparsity.
- Extending the models with automatic relevance determination, demonstrating the effectiveness of automatic relevance determination for model selection, and showing that it does not improve robustness to noise and sparsity.
- Provide a comparison of matrix factorisation and tri-factorisation, showing that even though the former converges faster and fits more closely to the training data, there is little to no difference in performance and robustness.

3.1 Models

3.1.1 Nonnegative matrix factorisation

We use the Bayesian nonnegative matrix factorisation model of Schmidt et al. [2009] as our starting point. It was introduced in Section 2.3.3, but we review it briefly below.

Nonnegative matrix factorisation (NMF) can be formulated as decomposing a matrix $\boldsymbol{R} \in \mathbb{R}^{I \times J}$ into two latent matrices $\boldsymbol{U} \in \mathbb{R}_+^{I \times K}$ and $\boldsymbol{V} \in \mathbb{R}_+^{J \times K}$, whose values are constrained to be positive. In other words, solving $\boldsymbol{R} = \boldsymbol{U}\boldsymbol{V}^T + \boldsymbol{E}$, where noise is captured by matrix $\boldsymbol{E} \in \mathbb{R}^{I \times J}$. The dataset $\boldsymbol{R}$ need not be complete: the indices of observed entries can be represented by the set $\Omega = \{(i, j) \mid R_{ij} \text{ is observed}\}$, and these entries can then be predicted by $\boldsymbol{U}\boldsymbol{V}^T$. We take a probabilistic approach to this problem, by expressing a likelihood function for the observed data and treating the latent matrices as random variables. As the likelihood we assume each value of $\boldsymbol{R}$ comes from the product of $\boldsymbol{U}$ and $\boldsymbol{V}$, with some Gaussian noise added of precision τ. We place exponential priors over $\boldsymbol{U}$ and $\boldsymbol{V}$, so that each element in U and V is assumed to

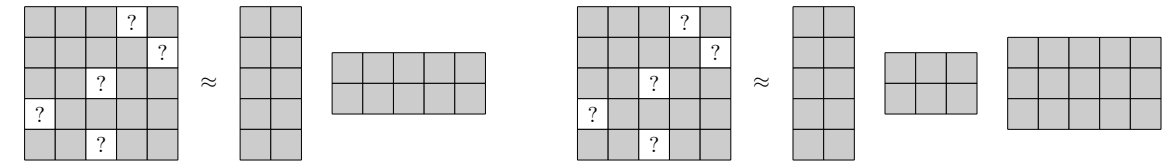


Figure 3.1 Overview of matrix factorisation and matrix tri-factorisation methods, with missing values indicated as question marks. Repeated for clarity from Section 2.3.

be independently exponentially distributed with rate parameters $\lambda_U, \lambda_V > 0$. Finally, for the precision τ we use a Gamma distribution with shape $\alpha_\tau > 0$ and rate $\beta_\tau > 0$, giving

$$R_{ij} \sim \mathcal{N}(R_{ij}|\boldsymbol{U}_i\boldsymbol{V}_j, \tau^{-1}) \qquad \tau \sim \mathcal{G}(\tau|\alpha_\tau, \beta_\tau) \qquad U_{ik} \sim \mathcal{E}(U_{ik}|\lambda_U) \qquad V_{jk} \sim \mathcal{E}(V_{jk}|\lambda_V).$$

3.1.2 Nonnegative matrix tri-factorisation

The problem of nonnegative matrix tri-factorisation (NMTF) can be formulated similarly to that of nonnegative matrix factorisation. We now decompose $\boldsymbol{R}$ into three matrices $\boldsymbol{F} \in \mathbb{R}_+^{I\times K}$, $\boldsymbol{S} \in \mathbb{R}_+^{K\times L}$, $\boldsymbol{G} \in \mathbb{R}_+^{J\times L}$, so that $\boldsymbol{R} = \boldsymbol{F}\boldsymbol{S}\boldsymbol{G}^T + \boldsymbol{E}$. We can extend the Bayesian nonnegative matrix factorisation to tri-factorisation using the prior distributions, again using a Gaussian likelihood and exponential priors for the latent matrices,

$$\begin{aligned} &R_{ij} \sim \mathcal{N}(R_{ij}|\boldsymbol{F}_i\boldsymbol{S}\boldsymbol{G}_j, \tau^{-1}) && \tau \sim \mathcal{G}(\tau|\alpha_\tau, \beta_\tau) \\ &F_{ik} \sim \mathcal{E}(F_{ik}|\lambda_F) \qquad S_{kl} \sim \mathcal{E}(S_{kl}|\lambda_S) && G_{jl} \sim \mathcal{E}(G_{jl}|\lambda_G). \end{aligned}$$

3.1.3 Automatic relevance determination

Automatic relevance determination (ARD) is a Bayesian prior which helps perform automatic model selection. It works by replacing the individual λ parameters in the exponential priors for the factor matrices by one that is shared by all entries in the same column (in other words, shared for each factor). We then place a further Gamma prior over all these λ_k parameters. This has the advantage that factors that are used by only a few datapoints will be pushed further to zero and made inactive. We will see in the experiments that this helps massively with model selection. In Section 2.3.7 we showed how this can be done for real-valued matrix factorisation with Gaussian priors

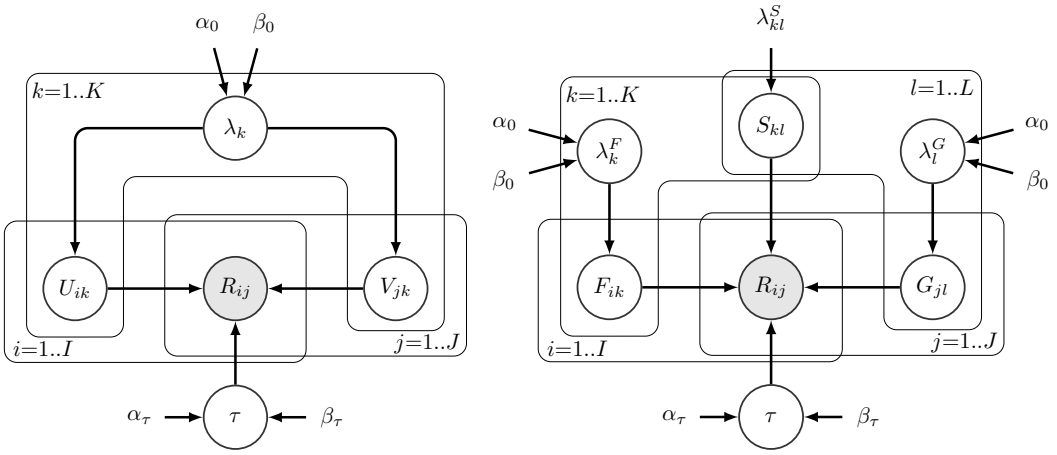


Figure 3.2 Graphical model representation of Bayesian nonnegative matrix factorisation (left) and tri-factorisation (right), with ARD. The variables are introduced in Section 3.1.

over the factor matrices. For NMF it can be added it similarly, changing the priors to

$$U_{ik} \sim \mathcal{E}(U_{ik}|\lambda_k) \qquad V_{jk} \sim \mathcal{E}(V_{jk}|\lambda_k) \qquad \lambda_k \sim \mathcal{G}(\lambda_k|\alpha_0, \beta_0).$$

For NMTF we modify our model by employing two ARD's: one over the $\boldsymbol{F}$ matrix (λ^F_k) and another for $\boldsymbol{G}$ (λ^G_l), with the $\boldsymbol{S}$ matrix remaining the same,

$$F_{ik} \sim \mathcal{E}(F_{ik}|\lambda^F_k) \qquad \lambda^F_k \sim \mathcal{G}(\lambda^F_k|\alpha_0, \beta_0) \qquad G_{jl} \sim \mathcal{E}(G_{jl}|\lambda^G_l) \qquad \lambda^G_l \sim \mathcal{G}(\lambda^G_l|\alpha_0, \beta_0).$$

The graphical models for Bayesian NMF and NMTF, with ARD, are given in Figure 3.2.

3.2 Inference approaches

In this section we give details for four different types of inference for nonnegative matrix factorisation (NMF) and tri-factorisation (NMTF) models. Non-probabilistic inference gives a point estimate solution. Gibbs sampling and variational Bayesian inference both give a full posterior estimate, whereas iterated conditional modes gives a maximum a posteriori (MAP) point estimate.

3.2.1 Non-probabilistic inference

We use the popular model in Lee and Seung [2000] for the non-probabilistic inference, which we reviewed in Section 2.3.1, but extend here with missing values. Their algorithm relies on multiplicative updates, where at each iteration the values in the $\boldsymbol{U}$ and $\boldsymbol{V}$ matrices are updated using the following equations,

$$U_{ik} = U_{ik} \cdot \frac{\sum_{j \in \Omega_i} R_{ij} V_{jk} / (\boldsymbol{U}_i \boldsymbol{V}_j)}{\sum_{j \in \Omega_i} V_{jk}} \qquad V_{jk} = V_{jk} \cdot \frac{\sum_{i \in \Omega_j} R_{ij} U_{ik} / (\boldsymbol{U}_i \boldsymbol{V}_j)}{\sum_{i \in \Omega_j} U_{ik}},$$

where $\Omega_i^1 = \{j \mid (i,j) \in \Omega\}$ and $\Omega_j^2 = \{i \mid (i,j) \in \Omega\}$. These updates can be shown to minimise the I-divergence (generalised KL-divergence),

$$D(\boldsymbol{R} || \boldsymbol{U}\boldsymbol{V}^T) = \sum_{(i,j) \in \Omega} \left(R_{ij} \log \frac{R_{ij}}{(\boldsymbol{U}\boldsymbol{V}^T)_{ij}} - R_{ij} + (\boldsymbol{U}\boldsymbol{V}^T)_{ij} \right).$$

Yoo and Choi [2009] extended this approach to NMTF, giving the following multiplicative updates, with $\boldsymbol{S}_{\cdot l}$ denoting the lth column of $\boldsymbol{S}$,

$$F_{ik} = F_{ik} \cdot \frac{\sum_{j \in \Omega_i} R_{ij} (\boldsymbol{S}_k \boldsymbol{G}_j) / (\boldsymbol{F}_i \boldsymbol{S} \boldsymbol{G}_j)}{\sum_{j \in \Omega_i} (\boldsymbol{S}_k \boldsymbol{G}_j)} \qquad G_{jl} = G_{jl} \cdot \frac{\sum_{i \in \Omega_j} R_{ij} (\boldsymbol{F}_i \boldsymbol{S}_{\cdot l}) / (\boldsymbol{F}_i \boldsymbol{S} \boldsymbol{G}_j)}{\sum_{i \in \Omega_j} (\boldsymbol{F}_i \boldsymbol{S}_{\cdot l})}$$

$$S_{kl} = S_{kl} \cdot \frac{\sum_{(i,j) \in \Omega} R_{ij} F_{ik} G_{jl} / (\boldsymbol{F}_i \boldsymbol{S} \boldsymbol{G}_j)}{\sum_{(i,j) \in \Omega} F_{ik} G_{jl}},$$

which minimises the I-divergence

$$D(\boldsymbol{R} || \boldsymbol{F}\boldsymbol{S}\boldsymbol{G}^T) = \sum_{(i,j) \in \Omega} \left(R_{ij} \log \frac{R_{ij}}{(\boldsymbol{F}\boldsymbol{S}\boldsymbol{G}^T)_{ij}} - R_{ij} + (\boldsymbol{F}\boldsymbol{S}\boldsymbol{G}^T)_{ij} \right).$$

3.2.2 Gibbs sampling

Bayesian nonnegative matrix factorisation

Schmidt et al. [2009] introduced a Gibbs sampling algorithm for approximating the posterior distribution. Recall from Section 2.2.1 that Gibbs sampling works by sampling new values for each parameter θ_i from its marginal distribution given the current values of the other parameters $\boldsymbol{\theta}_{-i}$, and the observed data D. For the NMF model this means

that we need to be able to draw from the following distributions:

$$p(\tau|\boldsymbol{U},\boldsymbol{V},\boldsymbol{\lambda},D) \quad p(\lambda_k|\tau,\boldsymbol{U},\boldsymbol{V},D) \quad p(U_{ik}|\tau,\boldsymbol{U}_{-ik},\boldsymbol{V},\boldsymbol{\lambda},D) \quad p(V_{jk}|\tau,\boldsymbol{U},\boldsymbol{V}_{-jk},\boldsymbol{\lambda},D)$$

where $\boldsymbol{U}_{-ik}$ denotes all elements in $\boldsymbol{U}$ except U_{ik}, and similarly for $\boldsymbol{V}_{-jk}$. $\boldsymbol{\lambda}$ is a vector including all λ_k values. Using Bayes theorem we obtain the following posterior distributions, where $\mathcal{TN}$ is a truncated normal distribution,

$$\begin{aligned} p(\tau|\boldsymbol{U},\boldsymbol{V},\boldsymbol{\lambda},D) &= \mathcal{G}(\tau|\alpha^*_\tau,\beta^*_\tau) & p(U_{ik}|\tau,\boldsymbol{U}_{-ik},\boldsymbol{V},\boldsymbol{\lambda},D) &= \mathcal{TN}(U_{ik}|\mu^U_{ik},\tau^U_{ik}) \\ p(\lambda_k|\tau,\boldsymbol{U},\boldsymbol{V},D) &= \mathcal{G}(\lambda_k|\alpha^*_k,\beta^*_k) & p(V_{jk}|\tau,\boldsymbol{U},\boldsymbol{V}_{-jk},\boldsymbol{\lambda},D) &= \mathcal{TN}(V_{jk}|\mu^V_{jk},\tau^V_{jk}) \end{aligned}$$

We show the derivation for U_{ik} below.

$$\begin{aligned} p(U_{ik}|\tau,\boldsymbol{R},\boldsymbol{U}_{-ik},\boldsymbol{V},\boldsymbol{\lambda}) &\propto p(\boldsymbol{R}|\tau,\boldsymbol{U},\boldsymbol{V})\cdot p(U_{ik}|\lambda_k) \\ &\propto \prod_{j\in\Omega^1_i}\mathcal{N}(R_{ij}|\boldsymbol{U}_i\cdot\boldsymbol{V}_j,\tau^{-1})\cdot\mathcal{E}(U_{ik}|\lambda_k) \\ &\propto \exp\left\{-\frac{\tau}{2}\sum_{j\in\Omega^1_i}(R_{ij}-\boldsymbol{U}_i\boldsymbol{V}_j)^2\right\}\cdot\exp\left\{-\lambda_k U_{ik}\right\}\cdot u(x) \\ &\propto \exp\left\{-\frac{U^2_{ik}}{2}\left[\tau\sum_{j\in\Omega^1_i}V^2_{jk}\right]\right. \\ &\qquad \left. +U_{ik}\left[-\lambda_k+\tau\sum_{j\in\Omega^1_i}\left(R_{ij}-\sum_{k'\neq k}U_{ik'}V_{jk'}\right)V_{jk}\right]\right\}\cdot u(x) \\ &\propto \exp\left\{-\frac{\tau^U_{ik}}{2}(U_{ik}-\mu^U_{ik})^2\right\}\cdot u(x) \\ &\propto \mathcal{TN}(U_{ik}|\mu^U_{ik},\tau^U_{ik}). \end{aligned}$$

The parameter values for the NMF Gibbs sampling algorithm are given below. The Gibbs sampling algorithm then simply draws a new value for each random variable in turn from these conditional posterior distributions. It takes a while to converge to the right posterior parameter space, so we need to remove the first n samples (*burn-in*), and since consecutive draws are correlated we only use every ith value (*thinning*).

$$\alpha^*_\tau = \alpha_\tau + \frac{|\Omega|}{2} \qquad \beta^*_\tau = \beta_\tau + \frac{1}{2}\sum_{(i,j)\in\Omega}(R_{ij}-\boldsymbol{U}_i\boldsymbol{V}_j)^2$$

$$\alpha_k^* = \alpha_0 + I + J \qquad \beta_k^* = \beta_0 + \sum_{i=1}^{I} U_{ik} + \sum_{j=1}^{J} V_{jk}$$

$$\tau_{ik}^U = \tau \sum_{j \in \Omega_i^1} V_{jk}^2 \qquad \mu_{ik}^U = \frac{1}{\tau_{ik}^U} \left(-\lambda_k + \tau \sum_{j \in \Omega_i^1} \left(R_{ij} - \sum_{k' \neq k} U_{ik'} V_{jk'} \right) V_{jk} \right)$$

$$\tau_{jk}^V = \tau \sum_{i \in \Omega_j^2} U_{ik}^2 \qquad \mu_{jk}^V = \frac{1}{\tau_{jk}^V} \left(-\lambda_k + \tau \sum_{i \in \Omega_j^2} \left(R_{ij} - \sum_{k' \neq k} U_{ik'} V_{jk'} \right) U_{ik} \right).$$

Bayesian nonnegative matrix tri-factorisation

For the NMTF Gibbs sampling algorithm we need to sample from the following posteriors, where $\boldsymbol{\lambda}^F$ is a vector including all λ_k^F values, $\boldsymbol{\lambda}_{-k}^F$ excludes λ_k^F, and similarly for $\boldsymbol{\lambda}^G$.

$$\begin{aligned}
p(\tau | \boldsymbol{F}, \boldsymbol{S}, \boldsymbol{G}, \boldsymbol{\lambda}^F, \boldsymbol{\lambda}^G, D) &= \mathcal{G}(\tau | \alpha_\tau^*, \beta_\tau^*) \\
p(\lambda_k^F | \boldsymbol{F}, \boldsymbol{S}, \boldsymbol{G}, \boldsymbol{\lambda}_{-k}^F, \boldsymbol{\lambda}^G, D) &= \mathcal{G}(\lambda_k^F | \alpha_k^{F*}, \beta_k^{F*}) \\
p(\lambda_l^G | \boldsymbol{F}, \boldsymbol{S}, \boldsymbol{G}, \boldsymbol{\lambda}^F, \boldsymbol{\lambda}_{-l}^G, D) &= \mathcal{G}(\lambda_l^G | \alpha_l^{G*}, \beta_l^{G*}) \\
p(F_{ik} | \tau, \boldsymbol{F}_{-ik}, \boldsymbol{S}, \boldsymbol{G}, \boldsymbol{\lambda}^F, \boldsymbol{\lambda}^G, D) &= \mathcal{TN}(F_{ik} | \mu_{ik}^F, \tau_{ik}^F) \\
p(S_{kl} | \tau, \boldsymbol{F}, \boldsymbol{S}_{-kl}, \boldsymbol{G}, \boldsymbol{\lambda}^F, \boldsymbol{\lambda}^G, D) &= \mathcal{TN}(S_{kl} | \mu_{kl}^S, \tau_{kl}^S) \\
p(G_{jl} | \tau, \boldsymbol{F}, \boldsymbol{S}, \boldsymbol{G}_{-jl}, \boldsymbol{\lambda}^F, \boldsymbol{\lambda}^G, D) &= \mathcal{TN}(G_{jl} | \mu_{jl}^G, \tau_{jl}^G),
\end{aligned}$$

with

$$\alpha_\tau^* = \alpha_\tau + \frac{|\Omega|}{2} \qquad \beta_\tau^* = \beta_\tau + \frac{1}{2} \sum_{(i,j) \in \Omega} (R_{ij} - \boldsymbol{F}_i \cdot \boldsymbol{S} \cdot \boldsymbol{G}_j)^2$$

$$\alpha_k^{F*} = \alpha_0 + I \qquad \beta_k^{F*} = \beta_0 + \sum_{i=1}^{I} F_{ik}$$

$$\alpha_l^{G*} = \alpha_0 + J \qquad \beta_l^{G*} = \beta_0 + \sum_{j=1}^{J} G_{jl}$$

$$\tau_{ik}^F = \tau \sum_{j \in \Omega_i^1} (\boldsymbol{S}_k \cdot \boldsymbol{G}_j)^2 \qquad \mu_{ik}^F = \frac{1}{\tau_{ik}^F} \left(-\lambda_k^F + \tau \sum_{j \in \Omega_i^1} \left(R_{ij} - \sum_{k' \neq k} \sum_{l=1}^{L} F_{ik'} S_{k'l} G_{jl} \right) (\boldsymbol{S}_k \cdot \boldsymbol{G}_j) \right)$$

$$\tau_{kl}^S = \tau \sum_{(i,j) \in \Omega} F_{ik}^2 G_{jl}^2 \qquad \mu_{kl}^S = \frac{1}{\tau_{kl}^S} \left(-\lambda_S + \tau \sum_{(i,j) \in \Omega} \left(R_{ij} - \sum_{(k',l') \neq (k,l)} F_{ik'} S_{k'l'} G_{jl'} \right) F_{ik} G_{jl} \right)$$

$$\tau_{jl}^G = \tau \sum_{i \in \Omega_j^2} (\boldsymbol{F}_i \cdot \boldsymbol{S}_{\cdot,l})^2 \qquad \mu_{jl}^G = \frac{1}{\tau_{jl}^G} \left(-\lambda_l^G + \tau \sum_{i \in \Omega_j^2} \left(R_{ij} - \sum_{k=1}^{K} \sum_{l' \neq l} F_{ik} S_{kl'} G_{jl'} \right) (\boldsymbol{F}_i \cdot \boldsymbol{S}_{\cdot,l}) \right).$$

3.2.3 Iterated conditional modes

The iterated conditional models (ICM) algorithm for inference in the NMF model was also given in Schmidt et al. [2009]. This algorithm works very similarly to the Gibbs sampler, but instead of randomly drawing a value from the conditional posteriors, we take the mode at each iteration. This gives a maximum a posteriori (MAP) point estimate $\boldsymbol{\theta}_{\text{MAP}} = \max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|D)$, rather than a full posterior distribution. For random variables $X \sim \mathcal{G}(a, b)$, $Y \sim \mathcal{TN}(\mu, \tau)$, the modes are $\frac{a-1}{b}$ and $\max(0, \mu)$, respectively.

In practice, the ICM methods often converge to solutions where multiple columns in the matrices are all zeros, leading to poor approximations. We have addressed this issue by resetting zeros to a small positive value like 0.1 at each iteration.

3.2.4 Variational Bayesian inference

Variational Bayesian inference (VB) has been used for other matrix factorisation models before (Gönen [2012]), but not for the nonnegative model in this paper. We therefore now introduce a new VB algorithm for our model. Like Gibbs sampling, VB is a way to approximate the true posterior $p(\boldsymbol{\theta}|D)$. Recall from Section 2.2.2 that the idea behind VB is to introduce an approximation $q(\boldsymbol{\theta})$ to the true posterior that is easier to compute, and to make our variational distribution $q(\boldsymbol{\theta})$ as similar to $p(\boldsymbol{\theta}|D)$ as possible (as measured by the KL-divergence). We make the mean-field assumption, so that the variational distribution $q(\boldsymbol{\theta})$ factorises completely and all variables are independent in the approximation of the posterior, $q(\boldsymbol{\theta}) = \prod_{\theta_i \in \boldsymbol{\theta}} q(\theta_i)$.

Bayesian nonnegative matrix factorisation

We can derive the forms of $q(\theta_i)$ using the expression given in Equation 2.2.2, which turn out to be the same as the posterior sampling distributions in the Gibbs sampling algorithms,

$$q(\tau) = \mathcal{G}(\tau|\alpha_\tau^*, \beta_\tau^*) \qquad q(\lambda_k) = \mathcal{G}(\lambda_k|\alpha_k^*, \beta_k^*)$$
$$q(U_{ik}) = \mathcal{TN}(U_{ik}|\mu_{ik}^U, \tau_{ik}^U) \qquad q(V_{jk}) = \mathcal{TN}(V_{jk}|\mu_{jk}^V, \tau_{jk}^V).$$

Variational Bayesian inference works by iteratively updating the posterior approximations $q(\theta_i)$ to all the random variables θ_i in the model. We do this by computing the variational parameters above for each variable, and then updating the expectation and variance with respect to the variational approximation q.

The derivation for the variational parameter values for U_{ik} is given below. We use $\widetilde{f(X)}$ as a shorthand for $\mathbb{E}_q[f(X)]$, where X is a random variable and f is a function over X.

$$
\begin{aligned}
q^*(U_{ik}) &\propto \exp\left\{\mathbb{E}_{q(\boldsymbol{\theta}_{-U_{ik}})}\left[\log p(D|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta})\right]\right\} \\
&\propto \exp\left\{\mathbb{E}_{q(\boldsymbol{\theta}_{-U_{ik}})}\left[\sum_{j\in\Omega_i^1} \log p(R_{ij}|\boldsymbol{U},\boldsymbol{V}) + \log p(U_{ik}|\lambda_k)\right]\right\} \\
&\propto \exp\left\{\mathbb{E}_{q(\boldsymbol{\theta}_{-U_{ik}})}\left[\sum_{j\in\Omega_i^1} \log\left[\sqrt{\frac{\tau}{2\pi}}\exp\left\{-\frac{\tau}{2}\left(R_{ij}-\boldsymbol{U}_i\boldsymbol{V}_j\right)^2\right\}\right]\right.\right. \\
&\qquad\qquad \left.\left. + \log\left[\lambda_k \exp\left\{-\lambda_k U_{ik}\right\}\right]\right]\right\} \cdot u(x) \\
&\propto \exp\left\{\mathbb{E}_{q(\boldsymbol{\theta}_{-U_{ik}})}\left[\sum_{j\in\Omega_i^1} -\frac{\tau}{2}\left(R_{ij}-\boldsymbol{U}_i\boldsymbol{V}_j\right)^2 - \lambda_k U_{ik}\right]\right\} \cdot u(x) \\
&\propto \exp\left\{-\frac{U_{ik}^2}{2}\left[\widetilde{\tau}\sum_{j\in\Omega_i^1}\widetilde{V_{jk}}^2\right]\right. \\
&\qquad \left. + U_{ik}\left[-\widetilde{\lambda_k} + \widetilde{\tau}\sum_{j\in\Omega_i^1}\left(R_{ij} - \sum_{k'\neq k}\widetilde{U_{ik'}}\,\widetilde{V_{jk'}}\right)\widetilde{V_{jk}}\right]\right\} \cdot u(x) \\
&\propto \exp\left\{-\frac{\tau_{ik}^U}{2}(U_{ik}-\mu_{ik}^U)^2\right\} \cdot u(x) \\
&\propto \mathcal{TN}(U_{ik}|\mu_{ik}^U, \tau_{ik}^U).
\end{aligned}
$$

Note that we use the expectation of the other variables in these updates. The expectation and variance for random variables $X \sim \mathcal{G}(a,b)$ and $Y \sim \mathcal{TN}(\mu,\tau)$ are

$$
\widetilde{X} = \frac{a}{b} \qquad \widetilde{Y} = \mu + \frac{1}{\sqrt{\tau}}\lambda\left(-\mu\sqrt{\tau}\right) \qquad \mathrm{Var}[Y] = \frac{1}{\tau}\left[1 - \delta\left(-\mu\sqrt{\tau}\right)\right],
$$

where $\psi(x) = \frac{d}{dx}\log\Gamma(x)$ is the digamma function, $\lambda(x) = \phi(x)/[1-\Phi(x)]$, and $\delta(x) = \lambda(x)[\lambda(x)-x]$. $\phi(x) = \frac{1}{\sqrt{2\pi}}\exp\{-\frac{1}{2}x^2\}$ is the density function of $\mathcal{N}(0,1)$.

All the variational parameter values for NMF are given below. In the code implementation of these updates we make use of the identity $\widetilde{X^2} = \widetilde{X}^2 + \mathrm{Var}_q[X]$. Finally, note that $\mathbb{E}_q[(R_{ij} - \boldsymbol{U}_i\boldsymbol{V}_j)^2] = \left(R_{ij} - \sum_{k=1}^K \widetilde{U_{ik}}\,\widetilde{V_{jk}}\right)^2 + \sum_{k=1}^K \left(\widetilde{U_{ik}^2}\widetilde{V_{jk}^2} - \widetilde{U_{ik}}^2\widetilde{V_{jk}}^2\right)$.

$$\alpha_\tau^* = \alpha_\tau + \frac{|\Omega|}{2} \qquad \beta_\tau^* = \beta_\tau + \frac{1}{2}\sum_{(i,j)\in\Omega} \mathbb{E}_q\left[(R_{ij} - \boldsymbol{U}_i\boldsymbol{V}_j)^2\right]$$

$$\alpha_k^* = \alpha_0 + I + J \qquad \beta_k^* = \beta_0 + \sum_{i=1}^I \widetilde{U_{ik}} + \sum_{j=1}^J \widetilde{V_{jk}}$$

$$\tau_{ik}^U = \widetilde{\tau}\sum_{j\in\Omega_i^1} \widetilde{V_{jk}^2} \qquad \mu_{ik}^U = \frac{1}{\tau_{ik}^U}\left(-\widetilde{\lambda}_k + \widetilde{\tau}\sum_{j\in\Omega_i^1}\left(R_{ij} - \sum_{k'\neq k}\widetilde{U_{ik'}}\,\widetilde{V_{jk'}}\right)\widetilde{V_{jk}}\right)$$

$$\tau_{jk}^V = \widetilde{\tau}\sum_{i\in\Omega_j^2} \widetilde{U_{ik}^2} \qquad \mu_{jk}^V = \frac{1}{\tau_{jk}^V}\left(-\widetilde{\lambda}_k + \widetilde{\tau}\sum_{i\in\Omega_j^2}\left(R_{ij} - \sum_{k'\neq k}\widetilde{U_{ik'}}\,\widetilde{V_{jk'}}\right)\widetilde{U_{ik}}\right).$$

We can see the similarity with the Gibbs sampling algorithm: the posterior parameter values are identical, except in this algorithm we use the expectation of random variables (as opposed to their current draw value), and the expectation term $\mathbb{E}_q[(R_{ij} - \boldsymbol{U}_i\boldsymbol{V}_j)^2]$ yields an additional variance term.

Bayesian nonnegative matrix tri-factorisation

Our VB algorithm for NMTF follows the same steps as for NMF, but now has an added complexity due to the term $\mathbb{E}_q[(R_{ij} - \boldsymbol{F}_i\boldsymbol{S}\boldsymbol{G}_j)^2]$. For NMF all covariance terms for $k' \neq k$ are zero due to the factorisation in q, but we now obtain some additional non-zero covariance terms,

$$\begin{aligned}\mathbb{E}_q\left[(R_{ij} - \boldsymbol{F}_i\boldsymbol{S}\boldsymbol{G}_j)^2\right] = &\left(R_{ij} - \sum_{k=1}^K\sum_{l=1}^L \widetilde{F_{ik}}\widetilde{S_{kl}}\widetilde{G_{jl}}\right)^2 \\ &+ \sum_{k=1}^K\sum_{l=1}^L \mathrm{Var}_q\left[F_{ik}S_{kl}G_{jl}\right] &(3.1)\\ &+ \sum_{k=1}^K\sum_{l=1}^L\sum_{k'\neq k} \mathrm{Cov}\left[F_{ik}S_{kl}G_{jl}, F_{ik'}S_{k'l}G_{jl}\right] &(3.2)\\ &+ \sum_{k=1}^K\sum_{l=1}^L\sum_{l'\neq l} \mathrm{Cov}\left[F_{ik}S_{kl}G_{jl}, F_{ik}S_{kl'}G_{jl'}\right]. &(3.3)\end{aligned}$$

The above variance and covariance terms (3.1, 3.2, 3.3) are equal to the following, respectively.

$$\widetilde{F_{ik}^2}\widetilde{S_{kl}^2}\widetilde{G_{jl}^2} - \widetilde{F_{ik}}^2\widetilde{S_{kl}}^2\widetilde{G_{jl}}^2, \qquad \mathrm{Var}_q\left[F_{ik}\right]\widetilde{S_{kl}}\widetilde{G_{jl}}\widetilde{S_{kl'}}\widetilde{G_{jl'}}, \qquad \widetilde{F_{ik}}\widetilde{S_{kl}}\mathrm{Var}_q\left[G_{jl}\right]\widetilde{F_{ik'}}\widetilde{S_{k'l}}.$$

We have the following approximations to the posteriors for the NMTF Variational Bayes algorithm:

$$q(\tau) = \mathcal{G}(\tau|\alpha_\tau^*, \beta_\tau^*) \qquad q(\lambda_k^F) = \mathcal{G}(\lambda_k^F|\alpha_k^{F*}, \beta_k^{F*}) \qquad q(\lambda_l^G) = \mathcal{G}(\lambda_l^G|\alpha_l^{G*}, \beta_l^{G*})$$
$$q(F_{ik}) = \mathcal{TN}(F_{ik}|\mu_{ik}^F, \tau_{ik}^F) \qquad q(S_{kl}) = \mathcal{TN}(S_{kl}|\mu_{kl}^S, \tau_{kl}^S) \qquad q(G_{jl}) = \mathcal{TN}(G_{jl}|\mu_{jl}^G, \tau_{jl}^G),$$

with

$$\alpha_\tau^* = \alpha_\tau + \frac{|\Omega|}{2} \qquad \beta_\tau^* = \beta_\tau + \frac{1}{2}\sum_{(i,j)\in\Omega}\mathbb{E}_q\left[(R_{ij} - \boldsymbol{F}_i\boldsymbol{S}\boldsymbol{G}_j)^2\right]$$

$$\alpha_k^{F*} = \alpha_0 + I \qquad \beta_k^{F*} = \beta_0 + \sum_{i=1}^{I}\widetilde{F_{ik}}$$

$$\alpha_l^{G*} = \alpha_0 + J \qquad \beta_l^{G*} = \beta_0 + \sum_{j=1}^{J}\widetilde{G_{jl}}$$

$$\tau_{ik}^F = \widetilde{\tau}\sum_{j\in\Omega_i^1}\left[\left(\sum_{l=1}^{L}\widetilde{S_{kl}}\widetilde{G_{jl}}\right)^2 + \sum_{l=1}^{L}\left(\widetilde{S_{kl}^2}\widetilde{G_{jl}^2} - \widetilde{S_{kl}}^2\widetilde{G_{jl}}^2\right)\right]$$

$$\mu_{ik}^F = \frac{1}{\tau_{ik}^F}\left(-\widetilde{\lambda_k^F} + \widetilde{\tau}\sum_{j\in\Omega_i^1}\left[\left(R_{ij} - \sum_{k'\neq k}\sum_{l=1}^{L}\widetilde{F_{ik'}}\widetilde{S_{k'l}}\widetilde{G_{jl}}\right)\sum_{l=1}^{L}\widetilde{S_{kl}}\widetilde{G_{jl}} - \sum_{l=1}^{L}\widetilde{S_{kl}}\mathrm{Var}_q\left[G_{jl}\right]\sum_{k'\neq k}\widetilde{F_{ik'}}\widetilde{S_{k'l}}\right]\right)$$

$$\tau_{kl}^S = \widetilde{\tau}\sum_{(i,j)\in\Omega}\widetilde{F_{ik}^2}\widetilde{G_{jl}^2}$$

$$\mu_{kl}^S = \frac{1}{\tau_{kl}^S}\left(-\lambda_{kl}^S + \widetilde{\tau}\sum_{(i,j)\in\Omega}\left[\left(R_{ij} - \sum_{(k',l')\neq(k,l)}\widetilde{F_{ik'}}\widetilde{S_{k'l'}}\widetilde{G_{jl'}}\right)\widetilde{F_{ik}}\widetilde{G_{jl}} - \widetilde{F_{ik}}\mathrm{Var}_q\left[G_{jl}\right]\sum_{k'\neq k}\widetilde{F_{ik'}}\widetilde{S_{k'l}} - \mathrm{Var}_q\left[F_{ik}\right]\widetilde{G_{jl}}\sum_{l'\neq l}\widetilde{S_{kl'}}\widetilde{G_{jl'}}\right]\right)$$

$$\tau_{jl}^G = \widetilde{\tau}\sum_{i\in\Omega_j^2}\left[\left(\sum_{k=1}^{K}\widetilde{F_{ik}}\widetilde{S_{kl}}\right)^2 + \sum_{k=1}^{K}\left(\widetilde{F_{ik}^2}\widetilde{S_{kl}^2} - \widetilde{F_{ik}}^2\widetilde{S_{kl}}^2\right)\right]$$

$$\mu_{jl}^G = \frac{1}{\tau_{jl}^G}\left(-\widetilde{\lambda_l^G} + \widetilde{\tau}\sum_{i\in\Omega_j^2}\left[\left(R_{ij} - \sum_{k=1}^{K}\sum_{l'\neq l}\widetilde{F_{ik}}\widetilde{S_{kl'}}\widetilde{G_{jl'}}\right)\sum_{k=1}^{K}\widetilde{F_{ik}}\widetilde{S_{kl}} - \sum_{k=1}^{K}\mathrm{Var}_q\left[F_{ik}\right]\widetilde{S_{kl}}\sum_{l'\neq l}\widetilde{S_{kl'}}\widetilde{G_{jl'}}\right]\right).$$

Again note the similarity with the posterior parameter values for the Gibbs sampler for NMTF. However, the VB algorithm now has several additional terms, due to the extra dependencies (variance and covariance terms) that come with the tri-factorisation.

3.3 Implementation details

3.3.1 Software implementation

Implementations of all methods, the datasets, and experiments described in the next section, are available at https://github.com/ThomasBrouwer/BNMTF_ARD. We used the Python language. The *numpy* package was used for fast matrix operations, and for random draws of the truncated normal distribution we used the Python package *rtnorm* by C. Lassner (http://miv.u-strasbg.fr/mazet/rtnorm/), giving more efficient draws than the standard libraries and dealing with rounding errors.

The mean and variance of the truncated normal involve operations prone to numerical errors when μ takes high negative values. To deal with this we observe that when $\mu\tau \ll 0$ the truncated normal distribution approximates an exponential one with rate $|\mu\tau|$, and therefore has mean $1/|\mu\tau|$ and variance $1/|\mu\tau|^2$.

All experiments in Section 3.5 were run on a MacBook Pro laptop, with 2.2 GHz Intel Core i7 processor, 16 GB 1600 MHz DDR3 memory, and an Intel Iris Pro 1536 MB Graphics card.

3.3.2 Computational complexity

Each of the four approaches have the same time complexities, but vary in how efficiently the updates can be computed, and how quickly they converge. The time complexity per iteration for NMF is $\mathcal{O}(IJK^2)$, and $\mathcal{O}(IJ(K^2L + KL^2))$ per iteration for NMTF. However, the updates in each column of $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{F}, \boldsymbol{G}$ are independent of each other and can therefore be updated in parallel. This means we can joinly compute and update the (parameter values of the) columns using a single matrix operation. Modern computer

architectures can exploit this using vector processors, leading to a great speedup. For the Gibbs sampler and ICM algorithms we can also draw the new values in parallel.

Furthermore, after the VB algorithm converges we have our approximation to the posterior distributions immediately, whereas with Gibbs and ICM we need to obtain further draws after convergence and use a thinning rate to obtain an accurate MAP (ICM) or posterior (Gibbs) estimate. This deterministic behaviour of VB and NP makes them easier to use. Although additional variables need to be stored to represent the posteriors, this does not result in a worse space complexity, as the Gibbs sampler needs to store draws over time.

3.3.3 Initialisation

Initialising the parameters of the models can vastly influence the quality of convergence. This can be done by using the hyperparameters (λ_U, λ_V, λ_F, λ_S, λ_G, α, β, α_0, β_0, α_0^F, β_0^F, α_0^G, β_0^G) to set the initial values to the mean of the priors of the model, or using random draws. We found that random draws tend to give faster and better convergence than the expectation. For matrix tri-factorisation we can also initialise $\boldsymbol{F}$ by running the K-means clustering algorithm on the rows as datapoints, and similarly $\boldsymbol{G}$ for the columns, as suggested by Ding et al. [2006]. For the VB and NP algorithms we then set the μ parameters to the cluster indicators, and for Gibbs and ICM we set the matrices to the cluster indicators, plus 0.2 for smoothing. We found that this improved the convergence as well, with $\boldsymbol{S}$ initialised using random draws. These findings are confirmed by the initialisation experiment in the last chapter (5.6.1).

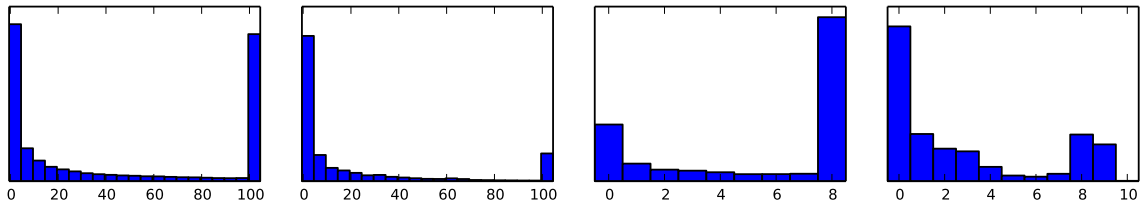
3.4 Data preprocessing

We consider a total of six different datasets: two synthetic ones (randomly generated from the NMF and NMTF model distributions), and four drug sensitivity datasets.

For the synthetic datasets we generated the latent matrices using unit mean exponential distributions, and adding zero mean unit variance Gaussian noise to the resulting product. For the matrix factorisation model we used $I = 100, J = 80, K = 10$, and for the matrix tri-factorisation $I = 100, J = 80, K = 5, L = 5$. These datasets perfectly satisfy the Bayesian models' priors and likelihood choice, and therefore we would expect Gibbs sampling, ICM, and VB to do especially well on them.

Table 3.1 Overview of the four drug sensitivity datasets, giving the number of cell lines (rows), drugs (columns), and the fraction of entries that are observed.

Dataset	Cell lines	Drugs	Fraction observed
GDSC IC_{50}	707	139	0.806
CTRP EC_{50}	887	545	0.801
CCLE IC_{50}	504	24	0.965
CCLE EC_{50}	504	24	0.630



(a) GDSC IC_{50} (b) CTRP EC_{50} (c) CCLE IC_{50} (d) CCLE EC_{50}

Figure 3.3 Plots of the distribution of values in the drug sensitivity datasets, after capping the extremely high values in the CTRP EC_{50} and GDSC IC_{50} datasets to 100.

We considered the four drug sensitivity datasets introduced in Section 2.4.1, each detailing the effectiveness (IC_{50} or EC_{50} values) of a range of drugs on different cell lines for cancer and tissue types, where some of the entries are missing. In particular, we used the Genomics of Drug Sensitivity in Cancer (GDSC v5.0, Yang et al. [2013], IC_{50}), Cancer Therapeutics Response Portal (CTRP v2, Seashore-Ludlow et al. [2015], EC_{50}), and Cancer Cell Line Encyclopedia (CCLE, Barretina et al. [2012], both IC_{50} and EC_{50}) datasets. The values in the CCLE datasets were in the range [0,8] for IC_{50}, and [0,10] for EC_{50}. The CTRP values were all nonnegative, with some very high values, so we capped them at 100. For the GDSC dataset we undid the natural log transform by taking the exponent, making all values nonnegative, and then also capped high values at 100. For this dataset we also had two cell lines with only one and two observed entries, so we filtered them out.

A summary of the datasets can also be found in Table 3.1. Distributions of the values are plotted in Figure 3.3.

3.5 Experiments

To demonstrate the trade-offs between the four inference methods presented, we conducted experiments on the two synthetic and four real-world drug sensitivity datasets.

We compare the convergence speeds, robustness to noise, robustness to sparsity, effectiveness of the ARD for model selection, and sensitivity to the hyperparameter values. At the end of each section we provide a short summary of our findings for that experiment.

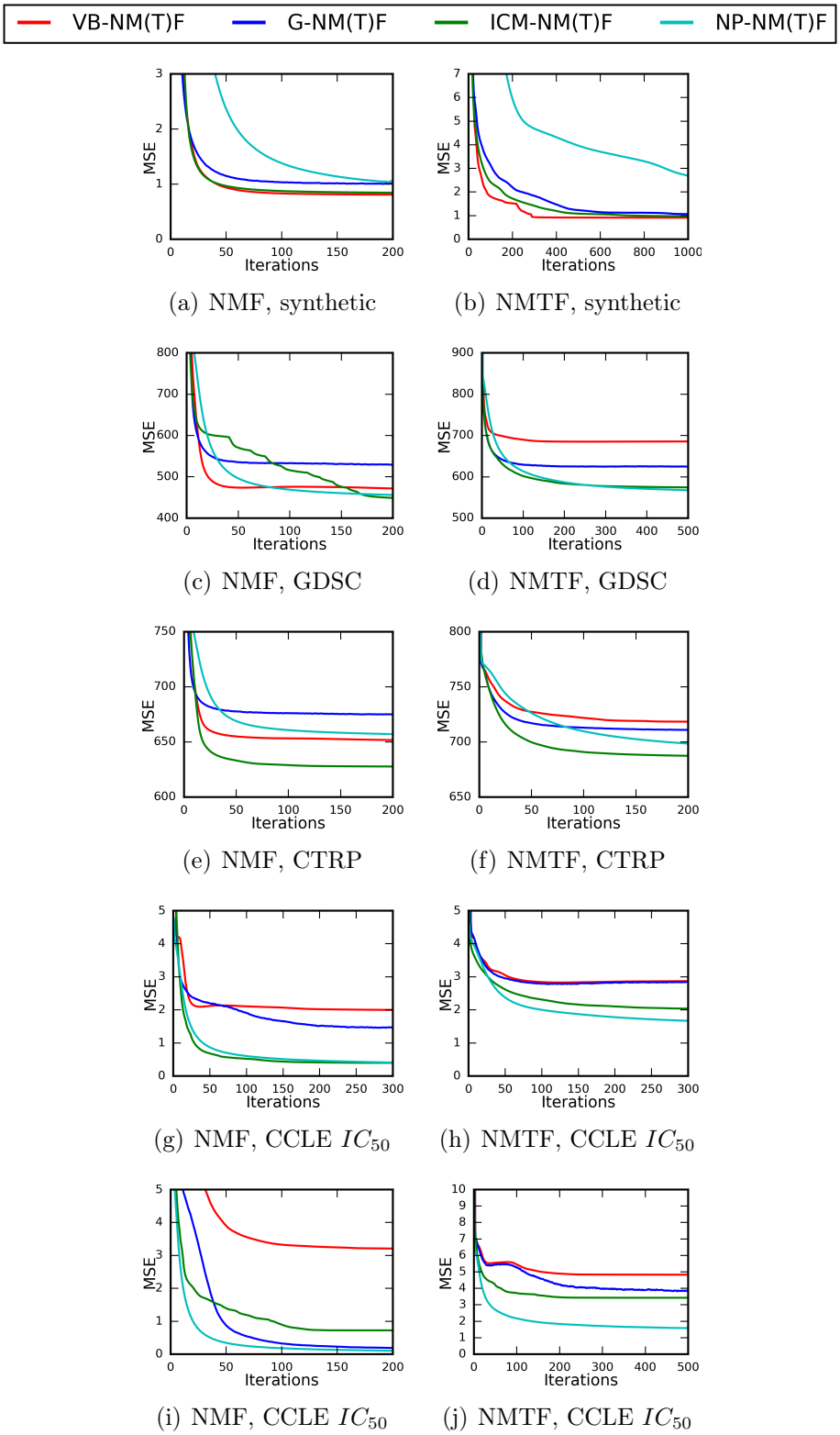
For all models we used weak priors ($\lambda = 0.1$, $\alpha_\tau = \beta_\tau = \alpha_0 = \beta_0 = 1$). Unless explicitly stated otherwise, we used the models without ARD. We generally ran each algorithm for 1000 iterations (burn-in 800, thinning rate 5), and sometimes longer, to ensure all models properly converged.

3.5.1 Convergence and runtime speed

We firstly measured the convergence speeds of the different inference methods on the datasets, using the versions of NMF and NMTF without ARD. Convergence plots on all datasets are given in Figure 3.4, plotting the error (mean squared error) on the training data against the number of iterations taken, for the NMF (left column) and NMTF (right column) models. For the synthetic data we used the correct number of factors, and for the drug sensitivity datasets we used $K = 20$ for NMF and $K = L = 10$ for NMTF. We ran each method 20 times, taking the average training errors and timestamps.

Although the results are empirical, they illustrate that different inference approaches have different convergence speeds and depths (the final training error reached). On the synthetic data (left column), VB is the fastest against iterations, followed by ICM and Gibbs, and finally NP. All methods reach the optimal MSE of 1 (which is the level of noise added). On the real-world drug sensitivity datasets, all methods reach their lowest depth at roughly the same number of iterations. However, ICM and NP generally converge much deeper than VB and Gibbs. Although this initially seems good, this is a sign of overfitting to the training data, and can lead to poor predictions for unseen data. We will see this later in the noise and sparsity experiments (Sections 3.5.3 and 3.5.4), where VB and Gibbs are more robust than ICM and NP.

The average time taken per iteration for each of the methods on the five datasets are given in Table 3.2, which demonstrate that the ICM and NP methods can be implemented much more efficiently than the fully Bayesian models. The NP approach takes the least amount of time per iteration, followed by ICM, VB, and then Gibbs. We also plotted the convergence against time taken (in seconds), which is given in



(a) NMF, synthetic (b) NMTF, synthetic

(c) NMF, GDSC (d) NMTF, GDSC

(e) NMF, CTRP (f) NMTF, CTRP

(g) NMF, CCLE IC_{50} (h) NMTF, CCLE IC_{50}

(i) NMF, CCLE IC_{50} (j) NMTF, CCLE IC_{50}

Figure 3.4 Convergence of the different inference approaches on the synthetic and drug sensitivity datasets, measuring the training data fit (mean square error) against iterations, for NMF (left column) and NMTF (right column).

Table 3.2 Average time (in seconds) taken per iteration for the four Bayesian matrix factorisation and tri-factorisation inference methods, on the synthetic and four drug sensitivity datasets.

Method	Synthetic	GDSC IC_{50}	CTRP EC_{50}	CCLE $IC50$	CCLE $EC50$
NMF VB	0.015	0.125	0.387	0.067	0.064
NMF Gibbs	0.024	0.251	0.655	0.175	0.143
NMF ICM	0.003	0.047	0.279	0.012	0.012
NMF NP	0.002	0.042	0.268	0.010	0.013
NMTF VB	0.019	0.298	1.703	0.114	0.111
NMTF Gibbs	0.014	0.264	1.557	0.107	0.107
NMTF ICM	0.005	0.173	1.259	0.035	0.034
NMTF NP	0.004	0.124	0.697	0.026	0.030

Figure 3.5. Note the weird fitting behaviour of NP-NMF on the synthetic data, which occasionally happens.

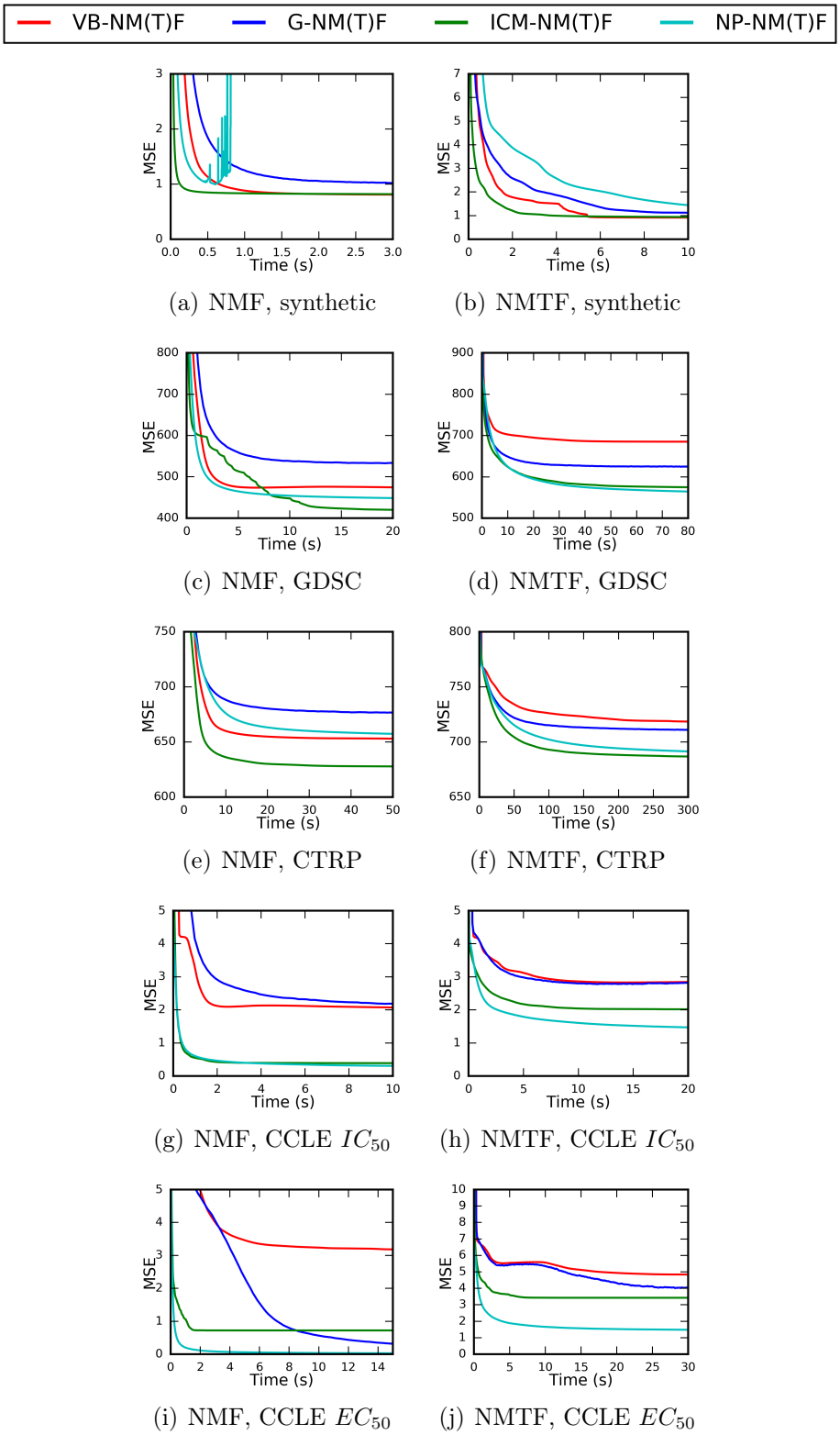
Finally, note that the matrix factorisation models generally converge deeper than the matrix tri-factorisation ones (for example on GDSC, NMF reaches a MSE of 450, whereas NMTF only gets to 550).

Summary: ICM and NP give the fastest convergence, followed by VB, and then Gibbs. ICM and NP converge the deepest, followed by Gibbs, and then VB. Finally, NMF converges deeper than NMTF.

3.5.2 Cross-validation performance

Next we measured the cross-validation performances of the methods on the four drug sensitivity datasets. For each method we performed 10-fold nested cross-validation (nested to pick the dimensionality K; for simplicity we used $L = K$ for the NMTF models), giving the average performance in Figure 3.6. For the ARD models we did not need to pick the dimensionality, instead using $K = 20$ for NMF, and $K = 10, L = 10$ for NMTF.

We can see that most models perform very similarly, with little to no difference between the matrix factorisation and tri-factorisation versions. Using the ARD models generally works equally well as without ARD, but with the added benefit of not having to run nested cross-validation to choose the dimensionality, reducing the running time from hours to minutes. We will see in Section 3.5.5 that the ARD is actually very efficient



(a) NMF, synthetic (b) NMTF, synthetic

(c) NMF, GDSC (d) NMTF, GDSC

(e) NMF, CTRP (f) NMTF, CTRP

(g) NMF, CCLE IC_{50} (h) NMTF, CCLE IC_{50}

(i) NMF, CCLE EC_{50} (j) NMTF, CCLE EC_{50}

Figure 3.5 Convergence of the different inference approaches on the synthetic and drug sensitivity datasets, measuring the training data fit (mean square error) against time taken (in seconds), for NMF (top row) and NMTF (bottom row).

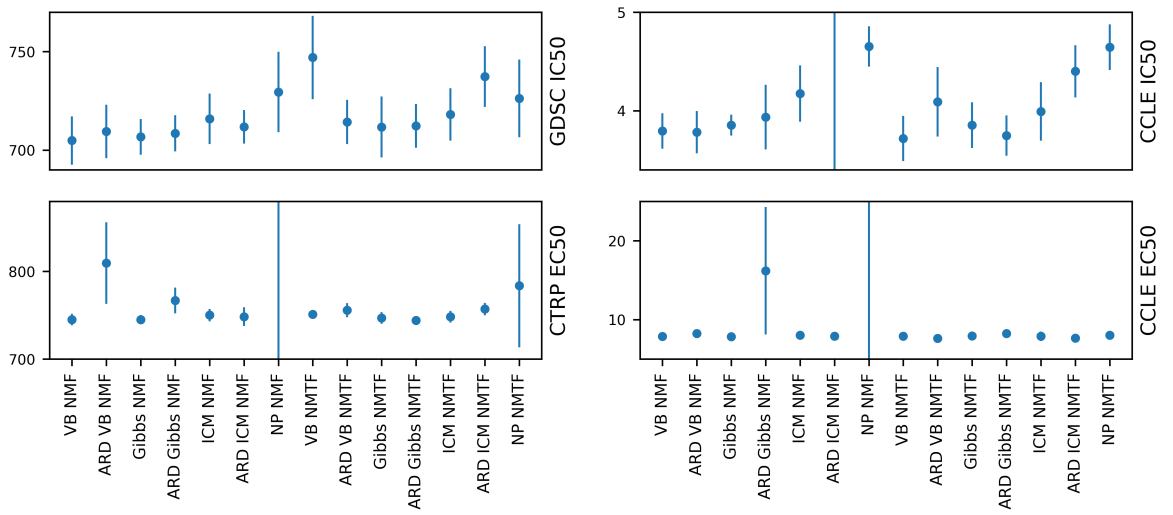


Figure 3.6 10-fold cross-validation results (mean squared error) for drug sensitivity predictions on each of the four datasets. Each boxplot gives the mean and standard deviation.

at turning off unnecessary factors and reducing overfitting. However, sometimes the ARD does not work as desired and overfits (for example VB NMF on CTRP EC_{50}, ICM NMF on CCLE IC_{50}, and Gibbs NMF on CCLE EC_{50}).

We can also see that the VB and Gibbs models often do better than the NP and ICM versions, such as on the CCLE IC_{50} and GDSC IC_{50} datasets. On the CTRP and CCLE EC_{50} datasets the NP NMF model overfits massively on some of the folds, leading to extremely high predictive errors.

In addition to the predictive performances, we give the most common dimensionalities found in nested cross-validation in Table 3.3. This is used for the sparsity tests in Section 3.5.4, but also provides us with some interesting insights. First of all, note that the best dimensionality is roughly the same for NMF and NMTF. Furthermore, VB and Gibbs have the highest values, because they overfit less when given more factors and can therefore leverage more of them. Finally, the CCLE IC_{50} dataset has dimensionality 1 for all methods, indicating that no sensible predictions can be made other than a weighted row and column average. We see this in the crossvalidation results as well, where most models achieve identical performances.

Summary: the fully Bayesian models tend to outperform the NP and ICM ones; the ARD models are often as efficient as the non-ARD ones using nested cross-validation;

Table 3.3 Average nested cross-validation dimensionality (K for NMF, K, L for NMTF) of the inference approaches on the four drug sensitivity datasets.

Method	GDSC IC_{50}	CTRP EC_{50}	CCLE $IC50$	CCLE $EC50$
NMF VB	7	6	5	1
NMF Gibbs	8	7	5	1
NMF ICM	5	4	4	1
NMF NP	6	3	1	1
NMTF VB	5,5	9,9	7,7	1,1
NMTF Gibbs	10,10	8,8	7,7	1,1
NMTF ICM	6,6	6,6	4,4	1,1
NMTF NP	6,6	4,4	1,1	1,1

and there is little to no difference between the matrix factorisation and tri-factorisation models.

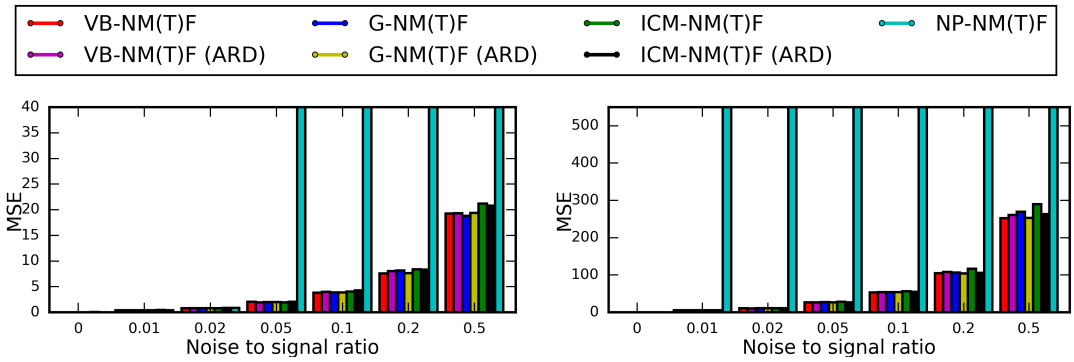
3.5.3 Noise test

We conducted a noise test on the synthetic data, to measure the robustness of the different methods. We added different levels of Gaussian noise to the data, with the noise-to-signal ratio being given by the ratio of the variance of the Gaussian noise we add, to the variance of the generated data. For each noise level we split the datapoints randomly into ten folds, and measure the predictive performance of the models on one held-out set. The results are given in Figures 3.7(a) (NMF) and 3.7(b) (NMTF), where we can see that the non-probabilistic approach starts overfitting heavily at low levels of noise, whereas the fully Bayesian approaches achieve the best possible predictive performances even at high levels of noise. ICM only does slightly worse in the very noisy cases. Note that the addition of ARD to the models does not make a difference for the robustness of the Bayesian models.

Summary: VB, Gibbs, and ICM are all very robust to noise; NP starts overfitting very quickly; ARD makes no difference for robustness.

3.5.4 Sparse predictions

We furthermore measured the robustness to sparsity of the data for each inference technique. For different fractions of missing values, varying between 10% and 90%, we randomly split the data ten times into train and test sets using those proportions, and measure the average predictive error. We conducted this experiment on the synthetic



(a) NMF (b) NMTF

Figure 3.7 Noise test performances on the drug sensitivity datasets, measured by average predictive performance on test set (mean square error) for different noise-to-signal ratios.

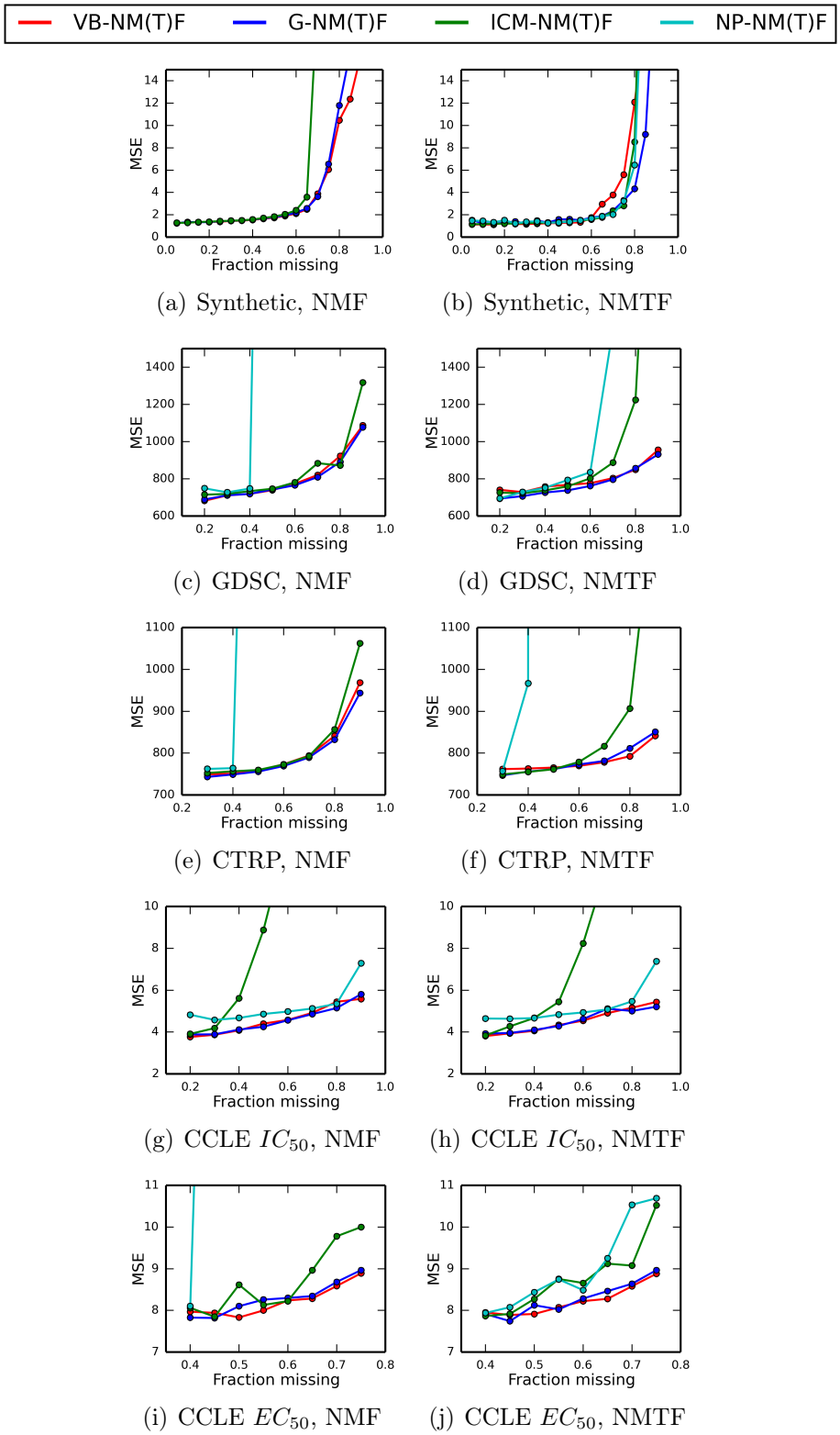
data, using the true dimensionality K (and L) for each model. We also performed it on the four drug sensitivity datasets, using the most common dimensionalities in the cross-validation from Section 3.5.2, given in Table 3.3.

The results are given in Figure 3.8 for the models without ARD. We can see that the non-probabilistic models generally start overfitting even on very low sparsity levels—in Figure 3.8(a) we cannot even see the line as the model overfits at each level of sparsity. The ICM models are also less robust when the sparsity is high. In contrast, the Gibbs sampling model achieves very good predictive performance even under extreme sparsity. The VB models are similar, but for sparser data it can sometimes not find the best solution, as can be seen in Figure 3.8(b). Finally, we repeated this experiment for the models with ARD (results omitted), but found that it makes no difference to the robustness to sparsity.

Summary: Gibbs is most robust to sparsity, closely followed by VB; NP is not robust to sparsity at all; and ICM is somewhere in the middle.

3.5.5 Model selection

We also conducted an experiment to see the extent of overfitting if the model is given a high dimensionality K, and whether this is remedied through the use of ARD. If we give a model a higher dimensionality, it can fit more closely to the data, but this can lead to overfitting and a higher predictive error. ARD can remedy this by turning off scarsely used factors, hopefully leading to less overfitting.



(a) Synthetic, NMF (b) Synthetic, NMTF

(c) GDSC, NMF (d) GDSC, NMTF

(e) CTRP, NMF (f) CTRP, NMTF

(g) CCLE IC_{50}, NMF (h) CCLE IC_{50}, NMTF

(i) CCLE EC_{50}, NMF (j) CCLE EC_{50}, NMTF

Figure 3.8 Sparsity test performances on the drug sensitivity datasets, measured by average predictive performance on test set (mean square error) for different sparsity levels. The left column gives the performances for NMF, and the right for NMTF.

On each of the four drug sensitivity datasets, we performed 10-fold cross-validation for different values of K (and L for NMTF, using $K = L$) for VB (red, top row), Gibbs (blue, middle row), and ICM (green, bottom row). We show these results in Figures 3.9 for NMF, and 3.10 for NMTF, where the results for models without ARD are given by dotted lines with crosses (x), and with ARD by solid lines with circles (o).

We can see that in most graphs, the models with ARD have a much flatter line as the dimensionality increases, hence reducing overfitting. This effect is more apparent for the NMF models than for the NTMF ones, probably because the NMF models generally converge deeper and therefore more easily overfit. The NMF ICM models in particular make very effective use of the ARD. The only exception to ARD improving the performances in model selection is NMTF ICM on the GDSC dataset (bottom row of Figure 3.10(a)), where the ARD is potentially preventing the model from fitting as much to the data, hence leading to poor predictive results. However, we did not observe this behaviour for any of the other inference methods or datasets.

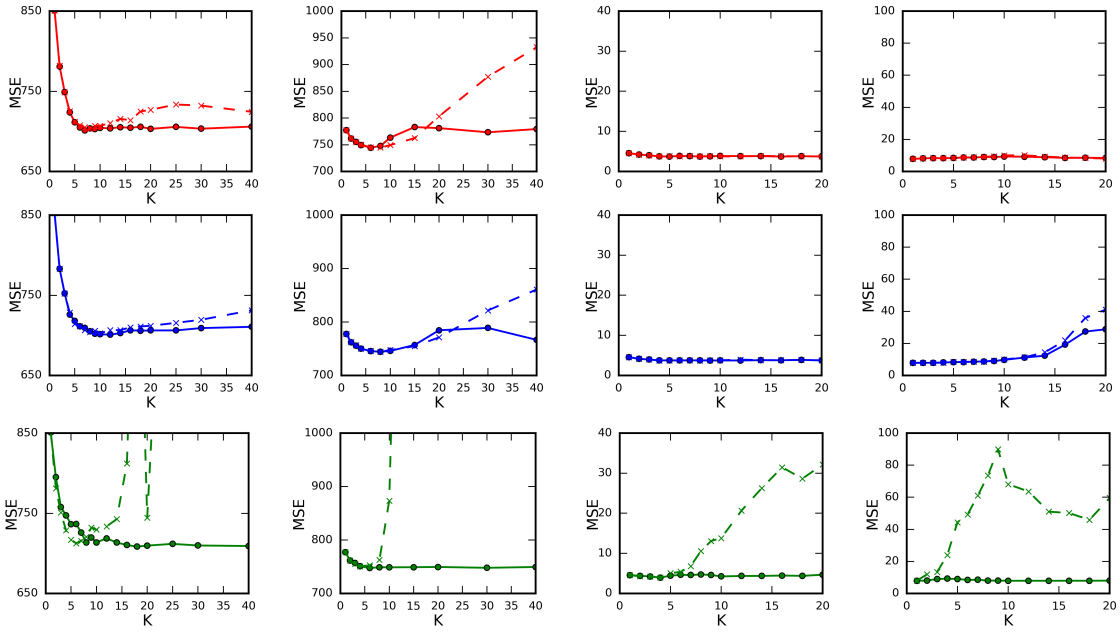
On the CCLE IC_{50} and EC_{50} datasets the Gibbs and VB models without ARD already have a flat line, demonstrating that the fully Bayesian approaches are naturally robust to overfitting (although it can help—as can be seen on the CTRP and GDSC datasets).

Summary: ARD can greatly improve the robustness of the models to overfitting in model selection, and especially for ICM.

3.5.6 Hyperparameter values

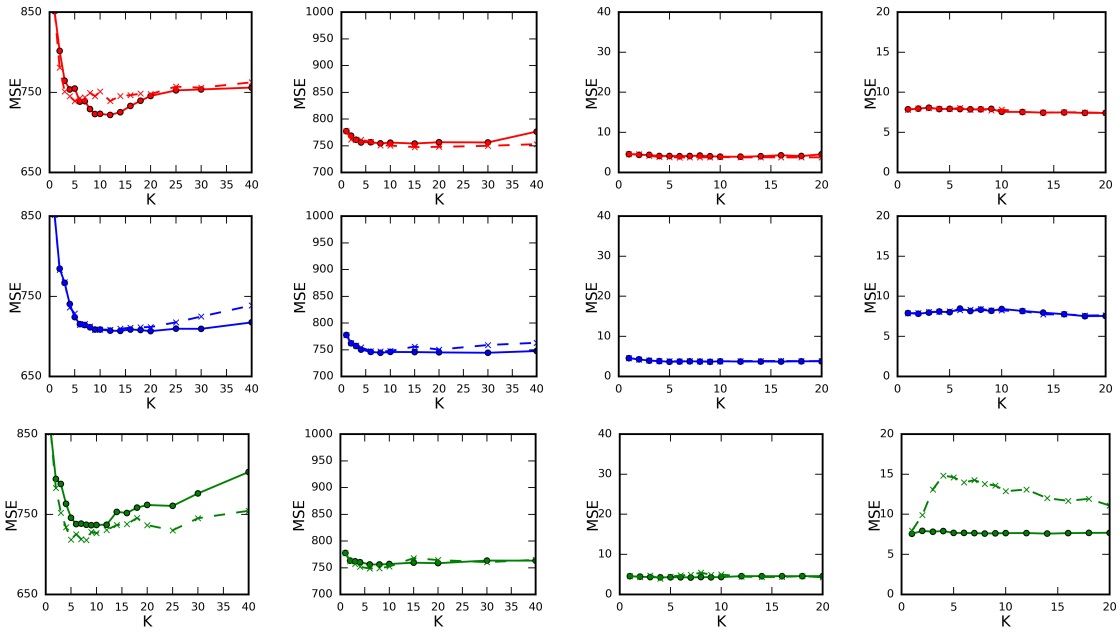
Finally, we consider the effect that the choice of the hyperparameter values λ_U, λ_V, λ_F, λ_S, λ_G have on predictive performance, and in particular how sensitive the choice of these hyperparameters is. Lower values for λ correspond to a weaker prior belief, which could lead to overfitting when the sparsity of the methods is high—we saw before that the advantage of the Bayesian approach is that the prior belief prevents overfitting. High values correspond to a strong belief that all factor values are close to zero, which can also lead to high predictive errors as we cannot fit well to the data.

For three different sparsity levels—20% unobserved entries, 50%, and 80%—we measured the predictive error on the GDSC dataset of Gibbs, ICM, and VB. We used $K = 10$ for NMF, and $K = L = 10$ for NMTF. We repeated the random training-test data splits ten times, and plot the average predictive performances in Figure 3.11.



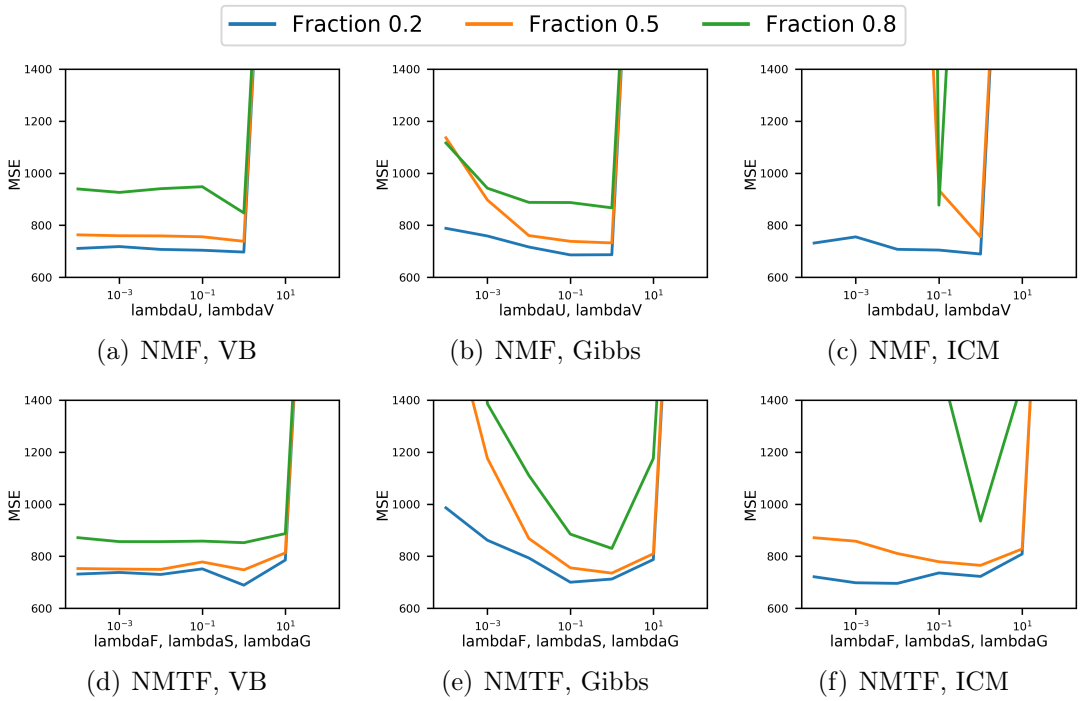
(a) GDSC, NMF (b) CTRP, NMF (c) CCLE IC_{50}, NMF (d) CCLE EC_{50}, NMF

Figure 3.9 Model selection experiment results, giving 10-fold cross-validation performances of the Bayesian NMF models on the drug sensitivity datasets, where we vary the dimensionality K and L ($L = K$). We plot results for VB (top), Gibbs (middle), and ICM (bottom), comparing the models with ARD (o) with those without (x).



(a) GDSC, NMTF (b) CTRP, NMTF (c) CCLE IC_{50}, NMTF (d) CCLE EC_{50}, NMTF

Figure 3.10 Model selection experiment results, giving 10-fold cross-validation performances of the Bayesian NMTF models on the drug sensitivity datasets, where we vary the dimensionality K and L ($L = K$). We plot results for VB (top), Gibbs (middle), and ICM (bottom), comparing the models with ARD (o) with those without (x).



(a) NMF, VB (b) NMF, Gibbs (c) NMF, ICM

(d) NMTF, VB (e) NMTF, Gibbs (f) NMTF, ICM

Figure 3.11 Hyperparameter experiment results for different inference approaches for Bayesian nonnegative matrix factorisation and tri-factorisation. We plot the average predictive error (across ten repeats) on the GDSC drug sensitivity datasets for three different sparsity levels (20% unobserved entries, 50%, and 80%), and different values for the hyperparameters λ.

As expected, the predictive performance becomes very poor as the prior becomes too strong (high values for λ), and it also increases as the prior becomes too weak (low values for λ). The Gibbs NMF, VB NMF, and VB NMTF inference approaches are fairly insensitive to the choice of λ, even when the sparsity is high (80%). In contrast, Gibbs NMTF, ICM NMF, and ICM NMTF are more sensitive to the choice under high sparsity. This again demonstrates that the fully Bayesian methods are more robust to sparsity and overfitting.

Summary: Gibbs and VB are fairly insensitive to the choice of the hyperparameter λ, even under high sparsity levels. A weak prior belief ($\lambda = 0.1$) usually works well. ICM is more prone to overfitting if λ is not chosen carefully.

Table 3.4 Qualitative comparison of inference methods.

Method	Estimate	Requires sampling	Speed of convergence	Robustness
Non-probabilistic	Point	No	High	Low
Iterated conditional modes	Point (MAP)	No	High	Medium
Gibbs sampling	Full posterior	Yes	Low	High
Variational Bayes	Full posterior	No	Medium	Fairly high

3.6 Conclusion

We have studied the trade-offs between different inference approaches for Bayesian nonnegative matrix factorisation and tri-factorisation models. We considered three methods, namely Gibbs sampling, iterated conditional modes, and non-probabilistic inference, and introduced a fourth one based on variational Bayesian inference. We furthermore extended these models with the Bayesian automatic relevance determination prior, to perform automatic model selection. Through experiments on both synthetic data, and real-world drug sensitivity datasets, we explored the trade-offs in convergence, robustness to noise, and robustness to sparsity.

A qualitative summary based on our quantitative findings can be found in Table 3.4. We found that the non-probabilistic methods are not very robust to noise and sparsity. Gibbs sampling is the most robust of the methods, especially for sparse datasets, and gives a full Bayesian posterior estimate. However, it converges slowly, and requires additional samples to estimate the posterior. Iterated conditional modes offers a much faster convergence and run-time speed, but sacrifices some robustness, still requires sampling, no longer returns a full posterior (giving a MAP estimate instead), and is very sensitive to the hyperparameter choice. Our variational Bayesian inference gives good convergence speeds while maintaining more robustness properties.

In many ways, these results confirm our prior expectations. Fully Bayesian inference methods such as variational Bayes maintain more uncertainty information about their current estimates of the variables than maximum a posteriori or maximum likelihood versions, and Gibbs sampling explores more of the posterior space before committing to a high density region. As a result, they are less likely to overfit, especially when the uncertainty increases—such as under high sparsity or noise levels. Due to the widespread usage of non-probabilistic models for matrix factorisation, this study provides important evidence of the importance of using Bayesian models. We will

see in Chapter 4 that MAP solutions in Bayesian models are equivalent to adding regularisation to non-probabilistic models. This chapter demonstates that this vastly improves the robustness of our models. However, the fully Bayesian approaches give an additional performance gain by obtaining a full posterior estimate, something the non-probabilistic models with regularisation cannot achieve. This is most likely due to the better exploration of the posterior space.

Furthermore, we found that there is little to no difference in predictive performance between the matrix factorisation and tri-factorisation versions. The former fitted more closely to the drug sensitivity datasets, but this did not lead to better or worse predictive performances in any of the experiments.

Finally, we have shown that ARD is an effective way of reducing overfitting when using the wrong dimensionality in matrix factorisation models. This can eliminate the need for performing model selection, or nested cross-validation. We also discovered that adding ARD has little impact on performance, or on the robustness of the models to sparsity and noise.

These experiments were conducted for a specific version of Bayesian matrix factorisation and tri-factorisation, but we believe they offer insights into the trade-offs between different inference techniques in other matrix factorisation models, as well as tensor and Tucker decomposition methods. For future work it would be interesting to see how these findings extend to real-valued matrix factorisation models (for example with Gaussian priors), and Poisson-likelihood based models.

Chapter 4

Prior and likelihood choices for Bayesian matrix factorisation

As we saw in the previous chapter, the technique to infer the distribution over the factor matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ can have a great impact on the performance of our Bayesian (and non-Bayesian) matrix factorisation models. In this chapter, we study the effects of the likelihood and prior choice on predictive performance. These choices can be grouped based on the constraints they place on the factor matrices. Firstly, many approaches place no constraints, using real-valued factor matrices (commonly done in the Bayesian literature—Gönen [2012]; Salakhutdinov and Mnih [2008]). Instead, as in the last chapter we could constrain them to be nonnegative (as is popular in the non-probabilistic literature—Lee and Seung [2000]; Tan and Févotte [2013]), limiting its applicability to nonnegative datasets, but making it easier to interpret the factors and potentially also making the method more robust to overfitting. Thirdly, semi-nonnegative variants constrain one factor matrix to be nonnegative, leaving the other real-valued (Ding et al. [2010]; Wang et al. [2008]). Finally, some versions work only on count data.

In the Bayesian setting, the first three groups of methods all generally use a Gaussian likelihood for noise, and place either real-valued or nonnegative priors over the matrices. For the former, Gaussian is a common choice (Gönen [2012]; Salakhutdinov and Mnih [2008]; Virtanen et al. [2011, 2012]), and for the latter options include the exponential distributions (Schmidt et al. [2009]). The fourth group uses a Poisson likelihood to capture count data (Gopalan and Blei [2014]; Gopalan et al. [2015]; Hu et al. [2015]).

All these models are often extended by using complicated hierarchical prior structures over the factor matrices, giving additional desired behaviour (such as automatic model selection). Some papers also introduce general exponential family models that can capture both Gaussian and Poisson likelihood approaches (Hayashi et al. [2009]; Klami et al. [2010]). However, these papers generally do not provide a thorough comparison between the different options.

This chapter offers the first systematic comparison between different Bayesian variants of matrix factorisation. Similar comparisons have been provided in other fields, such as for the regression parameter in Bayesian model averaging (Eicher et al. [2011]; Ley and Steel [2009]), where it was demonstrated that the choice of prior can greatly influence the predictive performance of these models. However, a similar study for Bayesian matrix factorisation is still missing. Strikingly, many papers that introduce new matrix factorisation models do not provide a thorough comparison with competing approaches, or popular non-probabilistic ones such as Lee and Seung [2000]—for example, the seminal paper by Salakhutdinov and Mnih [2008] compares their approach with only one other matrix factorisation method; although Gopalan et al. [2015] compares with three others.

In this chapter we give an overview of some of the different approaches that can be found in the literature, including hierarchical priors, and then study the effects of these different Bayesian prior and likelihood choices. We furthermore discuss the prior choices and their relation to different sparsity types and matrix norms. We aim to make general statements about the behaviour of the four different groups of methods on small real-world datasets (up to a million observed entries), by considering eight datasets across three different applications—four drug effectiveness datasets, two collaborative filtering datasets, and two methylation expression datasets. Our experiments consider each model's convergence and runtime speed, cross-validation performance, sparse and noisy prediction performance, and model selection effectiveness. This study offers novel insights into the differences between the four approaches, and the effects of popular hierarchical priors. The novelty of this chapter is not proposing new models, as we mainly reviewed popular models for the literature. Instead, it is bringing all these models into the same space of experiments.

Most interestingly, where Poisson matrix factorisation models are often claimed to provide faster inference and better predictive performances (Gopalan et al. [2015]), we found that on the smaller datasets that we considered, the opposite was true—with

all Gaussian models consistently outperforming the Poisson ones. We provide several further insights in Section 4.6.

We note that there is a rich literature of Bayesian nonparametric matrix factorisation models, which learn the size of the factor matrices automatically. However, these models often require complex inference approaches to find good solutions, and hence their predictive performance is more determined by the inference method than the precise model choices (such as likelihood and prior). In this paper we therefore focus on parametric matrix factorisation models, to isolate the effects of likelihood and prior choices.

Finally, we acknowledge that the models we study were generally introduced for a specific application domain, and that this makes it hard to make general statements about the behaviour of these methods on different datasets. However, we believe that it is essential to provide a cross-application comparison of the different approaches, as this teaches us valuable lessons for the applications studied, and they are likely to apply to different areas as well. The lack of other studies exploring the trade-offs between likelihood and prior choices for Bayesian matrix factorisation make this an essential study.

In summary, the contributions of this chapter are as follows:

- We review a large number of Bayesian matrix factorisation models from the literature, and group them into four categories.
- We discuss the relation of different Bayesian priors to sparsity types and matrix norms.
- On three different application areas we compare the models' convergence and runtime speeds, and cross-validation, noise, sparsity, and model selection performances. This yields several new insights into the advantages and disadvantages of the four groups of methods.

4.1 Models and inference

There are three types of choices we make that determine the type of matrix factorisation model we use: the likelihood function, the priors we place over the factor matrices $\boldsymbol{U}$ and $\boldsymbol{V}$, and whether we use any further hierarchical priors (such as automatic

Table 4.1 Overview of the Bayesian matrix factorisation models.

Category	Name	Likelihood	Prior $\boldsymbol{U}$	Prior $\boldsymbol{V}$	Hierarchical prior
Real-valued	GGG	$\mathcal{N}(R_{ij}\|\boldsymbol{U_i}\boldsymbol{V_j}, \tau^{-1})$	$\mathcal{N}(\boldsymbol{U_i}\|\boldsymbol{0}, \lambda^{-1}\mathbf{I})$	$\mathcal{N}(\boldsymbol{V_j}\|\boldsymbol{0}, \lambda^{-1}\mathbf{I})$	-
	GGGU	$\mathcal{N}(R_{ij}\|\boldsymbol{U_i}\boldsymbol{V_j}, \tau^{-1})$	$\mathcal{N}(\boldsymbol{U_i}\|\boldsymbol{0}, \lambda^{-1}\mathbf{I})$	$\mathcal{N}(\boldsymbol{V_j}\|\boldsymbol{0}, \lambda^{-1}\mathbf{I})$	-
	GGGA	$\mathcal{N}(R_{ij}\|\boldsymbol{U_i}\boldsymbol{V_j}, \tau^{-1})$	$\mathcal{N}(\boldsymbol{U_i}\|\boldsymbol{0}, \mathrm{diag}(\boldsymbol{\lambda}^{-1}))$	$\mathcal{N}(\boldsymbol{V_j}\|\boldsymbol{0}, \mathrm{diag}(\boldsymbol{\lambda}^{-1}))$	$\lambda_k \sim \mathcal{G}(\alpha_0, \beta_0)$
	GGGW	$\mathcal{N}(R_{ij}\|\boldsymbol{U_i}\boldsymbol{V_j}, \tau^{-1})$	$\mathcal{N}(\boldsymbol{U_i}\|\boldsymbol{\mu_U}, \boldsymbol{\Sigma_U})$	$\mathcal{N}(\boldsymbol{V_j}\|\boldsymbol{\mu_V}, \boldsymbol{\Sigma_V})$	$(\boldsymbol{\mu_U}, \boldsymbol{\Sigma_U})$ and $(\boldsymbol{\mu_V}, \boldsymbol{\Sigma_V})$ $\sim \mathcal{NIW}(\boldsymbol{\mu_0}, \beta_0, \nu_0, \boldsymbol{W_0})$
	GLL	$\mathcal{N}(R_{ij}\|\boldsymbol{U_i}\boldsymbol{V_j}, \tau^{-1})$	$\mathcal{L}(U_{ik}\|0, \eta)$	$\mathcal{L}(V_{jk}\|0, \eta)$	-
	GLLI	$\mathcal{N}(R_{ij}\|\boldsymbol{U_i}\boldsymbol{V_j}, \tau^{-1})$	$\mathcal{L}(U_{ik}\|0, \eta_{ik}^U)$	$\mathcal{L}(V_{jk}\|0, \eta_{jk}^V)$	η_{ik}^U and $\eta_{jk}^V \sim \mathcal{IG}(\mu, \lambda)$
	GVG	$\mathcal{N}(R_{ij}\|\boldsymbol{U_i}\boldsymbol{V_j}, \tau^{-1})$	$p(\boldsymbol{U}) \propto$ $\exp\{-\gamma \det(\boldsymbol{U}^T\boldsymbol{U})\}$	$\mathcal{N}(\boldsymbol{V_j}\|\boldsymbol{0}, \lambda^{-1}\mathbf{I})$	-
Nonnegative	GEE	$\mathcal{N}(R_{ij}\|\boldsymbol{U_i}\boldsymbol{V_j}, \tau^{-1})$	$\mathcal{E}(U_{ik}\|\lambda)$	$\mathcal{E}(V_{jk}\|\lambda)$	-
	GEEA	$\mathcal{N}(R_{ij}\|\boldsymbol{U_i}\boldsymbol{V_j}, \tau^{-1})$	$\mathcal{E}(U_{ik}\|\lambda_k)$	$\mathcal{E}(V_{jk}\|\lambda_k)$	$\lambda_k \sim \mathcal{G}(\alpha_0, \beta_0)$
	GTT	$\mathcal{N}(R_{ij}\|\boldsymbol{U_i}\boldsymbol{V_j}, \tau^{-1})$	$\mathcal{TN}(U_{ik}\|\mu_U, \tau_U)$	$\mathcal{TN}(V_{jk}\|\mu_V, \tau_V)$	-
	GTTN	$\mathcal{N}(R_{ij}\|\boldsymbol{U_i}\boldsymbol{V_j}, \tau^{-1})$	$\mathcal{TN}(U_{ik}\|\mu_U, \tau_U)$	$\mathcal{TN}(V_{jk}\|\mu_V, \tau_V)$	$p(\mu_{ik}^U, \tau_{ik}^U\|\mu_\mu, \tau_\mu, a, b) \propto$ $\frac{1}{\sqrt{\tau_{ik}^U}}\left(1 - \Phi(-\mu_{ik}^U\sqrt{\tau_{ik}^U})\right)$ $\mathcal{N}(\mu_{ik}^U\|\mu_\mu, \tau_\mu^{-1})\mathcal{G}(\tau_{ik}^U\|a, b)$
	GL_1^2	$\mathcal{N}(R_{ij}\|\boldsymbol{U_i}\boldsymbol{V_j}, \tau^{-1})$	$p(\boldsymbol{U}) \propto \exp$ $\{-\frac{\lambda}{2}\sum_i(\sum_k U_{ik})^2\}$ with $U_{ik} \geq 0$	$p(\boldsymbol{V}) \propto \exp$ $\{-\frac{\lambda}{2}\sum_j(\sum_k V_{jk})^2\}$ with $V_{jk} \geq 0$	-
Semi-nonnegative	GEG	$\mathcal{N}(R_{ij}\|\boldsymbol{U_i}\boldsymbol{V_j}, \tau^{-1})$	$\mathcal{E}(U_{ik}\|\lambda)$	$\mathcal{N}(\boldsymbol{V_j}\|\boldsymbol{0}, \lambda^{-1}\mathbf{I})$	-
	GVnG	$\mathcal{N}(R_{ij}\|\boldsymbol{U_i}\boldsymbol{V_j}, \tau^{-1})$	GVG with $U_{ik} \geq 0$	$\mathcal{N}(\boldsymbol{V_j}\|\boldsymbol{0}, \lambda^{-1}\mathbf{I})$	-
Poisson	PGG	$\mathcal{P}(R_{ij}\|\boldsymbol{U_i}\boldsymbol{V_j})$	$\mathcal{G}(U_{ik}\|a, b)$	$\mathcal{G}(V_{jk}\|a, b)$	-
	PGGG	$\mathcal{P}(R_{ij}\|\boldsymbol{U_i}\boldsymbol{V_j})$	$\mathcal{G}(U_{ik}\|a, h_i^U)$	$\mathcal{G}(V_{jk}\|a, h_j^V)$	h_i^U and $h_j^V \sim \mathcal{G}(a', \frac{a'}{b'})$

relevance determination). We have identified four different groups of Bayesian matrix factorisation approaches in the literature based on these choices: Gaussian-likelihood with real-valued priors, nonnegative priors (constraining the matrices $\boldsymbol{U}, \boldsymbol{V}$ to be nonnegative), semi-nonnegative models (constraining one of the two factor matrices to be nonnegative), and finally Poisson-likelihood approaches. Models within each group use different priors and hierarchical priors, and many choices can be found in the literature. We consider a total of sixteen models, as summarised in Table 4.1. We have focused on fully conjugate models to ensure inference for each model is guaranteed to work well, so that all performance differences in Section 4.5 come entirely from the choice of likelihood and priors.

In the following subsections we will define the model choices for each of the sixteen Bayesian matrix factorisation models. We also give the Gibbs sampling posterior distributions for $\boldsymbol{U}$, which can be derived using Bayes' theorem (see Section 3.2.2 for an example). Expressions for $\boldsymbol{V}$ are symmetrical, and hence omitted.

4.1.1 Real-valued matrix factorisation

The first group of methods use a Gaussian likelihood for noise, and place real-valued prior distributions over $\boldsymbol{U}$ and $\boldsymbol{V}$, typically Gaussian as well. We assume each value in $\boldsymbol{R}$ comes from the product of $\boldsymbol{U}$ and $\boldsymbol{V}$, with Gaussian noise added with precision τ (as in the previous chapter),

$$R_{ij} \sim \mathcal{N}(R_{ij}|\boldsymbol{U_i}\boldsymbol{V_j}, \tau^{-1}) \qquad \tau \sim \mathcal{G}(\tau|\alpha_\tau, \beta_\tau)$$

In the Gibbs sampling algorithm, the posterior for the noise parameter is

$$\tau \sim \mathcal{G}(\tau|\alpha^*_\tau, \beta^*_\tau) \qquad \alpha^*_\tau = \alpha_\tau + \frac{|\Omega|}{2} \qquad \beta^*_\tau = \beta_\tau + \frac{1}{2}\sum_{(i,j)\in\Omega}(R_{ij} - \boldsymbol{U_i}\boldsymbol{V_j})^2.$$

All Gaussian model (GGG)

The most common approach is to use independent zero-mean Gaussian priors for $\boldsymbol{U}, \boldsymbol{V}$ (Gönen [2012]; Salakhutdinov and Mnih [2008]; Virtanen et al. [2011, 2012]) with hyperparameter λ,

$$\boldsymbol{U_i} \sim \mathcal{N}(\boldsymbol{U_i}|\boldsymbol{0}, \lambda^{-1}\mathbf{I}) \qquad \boldsymbol{V_j} \sim \mathcal{N}(\boldsymbol{V_j}|\boldsymbol{0}, \lambda^{-1}\mathbf{I})$$

where $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = |\boldsymbol{\Sigma}|^{-\frac{1}{2}}(2\pi)^{-\frac{K}{2}}\exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right\}$ is the density of a K-dimensional multivariate Gaussian distribution, and $\mathbf{I}$ is the identity matrix. The conditional posterior distributions we obtain in the Gibbs sampling algorithm are also multivariate Gaussians. The parameter values are given below, with $\Omega^1_i = \{j \mid (i,j) \in \Omega\}$.

$$\boldsymbol{U}_i \sim \mathcal{N}(\boldsymbol{U_i}|\boldsymbol{\mu}^U_{\boldsymbol{i}}, \boldsymbol{\Sigma}^U_{\boldsymbol{i}}) \qquad \boldsymbol{\mu}^U_{\boldsymbol{i}} = \boldsymbol{\Sigma}^U_{\boldsymbol{i}} \cdot \left[\tau\sum_{j\in\Omega^1_i} R_{ij}\boldsymbol{V_j}\right] \qquad \boldsymbol{\Sigma}^U_{\boldsymbol{i}} = \left[\lambda\mathbf{I} + \tau\sum_{j\in\Omega^1_i}(\boldsymbol{V_j}\otimes\boldsymbol{V_j})\right]^{-1}.$$

All Gaussian model with univariate posterior (GGGU)

It is also possible to have a univariate posterior for the Gibbs sampler. We expect this to have little effect, but it will be interesting to see whether there is any difference in

performance.

$$U_{ik} \sim \mathcal{N}(U_{ik}|\mu_{ik}^U, (\tau_{ik}^U)^{-1}) \qquad \mu_{ik}^U = \frac{1}{\tau_{ik}^U}\left[\tau \sum_{j\in\Omega_i^1}\left(R_{ij} - \sum_{k'\neq k} U_{ik'}V_{jk'}\right)V_{jk}\right]$$
$$\tau_{ik}^U = \lambda + \tau \sum_{j\in\Omega_i^1} V_{jk}^2.$$

All Gaussian model with ARD hierarchical prior (GGGA)

We reviewed in Section 2.3.7 how to extend the all Gaussian model with automatic relevance determination (ARD), by replacing the λ hyperparameter by a factor-specific variable λ_k, which has a further Gamma prior (Virtanen et al. [2011, 2012]),

$$\boldsymbol{U_i} \sim \mathcal{N}(\boldsymbol{U_i}|\mathbf{0}, \text{diag}(\boldsymbol{\lambda}^{-1})) \qquad \boldsymbol{V_j} \sim \mathcal{N}(\boldsymbol{V_j}|\mathbf{0}, \text{diag}(\boldsymbol{\lambda}^{-1})) \qquad \lambda_k \sim \mathcal{G}(\lambda_k|\alpha_0, \beta_0).$$

The Gibbs sampling posteriors for $\boldsymbol{U_i}$ and $\boldsymbol{V_j}$ are largely unchanged compared to the GGG model, replacing $\lambda\mathbf{I}$ in the expressions for $\boldsymbol{\Sigma_i^U}, \boldsymbol{\Sigma_j^V}$ by $\text{diag}(\lambda_1, .., \lambda_K)$. The posterior for λ_k is

$$\lambda_k \sim \mathcal{G}(\lambda_k|\alpha_0^*, \beta_0^*) \qquad \alpha_0^* = \alpha_0 + \frac{I}{2} + \frac{J}{2} \qquad \beta_0^* = \beta_0 + \frac{1}{2}\sum_{i=1}^{I} U_{ik}^2 + \frac{1}{2}\sum_{j=1}^{J} V_{jk}^2.$$

All Gaussian model with Wishart hierarchical prior (GGGW)

Another hierarchical prior was introduced in the seminal paper of Salakhutdinov and Mnih [2008]. Instead of assuming independence of each entry in $\boldsymbol{U}, \boldsymbol{V}$, we now assume each row of $\boldsymbol{U}$ comes from a multivariate Gaussian with row mean $\boldsymbol{\mu_U}$ and covariance $\boldsymbol{\Sigma_U}$, and similarly for $\boldsymbol{V}$. We place a further Normal-Inverse Wishart prior over these parameters,

$$\boldsymbol{U_i} \sim \mathcal{N}(\boldsymbol{U_i}|\boldsymbol{\mu_U}, \boldsymbol{\Sigma_U}) \qquad \boldsymbol{\mu_U}, \boldsymbol{\Sigma_U} \sim \mathcal{NIW}(\boldsymbol{\mu_U}, \boldsymbol{\Sigma_U}|\boldsymbol{\mu_0}, \beta_0, \nu_0, \boldsymbol{W_0})$$
$$\boldsymbol{V_j} \sim \mathcal{N}(\boldsymbol{V_j}|\boldsymbol{\mu_V}, \boldsymbol{\Sigma_V}) \qquad \boldsymbol{\mu_V}, \boldsymbol{\Sigma_V} \sim \mathcal{NIW}(\boldsymbol{\mu_V}, \boldsymbol{\Sigma_V}|\boldsymbol{\mu_0}, \beta_0, \nu_0, \boldsymbol{W_0})$$

where $\mathcal{NIW}(\boldsymbol{\mu_U}, \boldsymbol{\Sigma_U}|\boldsymbol{\mu_0}, \beta_0, \nu_0, \boldsymbol{W_0}) = \mathcal{N}(\boldsymbol{\mu}|\boldsymbol{\mu_0}, \frac{1}{\beta_0}\mathbf{I})\mathcal{W}^{-1}(\boldsymbol{\Sigma}|\nu_0, \boldsymbol{W_0})$ is the density of a normal-inverse Wishart distribution, and $\mathcal{W}^{-1}(\boldsymbol{\Sigma}|\nu_0, \boldsymbol{W_0})$ is the inverse Wishart distribution.

For the Gibbs sampling algorithm we obtain the posteriors $\boldsymbol{U_i} \sim \mathcal{N}(\boldsymbol{U_i}|\boldsymbol{\mu_i^U}, \boldsymbol{\Sigma_i^U})$ and $\boldsymbol{\mu_U}, \boldsymbol{\Sigma_U} \sim \mathcal{NIW}(\boldsymbol{\mu_U}, \boldsymbol{\Sigma_U}|\boldsymbol{\mu_0^*}, \beta_0^*, \nu_0^*, \boldsymbol{W_0^*})$, with

$$\boldsymbol{\mu_i^U} = \boldsymbol{\Sigma_i^U} \cdot \left[\boldsymbol{\Sigma_U^{-1}}\boldsymbol{\mu_U} + \tau \sum_{j \in \Omega_i^1} R_{ij}\boldsymbol{V_j}\right] \qquad \boldsymbol{\Sigma_i^U} = \left[\boldsymbol{\Sigma_U^{-1}} + \tau \sum_{j \in \Omega_i^1} (\boldsymbol{V_j} \otimes \boldsymbol{V_j})\right]^{-1}$$

$$\beta_0^* = \beta_0 + I \qquad \nu_0^* = \nu_0 + I \qquad \boldsymbol{\mu_0^*} = \frac{\beta_0 \boldsymbol{\mu_0} + I\bar{\boldsymbol{U}}}{\beta_0 + I} \qquad \bar{\boldsymbol{U}} = \frac{1}{I}\sum_{i=1}^{I} \boldsymbol{U_i}$$

$$\boldsymbol{W_0^*} = \boldsymbol{W_0} + I\bar{\boldsymbol{S}} + \frac{\beta_0 I}{\beta_0 + I}(\boldsymbol{\mu_0} - \bar{\boldsymbol{U}}) \otimes (\boldsymbol{\mu_0} - \bar{\boldsymbol{U}}) \qquad \bar{\boldsymbol{S}} = \frac{1}{I}\sum_{i=1}^{I}(\boldsymbol{U_i} \otimes \boldsymbol{U_i}).$$

Gaussian likelihood with Laplace priors (GLL)

An alternative to the Gaussian prior is to use the Laplace distribution $\mathcal{L}(x|\mu, \rho) = \frac{1}{2\rho}\exp\left\{-\frac{|x-\mu|}{\rho}\right\}$, which has a much more pointy distribution than Gaussian around $x = \mu$ (Jing et al. [2015]). This leads to more sparse solutions, as more factors are set to low values. The priors are

$$U_{ik} \sim \mathcal{L}(U_{ik}|0, \eta) \qquad V_{jk} \sim \mathcal{L}(V_{jk}|0, \eta)$$

To simplify inference we introduce a new variable λ_{ik}^U for each U_{ik}, with prior $\lambda_{ik}^U \sim \mathcal{E}(\lambda_{ik}^U|\eta)$. The idea behind this is that we can rewrite a Laplace distribution as

$$\mathcal{L}(x|\mu, \rho) = \int_{\epsilon=0}^{\infty} \mathcal{N}(x|\mu, \epsilon)\mathcal{E}(\epsilon|\frac{\rho}{2})d\epsilon$$

This leads to the following Gibbs sampling posteriors:

$$\boldsymbol{U}_i \sim \mathcal{N}(\boldsymbol{U_i}|\boldsymbol{\mu_i^U}, \boldsymbol{\Sigma_i^U}) \qquad \boldsymbol{\mu_i^U} = \boldsymbol{\Sigma_i^U} \cdot \left[\tau \sum_{j \in \Omega_i^1} R_{ij}\boldsymbol{V_j}\right]$$

$$\boldsymbol{\Sigma_i^U} = \mathrm{diag}((\boldsymbol{\lambda_i^U})^{-1}) + \left[\tau \sum_{j \in \Omega_i^1} (\boldsymbol{V_j} \otimes \boldsymbol{V_j})\right]^{-1}$$

$$\frac{1}{\lambda_{ik}^U} \sim \mathcal{IG}(x|\mu_{ik}^U, \lambda_{ik}^U) \qquad \mu_{ik}^U = \frac{\sqrt{\eta}}{|U_{ik}|} \qquad \lambda_{ik}^U = \eta.$$

Gaussian and Laplace model with hierarchical inverse Gaussian priors (GLLI)

We can place a further hierarchical prior over the η parameters, which Jing et al. [2015] claim helps with variable selection. We replace each η with η^U_{ik}, η^V_{jk}, and use priors

$$\eta^U_{ik} \sim \mathcal{IG}(\mu, \lambda) \qquad \eta^V_{jk} \sim \mathcal{IG}(\mu, \lambda).$$

That paper originally placed a Generalised Inverse Gaussian $\mathcal{GIG}(\gamma, a, b)$ prior over the η parameters, but then used $\gamma = -\frac{1}{2}$, which reduces the prior to the Inverse Gaussian above with $\mu = \sqrt{b/a}, \lambda = b$ (or $a = 1/\mu, b = \lambda$).

The posteriors for $\boldsymbol{U}, \boldsymbol{V}$ are identical, and for λ^U_{ik} we only replace η with η^U_{ik}. We obtain another Inverse Gaussian posterior for the η^U_{ik} parameters,

$$\eta^U_{ik} \sim \mathcal{IG}(\eta^U_{ik} | \mu^\eta_{ik}, \lambda^\eta_{ik}) \qquad \mu^U_{ik} = \sqrt{\frac{\lambda^U_{ik} + 1/\mu}{\lambda}} \qquad \lambda^U_{ik} = 1/\mu.$$

Gaussian likelihood with volume prior (GVG)

The final real-valued model we consider was presented by Arngren et al. [2011]. The prior over $\boldsymbol{V}$ is Gaussian, as in the GGG model, but we now use a so-called volume prior (VP) for the $\boldsymbol{U}$ matrix, with density $p(\boldsymbol{U}) \propto \exp\{-\gamma \det(\boldsymbol{U}^T\boldsymbol{U})\}$. The γ hyperparameter determines the strength of the volume penalty (higher means stronger prior). This model leads to the posterior

$$U_{ik} \sim \mathcal{N}(U_{ik} | \mu^U_{ik}, (\tau^U_{ik})^{-1}) \qquad \mu^U_{ik} = \frac{1}{\tau^U_{ik}} \left[\gamma \boldsymbol{U}_{i\tilde{\boldsymbol{k}}} \boldsymbol{A}_{\tilde{\boldsymbol{k}}\tilde{\boldsymbol{k}}} (\boldsymbol{U}^{\boldsymbol{T}}_{\tilde{\boldsymbol{i}}\tilde{\boldsymbol{k}}} \boldsymbol{U}_{\tilde{\boldsymbol{i}}\boldsymbol{k}}) + \tau \sum_{j \in \Omega^1_i} \left(R_{ij} - \sum_{k' \neq k} U_{ik'} V_{jk'} \right) V_{jk} \right]$$

$$\tau^U_{ik} = \tau \sum_{j \in \Omega^1_i} V^2_{jk} + \gamma (D_{\tilde{k}\tilde{k}} - \boldsymbol{U}_{i\tilde{\boldsymbol{k}}} \boldsymbol{A}_{\tilde{\boldsymbol{k}}\tilde{\boldsymbol{k}}} \boldsymbol{U}^{\boldsymbol{T}}_{i\tilde{\boldsymbol{k}}}).$$

In the above, vector $\boldsymbol{U}_{i\tilde{\boldsymbol{k}}}$ is the ith row of $\boldsymbol{U}$ excluding column k; vector $\boldsymbol{U}_{\tilde{\boldsymbol{i}}\boldsymbol{k}}$ is the kth column of $\boldsymbol{U}$ excluding row i; matrix $\boldsymbol{U}_{\tilde{\boldsymbol{i}}\tilde{\boldsymbol{k}}}$ is $\boldsymbol{U}$ excluding row i and column k; matrix $\boldsymbol{U}_{\cdot\tilde{\boldsymbol{k}}}$ is $\boldsymbol{U}$ excluding column k; $D_{\tilde{k}\tilde{k}} = \det\{\boldsymbol{U}^{\boldsymbol{T}}_{\cdot\tilde{\boldsymbol{k}}} \boldsymbol{U}_{\cdot\tilde{\boldsymbol{k}}}\}$; and matrix $\boldsymbol{A}_{\tilde{\boldsymbol{k}}\tilde{\boldsymbol{k}}} = \det\{\boldsymbol{U}^{\boldsymbol{T}}_{\cdot\tilde{\boldsymbol{k}}} \boldsymbol{U}_{\cdot\tilde{\boldsymbol{k}}}\}$ is the matrix adjugate.

4.1.2 Nonnegative matrix factorisation

Nonnegative matrix factorisation models use the same Gaussian noise model as the real-valued ones, but placing nonnegative prior distributions over entries in $\boldsymbol{U}$ and $\boldsymbol{V}$. As a result these models can only deal with nonnegative datasets, but by making the factor matrices constrained it is hoped that this will lead to less overfitting to noise and sparsity. Furthermore, the factor values may be easier to interpret, for example as cluster indicators—negative factor values are much harder to understand in that context.

Gaussian likelihood with exponential priors (GEE)

This is the model used for the previous chapter, introduced by Schmidt et al. [2009]. We place independent Exponential priors over the entries in $\boldsymbol{U}, \boldsymbol{V}$,

$$U_{ik} \sim \mathcal{E}(U_{ik}|\lambda) \qquad V_{jk} \sim \mathcal{E}(V_{jk}|\lambda),$$

with posteriors

$$U_{ik} \sim \mathcal{TN}(U_{ik}|\mu^U_{ik}, \tau^U_{ik}) \qquad \mu^U_{ik} = \frac{1}{\tau^U_{ik}}\left[-\lambda + \tau \sum_{j \in \Omega^1_i} \left(R_{ij} - \sum_{k' \neq k} U_{ik'} V_{jk'}\right) V_{jk}\right]$$
$$\tau^U_{ik} = \tau \sum_{j \in \Omega^1_i} V^2_{jk}.$$

Gaussian likelihood with exponential prior and ARD (GEEA)

As before, we can extend this with the ARD priors,

$$U_{ik} \sim \mathcal{E}(U_{ik}|\lambda_k) \qquad V_{jk} \sim \mathcal{E}(V_{jk}|\lambda_k) \qquad \lambda_k \sim \mathcal{G}(\lambda_k|\alpha_0, \beta_0).$$

The posteriors for $\boldsymbol{U}$ and $\boldsymbol{V}$ are the same as in the GEE model, but replacing λ by λ_k. The posteriors for λ_k become

$$\lambda_k \sim \mathcal{G}(\lambda_k|\alpha^*_0, \beta^*_0) \qquad \alpha^*_0 = \alpha_0 + I + J \qquad \beta^*_0 = \beta_0 + \sum_{i=1}^{I} U_{ik} + \sum_{j=1}^{J} V_{jk}.$$

Gaussian likelihood with truncated normal priors (GTT)

We can also use the truncated normal distribution directly as the priors for $\boldsymbol{U}$ and $\boldsymbol{V}$ (Schmidt and Mohamed [2009]),

$$U_{ik} \sim \mathcal{TN}(U_{ik}|\mu_U, \tau_U) \qquad V_{jk} \sim \mathcal{TN}(V_{jk}|\mu_V, \tau_V)$$

This again gives a truncated normal posterior, but with slightly different values.

$$U_{ik} \sim \mathcal{TN}(U_{ik}|\mu_{ik}^U, (\tau_{ik}^U)^{-1}) \qquad \mu_{ik}^U = \frac{1}{\tau_{ik}^U}\left[\mu_U\tau_U + \tau\sum_{j\in\Omega_i^1}\left(R_{ij} - \sum_{k'\neq k} U_{ik'}V_{jk'}\right)V_{jk}\right]$$
$$\tau_{ik}^U = \tau_U + \tau\sum_{j\in\Omega_i^1} V_{jk}^2.$$

An alternative to the truncated normal distribution is the so-called half normal. If random variable y has density $\mathcal{N}(y|0,\sigma^2)$, and random variable $x = |y|$, then x follows a half normal distribution with density $\mathcal{HN}(x|\sigma) = \frac{\sqrt{2}}{\sigma\sqrt{\pi}}\exp\{-\frac{x^2}{2\sigma^2}\}u(x)$. Note however that when $\mu = 0$ in the truncated normal distribution, then these distributions are equivalent, with $\tau = \frac{1}{\sigma^2}$. Therefore, the GTT model is more general and includes the half normal prior.

Gaussian likelihood with truncated normal and hierarchical priors (GTTN)

We can place a further prior over the parameters of the truncated normal distributions (Schmidt and Mohamed [2009]),

$$U_{ik} \sim \mathcal{TN}(U_{ik}|\mu_{ik}^U, \tau_{ik}^U) \qquad V_{jk} \sim \mathcal{TN}(V_{jk}|\mu_{jk}^V, \tau_{jk}^V)$$
$$p(\mu_{ik}^U, \tau_{ik}^U|\mu_\mu, \tau\mu, a, b) \propto \frac{1}{\sqrt{\tau_{ik}^U}}\left(1 - \Phi(-\mu_{ik}^U\sqrt{\tau_{ik}^U})\right)\mathcal{N}(\mu_{ik}^U|\mu_\mu, \tau_\mu^{-1}).\mathcal{G}(\tau_{ik}^U|a,b)$$

The density for μ_{jk}^V, τ_{jk}^V is identical. Note that the distribution $p(\mu_{ik}^U, \tau_{ik}^U|\mu_\mu, \tau\mu, a, b)$ is not the same as the product of a normal and gamma distribution. We cannot easily sample from it due to its nonstandard form, but it can be used as a hierarchical prior. The posteriors for U_{ik} remain the same as in the GTT model (replacing μ_U and τ_U by μ_{ik}^U and τ_{ik}^U), and for μ_{ik}^U and τ_{ik}^U we obtain posteriors

$$\mu_{ik}^U \sim \mathcal{N}(\mu_{ik}^U|m_\mu, t_\mu^{-1}) \qquad m_\mu = \frac{1}{t_\mu}\left[\tau_{ik}^U U_{ik} + \mu_\mu\tau_\mu\right] \qquad t_\mu = \tau_{ik}^U + \tau_\mu$$

$$\tau_{ik}^U \sim \mathcal{G}(\tau_{ik}^U | a^*, b^*) \qquad a^* = a + \frac{1}{2} \qquad b^* = b + \frac{(U_{ik} - \mu_{ik}^U)^2}{2}.$$

Gaussian likelihood with L_1^2 norm priors (GL_1^2)

Finally, we can use a prior inspired by the L_1^2 norm (discussed further in Section 4.3) for both $\boldsymbol{U}$ and $\boldsymbol{V}$, giving prior densities

$$p(\boldsymbol{U}) \propto \begin{cases} \exp\left\{-\frac{\lambda}{2}\sum_i \left(\sum_k U_{ik}\right)^2\right\} & \text{if } U_{ik} \geq 0 \text{ for all } i,k \\ 0 & \text{if any } U_{ik} < 0 \end{cases}$$

$$p(\boldsymbol{V}) \propto \begin{cases} \exp\left\{-\frac{\lambda}{2}\sum_j \left(\sum_k V_{jk}\right)^2\right\} & \text{if } V_{jk} \geq 0 \text{ for all } j,k \\ 0 & \text{if any } V_{jk} < 0 \end{cases}$$

The nonnegativity constraint is used to address the fact that the L_1^2 norm uses the absolute value of entries in $\boldsymbol{U}, \boldsymbol{V}$, which makes inference impossible unless we constrain them to be nonnegative (in which case the values are automatically absolute).

The posteriors are similar to the GEE and GTT models, but now adding a term that depends on the other entries in the ith (or jth) row of $\boldsymbol{U}$,

$$U_{ik} \sim \mathcal{TN}(U_{ik} | \mu_{ik}^U, \tau_{ik}^U) \qquad \mu_{ik}^U = \frac{1}{\tau_{ik}^U}\left[-\lambda \sum_{k' \neq k} U_{ik'} + \tau \sum_{j \in \Omega_i^1} \left(R_{ij} - \sum_{k' \neq k} U_{ik'} V_{jk'}\right) V_{jk}\right]$$
$$\tau_{ik}^U = \lambda + \tau \sum_{j \in \Omega_i^1} V_{jk}^2.$$

4.1.3 Semi-nonnegative matrix factorisation

In the nonnegative matrix factorisation models we place nonnegative priors over both $\boldsymbol{U}$ and $\boldsymbol{V}$. Instead, we could constrain only one to be nonnegative, as was done in Wang et al. [2008] and Ding et al. [2010]. In the Bayesian setting this is done by placing a real-valued prior over one matrix, and a nonnegative prior over the other. The major advantage is that we can handle real-valued datasets, while still enforcing some nonnegativity. However, we will see in Section 4.5 that their performances are identical to the real-valued approaches.

Gaussian likelihood with exponential and Gaussian priors (GEG)

For this model we use an exponential prior for entries in $\boldsymbol{U}$, and a Gaussian for $\boldsymbol{V}$. The posteriors are given in the GEE and GGG model sections.

Gaussian likelihood with nonnegative volume prior (GVnG)

The volume prior discussed earlier was originally formulated to be nonnegative. In particular, the probability distribution over $\boldsymbol{U}$ was

$$p(\boldsymbol{U}) \propto \begin{cases} \exp\{-\gamma \det(\boldsymbol{U}^T\boldsymbol{U})\} & \text{if } U_{ik} \geq 0 \text{ for all } i,k \\ 0 & \text{if any } U_{ik} < 0 \end{cases}$$

The posterior parameters are the same as for the GVG model, but drawing new values from a truncated normal, rather than normal. For $\boldsymbol{V}$ we again use a Gaussian.

4.1.4 Poisson-likelihood matrix factorisation

The final category of matrix factorisation models do not use a Gaussian likelihood, instead opting for a Poisson one. This only works for nonnegative count data, with $\boldsymbol{R} \in \mathbb{N}^{I\times J}$, but has been studied extensively in the literature due to the popularity and prevalence of datasets like the Netflix Prize. We again assume each value in $\boldsymbol{R}$ comes from the product of $\boldsymbol{U}$ and $\boldsymbol{V}$, $R_{ij} \sim \mathcal{P}(R_{ij}|\boldsymbol{U_i}\boldsymbol{V_j})$, where $\mathcal{P}(x|\lambda) = \frac{\lambda^x \exp\{-\lambda\}}{x!}$ is the density of a Poisson distribution. We will consider two Poisson matrix factorisation models.

Poisson likelihood with Gamma priors (PGG)

The classical Poisson matrix factorisation model (Gopalan and Blei [2014]; Gopalan et al. [2015]; Hu et al. [2015]) uses independent Gamma priors over the entries in $\boldsymbol{U}$ and $\boldsymbol{V}$. To make inference simpler, we also introduce random variables Z_{ijk} such that $R_{ij} = \sum_{k=1}^{K} Z_{ijk}$, each effectively accounting for the contribution of factor k to R_{ij}. We use the following distributions and priors:

$$Z_{ijk} \sim \mathcal{P}(Z_{ijk}|U_{ik}V_{jk}) \qquad U_{ik} \sim \mathcal{G}(U_{ik}|a,b) \qquad V_{jk} \sim \mathcal{G}(V_{jk}|a,b).$$

Note that we can do this because the sum of Poisson distributed random variables (like Z_{ijk}) is again Poisson distributed, with rate λ equal to the sum of rates of the Z_{ijk}, giving us the original Poisson likelihood for R_{ij}. The above is also equivalent

to saying that $\boldsymbol{Z_{ij}} \sim \text{Mult}(\boldsymbol{Z_{ij}}|n, \boldsymbol{p})$ with $n = R_{ij}$ and $\boldsymbol{p} = (\frac{U_{i1}V_{j1}}{\boldsymbol{U_i V_j}}, .., \frac{U_{iK}V_{jK}}{\boldsymbol{U_i V_j}})$, where $\boldsymbol{Z_{ij}}$ is a vector containing $Z_{ij1}, .., Z_{ijK}$, and $\text{Mult}(\boldsymbol{x}|n, \boldsymbol{p}) = \frac{n!}{x_1!..x_K!} p_1^{x_1} p_2^{x_2} .. p_K^{x_K}$ is a K-dimensional multinomial distribution. Using the above trick, the posteriors are

$$\begin{aligned} &\boldsymbol{Z_{ij}} \sim \text{Mult}(\boldsymbol{Z_{ij}}|n, \boldsymbol{p}) \qquad n = R_{ij} \qquad \boldsymbol{p} = (\frac{U_{i1}V_{j1}}{\boldsymbol{U_i V_j}}, .., \frac{U_{iK}V_{jK}}{\boldsymbol{U_i V_j}}) \\ &U_{ik} \sim \mathcal{P}(U_{ik}|a^*_{ik}, b^*_{ik}) \qquad a^*_{ik} = a + \sum_{j \in \Omega_i^1} Z_{ijk} \qquad b^*_{ik} = b + \sum_{j \in \Omega_i^1} V_{jk}. \end{aligned}$$

Poisson likelihood with Gamma and hierarchical Gamma priors (PGGG)

Gopalan et al. [2015] introduced a Poisson matrix factorisation model with hierarchical priors. They presented a variational Bayesian algorithm for inference—we derived a Gibbs sampling version for this chapter. The priors are

$$U_{ik} \sim \mathcal{G}(U_{ik}|a, h_i^U) \qquad V_{jk} \sim \mathcal{G}(V_{jk}|a, h_j^V) \qquad h_i^U \sim \mathcal{G}(a', \frac{a'}{b'}) \qquad h_j^V \sim \mathcal{G}(a', \frac{a'}{b'})$$

The posteriors for U_{ik} are identical, but replacing b with h_i^U in the expression for b^*_{ik}. For the hierarchical part we obtain the following posteriors:

$$h_i^U \sim \mathcal{G}(h_i^U|a^*_i, b^*_i) \qquad a^*_i = a' + Ka \qquad b^*_i = \frac{a'}{b'} + \sum_{k=1}^{K} U_{ik}.$$

4.2 Implementation details

4.2.1 Software implementation

We provide an open-source Python implementation of all models at https://github.com/ThomasBrouwer/HMF, as well as all datasets, preprocessing scripts, and Python code for the experiments in Section 4.5.

4.2.2 Computational complexity

The different matrix factorisation models have different time complexities for computing the parameter values and sample new values for $\boldsymbol{U}$, $\boldsymbol{V}$, and any other random variables. The space complexity for all models is $\mathcal{O}(IK + JK)$ per iteration, with an additional $K|\Omega|$ term for the Poisson models (for the Z_{ijk}).

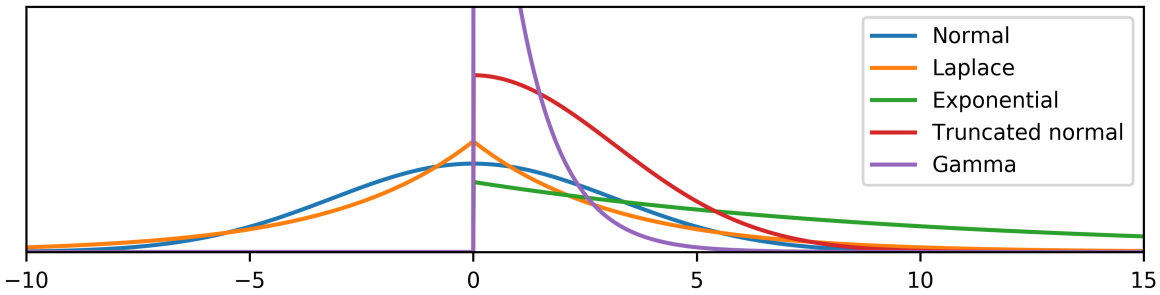


Figure 4.1 Plots of the prior distributions with hyperparameters from Section 4.2.3.

The time complexities per iteration for the multivariate Gaussian posterior models (GGG, GGGA, GGGW, GLL, GLLI) is $\mathcal{O}((I+J)K^3 + IJK^2)$. However, these row draws and parameter value computations can all be done in parallel. The univariate posterior models (GGGU, GEE, GEEA, GTT, GTTN, GL_1^2) have complexity $\mathcal{O}(IJK^2)$, but the parameters can be computed efficiently per column. The volume prior models (GVG, GVnG) have the highest complexity, with $\mathcal{O}(I^2JK^2)$. Finally, the Poisson models are $\mathcal{O}(IJK)$, but this hides a big constant that effectively makes it the slowest model for low values of K.

4.2.3 Hyperparameters

The hyperparameter values we choose for each model can influence their performance, especially when the data is sparse. The hierarchical models try to automatically choose the correct values, by placing a prior over the original hyperparameters. This introduces new hyperparameters, but the models are generally less sensitive to these. However, as we saw in Section 3.5.6, the models without hierarchical priors are usually not very sensitive to this choice, as long as we use fairly weak priors. In particular, we used $\lambda = 0.1$ (GGG, GGGU, GEE, GL_1^2, GEG), $\mu_U = 0, \tau_U = 0.1$ (GTT), $\eta = \sqrt{10}$ (GLL), and $a = 1, b = 1$ (PGG). The distributions with these hyperparameter values are plotted in Figure 4.1.

For the other models we used: $\alpha_\tau = \beta_\tau = 1$ (Gaussian likelihood); $\alpha_0 = \beta_0 = 1$ (GGGA, GEEA); $\boldsymbol{\mu_0} = \mathbf{0}, \beta_0 = 1, \nu_0 = K, \boldsymbol{W_0} = \mathbf{I}$ (GGGW); $\mu = \lambda = K$ (GLLI), $\mu_\mu = 0, \tau_\mu = 0.1, a = b = 1$ (GTTN), $a = a' = b' = 1$ (PGGG).

We did find that the volume prior models (GVG, GVnG) were very sensitive to the hyperparameter choice γ. The following values were chosen by trying a range on each

dataset (see Section 4.4) and choosing the best one: $\gamma = 10^{\{-30,-20,-10,-10,0,0,0,0\}}$ for {GDSC,CTRP,CCLE IC_{50},EC_{50},MovieLens 100K,1M,GM,PM}.

4.3 Priors and norms

The prior distributions in Bayesian models act as a regulariser that prevents us from overfitting to the data and obtaining poor predictive performance. We can write out the expression of the log posterior of the parameters, which for a Gaussian likelihood and no hierarchical priors becomes

$$\begin{aligned}\log p(\boldsymbol{\theta}|D) &= \log p(D|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) + C_1 \\ &= \sum_{(i,j)\in\Omega} \log p(R_{ij}|\boldsymbol{U_i}\boldsymbol{V_j}, \tau^{-1}) + \log p(\boldsymbol{U}, \boldsymbol{V}) + C_2 \\ &= -\frac{\tau}{2} \sum_{(i,j)\in\Omega} (R_{ij} - \boldsymbol{U_i}\boldsymbol{V_j})^2 + \log p(\boldsymbol{U}, \boldsymbol{V}) + C_3\end{aligned}$$

for some constants C_i. Note that this last expression is proportional to the negative Frobenius norm (squared error) of the training fit, plus a regularisation term over the matrices $\boldsymbol{U}, \boldsymbol{V}$. This training error is frequently used in the nonprobabilistic matrix factorisation literature (Lee and Seung [2000]; Pauca et al. [2004, 2006]), where different regularisation terms are used. These are often based on row-wise **matrix norms**, such as

$$\mathrm{L}_1 = \sum_{i=1}^{I}\sum_{k=1}^{K} U_{ik} \qquad \mathrm{L}_2 = \sum_{i=1}^{I}\sqrt{\sum_{k=1}^{K} U_{ik}^2} \qquad \mathrm{L}_1^2 = \sum_{i=1}^{I}(\sum_{k=1}^{K} U_{ik})^2 \qquad \mathrm{L}_2^2 = \sum_{i=1}^{I}\sum_{k=1}^{K} U_{ik}^2.$$

This offers some interesting insights: the L_2^2 norm is equivalent to an independent Gaussian prior (GGG), due to the square in the exponential of the Gaussian prior; the L_1 norm is equivalent to a Laplace prior distribution (GLL); if we constrain $\boldsymbol{U}, \boldsymbol{V}$ to be nonnegative then the L_1 norm is equivalent to an exponential prior distribution (GEE); and finally, the L_1^2 norm can be formulated as a nonnegative prior distribution, which we use for the GL_1^2 model (see Table 4.1).

The type of priors chosen for Bayesian matrix factorisation determine the type of sparsity that we add to the model. The L_1 norm causes the rows of the factor matrices to have very few non-zero entries because reducing a value U_{ik} causes a linear reduction in the penalisation term regardless of how close it is to zero. In contrast, the L_2^2 norm

Table 4.2 Overview of the four drug sensitivity, two MovieLens, and two methylation datasets, giving the number of rows (cell lines, users, genes), columns (drugs, movies, patients), and the fraction of entries that are observed.

Dataset	Rows	Columns	Fraction obs.
GDSC IC_{50}	707	139	0.806
CTRP EC_{50}	887	545	0.801
CCLE IC_{50}	504	24	0.965
CCLE EC_{50}	502	24	0.632
MovieLens 100K	943	1473	0.072
MovieLens 1M	6040	3503	0.047
Gene body methylation (GM)	160	254	1.000
Promoter methylation (PM)	160	254	1.000

gives a smaller reduction the closer U_{ik} gets to zero. Finally, the reduction for the L_1^2 norm depends on the values of the other entries $U_{ik'}$ in the same row.

Note that, although the cost function minimised by the nonprobabilistic approach with additional regularisation terms is the same as the log likelihood function of Bayesian approaches, the latter holds additional advantages. This is because in Bayesian inference we approximate the full posterior distribution, whereas the nonprobabilistic methods still only find a point estimate (in this case, a maximum a posteriori one); we saw in Chapter 3 that this is less robust than fully Bayesian solutions. In addition, we can use hierarchical priors to model further desired behaviour (such as ARD), which is much harder in nonprobabilistic models.

4.4 Data preprocessing

We conduct our experiments on a total of eight real-world datasets across three different applications, allowing us to see whether our observations on one dataset or application also hold more generally. We will focus on a couple of these datasets for some specific experiments. Also note that we make sure all datasets contain only positive integers, so that we can compare all four groups of Bayesian matrix factorisation approaches.

Firstly we used the four drug sensitivity datasets, as in last chapter. We preprocessed them nearly the same way: undoing the natural log transform of the GDSC dataset; capping high values to 100 for GDSC and CTRP; and then casting them as integers. We also filtered out rows and columns with only one or two observed datapoints.

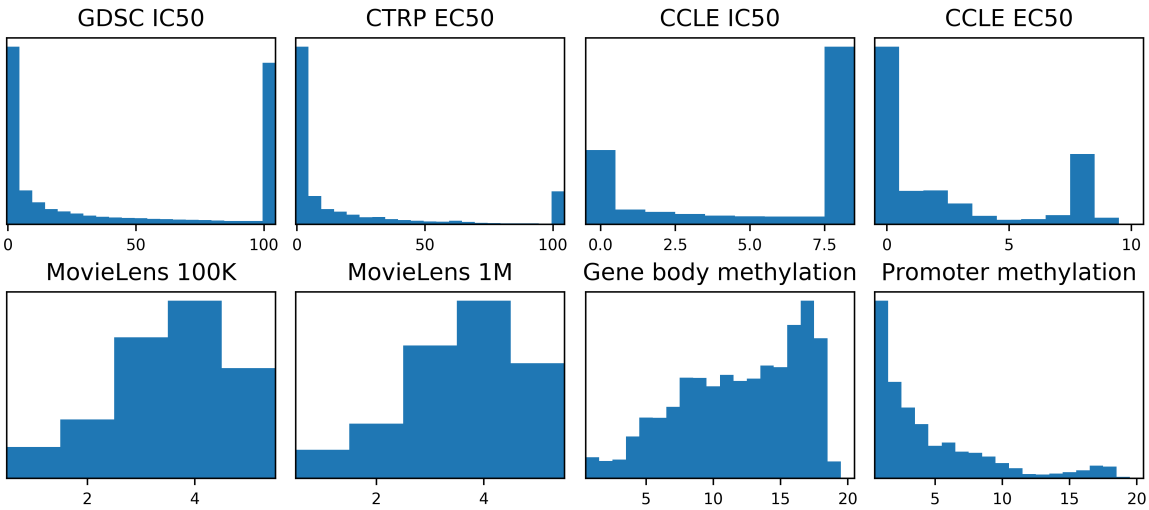


Figure 4.2 Distributions of the values of the eight datasets: GDSC IC_{50}, CTRP EC_{50}, CCLE IC_{50}, CCLE EC_{50}, MovieLens 100K, MovieLens 1M, gene body methylation, and promoter-region methylation.

The second application is collaborative filtering, where we are given movie ratings for different users (one to five stars) and we wish to predict the number of stars a user will give to an unseen movie. We use the MovieLens 100K and 1M datasets (Harper and Konstan [2015]), with 100,000 and 1,000,000 ratings respectively.

Finally, another bioinformatics application, this time looking at methylation expression profiles (Koboldt et al. [2012]). These datasets give the amount of methylation measured in either the body region of genes (gene body methylation) or the promoter region (promoter methylation) for 254 different patients. We focused on 160 cancer driver genes given by the IntOGen database (Gonzalez-Perez et al. [2013]). We multiplied all values by twenty and cast them as integers.

The datasets are summarised in Table 4.2, and the distribution of values for each dataset is visualised in Figure 4.2. This shows us that the drug sensitivity datasets tend to be bimodal, whereas the MovieLens and methylation datasets are more normally distributed. We can also see that the MovieLens datasets tend to be large and sparse, whereas the others are well-observed and relatively small.

4.5 Experiments

We conducted experiments to compare the four different groups of approaches. In particular, we measured their convergence and runtime speed, cross-validation perfor-

mance, sparse prediction performance, and model selection effectiveness. We sometimes focus on a selection of the methods for clarity. To make the comparison complete, we also added a non-probabilistic nonnegative matrix factorisation model (NMF, Lee and Seung [2000]) as a baseline.

4.5.1 Convergence and runtime speed

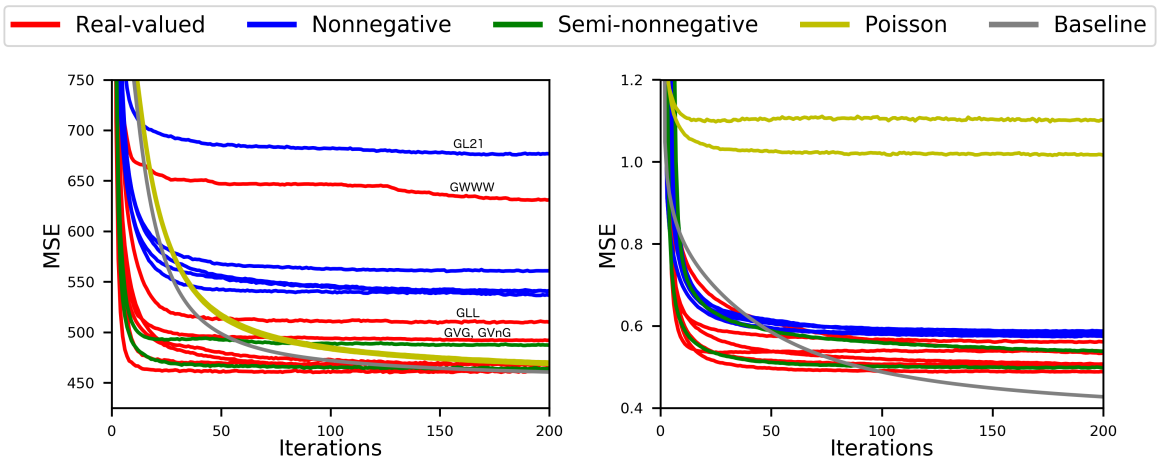
Firstly we compared the convergence speed of the models on the GDSC and MovieLens 100K datasets. We ran each model with $K = 20$, and measured the average mean squared error on the training data across ten runs. We plotted the results in Figure 4.3, where each group is plotted as the same colour: red for real-valued, blue for nonnegative, green for semi-nonnegative, yellow for Poisson, and grey for the non-probabilistic baseline. We can see that the Gaussian-likelihood methods all converge after the same number of iterations, but they differ in their depth. Real-valued methods converge the deepest, and nonnegative ones converge less deep, as they are more constrained. The semi-nonnegative models converge similarly to the real-valued ones. The Poisson versions converge either slowly (on GDSC) or less deep (on MovieLens 100K).

Furthermore, the average runtime (in seconds) per iteration is given in Table 4.3, for different values of K on the GDSC drug sensitivity and MovieLens 100K datasets. Here we see that the univariate posterior models (GGGU, GEE, GEEA, GTT, GTTN, GL_1^2, GEG) are faster than the multivariate ones (GGG, GGGA, GGGW, GLL, GLLI). GGGU is faster than the other univariate posterior versions because sampling from distributions like a truncated normal is slow. We can also see that the hierarchical priors are not noticably slower than their non-hierarchical counterparts. The volume prior models are by far the slowest, due to their higher time complexity. Finally, the Poisson models are slow for low K, but at $K = 50$ this is no longer true.

4.5.2 Cross-validation performance

Next we measured the 5-fold cross-validation performance on each of the eight datasets. We used the hyperparameter values from Section 4.2.3, and used 5-fold nested cross-validation to choose the dimensionality K. The average mean squared error of predictions are given in Figure 4.4 for all eight datasets. The average dimensionality found in nested cross-validation can be found in Table 4.4.

Firstly, we can see that the Poisson and non-probabilistic models generally perform much more poorly than all other models. The dimensionalities used are also much lower,



(a) GDSC drug sensitivity (b) MovieLens 100K

Figure 4.3 Convergence of the Bayesian matrix factorisation models on the GDSC drug sensitivity (left) and MovieLens 100K (right) datasets, measuring the training data fit (mean square error).

Table 4.3 Average runtime per iteration (in seconds) of the different Bayesian matrix factorisation models on GDSC drug sensitivity and MovieLens 100K.

	GDSC drug sensitivity				MovieLens 100K			
Method	$K = 5$	$= 10$	$= 20$	$= 50$	$K = 5$	$= 10$	$= 20$	$= 50$
GGG	0.14	0.18	0.29	0.93	1.04	1.29	1.86	5.07
GGGU	0.02	0.03	0.07	0.16	0.48	0.79	1.49	3.99
GGGA	0.14	0.17	0.28	0.93	1.03	1.30	1.88	5.75
GGGW	0.13	0.17	0.26	0.82	1.27	1.23	1.84	5.06
GLL	0.24	0.30	0.38	0.95	0.81	0.94	1.30	3.01
GLLI	0.22	0.29	0.42	0.98	0.82	0.99	1.49	3.26
GVG	0.42	1.02	2.58	22.1	1.22	2.86	5.94	40.9
GEE	0.06	0.12	0.25	0.66	0.49	0.86	1.68	4.16
GEEA	0.06	0.12	0.24	0.63	0.68	1.27	2.53	6.50
GTT	0.06	0.12	0.25	0.68	0.70	1.27	2.34	6.00
GTTN	0.07	0.14	0.29	0.77	0.71	1.22	2.55	6.56
GL_1^2	0.06	0.13	0.26	0.62	0.43	0.80	1.56	3.88
GVnG	0.45	1.16	2.88	22.9	2.02	5.08	9.63	53.2
GEG	0.07	0.12	0.22	0.61	0.96	1.30	1.34	3.51
PGG	0.36	0.40	0.50	0.78	1.32	1.96	3.36	7.07
PGGG	0.56	0.49	0.50	0.78	1.34	2.02	3.40	7.19
NMF-NP	0.01	0.04	0.04	0.16	0.32	0.52	1.11	2.62

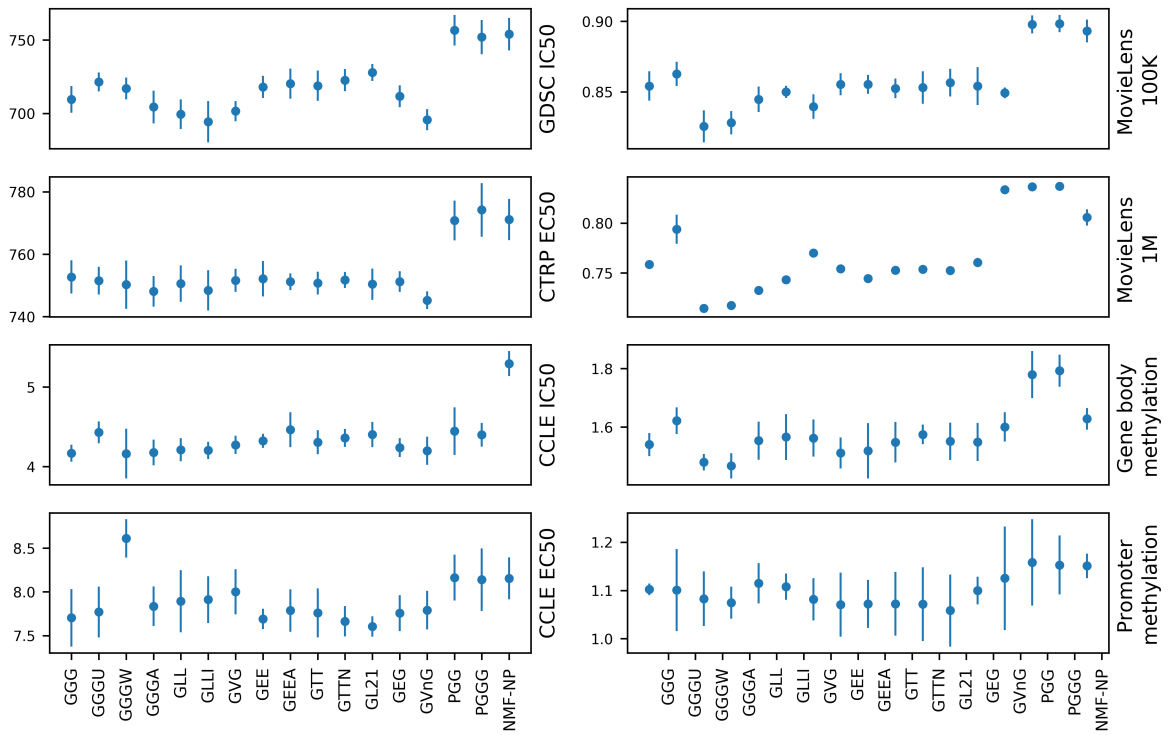


Figure 4.4 Average mean squared error of 5-fold nested cross-validation for the seventeen methods on the eight datasets. We also plot the standard deviation of errors across the folds.

except on CCLE IC_{50}. Secondly, we can see that most other methods have very similar performances. The Wishart, ARD, and Laplace models (GGGW, GGGA, GLL, GLLI) sometimes offer better performances (such as on GDSC, MovieLens 100K and 1M, and gene body methylation) but not always (such as GGGW on CCLE EC_{50}). Finally, we can see that the nonnegative models often have to use a higher dimensionality K to achieve the same predictive performance as the default GGG model. This seems to imply that they are more constrained and can therefore not use each factor as freely, which makes sense: they cannot form the same complex patterns of positive and negative values. We see something similar for the Laplace, Wishart, and volume prior models.

4.5.3 Noise test

We then measured the predictive performance when the datasets are very noisy. We added different levels of Gaussian noise to the data, with the noise-to-signal ratio being given by the ratio of the variance of the Gaussian noise we add, to the standard deviation of the generated data. For each noise level we split the datapoints randomly

Table 4.4 Average dimensionality found in 5-fold nested cross-validation for each Bayesian matrix factorisation model on the eight datasets.

	Drug sensitivity				MovieLens		Methylation	
Method	GDSC	CTRP	CCLE IC_{50}	CCLE EC_{50}	100K	1M	GM	PM
GGG	6	4	5	1	2	5	4	3
GGGU	6	5	5	1	2	2	3	3
GGGA	10	6	5	1	4	10	6	3
GGGW	16	8	6	2	5	13	7	3
GLL	10	6	5	1	3	8	4	2
GLLI	10	6	5	1	2	7	4	2
GVG	10	5	6	2	3	3	4	4
GEE	8	6	5	1	2	8	6	5
GEEA	10	6	5	1	2	10	6	4
GTT	9	6	5	1	2	8	5	4
GTTN	8	6	5	1	2	8	5	4
GL_1^2	6	6	4	1	2	8	5	5
GEG	6	5	5	1	2	5	4	3
GVnG	11	8	5	2	2	1	3	3
PGG	4	2	13	1	1	1	2	3
PGGG	5	2	12	1	1	1	2	3
NMF	4	2	1	1	1	3	2	2

into ten folds, and measured the predictive performance of the models on one held-out set at a time. We used $K = 5$ for all methods. The results for the GDSC drug sensitivity dataset are given in Figure 4.5, where we plot the ratio of the variance of the data to the mean squared error of the predictions—higher values are better, and using the row average gives a performance of one.

This shows that the Poisson and nonprobabilistic models have a significantly lower ratio, and are outperformed by the majority of the other models on all noise levels. The other models generally perform similarly and their variance can be explained by randomness. The Wishart model does seem to perform poorly, but as we saw in Table 4.4 this model requires a much higher dimensionality than $K = 5$ to obtain optimal predictions. As the noise increases it becomes harder for the models to give good predictions, although even at 200% noise they still pick up some signals.

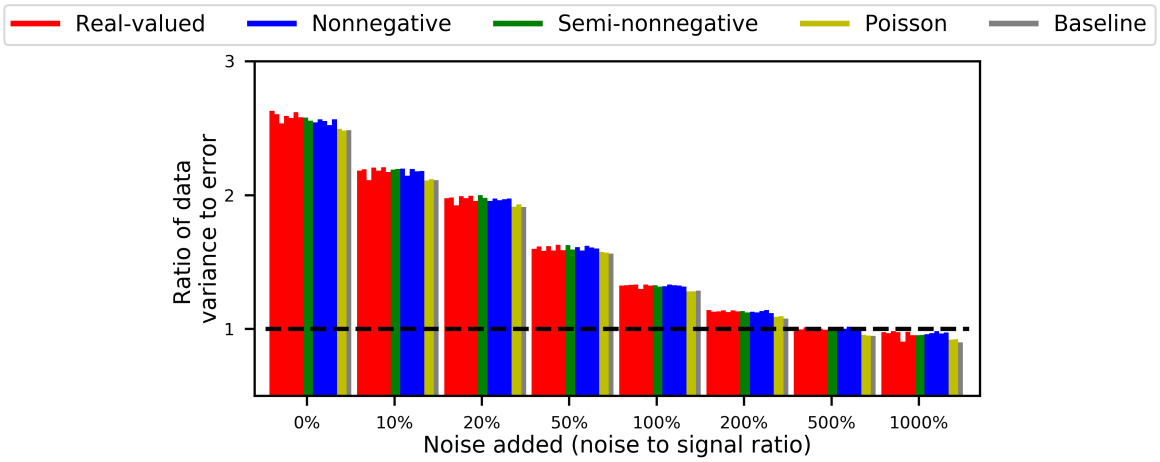


Figure 4.5 Noise experiment results on the GDSC drug sensitivity dataset. We added different levels of Gaussian noise to the data, and measured the 10-fold cross-validation performance.

4.5.4 Sparse predictions

Next we measured the predictive performances when the sparsity of the data increases. For different fractions of unobserved data, we randomly split the data based on that fraction, trained the model on the observed data, and measured the performance on the held-out test data. We used $K = 5$ for all models. The average mean squared error of ten repeats is given in Figure 4.6, showing the performances on the GDSC drug sensitivity, gene body methylation, and MovieLens 100K datasets. For each we provide a colour-coded global comparison of all models (Figures 4.6(a)(f)(k)) as well as comparing a couple of models at a time.

There are a couple of interesting observations. First of all, the Poisson models do not do well under high sparsity levels on the GDSC dataset. On the MovieLens 100K one they initially do poorly, but as the sparsity increases they start outperforming the other models. This implies that for large and extremely sparse datasets a Poisson likelihood might be better. We also see in Figures 4.6(c)(h)(m) that the standard nonnegative models (GEE, GTT, GL_1^2) are more robust to sparsity than the real-valued one (GGG).

Although some hierarchical models (such as Wishart) can give advantages, we observe little to no difference for some others—GLLI, GTTN, and PGGG give nearly identical performances in Figures 4.6(d)(i)(n) to GLL, GTT, and PGG. Similarly, we observe no difference between real-valued and semi-nonnegative models in Figure 4.6(e)(j)(o).

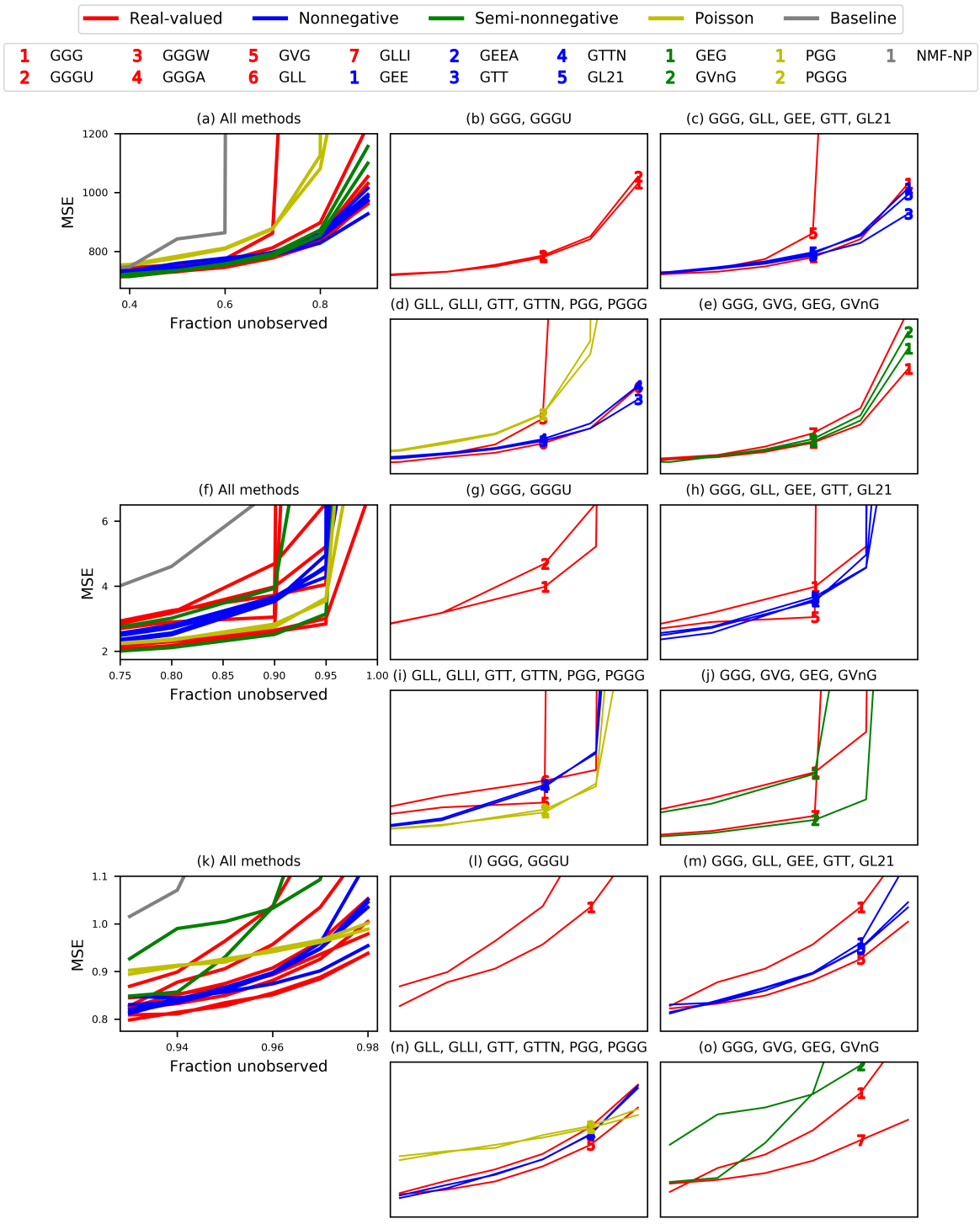


Figure 4.6 Sparsity experiment results on the GDSC drug sensitivity (top, a-e) gene body methylation (middle, f-j), and MovieLens 100K (bottom, k-o) datasets. We measure the predictive performance (mean squared error) on a held-out dataset for different fractions of unobserved data.

4.5.5 Model selection

We also measured the robustness of the models to overfitting if the dimensionality K is high. As a result, most models will fit very well to the training data, but give poor predictions on the test data. Here, we vary the dimensionality K for each of the models on the GDSC drug sensitivity and MovieLens 100K datasets, randomly taking out 10% as test data, and repeating ten times. The results are given in Figure 4.7. The model-level comparisons in 4.7(a)(f)(k) shows that the nonnegative models tend to have a flatter line as K increases than the real-valued ones, showing that they are more robust to overfitting. However, we also observe a few red lines that go much lower than the blue ones, such as the GGGW and GGGA models, which then outperform the nonnegative ones.

Last chapter we saw that ARD was effective at reducing overfitting for nonnegative matrix factorisation, and Figures 4.7(b)(c)(g)(h)(l)(m) show that the same holds for the real-valued model (GGG). The Wishart model (GGGW) also gives a similar effect.

As with the sparsity experiment, some of the hierarchical models have performances identical to their non-hierarchical counterparts (Figures 4.7(d)(i)(n)) and similarly for the semi-nonnegative and real-valued models (Figures 4.7(e)(j)(o)). Finally, we observe no difference in predictive performance between the multivariate and univariate posterior models (GGG, GGGU), as shown in Figures 4.7(b)(g)(l).

4.5.6 Factor usage

Finally, we consider the difference in factor values between the different models. When performing factor analysis we try to identify clusters in the rows and columns of the datasets, by looking at the values in the $\boldsymbol{U}$ and $\boldsymbol{V}$ matrices. Each factor can be seen as a cluster, and the value U_{ik} as the degree of membership of row i to cluster k.

In practice we want that similar datapoints have similar factor values, and that those in different clusters have dissimilar ones. To measure this, we can construct a similarity kernel based on the rows of the $\boldsymbol{U}$ and $\boldsymbol{V}$ matrices, computing the Pearson correlation for each. We therefore ran each model ten times on the GDSC drug sensitivity dataset, and plot the average similarity kernel across those ten runs. We used the absolute value of the factor matrices (it is the strength of the factor values that matters, not their sign), and also of the similarity kernels (a negative correlation still indicates similarity).

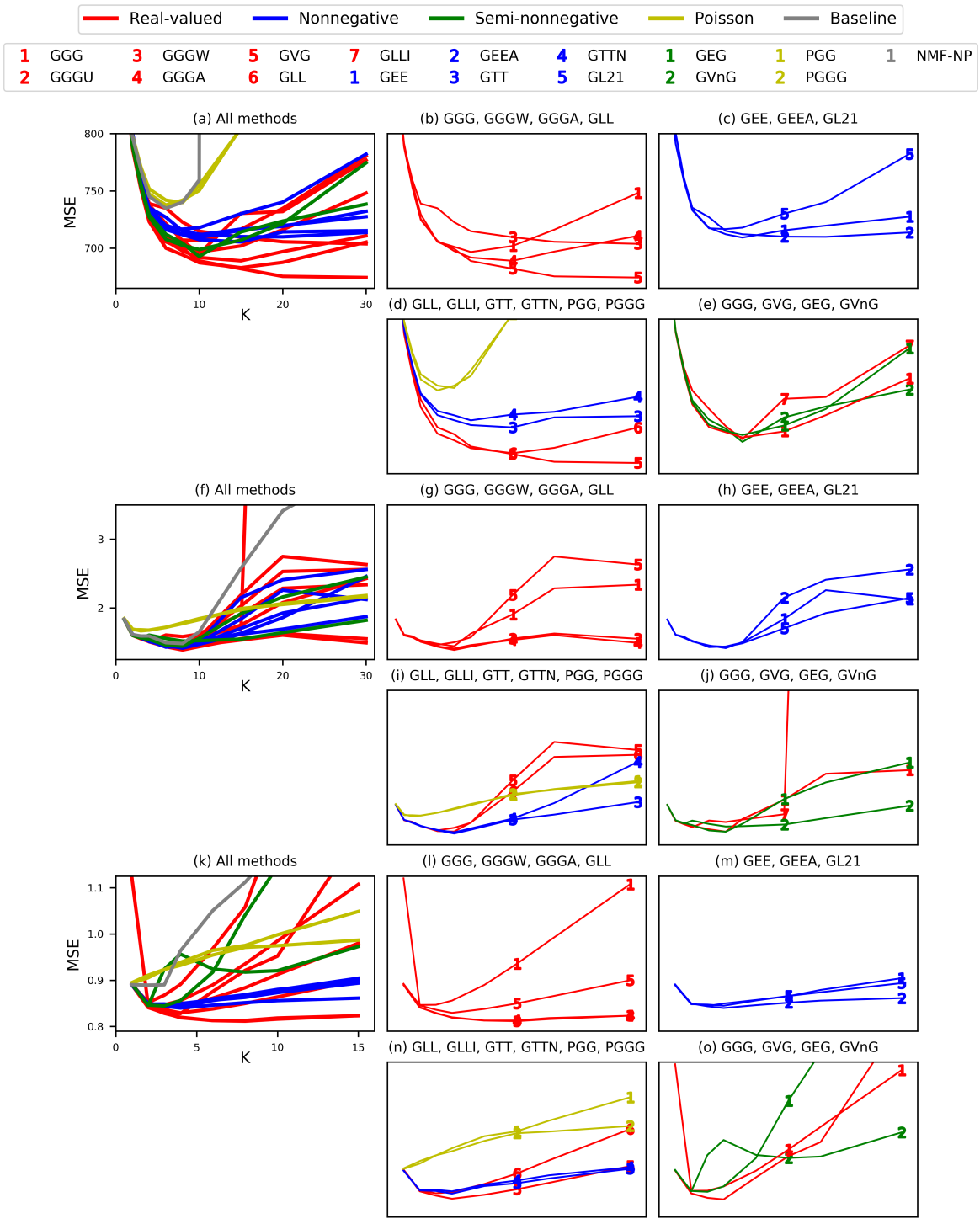
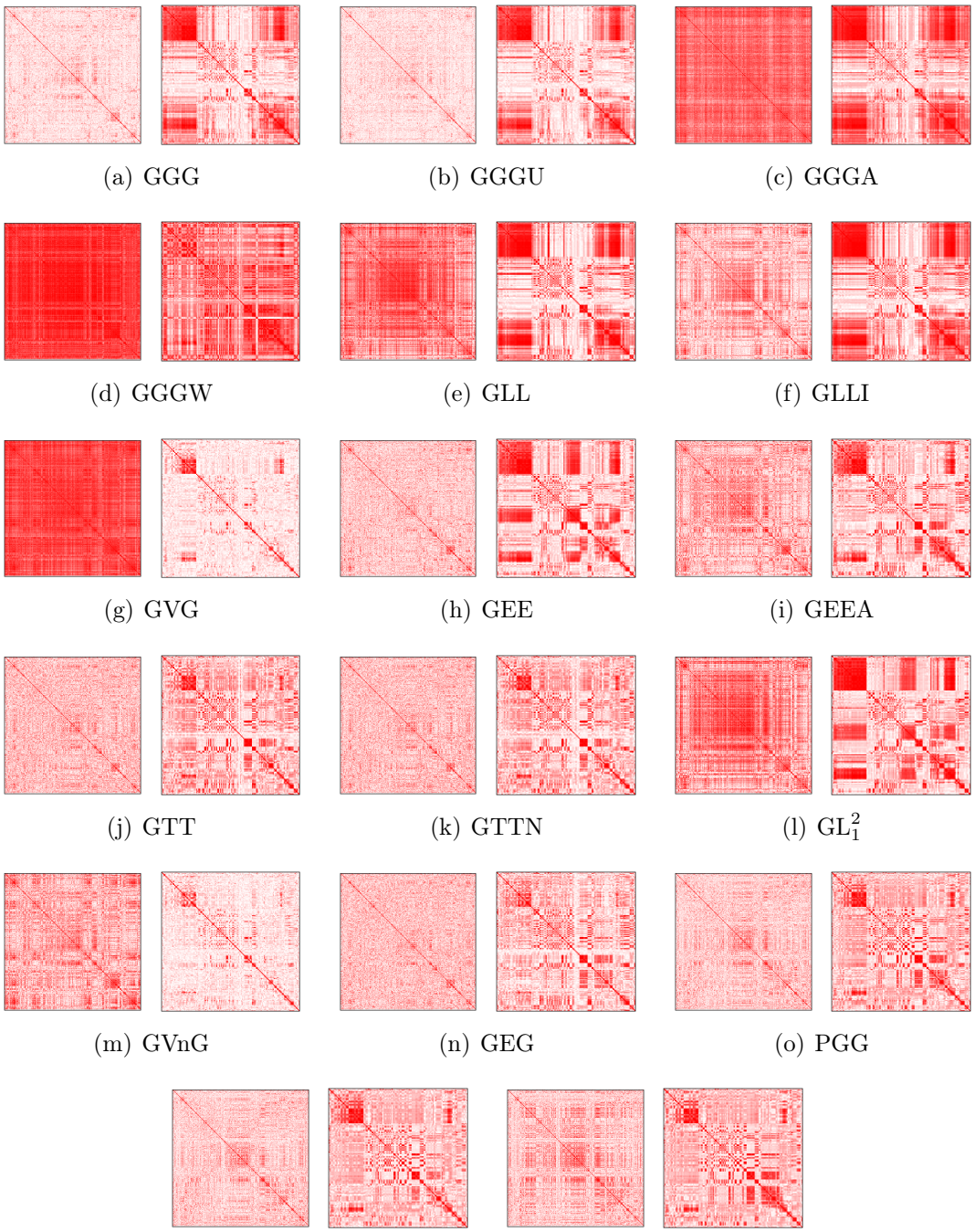


Figure 4.7 Model selection experiment results on the GDSC drug sensitivity (top, a-e), gene body methylation (middle, f-j), and MovieLens 100K (bottom, k-o) datasets. We measure the predictive performance (mean squared error) on a held-out dataset for different dimensionalities K.



(a) GGG (b) GGGU (c) GGGA

(d) GGGW (e) GLL (f) GLLI

(g) GVG (h) GEE (i) GEEA

(j) GTT (k) GTTN (l) GL_1^2

(m) GVnG (n) GEG (o) PGG

(p) PGGG (q) NMF-NP

Figure 4.8 Plots of the average Pearson correlation similarity kernels based on the $\boldsymbol{U}$ (left) and $\boldsymbol{V}$ (right) factor matrices for each method, on the GDSC drug sensitivity dataset. We took the absolute of all factor values, and also of the correlations. Red indicates stronger correlation.

We also reordered the rows of the GDSC dataset according to a hierarchical clustering (UPGMC) of the data, and similarly for the columns, to better display the patterns.

In Figure 4.8 we plot these average similarity kernels. If a method gives the sparsity that we want, we should expect small blocks of red (indicating high similarity) for datapoints close to each other, and other than that low values. Methods that give similar factor values for all datapoints will have nearly entirely red similarity kernels.

Some models display the desired behaviour on both matrices: GGG, GGGU, GLLI, GEE, GTT, GTTN, GVnG, GEG, PGG, PGGG, and NMF-NP. In contrast, GLL, GVG, and $\mathbf{GL}_1^2$ have a good $\boldsymbol{V}$ matrix, but the $\boldsymbol{U}$ is largely red. The worst models are GGGA, GGGW and GVG. The first two are especially interesting, because we saw before that the ARD and Wishart hierarchical priors can massively help with predictive tasks. However, this experiment shows that the factor values for each row and column are much more similar, hence making the factor matrices harder to analyse. This makes us believe there may be some trade-off between giving the models the freedom to best fit to the data for predictive performances on the one hand, and giving the desired row- and column-wise sparsity for good factor analysis on the other.

4.6 Conclusion

In this chapter we reviewed sixteen different Bayesian matrix factorisation models, that all use different likelihood and prior distributions. We provided the first systematic study of the effects of these choices on predictive performance, considering convergence and runtime speed, cross-validation performance, robustness to noise and sparsity, and model selection effectiveness. The novelty of this chapter is not in proposing new models, as we mainly reviewed popular models for the literature. Instead, it is bringing all these models into the same space of experiments. From the results shown in the previous section, we were able to draw the following conclusions. Although they are specific to the applications and dataset sizes studied, we believe that general insights can be drawn from them about the behaviour of the four different groups of Bayesian matrix factorisation models, which can assist future researchers in their model design.

Poisson likelihood methods perform poorly compared to the Gaussian likelihood—they overfit quickly, give worse predictive performances in cross-validation and under noisy conditions, partly because they cannot converge as deep as the other methods. At high sparsity levels they can start to perform better, seeming to indicate

that for larger and more sparse datasets they could start outperforming Gaussian models, as some papers (Gopalan et al. [2015]) claim they do. However, for small and well-observed datasets we found the opposite to be true.

Nonnegative models are more constrained than the real-valued ones, causing them to converge less deep, and to be less likely to overfit to high sparsity levels than the standard GGG model. However, the right hierarchical prior for a real-valued model (such as Wishart) can bridge this gap.

The automatic relevance determination and Wishart hierarchical priors are effective ways of preventing overfitting: the GGGA, GGGW, and GEEA models maintain a low predictive error as the dimensionality K increase, whereas the GGG and GEE models start overfitting more. We saw the same behaviour last chapter for nonnegative models, but the effect is even stronger for the real-valued ones. However, these hierarchical priors make the factor values for each row and column entity much more similar as well, and hence harder to analyse.

Similarly, the Laplace priors are good at reducing overfitting as the dimensionality grows, without requiring additional hierarchical priors.

Some other hierarchical priors do not make a difference, such as with GLLI, GTTN, and PGGG. They can help us automatically choose the hyperparameters, but in our experience the models are not very sensitive to this choice anyway.

There is no difference in performance between semi-nonnegative and real-valued matrix factorisation, as shown in the model selection and sparsity experiments: the performance for GGG and GEG, as well as GVG and GVnG, are nearly identical.

There is no difference in predictive performance between univariate and multivariate posteriors (GGG, GGGU), as shown in the model selection experiment.

We based the conclusions above on our experiments on three different applications. Some caution should be exercised about generalising these findings to other fields, and we do observe some variation in performance between the different datasets. It would be interesting to further explore some of these findings, for example finding out whether Poisson models really are better at larger and sparser datasets, and at what point they start outperforming the Gaussian ones.

Chapter 5

Bayesian hybrid matrix factorisation for data integration

In the previous two chapters we studied the trade-offs between inference approaches, as well as prior and likelihood choices, for Bayesian matrix factorisation. We now turn our attention to the final research question: how can we best integrate multiple datasets to improve our predictions? In this chapter we introduce a new Bayesian multiple matrix factorisation model, that can be used to integrate multiple datasets and to make both in- and out-of-matrix predictions. The model is very general and can be used to integrate many datasets across different entity types, including repeated experiments, similarity matrices, and very sparse datasets.

In our model, each dataset can either be decomposed into two matrices, in which case only the row factor matrices are shared, or into three, in which case the row and column matrices are shared. This results in a hybrid model between matrix factorisation and tri-factorisation. Additionally, each of the latent matrices can have nonnegative or real-valued factors, giving a hybrid between nonnegative, semi-nonnegative, and real-valued factorisations. By using a probabilistic approach, our method can effectively handle missing values and predict them, for both in- and out-of-matrix predictions, and as we saw before the Bayesian approach is much less prone to overfitting than non-probabilistic models. Furthermore, the rank of each matrix is automatically chosen using automatic relevance determination, eliminating the need to perform model selection. We will show that this hybrid combination of multiple matrix factorisation and tri-factorisation gives a more general model than tensor and Tucker decompositions.

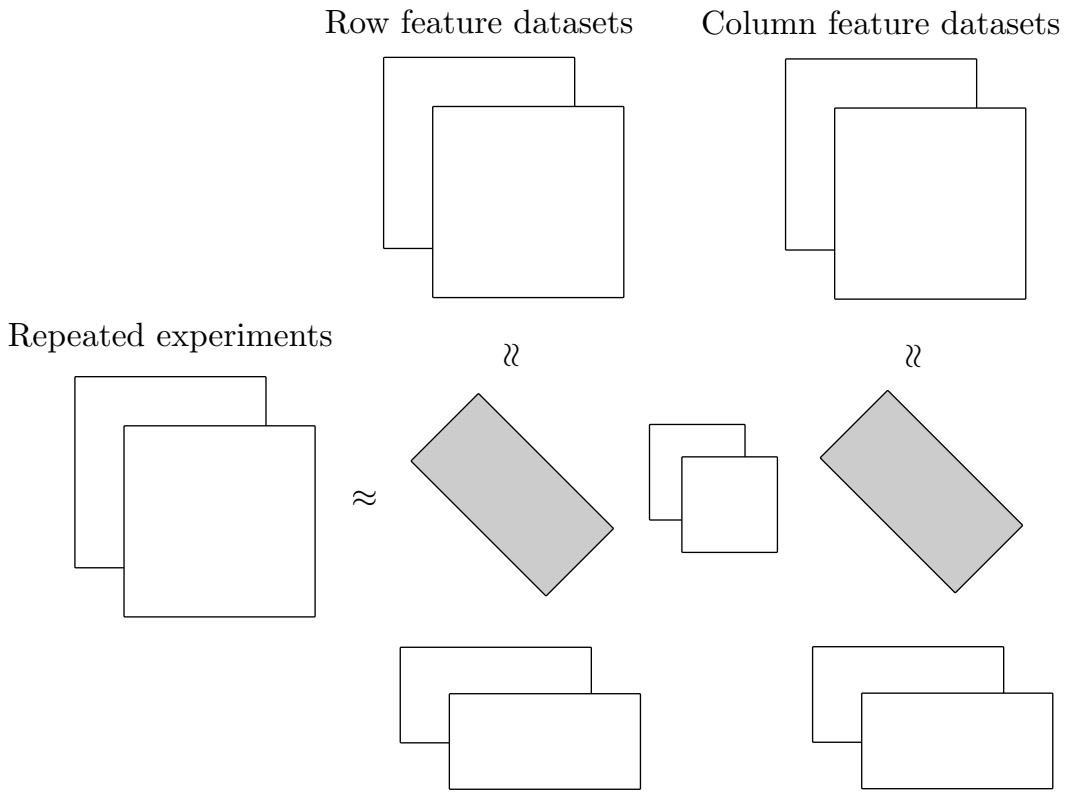


Figure 5.1 Overview of hybrid matrix factorisation, combining the multiple matrix tri-factorisation of two repeated experiments with multiple matrix factorisations of row and column feature datasets. Shared factor matrices are grey.

To demonstrate the effectiveness of our method for predicting missing values, we apply it to two different settings. Firstly, we again consider the four drug sensitivity datasets, where the matrices are similar (high correlation) and hence have high predictivity (low predictive error). We measure the in-matrix predictive performance of our method, as well as Bayesian and non-probabilistic matrix factorisation methods, and several state-of-the-art machine learning methods. Our model consistently outperforms all other methods, especially when the sparsity of the data increases. Secondly, we integrate gene expression, promoter region methylation, and gene body methylation profiles for breast cancer patients. These datasets are much more dissimilar, hence predicting one dataset given the others is much harder. However, out-of-matrix prediction experiments show that our method achieves better performance than state-of-the-art machine learning methods on two of the three combinations.

Our method is novel in several aspects. Firstly, it is the first general hybrid model between matrix factorisation and tri-factorisation. A non-probabilistic hybrid model

can be found in Zhu et al. [2007], but it only combined a single matrix tri-factorisation with a single matrix factorisation. Secondly, our model is a hybrid between nonnegative and real-valued factors: if multiple datasets are jointly decomposed, one can be a nonnegative matrix factorisation, where another can be semi-nonnegative, and another can be real-valued. Finally, through formulating the method as a Bayesian probabilistic model, it can deal with missing values, perform automatic model selection, and is much less prone to overfitting (especially for sparse datasets).

The contributions of this chapter are as follows.

- We introduce a novel Bayesian matrix factorisation model, that hybridly combines matrix factorisation and tri-factorisation, as well as real-valued, nonnegative, and semi-nonnegative factorisations.
- We demonstrate that this model can vastly improve both in- and out-of-matrix predictions.
- Finally, we explore several model choices, such as initialisation, hyperparameter values, and factorisation type trade-offs.

5.1 Hybrid matrix factorisation

The idea behind hybrid matrix factorisation (HMF) is to integrate multiple datasets by jointly decomposing them, and sharing their latent factors. We consider three different types of datasets, and each is decomposed using a different type of factorisation. These datasets span a number of different entity types (such as drugs, cell lines, targets) and each type has its own matrix of factor values that will be shared across the different factorisations that relate this entity type.

Formally, we are given a number of datasets spanning T different entity types $E_1, .., E_T$. Each entity type E_t has I_t instances, K_t factors, and a factor matrix $\boldsymbol{F}^t \in \mathbb{R}^{I_t \times K_t}$, which is shared across the relevant factorisations. We decompose the three dataset types as follows (see Figure 5.2):

1. Main datasets $\boldsymbol{R} = \{\boldsymbol{R}^1, .., \boldsymbol{R}^N\}$, relating two different entity types. For both entity type we have other matrices that we factorise, and therefore we share their factor matrices across different factorisations. By using matrix tri-factorisation here, and hence sharing two factor matrices, we can share more common patterns.

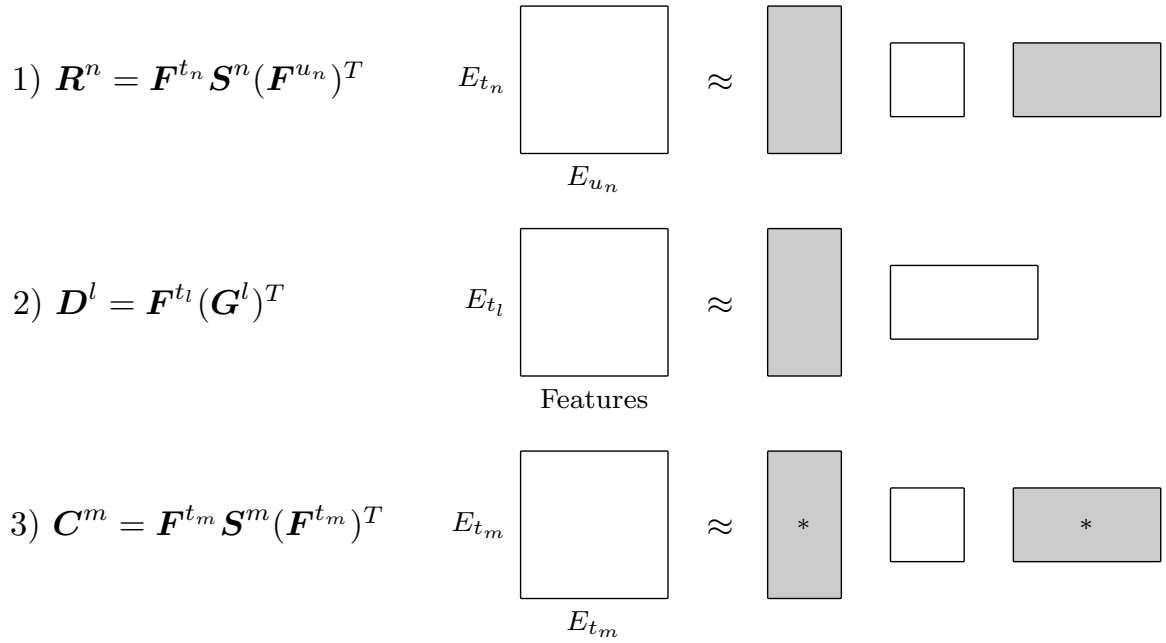


Figure 5.2 The three different types of datasets and factorisations used in hybrid matrix factorisation. Shared factor matrices are grey, and dataset-specific ones are white. The two grey matrices for the third factorisation type are the same (*).

Each dataset $\boldsymbol{R}^n \in \mathbb{R}^{I_{t_n} \times I_{u_n}}$ relates entity types E_{t_n}, E_{u_n}. We use matrix tri-factorisation to decompose it into two entity type factor matrices $\boldsymbol{F}^{t_n}, \boldsymbol{F}^{u_n}$, and a dataset-specific matrix $\boldsymbol{S}^n \in \mathbb{R}^{K_{t_n} \times K_{u_n}}$.

$$\boldsymbol{R}^n = \boldsymbol{F}^{t_n}\boldsymbol{S}^n(\boldsymbol{F}^{u_n})^T + \boldsymbol{E}^n.$$

2. Feature datasets $\boldsymbol{D} = \{\boldsymbol{D}^1, .., \boldsymbol{D}^L\}$ are similar to main datasets, but we do not have any other datasets relating the column entity type (the features) so we might as well use matrix factorisation rather than tri-factorisation—the third matrix would not be shared anyways.

Each dataset $\boldsymbol{D}^l \in \mathbb{R}^{I_{t_l} \times J_l}$ relates an entity type E_{t_l} to J_l features. We use matrix factorisation to decompose it into one entity type factor matrix $\boldsymbol{F}^{t_l}$, and a dataset-specific matrix $\boldsymbol{G}^l \in \mathbb{R}^{J_l \times K_{t_l}}$.

$$\boldsymbol{D}^l = \boldsymbol{F}^{t_l}(\boldsymbol{G}^l)^T + \boldsymbol{E}^l.$$

3. Similarity datasets $\boldsymbol{C} = \{\boldsymbol{C}^1, .., \boldsymbol{C}^M\}$ are those where the rows and the columns are the same entities, hence giving similarity values (such as Jaccard kernels). In the factorisation we use that entity type's factor matrix twice.

   Each dataset $\boldsymbol{C}^m \in \mathbb{R}^{I_{t_m} \times I_{t_m}}$ relates an entity type E_{t_m} to itself. We use matrix tri-factorisation to decompose it into a entity type factor matrix $\boldsymbol{F}^{t_m}$, a dataset-specific matrix $\boldsymbol{S}^m \in \mathbb{R}^{K_{t_m} \times K_{t_m}}$, and $\boldsymbol{F}^{t_m}$ again.

$$\boldsymbol{C}^m = \boldsymbol{F}^{t_m} \boldsymbol{S}^m (\boldsymbol{F}^{t_m})^T + \boldsymbol{E}^m.$$

Each dataset has a set indicating observed entries: $\Omega^n = \{(i,j)|\ R^n_{ij} \text{ observed}\}$, $\Omega^l = \{(i,j)|\ D^l_{ij} \text{ observed}\}$, $\Omega^m = \{(i,j)|\ C^m_{ij} \text{ observed}\}$, respectively.

For the similarity matrix factorisation we could have also decided to decompose $\boldsymbol{C} = \boldsymbol{F}\boldsymbol{F}^T + \boldsymbol{E}$, without the intermediate matrix $\boldsymbol{S}$. Ding et al. [2005b] includes a good discussion of the benefits to our approach (see Section 2.3.5). For this decomposition we do not consider diagonal entries of $\boldsymbol{C}$ (in other words, $(i,i) \notin \Omega$, $i = 1..I$) as this leads to third and fourth order terms in the posteriors and makes Gibbs sampling impossible. See Zhang and Yeung [2012] for a non-probabilistic approach that does consider these elements, leading to a very complicated optimisation problem.

The above formulation allows the user to very easily choose the kind of joint factorisation. By passing a set of matrices as $\boldsymbol{D}_1, .., \boldsymbol{D}_L$, multiple matrix factorisation is performed. Instead, passing them as $\boldsymbol{R}_1, .., \boldsymbol{R}_N$ gives multiple matrix tri-factorisation. A hybrid combination is also possible, as illustrated in Figure 5.1. Furthermore, each of the factor matrices can either be nonnegative (using an exponential prior), or real-valued (using a Gaussian prior), additionally giving a hybrid of nonnegative, semi-nonnegative and real-valued matrix factorisation.

Recall from Section 2.3.6 that if we wanted to integrate multiple datasets relating the same two entity types, and we use multiple matrix factorisation, we can only share one of the two factor matrices. Instead, by using multiple matrix tri-factorisation, more factor values can be shared between the datasets, with only small dataset-specific matrices ($\boldsymbol{S}^n$) in the middle. We expect that for datasets where the values are fairly similar (such as different biological labs conducting similar experiments—repeated experiments) this allows the model to more easily find the common patterns between the datasets and hence yield better predictions for missing values. For more dissimilar datasets, the multiple matrix factorisation approach might be better.

5.1.1 Model definition

Formally, the HMF model is defined as follows. The model likelihood functions are

$$
\begin{aligned}
R_{ij}^{n} &\sim \mathcal{N}(R_{ij}^{n} | \boldsymbol{F}_{i}^{t_n} \cdot \boldsymbol{S}^{n} \cdot \boldsymbol{F}_{j}^{u_n}, (\tau^{n})^{-1}) \\
D_{ij}^{l} &\sim \mathcal{N}(D_{ij}^{l} | \boldsymbol{F}_{i}^{t_l} \cdot \boldsymbol{G}_{j}^{l}, (\tau^{l})^{-1}) \\
C_{ij}^{m} &\sim \mathcal{N}(C_{ij}^{m} | \boldsymbol{F}_{i}^{t_m} \cdot \boldsymbol{S}^{m} \cdot \boldsymbol{F}_{j}^{t_m}, (\tau^{m})^{-1}),
\end{aligned}
$$

with Bayesian priors

$$
\begin{aligned}
&\tau^{n}, \tau^{l}, \tau^{m} \sim \mathcal{G}(\tau^{*} | \alpha_{\tau}, \beta_{\tau}) \\
&F_{ik}^{t} \sim \mathcal{E}(F_{ik}^{t} | \lambda_{k}^{t}) && \text{or} \quad F_{ik}^{t} \sim \mathcal{N}(F_{ik}^{t} | 0, (\lambda_{k}^{t})^{-1}) \\
&G_{jk}^{l} \sim \mathcal{E}(G_{jl}^{l} | \lambda_{k}^{t_l}) && \text{or} \quad G_{jk}^{l} \sim \mathcal{N}(G_{jl}^{l} | 0, (\lambda_{k}^{t_l})^{-1}) \\
&S_{kl}^{n} \sim \mathcal{E}(S_{kl}^{n} | \lambda_{S}^{n}) && \text{or} \quad S_{kl}^{n} \sim \mathcal{N}(S_{kl}^{n} | 0, (\lambda_{S}^{n})^{-1}) \\
&S_{kl}^{m} \sim \mathcal{E}(S_{kl}^{m} | \lambda_{S}^{m}) && \text{or} \quad S_{kl}^{m} \sim \mathcal{N}(S_{kl}^{m} | 0, (\lambda_{S}^{m})^{-1}).
\end{aligned}
$$

Recall from Chapter 4 that the Gaussian prior is equivalent to the L_2^2 norm, whereas the exponential one is the same as the nonnegative L_1 norm. This allows the user to choose the type of sparsity desired, with the strength of the norm determined by the hyperparameter values $\lambda_k^t, \lambda_S^n, \lambda_S^m, \lambda_k^{t_l}$.

We furthermore use the **automatic relevance determination** prior, to help perform automatic relevance determination. We demonstrated the effectiveness of this prior in Chapter 3 for both matrix factorisation and tri-factorisation. We use one ARD prior for each factor matrix belonging to an entity type, through the λ_k^t parameters in the prior of F_{ik}^t and G_{jk}^l. This parameter is shared by all entities of type E_t, and hence the entire factor k is either activated (if λ_k^t has a low value) or "turned off" (if λ_k^t has a high value). We place Gamma prior over each of these variables,

$$
\lambda_{k}^{t} \sim \mathcal{G}(\lambda_{k}^{t} | \alpha_{0}, \beta_{0}).
$$

Through this construction, factors that are active for only a few entities will be pushed further to zero, turning the factor off. Instead of having to choose the correct values for the K_t, we can give an upper bound and our model will automatically determine the number of factors to use.

One challenge with multiple matrix factorisation is that it relies on finding common patterns in multiple datasets. If two datasets are very different, the methods may end up finding a solution that fits one dataset much better, resulting in poor predictions for the other one. To address this, we add an **importance value** hyperparameter for each of the $\boldsymbol{R}^n, \boldsymbol{D}^l, \boldsymbol{C}^m$ datasets, respectively $\alpha^n, \alpha^l, \alpha^m$, to ensure that the method will converge to a solution that better fits datasets with higher importance values. We modify the likelihood of the model by using these importance values,

$$\begin{aligned} p(\boldsymbol{\theta}|\boldsymbol{R}, \boldsymbol{D}, \boldsymbol{C}) \propto p(\boldsymbol{\theta}) &\times \prod_{n=1}^{N} p(\boldsymbol{R}^n|\boldsymbol{F}^{t_n}, \boldsymbol{S}^n, \boldsymbol{F}^{u_n}, \tau^n)^{\alpha^n} \\ &\times \prod_{l=1}^{L} p(\boldsymbol{D}^l|\boldsymbol{F}^{t_l}, \boldsymbol{G}^l, \tau^l)^{\alpha^l} \\ &\times \prod_{m=1}^{M} p(\boldsymbol{C}^m|\boldsymbol{F}^{t_m}, \boldsymbol{S}^m, \tau^m)^{\alpha^m} \end{aligned}$$

where $\boldsymbol{\theta}$ is the set of model parameters. This technique was used by Remes et al. [2015] to ensure their model fits the binary training labels. It effectively makes the prior less strong if the importance value is higher, and since the prior acts as a regulariser it makes the model fit more to the data. It can be interpreted as repeating each of the values in the dataset $\boldsymbol{D}^l$ α^l times, although it is not the same as simply passing the dataset α^l times to the model, since then we would fit to each dataset separately. It is also not the same as simply multiplying the noise parameters τ^n, τ^l, τ^m by the importance value $\alpha^n, \alpha^l, \alpha^m$, as can be seen below.

$$\begin{aligned} \mathcal{N}(D_{ij}^l|\boldsymbol{F}_i^{t_l}\boldsymbol{G}_j^l, (\tau^l)^{-1})^{\alpha^l} &\propto \left[(\tau^l)^{\frac{1}{2}} \exp\left\{-\frac{\tau^l}{2}\left(D_{ij}^l - \boldsymbol{F}_i^{t_l}, \boldsymbol{G}_j^l\right)^2\right\}\right]^{\alpha^l} \\ &\propto (\tau^l)^{\frac{\alpha^l}{2}} \exp\left\{-\frac{\tau^l\alpha^l}{2}\left(D_{ij}^l - \boldsymbol{F}_i^{t_l}, \boldsymbol{G}_j^l\right)^2\right\} \\ &\neq (\tau^l\alpha^l)^{\frac{1}{2}} \exp\left\{-\frac{\tau^l\alpha^l}{2}\left(D_{ij}^l - \boldsymbol{F}_i^{t_l}, \boldsymbol{G}_j^l\right)^2\right\} \\ &\propto \mathcal{N}(D_{ij}^l|\boldsymbol{F}_i^{t_l}\boldsymbol{G}_j^l, (\tau^l\alpha^l)^{-1}) \end{aligned}$$

The latter approach would not work: if we multiplied the noise by a constant α^l as above, the model would simply learn a value for τ^l to balance it out (so multiplied by $\frac{1}{\alpha^l}$). The importance value approach does give the desired behaviour of fitting better to the data.

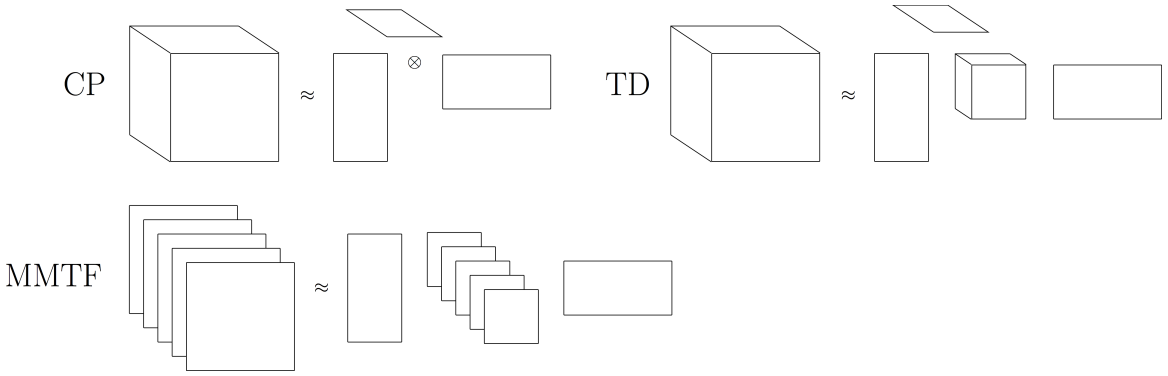


Figure 5.3 Overview of the CANDECOMP/PARAFAC (CP, top left), Tucker decomposition (TD, top right), and multiple matrix tri-factorisation (MMTF, bottom) methods. CP uses the outer product ($\otimes$), whereas TD and MMTF use the matrix product.

5.1.2 Relation to tensor decomposition

Multiple matrix factorisation and tri-factorisation methods are closely linked with tensor decomposition. Here, we explore some of these connections. In particular, we show that the CANDECOMP/PARAFAC (CP, Harshman [1970]) method is a less general version of the multiple matrix tri-factorisation (MMTF) part of our HMF model; and furthermore that the Tucker Decomposition (TD, Tucker [1966]), without its orthogonality constraints, is equivalent to MMTF. All three decompositions are illustrated in Figure 5.3, and we define them mathematically below. Our model is effectively more general than TD, due to its ability to incorporate datasets across more than three entity types (rather than just tensors), as well as performing both matrix factorisation and tri-factorisation at the same time.

Recall from Section 2.3.8 that the CP method decomposes a given tensor $\boldsymbol{R} \in \mathbb{R}^{I \times J \times N}$ into $\boldsymbol{R} = \boldsymbol{F}^1 \otimes \boldsymbol{F}^2 \otimes \boldsymbol{S}$, where $\otimes$ denotes the matrix outer product, and $\boldsymbol{F}^1 \in \mathbb{R}^{I \times K}, \boldsymbol{F}^2 \in \mathbb{R}^{J \times K}, \boldsymbol{S} \in \mathbb{R}^{N \times K}$. Each individual entry in $\boldsymbol{R}$ is decomposed as follows:

$$R_{ijn} = \sum_{k=1}^{K} F^1_{ik} \cdot F^2_{jk} \cdot S_{nk}. \tag{5.1}$$

The Tucker decomposition is defined similarly, with an additional core tensor $\boldsymbol{G} \in \mathbb{R}^{K \times L \times Q}$, and the factor matrices having its own number of latent factors K, L, Q; $\boldsymbol{F}^1 \in \mathbb{R}^{I \times K}, \boldsymbol{S} \in \mathbb{R}^{J \times L}, \boldsymbol{F}^2 \in \mathbb{R}^{N \times Q}$. We now factorise $\boldsymbol{R} = \boldsymbol{G} \bullet_1 \boldsymbol{F}^1 \bullet_2 \boldsymbol{F}^2 \bullet_3 \boldsymbol{S}$, where $\bullet_i$ denotes the matrix dot product using the ith dimension of tensor $\boldsymbol{G}$. Individual entries

in $\boldsymbol{R}$ are decomposed as:

$$R_{ijn} = \sum_{k=1}^{K} \sum_{l=1}^{L} \sum_{q=1}^{Q} F^1_{ik} \cdot F^2_{jl} \cdot S_{nq} \cdot G_{klq}. \tag{5.2}$$

Now consider the multiple matrix tri-factorisation part of our HMF model. Say we are given N datasets $\boldsymbol{R}^n$, all spanning the same two entity types E_1, E_2, with I rows, J columns, K row factors (for entity type E_1), and L row factors (for entity type E_2). Performing MMTF on these datasets can be seen as effectively concatenating the N matrices into one big tensor. Each entry is decomposed as:

$$R^n_{ij} = \sum_{k=1}^{K} \sum_{l=1}^{L} F^1_{ik} \cdot F^2_{jl} \cdot S^n_{kl}. \tag{5.3}$$

Firstly, compare this with the TD formulation. If we define a new matrix $H^n_{kl} = \sum_{q=1}^{Q} S_{nq} \cdot G_{klq} = \boldsymbol{S}_n \cdot \boldsymbol{G}_{kl}$, we can rewrite the TD expression as:

$$R_{ijn} = \sum_{k=1}^{K} \sum_{l=1}^{L} F^1_{ik} \cdot F^2_{jl} \cdot H^n_{kl}. \tag{5.4}$$

Note that this is now equivalent to our MMTF expression in Equation 5.3, with $F^1_{ik} \leftrightarrow F^1_{ik}, F^2_{jl} \leftrightarrow F^2_{jl}, S^n_{kl} \leftrightarrow H^n_{kl}$. Since both the $\boldsymbol{S}$ and $\boldsymbol{G}$ matrices have to be inferred by our model, we can in fact collapse them into one—as long as no further constraints are placed on them individually. Often in the TD method the three factor matrices $(\boldsymbol{F}^1, \boldsymbol{F}^2, \boldsymbol{S})$ have orthogonality constraints placed on them. If these constraints are dropped, TD is equivalent to MMTF.

Moving on to CP, consider constraining our MMTF model to have only diagonal entries in the $\boldsymbol{S}^n$ matrices (so that $S^n_{kl} = 0$ for $k \neq l$). Equation 5.3 then becomes:

$$R^n_{ij} = \sum_{k=1}^{K} F^1_{ik} \cdot F^2_{jk} \cdot S^n_{kk}. \tag{5.5}$$

This is equivalent to the CP formulation in Equation 5.1, with $F^1_{ik} \leftrightarrow F^1_{ik}, F^2_{jk} \leftrightarrow F^2_{jk}, S^n_{kk} \leftrightarrow S_{nk}$, showing that CP is a constrained version of MMTF, where the middle factor matrices S^n_{kl} are diagonal.

5.2 Gibbs sampling algorithm

Gibbs sampling can be used for inference due to the model's conjugacy, similarly to the models presented in the previous two chapters. Variational Bayesian inference is also possible, but as we discovered in Chapter 3, Gibbs sampling is more robust. We expect this to be especially the case in the multiple matrix factorisation case, as the optimisation landscape will be even more complex and therefore it is be easier to get stuck in a bad local minimum, which Gibbs sampling is better able to "jump out" of. As a result, we believe that the differences in performances due to different model choices will be minimally impacted by the inference method.

Although the inference is fairly straightforward for each individual factorisation, combining them into one general model is more tricky and requires careful notational definition. The datasets relating an entity type E_t are indicated by the following sets,

$$U_1^t = \{n \mid \boldsymbol{R}^n \in \boldsymbol{R} \ \wedge t_n = t \ \} \qquad V^t = \big\{l \mid \boldsymbol{D}^l \in \boldsymbol{D} \ \wedge t_l = t \ \big\}$$
$$U_2^t = \{n \mid \boldsymbol{R}^n \in \boldsymbol{R} \ \wedge u_n = t \ \} \qquad W^t = \{m \mid \boldsymbol{C}^m \in \boldsymbol{C} \ \wedge t_m = t \ \}.$$

Since the updates for the ARD can be different if a feature dataset is decomposed using negative factors or real-valued factors for $\boldsymbol{G}^l$, we also introduce the sets

$$V_+^t = \big\{l \in V^t \mid \boldsymbol{G}^l \text{ is nonnegative}\big\} \qquad V_-^t = \big\{l \in V^t \mid \boldsymbol{G}^l \text{ is real-valued}\big\}.$$

All $\boldsymbol{F}^t$ matrices are either real-valued or nonnegative, but if we wish to allow the user to specify this individually for each one, we can extend the model in the same way as we did for the $\boldsymbol{G}^l$. Observed entries per row i and column j are given by

$$\Omega_i^{n1} = \{j \ |(i,j) \in \Omega^n\} \qquad \Omega_i^{l1} = \big\{j \ |(i,j) \in \Omega^l\big\} \qquad \Omega_i^{m1} = \{j \ |(i,j) \in \Omega^m\}$$
$$\Omega_j^{n2} = \{i \ |(i,j) \in \Omega^n\} \qquad \Omega_j^{l2} = \big\{i \ |(i,j) \in \Omega^l\big\} \qquad \Omega_j^{m2} = \{i \ |(i,j) \in \Omega^m\}.$$

We obtain the following posterior distributions and parameter values.

Noise parameters

$$\tau^n \sim \mathcal{G}(\tau^n | \alpha_*^n, \beta_*^n) \qquad \alpha_*^n = \alpha_\tau + \alpha^n \frac{|\Omega^n|}{2}$$
$$\beta_*^n = \beta_\tau + \alpha^n \frac{1}{2} \sum_{(i,j) \in \Omega^n} (R_{ij}^n - \boldsymbol{F}_i^{t_n} \boldsymbol{S}^n \boldsymbol{F}_j^{u_n})^2$$

$$\tau^l \sim \mathcal{G}(\tau^l|\alpha^l_*, \beta^l_*) \qquad \alpha^l_* = \alpha_\tau + \alpha^l \frac{|\Omega^l|}{2}$$
$$\beta^l_* = \beta_\tau + \alpha^l \frac{1}{2} \sum_{(i,j)\in\Omega^l} (D^l_{ij} - \boldsymbol{F}^{t_l}_i \boldsymbol{G}^l_j)^2$$
$$\tau^m \sim \mathcal{G}(\tau^m|\alpha^m_*, \beta^m_*) \qquad \alpha^m_* = \alpha_\tau + \alpha^m \frac{|\Omega^m|}{2}$$
$$\beta^m_* = \beta_\tau + \alpha^m \frac{1}{2} \sum_{(i,j)\in\Omega^m} (C^m_{ij} - \boldsymbol{F}^{t_m}_i \boldsymbol{S}^m \boldsymbol{F}^{t_m}_j)^2$$

Note that this is the same as for the single-matrix factorisations and tri-factorisations.

ARD — If $\boldsymbol{F}^t$ contains nonnegative factors:

$$\lambda^t_k \sim \mathcal{G}(\lambda^t_k|\alpha^t_k, \beta^t_k) \qquad \alpha^t_k = \alpha_0 + I_t + \sum_{l\in V^t_+} I_{t_l} + \sum_{l\in V^t_-} \frac{I_{t_l}}{2}$$
$$\beta^t_k = \beta_0 + \sum_{i=1}^{I_t} F_{ik} + \sum_{l\in V^t_+} \sum_{j=1}^{J_l} G_{jk} + \sum_{l\in V^t_-} \frac{1}{2} \sum_{j=1}^{J_l} G^2_{jk}.$$

If $\boldsymbol{F}^t$ contains real-valued factors:

$$\lambda^t_k \sim \mathcal{G}(\lambda^t_k|\alpha^t_k, \beta^t_k) \qquad \alpha^t_k = \alpha_0 + \frac{I_t}{2} + \sum_{l\in V^t_+} I_{t_l} + \sum_{l\in V^t_-} \frac{I_{t_l}}{2}$$
$$\beta^t_k = \beta_0 + \frac{1}{2} \sum_{i=1}^{I_t} F^2_{ik} + \sum_{l\in V^t_+} \sum_{j=1}^{J_l} G_{jk} + \sum_{l\in V^t_-} \frac{1}{2} \sum_{j=1}^{J_l} G^2_{jk}.$$

These expressions generalise the ARD updates presented in Chapter 3 by considering all datasets that share the λ^t_k parameter: all R^n, D^l, C^m that use the factor matrix F^t in their factorisation.

Dataset-specific factor matrices — If $\boldsymbol{G}^l, \boldsymbol{S}^n, \boldsymbol{S}^m$ contain nonnegative factors:

$$G^l_{jk} \sim \mathcal{TN}(G^l_{jk}|\mu^l_{jk}, \tau^l_{jk})$$
$$\tau^l_{jk} = \tau^l \alpha^l \sum_{i\in\Omega^{l2}_j} (F^{t_l}_{ik})^2$$
$$\mu^l_{jk} = \frac{1}{\tau^l_{jk}} \left(-\lambda^{t_l}_k + \tau^l \alpha^l \sum_{i\in\Omega^{l2}_j} (D^l_{ij} - \sum_{k'\neq k} F^{t_l}_{ik'} G^l_{jk'}) F^{t_l}_{ik} \right)$$
$$S^n_{kl} \sim \mathcal{TN}(S^n_{kl}|\mu^n_{kl}, \tau^n_{kl})$$

$$\tau_{kl}^n = \tau^n \alpha^n \sum_{(i,j)\in\Omega^n} (F_{ik}^{t_n})^2 (F_{jl}^{u_n})^2$$

$$\mu_{kl}^n = \frac{1}{\tau_{kl}^n}\left(-\lambda_S^n + \tau^n\alpha^n \sum_{(i,j)\in\Omega^n} (R_{ij}^n - \sum_{(k',l')\neq(k,l)} F_{ik'}^{t_n} S_{k'l'}^n F_{jl'}^{u_n}) F_{ik}^{t_n} F_{jl}^{u_n}\right)$$

$$S_{kl}^m \sim \mathcal{TN}(S_{kl}^m | \mu_{kl}^m, \tau_{kl}^m)$$

$$\tau_{kl}^m = \tau^m \alpha^m \sum_{(i,j)\in\Omega^m} (F_{ik}^{t_m})^2 (F_{jl}^{t_m})^2$$

$$\mu_{kl}^m = \frac{1}{\tau_{kl}^m}\left(-\lambda_S^m + \tau^m\alpha^m \sum_{(i,j)\in\Omega^m} (C_{ij}^m - \sum_{(k',l')\neq(k,l)} F_{ik'}^{t_m} S_{k'l'}^m F_{jl'}^{t_m}) F_{ik}^{t_m} F_{jl}^{t_m}\right)$$

These expressions are also very similar to those presented in Chapter 3, with the notation updated according to this chapter's model definition.

If $\boldsymbol{G}^l, \boldsymbol{S}^n, \boldsymbol{S}^m$ contain real-valued factors:

$$G_{jk}^l \sim \mathcal{N}(G_{jk}^l | \mu_{jk}^l, (\tau_{jk}^l)^{-1}) \qquad \mu_{jk}^l = \frac{1}{\tau_{jk}^l}\left(\tau^l\alpha^l \sum_{i\in\Omega_j^{l2}} (D_{ij}^l - \sum_{k'\neq k} F_{ik'}^{t_l} G_{jk'}^l) F_{ik}^{t_l}\right)$$

$$\tau_{jk}^l = \lambda_k^{t_l} + \tau^l\alpha^l \sum_{i\in\Omega_j^{l2}} (F_{ik}^{t_l})^2$$

$$\boldsymbol{G}_j^l \sim \mathcal{N}(\boldsymbol{G}_j^l | \boldsymbol{\mu}_j^l, \boldsymbol{\Sigma}_j^l) \qquad \boldsymbol{\mu}_j^l = \boldsymbol{\Sigma}_j^l \cdot \left(\tau^l\alpha^l \sum_{i\in\Omega_j^{l2}} D_{ij}^l \boldsymbol{F}_i^{t_l}\right)$$

$$\boldsymbol{\Sigma}_j^l = \left(\mathrm{diag}(\boldsymbol{\lambda}^t) + \tau^l\alpha^l \sum_{i\in\Omega_j^{l2}} \boldsymbol{F}_i^{t_l} \otimes \boldsymbol{F}_i^{t_l}\right)^{-1}$$

$$S_{kl}^n \sim \mathcal{N}(S_{kl}^n | \mu_{kl}^n, (\tau_{kl}^n)^{-1}) \qquad \mu_{kl}^n = \frac{1}{\tau_{kl}^n}\left(\tau^n\alpha^n \sum_{(i,j)\in\Omega^n} (R_{ij}^n - \sum_{(k',l')\neq(k,l)} F_{ik'}^{t_n} S_{k'l'}^n F_{jl'}^{u_n}) F_{ik}^{t_n} F_{jl}^{u_n}\right)$$

$$\tau_{jk}^n = \lambda_S^n + \tau^n\alpha^n \sum_{(i,j)\in\Omega^n} (F_{ik}^{t_n})^2 (F_{jl}^{u_n})^2$$

$$\boldsymbol{S}_k^n \sim \mathcal{N}(\boldsymbol{S}_k^n | \boldsymbol{\mu}_k^n, \boldsymbol{\Sigma}_k^n) \qquad \boldsymbol{\mu}_k^n = \boldsymbol{\Sigma}_k^n \cdot \left(\tau^n\alpha^n \sum_{(i,j)\in\Omega^n} (R_{ij}^n - \sum_{k'\neq k} F_{ik'}^{t_n} (\boldsymbol{S}_{k'}^n \cdot \boldsymbol{F}_j^{u_n})) F_{ik}^{t_n} \boldsymbol{F}_j^{u_n}\right)$$

$$\boldsymbol{\Sigma}_k^n = \left(\lambda_S^n \boldsymbol{I} + \tau^n\alpha^n \sum_{(i,j)\in\Omega^n} (F_{ik}^{t_n})^2 (\boldsymbol{F}_j^{u_n} \otimes \boldsymbol{F}_j^{u_n})\right)^{-1}$$

$$S^m_{kl} \sim \mathcal{N}(S^m_{kl}|\mu^m_{kl}, (\tau^m_{kl})^{-1}) \qquad \mu^m_{kl} = \frac{1}{\tau^m_{kl}}\left(\tau^m\alpha^m \sum_{(i,j)\in\Omega^m}(C^m_{ij} - \sum_{(k',l')\neq(k,l)} F^{t_m}_{ik'} S^m_{k'l'} F^{t_m}_{jl'})F^{t_m}_{ik} F^{t_m}_{jl}\right)$$

$$\tau^m_{kl} = \lambda^m_S + \tau^m\alpha^m \sum_{(i,j)\in\Omega^m}(F^{t_m}_{ik})^2(F^{t_m}_{jl})^2$$

$$\boldsymbol{S}^m_k \sim \mathcal{N}(\boldsymbol{S}^m_k|\boldsymbol{\mu}^m_k, \boldsymbol{\Sigma}^m_k) \qquad \boldsymbol{\mu}^m_k = \boldsymbol{\Sigma}^m_k \cdot \left(\tau^m\alpha^m \sum_{(i,j)\in\Omega^m}(C^m_{ij} - \sum_{k'\neq k} F^{t_m}_{ik'}(\boldsymbol{S}^m_{k'} \cdot \boldsymbol{F}^{t_m}_j))F^{t_m}_{ik}\boldsymbol{F}^{t_m}_j\right)$$

$$\boldsymbol{\Sigma}^m_k = \left(\lambda^m_S \boldsymbol{I} + \tau^m\alpha^m \sum_{(i,j)\in\Omega^m}(F^{t_m}_{ik})^2(\boldsymbol{F}^{t_m}_j \otimes \boldsymbol{F}^{t_m}_j)\right)^{-1}$$

The updates for G^l_{jk} and $\boldsymbol{G}^l_j$ were introduced in Chapter 4, with updated notation here. For the remaining expressions Bayes' theorem was used in a similar way to derive the above parameter values. Although we saw in that chapter that there is little to no difference in performance between using a univariate or multivariate posterior, we still provide both options above.

Shared factor matrices — If $\boldsymbol{F}^t$ contains nonnegative factors:

$$F^t_{ik} \sim \mathcal{TN}(F^t_{ik}|\mu^t_{ik}, \tau^t_{ik})$$

$$\begin{aligned}\mu^t_{ik} = \frac{1}{\tau^t_{ik}}\Bigg(&-\lambda^t_k + \sum_{n\in U^t_1}\tau^n\alpha^n \sum_{j\in\Omega^{n1}_i}(R^n_{ij} - \sum_{k'\neq k} F^t_{ik'}(\boldsymbol{S}^n_{k'} \cdot \boldsymbol{F}^{u_n}_j))(\boldsymbol{S}^n_k \cdot \boldsymbol{F}^{u_n}_j)\\
&+ \sum_{n\in U^t_2}\tau^n\alpha^n \sum_{i'\in\Omega^{n2}_i}(R^n_{i'i} - \sum_{l\neq k} F^t_{il}(\boldsymbol{F}^{t_n}_{i'} \cdot \boldsymbol{S}^n_{\cdot,l}))(\boldsymbol{F}^{t_n}_{i'} \cdot \boldsymbol{S}^n_{\cdot,k})\\
&+ \sum_{l\in V^t}\tau^l\alpha^l \sum_{j\in\Omega^{l1}_i}(D^l_{ij} - \sum_{k'\neq k} F^t_{ik'}G^l_{jk'})G^l_{jk}\\
&+ \sum_{m\in W^t}\tau^m\alpha^m\Bigg[\sum_{j\in\Omega^{m1}_i}(C^m_{ij} - \sum_{k'\neq k} F^t_{ik'}(\boldsymbol{S}^m_{k'} \cdot \boldsymbol{F}^t_j))(\boldsymbol{S}^m_k \cdot \boldsymbol{F}^t_j)\\
&\qquad + \sum_{i'\in\Omega^{m2}_i}(C^m_{i'i} - \sum_{l\neq k} F^t_{il}(\boldsymbol{F}^t_{i'} \cdot \boldsymbol{S}^m_{\cdot,l}))(\boldsymbol{F}^t_{i'} \cdot \boldsymbol{S}^m_{\cdot,k})\Bigg]\Bigg)\end{aligned}$$

$$\begin{aligned}\tau^t_{ik} = &\sum_{n\in U^t_1}\tau^n\alpha^n \sum_{j\in\Omega^{n1}_i}(\boldsymbol{S}^n_k \cdot \boldsymbol{F}^{u_n}_j)^2 + \sum_{n\in U^t_2}\tau^n\alpha^n \sum_{i'\in\Omega^{n2}_i}(\boldsymbol{F}^{t_n}_{i'} \cdot \boldsymbol{S}^n_{\cdot,k})^2\\
&+ \sum_{l\in V^t}\tau^l\alpha^l \sum_{j\in\Omega^{l1}_i}(G^l_{jk})^2 + \sum_{m\in W^t}\tau^m\alpha^m\left[\sum_{j\in\Omega^{m1}_i}(\boldsymbol{S}^m_k \cdot \boldsymbol{F}^t_j)^2 + \sum_{i'\in\Omega^{m2}_i}(\boldsymbol{F}^t_{i'} \cdot \boldsymbol{S}^m_{\cdot,k})^2\right]\end{aligned}$$

This expression does the bulk of the work. Similar to the ARD parameters, the update for each entry in $\boldsymbol{F}^t$ has to consider all datasets that use this factor matrix for its decomposition. Note that we are effectively adding up the contribution of the prior and each individual dataset (using expressions similar to those presented in Chapter 3).

If $\boldsymbol{F}^t$ contains real-valued factors:

$$F_{ik}^t \sim \mathcal{N}(F_{ik}^t|\mu_{ik}^t,(\tau_{ik}^t)^{-1})$$

$$\mu_{ik}^t = \frac{1}{\tau_{ik}^t}\left(\sum_{n\in U_1^t}\tau^n\alpha^n\sum_{j\in\Omega_i^{n1}}(R_{ij}^n-\sum_{k'\neq k}F_{ik'}^t(\boldsymbol{S}_{k'}^n\cdot\boldsymbol{F}_j^{u_n}))(\boldsymbol{S}_k^n\cdot\boldsymbol{F}_j^{u_n})\right.$$
$$+\sum_{n\in U_2^t}\tau^n\alpha^n\sum_{i'\in\Omega_i^{n2}}(R_{i'i}^n-\sum_{l\neq k}F_{il}^t(\boldsymbol{F}_{i'}^{t_n}\cdot\boldsymbol{S}_{.,l}^n))(\boldsymbol{F}_{i'}^{t_n}\cdot\boldsymbol{S}_{.,k}^n)$$
$$+\sum_{l\in V^t}\tau^l\alpha^l\sum_{j\in\Omega_i^{l1}}(D_{ij}^l-\sum_{k'\neq k}F_{ik'}^tG_{jk'}^l)G_{jk}^l$$
$$+\sum_{m\in W^t}\tau^m\alpha^m\left[\sum_{j\in\Omega_i^{m1}}(C_{ij}^m-\sum_{k'\neq k}F_{ik'}^t(\boldsymbol{S}_{k'}^m\cdot\boldsymbol{F}_j^t))(\boldsymbol{S}_k^m\cdot\boldsymbol{F}_j^t)\right.$$
$$\left.\left.+\sum_{i'\in\Omega_i^{m2}}(C_{i'i}^m-\sum_{l\neq k}F_{il}^t(\boldsymbol{F}_{i'}^t\cdot\boldsymbol{S}_{.,l}^m))(\boldsymbol{F}_{i'}^t\cdot\boldsymbol{S}_{.,k}^m)\right]\right)$$

$$\tau_{ik}^t = \lambda_k^t+\sum_{n\in U_1^t}\tau^n\alpha^n\sum_{j\in\Omega_i^{n1}}(\boldsymbol{S}_k^n\cdot\boldsymbol{F}_j^{u_n})^2+\sum_{n\in U_2^t}\tau^n\alpha^n\sum_{i'\in\Omega_i^{n2}}(\boldsymbol{F}_{i'}^{t_n}\cdot\boldsymbol{S}_{.,k}^n)^2$$
$$+\sum_{l\in V^t}\tau^l\alpha^l\sum_{j\in\Omega_i^{l1}}(G_{jk}^l)^2+\sum_{m\in W^t}\tau^m\alpha^m\left[\sum_{j\in\Omega_i^{m1}}(\boldsymbol{S}_k^m\cdot\boldsymbol{F}_j^t)^2+\sum_{i'\in\Omega_i^{m2}}(\boldsymbol{F}_{i'}^t\cdot\boldsymbol{S}_{.,k}^m)^2\right]$$

$$\boldsymbol{F}_i^t \sim \mathcal{N}(\boldsymbol{F}_i^t|\boldsymbol{\mu}_i^t,\boldsymbol{\Sigma}_i^t)$$

$$\boldsymbol{\mu}_i^t = \boldsymbol{\Sigma}_i^t\cdot\left(\sum_{n\in U_1^t}\tau^n\alpha^n\sum_{j\in\Omega_i^{n1}}R_{ij}^n(\boldsymbol{S}^n\cdot\boldsymbol{F}_j^{u_n})+\sum_{n\in U_2^t}\tau^n\alpha^n\sum_{i'\in\Omega_i^{n2}}R_{i'i}^n(\boldsymbol{F}_{i'}^{t_n}\cdot\boldsymbol{S}^n)\right.$$
$$+\sum_{l\in V^t}\tau^l\alpha^l\sum_{j\in\Omega_i^{l1}}D_{ij}^l\boldsymbol{G}_j^l$$
$$\left.+\sum_{m\in W^t}\tau^m\alpha^m\left[\sum_{j\in\Omega_i^{m1}}C_{ij}^m(\boldsymbol{S}^m\cdot\boldsymbol{F}_j^t)+\sum_{i'\in\Omega_i^{l2}}C_{i'i}^m(\boldsymbol{F}_{i'}^t\cdot\boldsymbol{S}^m)\right]\right)$$

$$\boldsymbol{\Sigma}_i^t = \Bigg(\operatorname{diag}(\boldsymbol{\lambda}_k^t) + \sum_{n \in U_1^t} \tau^n \alpha^n \sum_{j \in \Omega_i^{n1}} (\boldsymbol{S}^n \cdot \boldsymbol{F}_j^{u_n}) \otimes (\boldsymbol{S}^n \cdot \boldsymbol{F}_j^{u_n}) + \sum_{n \in U_2^t} \tau^n \alpha^n \sum_{i' \in \Omega_i^{n2}} (\boldsymbol{F}_{i'}^{t_n} \cdot \boldsymbol{S}^n) \otimes (\boldsymbol{F}_{i'}^{t_n} \cdot \boldsymbol{S}^n) + \sum_{l \in V^t} \tau^l \alpha^l \sum_{j \in \Omega_i^{l1}} (\boldsymbol{G}_j^l \otimes \boldsymbol{G}_j^l) + \sum_{m \in W^t} \tau^m \alpha^m \Bigg[\sum_{j \in \Omega_i^{m1}} (\boldsymbol{S}^m \cdot \boldsymbol{F}_j^t) \otimes (\boldsymbol{S}^m \cdot \boldsymbol{F}_j^t) + \sum_{i' \in \Omega_i^{m2}} (\boldsymbol{F}_{i'}^t \cdot \boldsymbol{S}^m) \otimes (\boldsymbol{F}_{i'}^t \cdot \boldsymbol{S}^m) \Bigg] \Bigg)^{-1}$$

These parameter values are very similar to the nonnegative factors case, effectively just moving λ_k^t from μ_{ik}^t to τ_{ik}^t. Again we also provide the option of a multivariate posterior.

5.3 Implementation details

5.3.1 Software implementation

We provide an open-source Python implementation of all models discussed in the paper, available at https://github.com/ThomasBrouwer/HMF. We furthermore provide all datasets, preprocessing scripts, and Python code for the experiments.

5.3.2 Computational complexity

Recall that the updates for the Gibbs sampler for Bayesian matrix tri-factorisation have time complexity $\mathcal{O}(IJK^2L)$, compared to $\mathcal{O}(IJK^2)$ for Bayesian matrix factorisation. For HMF the complexity becomes $\mathcal{O}((N + M + L)I^2K^3)$ where $I = \max_t I_t$ and $K = \max_t K_t$. Notice that our model scales linearly in the number of observed datasets. Furthermore, the random draws for columns of the factor matrices are independent of each other, and therefore the parameter updates can be formulated as efficient joint matrix operations and new values drawn in parallel. Alternatively, the draws can be done per row by using a multivariate posterior, and then all these row-wise draws can be done in parallel as well.

5.3.3 Missing values and predictions

Missing values can be indicated to the model through the mask sets $\Omega^n, \Omega^l, \Omega^m$. Note that this also means that if specific feature values are missing for one of the entities, these features can still be included for the other entities, simply by marking them as unobserved when we do not know their value. This is much better than imputing those values, for example using the row or column average, as the model will still fit to those imputed values. This was the approach of previous multiple matrix tri-factorisation models (Žitnik and Zupan [2015]).

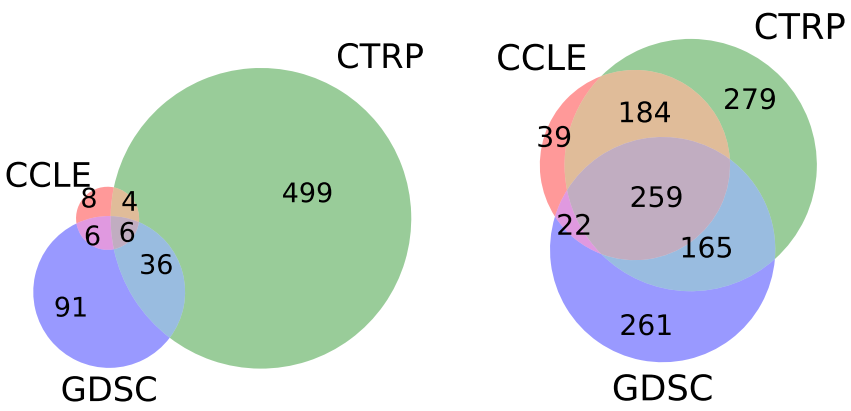
The missing values can then be predicted, by using the posterior draws of the Gibbs sampler (after burn-in and thinning) to estimate the posteriors of the factor matrices. For example, if we wish to predict missing values in the matrix $\boldsymbol{D}^l$, we estimate $\boldsymbol{F}^{t_l}$ and $\boldsymbol{G}^l$, and predictions for the missing entries are given by $\boldsymbol{F}^{t_l} \cdot (\boldsymbol{G}^l)^T$.

5.3.4 Initialisation strategies

Gibbs sampling can get stuck in a local maximum of the posterior, and therefore initialisation of the random variables is essential to obtain a good solution. As in the previous chapters, we can use the hyperparameters α_0, β_0, α_τ, β_τ, λ_S^n, λ_S^m to initialise the variables $\boldsymbol{F}^t$, $\boldsymbol{G}^l$, $\boldsymbol{S}^n$, $\boldsymbol{S}^m$, τ^n, τ^l, τ^m, λ_k^t either using the expectation of the prior model distribution, or by randomly drawing their value according to that distribution. Alternatively, we can initialise the entity type factor matrices $\boldsymbol{F}^t$ using K-means clustering, as suggested by Ding et al. [2006]. We can furthermore initialise the dataset-specific matrices $\boldsymbol{S}^n, \boldsymbol{S}^m, \boldsymbol{F}^l$ using least squares. This can be done using the Moore-Penrose pseudo-inverse ($^+$), as long as the dataset-specific matrices ($\boldsymbol{S}^n, \boldsymbol{S}^m, \boldsymbol{G}^l$) are real-valued. For example,

$$\boldsymbol{S}^n = (\boldsymbol{F}^{t_n})^+ \cdot \boldsymbol{R}^n \cdot ((\boldsymbol{F}^{u_n})^T)^+.$$

If the datasets are not real-valued, we can still initialise $\boldsymbol{S}^n$ or the other factor matrices in this way, but then set all values below zero to zero. We measure the effectiveness of the different initialisation methods in Section 5.6.1, which shows that this combination of K-means and least squares initialisation generally gives the fastest convergence.



(a) Overlap of drugs (b) Overlap of cell lines

Figure 5.4 Venn diagrams of the overlap of drugs and cell lines in the three drug sensitivity data sources.

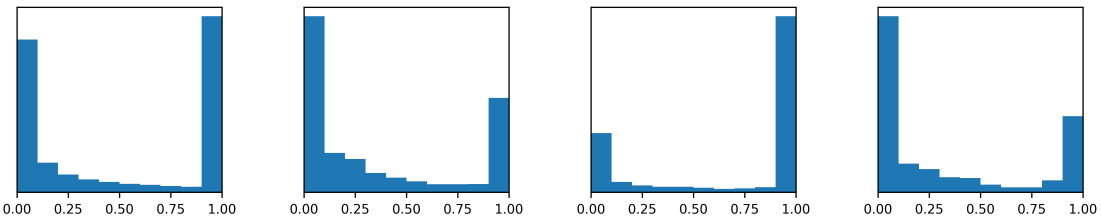
5.4 Data preprocessing

To demonstrate the advantages of our approach for missing values prediction, we consider two different applications. Firstly, integrating four drug sensitivity datasets, where the datasets are similar (high correlation) and hence predictivity of the datasets is high. Here we perform in-matrix predictions of missing values. Secondly, integrating gene expression and methylation level datasets for breast cancer patients and cancer driver genes, where the datasets are much more dissimilar. We perform out-of-matrix predictions, using the methylation levels of patients to predict gene expression values, and vice versa. We discuss the datasets and preprocessing in this section.

5.4.1 Drug sensitivity

We consider the same four drug sensitivity datasets as before—Genomics of Drug Sensitivity in Cancer (GDSC v5.0, Yang et al. [2013]), Cancer Therapeutics Response Portal (CTRP v2, Seashore-Ludlow et al. [2015]), Cancer Cell Line Encyclopedia (CCLE, Barretina et al. [2012])—but in this chapter we use different preprocessing on them so we can better measure the effectiveness of our data integration method.

In particular, we only considered drugs and cell lines that are present in at least two of the three data sources (considering CCLE IC_{50} and EC_{50} as one). Venn diagrams displaying the overlaps between drugs and cell lines are given in Figures 5.4(a) and



(a) GDSC IC_{50} (b) CTRP EC_{50} (c) CCLE IC_{50} (d) CCLE EC_{50}

Figure 5.5 Plots of the distribution of values in the four drug sensitivity datasets after preprocessing.

5.4(b), respectively. The CTRP dataset contains a large number of small molecule probes (311) causing very little intersection with the other datasets.

We also filtered out cell lines with no features available (to allow us to compare with other machine learning methods), which are provided by the GDSC dataset (gene expression levels, copy number variations, and mutation information). For the drugs we extracted 1D and 2D descriptors and structural fingerprints, obtained using the PaDeL-Descriptor software (http://www.yapcwsoft.com/dd/padeldescriptor/). We obtained primary protein targets from GDSC for 48 of the 52 drugs.

Recall that the CCLE and CTRP datasets all give the drug concentration levels, but the GDSC dataset gives the natural log transform of these values. We undid this transform by taking the exponent of each value. The drug sensitivity values for the CCLE IC_{50} and EC_{50} datasets lie in the range [0,8] and [0,10], but the other two datasets sporadically have extremely large values. This is a result of the curve fitting procedure used to approximate IC_{50} and EC_{50}, and in those cases it indicates an inefficient drug for the cell line. We cap all values above 20 to 20 to resolve this issue, and as a result obtain a similar shape of distribution of values to the CCLE datasets. Finally, we map the values in each row (per cell line) to the range [0,1]. The final distribution of values in each dataset is given in Figure 5.5, where we see that the data tends to be bimodal.

Before preprocessing the four datasets spanned 650 unique drugs and 1209 cell lines, and afterwards they cover 52 unique drugs and 399 cell lines, with 95.1% of the entries having at least one observed value, and 62.9% of the entries having at least two observed values. The information on the four datasets is summarised in Table 5.1, along with the fraction of overlapping observed entries between the datasets.

Table 5.1 Overview of the four drug sensitivity dataset after preprocessing.

Dataset	GDSC IC_{50}	CTRP EC_{50}	CCLE IC_{50}	CCLE EC_{50}
Number of cell lines	399	379	253	252
Number of drugs	48	46	16	16
Fraction observed	73.57%	86.03%	96.42%	58.88%
Overlap with GDSC IC_{50}	-	57.39%	44.19%	28.52%
Overlap with CTRP EC_{50}	52.25%	-	51.51%	31.87%
Overlap with CCLE IC_{50}	9.34%	11.96%	-	55.28%
Overlap with CCLE EC_{50}	6.00%	7.37%	55.06%	-

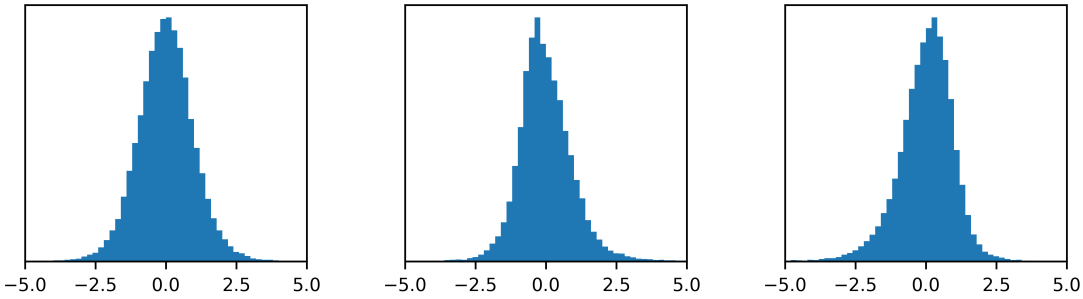
Finally, we constructed similarity kernels for each of the feature datasets, to compare predictive performances with a competing method. For binary data we used a Jaccard kernel, and for real-valued data we first standardised each feature to have zero mean and unit variance, and then used a Gaussian kernel to compute similarities, with as variance parameter the number of features.

5.4.2 Methylation and gene expression data

Our second application is that of integrating promoter-region methylation (PM) and gene body methylation (GM) datasets with a gene expression (GE) profile for breast cancer patients, coming from the The Cancer Genome Atlas (TCGA, Koboldt et al. [2012]). There are 254 different samples (both healthy and tumor tissues), across 13966 genes. We focus on 160 breast cancer driver genes, from the IntOGen database (Gonzalez-Perez et al. [2013]). We standardise the datasets to have zero mean and unit standard deviation per gene. Plots of the datasets containing the 160 genes after standardising can be found in Figure 5.6. Note that this dataset is not nonnegative. Finally, we constructed similarity kernels using these three datasets, giving similarity values between the cell lines. We used a Gaussian kernel with $\sigma^2 =$ no. genes.

5.5 Experiments

We measured the predictive performance of our method in both the in-matrix prediction setting, on the drug sensitivity data, and for out-of-matrix predictions, on the gene expression and methylation data. We compare our results against other matrix factorisation models, as well as several state-of-the-art machine learning methods.



(a) Gene expression (b) Promoter-region methylation (c) Gene body methylation

Figure 5.6 Plots of the distribution of the methylation datasets after standardising.

5.5.1 In-matrix predictions

We performed 10-fold cross-validation on each of the four drug sensitivity datasets to predict missing values. We tested two variants of our HMF model: multiple matrix tri-factorisation using all four drug sensitivity datasets (HMF D-MTF, $\boldsymbol{R}_n$), and multiple matrix factorisation on all four drug sensitivity datasets, sharing the cell line factors (HMF D-MF, $\boldsymbol{D}_l$). Many hybrid combinations in between these two models are possible, which we explore further in Section 5.6.4.

We compare our model to several state of the art methods. Since the four datasets are all nonnegative, we can use nonnegative matrix factorisation (NMF) and tri-factorisation (NMTF) models. We compare with non-probabilistic NMF by Lee and Seung [2000] (NP-NMF), Bayesian NMF by Schmidt et al. [2009] (BNMF), non-probabilistic NMTF by Yoo and Choi [2009] (NP-NMTF), Bayesian NMTF (BNMTF), and Multiple NMF (concatenating the datasets and using NMF-NP, sharing the cell line factors). We also applied several state-of-the-art machine learning models using the *skikit-learn* Python package, particularly: Linear Regression (LR), Random Forests (RF, 100 trees), and Support Vector Regression (SVR, *rbf* kernel). These methods were given the drug and cell line features for training. We considered a method called Kernelised Bayesian Matrix Factorisation (KBMF, Gönen and Kaski [2014]), which was used by Ammad-ud din et al. [2014] to predict drug sensitivity values for the GDSC dataset. This method leverages similarity kernels of the drugs and cell lines. Finally, we compared against the CP tensor decomposition model, to demonstrate that the more general multiple matrix factorisation and tri-factorisation approaches are better suited for making predictions. We modified our HMF D-MTF model by restricting the off-diagonal entries in $\boldsymbol{S}^n$ to be zero, giving the HMF CP model.

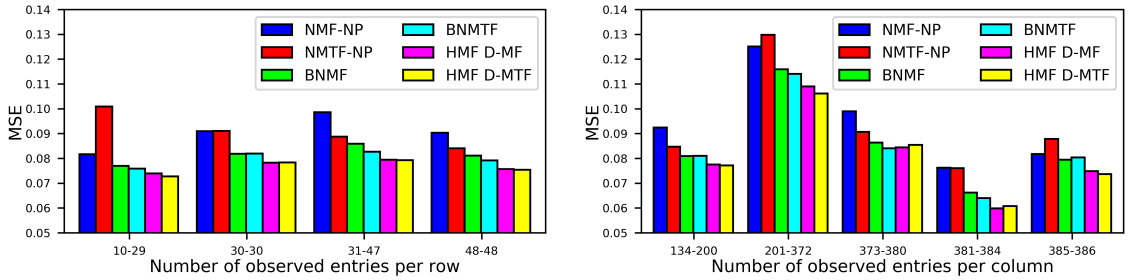
Table 5.2 Mean squared error (MSE) of 10-fold in-matrix cross-validation results on the drug sensitivity datasets. We also give the relative improvement (% impr.) compared to NMF. The best performances are highlighted in bold.

	GDSC IC_{50}		CTRP EC_{50}		CCLE IC_{50}		CCLE EC_{50}	
Method	MSE	% impr.	MSE	% impr.	MSE	% impr.	MSE	% impr.
NMF	0.0896	-	0.0959	-	0.0746	-	0.1535	-
NMTF	0.0879	1.91%	0.0954	0.44%	0.0747	-0.18%	0.1506	1.91%
Multiple NMF	0.0859	4.10%	0.0928	3.18%	0.0666	10.64%	0.1157	24.66%
BNMF	0.0805	10.20%	0.0919	4.05%	0.0594	20.29%	0.1318	14.19%
BNMTF	0.0799	10.81%	0.0920	4.03%	0.0593	20.52%	0.1292	15.84%
KBMF	0.0819	8.60%	0.0919	4.13%	0.0618	17.13%	0.1303	15.13%
LR	0.0886	1.10%	0.0949	1.00%	0.0719	3.62%	0.1342	12.60%
RF	0.0876	2.21%	0.0989	-3.15%	0.0668	10.47%	0.1219	20.62%
SVR	0.1091	-21.72%	0.1091	-13.80%	0.0916	-22.76%	0.1230	19.92%
HMF D-MF	0.0775	13.54%	0.0919	4.11%	0.0592	20.65%	**0.1062**	**30.81%**
HMF D-MTF	**0.0768**	**14.25%**	**0.0908**	**5.28%**	**0.0558**	**25.17%**	0.1073	30.12%
HMF CP	0.0796	11.16%	0.0913	4.80%	0.0560	24.93%	0.1104	28.08%

We performed nested cross-validation to select the dimensionality K for the matrix factorisation models and KBMF. In contrast, our model simply used $K_t = 10$ for each entity type E_t, and let the ARD choose the correct number of factors. We used nonnegative factors for the entity type factor matrices ($\boldsymbol{F}_t$), and real-valued for all other factors. We used K-means and least squares initialisation, and set all importance values to one. We explore these choices in depth in Section 5.6.

The results for cross-validation are given in Table 5.2. We see that our HMF models outperform all other methods, giving predictive gains of up to 30%. The multiple matrix tri-factorisation approach (HMF D-MTF) achieves the best performance on three of the datasets, and is a close second on the fourth. This makes sense: the four drug sensitivity datasets are highly correlated, so sharing more patterns by using multiple matrix tri-factorisation leads to better predictive performances. The tensor decomposition method (HMF CP) comes close, but restricting the off-diagonal elements to be zero resulted in a performance drop.

We also see that the Bayesian matrix factorisation models outperform both the non-probabilistic approaches, and the state-of-the-art machine learning methods, demonstrating that Bayesian matrix factorisation is a powerful paradigm for in-matrix predictions, with our proposed HMF model giving significant gains in predictive performance. The machine learning methods (LR, RF, SVR) have to rely on the feature



(a) Grouped per row

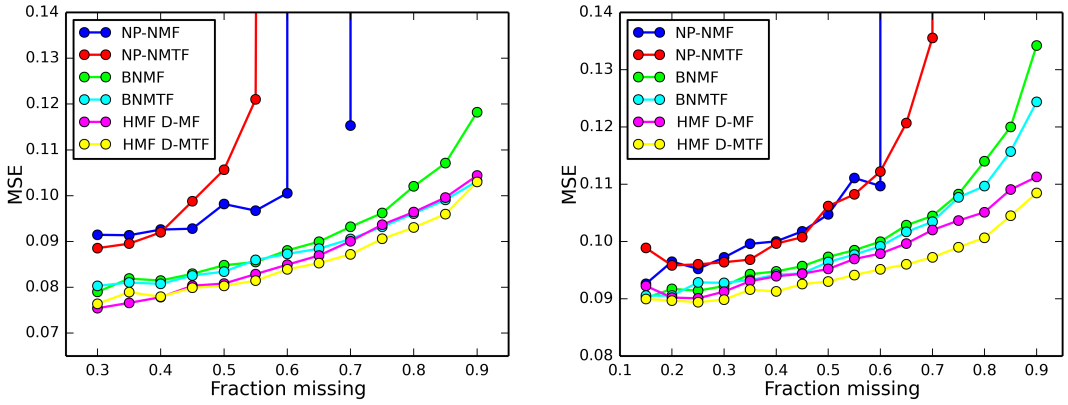
(b) Grouped per column

Figure 5.7 In-matrix cross-validation performances on the GDSC drug sensitivity dataset grouped by the number of observed datapoints—number of drugs per cell lines on the left, and number of cell lines per drug on the right.

values to predict missing values, which can often be a problem because the features may only be weakly predictive of the values in the matrix. In contrast, matrix factorisation models can directly leverage the hidden patterns in the matrix to predict them, leading to much better predictive performances.

We also experimented with using the feature datasets in our HMF model, either as similarity kernels $\boldsymbol{C}$ or feature datasets $\boldsymbol{D}$, but the two variants exploiting the four drug sensitivity datasets achieved the best performance. This is most likely because the feature datasets are not very predictive of the drug sensitivity values, as shown by the machine learning algorithms (LR, RF, SVR) achieving very poor performances. Multiple matrix factorisation methods rely on finding good common patterns between the datasets, and therefore the features did not contribute much.

To give a further intuition on where the improvements in predictions come from, we analysed the performances for the GDSC dataset and grouped them based on how many observed datapoints we have. In Figure 5.7(a) we grouped the rows based on how many cell lines (rows) they have observed entries for, and in Figure 5.7(b) for drugs (columns). We again ran ten-fold cross-validation, using the most common dimensionality found in nested cross-validation: $K = 2$ for NMF, $(K, L) = (4, 4)$ for NMTF, $K = 4$ for BNMF, and $(K, L) = (7, 7)$ for BNMTF. This shows us that the gain in performance for our proposed HMF models are very consistent across the board, regardless of how many datapoints are observed.



(a) GDSC (b) CTRP

Figure 5.8 Graphs showing average mean squared error (MSE) and standard deviation of in-matrix predictions on the GDSC (left) and CTRP (right) drug sensitivity datasets. We vary the fraction of missing entries, averaging performance across 20 random splits between train and test data, and compare our HMF models (HMF D-MF, HMF D-MTF) with several matrix factorisation models (NMF, NMTF, BNMF, BNMTF).

5.5.2 Sparse predictions

A very important use case is when there are few observed entries, leading to a sparse matrix. We measured the performances of in-matrix predictions on sparse matrices, focusing on the GDSC and CTRP drug sensitivity datasets as these are the largest. We vary the fraction of missing values and predict those entries, taking the average of twenty random training-test data splits per fraction. We compared our multiple matrix factorisation and tri-factorisation models (HMF D-MF and HMF D-MTF) with the other matrix factorisation models. For the dimensionality of HMF we used $K_t = 10$ as before, and for the matrix factorisation models (NMF, NMTF, BNMF, BNMTF) we used the most common dimensionality from the cross-validation in Section 5.5.1—GDSC: K = 2, (K,L) = (4,4), K = 4, (K,L) = (7,7). CTRP: K = 2, (K,L) = (2,4), K = 3, (K,L) = (3,3).

Figure 5.8 shows that the non-probabilistic models start overfitting very quickly as the sparsity levels of two datasets increase, on both the GDSC and CTRP datasets. The Bayesian versions perform a lot better, but our HMF models consistently outperform all other models, even when only 10% of the values are observed. The multiple matrix tri-factorisation model (HMF D-MTF) performs particularly well, again as expected.

Table 5.3 Mean squared error (MSE) of 10-fold out-of-matrix cross-validation results on the promoter-region methylation (PM), gene body methylation (GM), and gene expression (GE) datasets. We use two datasets as features, and predict values for new samples in the third dataset. The best results are highlighted in bold.

Method	GM, PM to GE	GE, GM to PM	GE, PM to GM
Gene average	1.009	1.008	1.009
LR	2.847	2.036	1.478
RF	0.811	0.799	0.714
SVR	**0.767**	0.749	0.657
HMF D-MF	0.788	**0.735**	**0.602**
HMF D-MTF	0.850	0.798	0.640
HMF S-MF	0.820	0.794	0.672
HMF CP	1.006	0.972	0.968

5.5.3 Out-of-matrix predictions

Finally, we performed three out-of-matrix prediction experiments on the methylation and gene expression data. We measured the predictive performances in ten-fold cross-validation, splitting the 254 samples into ten folds. Given the gene expression values of the other samples and both of the methylation datasets, we predicted the gene expression values for new samples (PM, GM to GE). We also did this for the other two combinations (GE, GM to PM; GE, PM to GM). Methylation data is known to be correlated with gene expression values (Kundaje et al. [2015]), although this correlation is generally weak. We therefore expected a weak predictive performance, but it is interesting to see which methods perform best.

We used the HMF D-MF and HMF D-MTF models described earlier, sharing the sample factors for HMF D-MF. We also considered the similarity dataset part of our model ($\boldsymbol{C}_m$) by using the similarity kernels of each dataset. We give the model the dataset we are trying to predict (e.g. GE), decomposing it using matrix factorisation, and also give it the similarity kernels for the other two (e.g. GM and PM). We call this approach HMF S-MF. We could have also used matrix tri-factorisation, but since the third matrix is not shared this is effectively the same model, and it gives no difference in predictive performance.

For the HMF D-MF models we used $K_t = 40$, 0.5 as the importance value for the dataset we are trying to predict, and 1.5 for the other two. For HMF D-MTF we used $K_t = 40$, and 0.5 as importance for all three datasets. Finally, for HMF CP and HMF

S-MF we used $K_t = 30$, and 1.0 as importance for all three datasets. For all four, we used nonnegative factors for shared matrices (K-means initialisation), and real-valued ones for private matrices (least squares initialisation).

We compared with the LR, RF, and SVR algorithms, giving two datasets as features, and the third as regression values. We additionally used the average value per gene as a baseline. Since the datasets are real-valued, we cannot compare with any nonnegative matrix factorisation models.

The results for this out-of-matrix cross-validation are given in Table 5.3. The HMF D-MF model outperforms all state-of-the-art machine learning methods on two of the three datasets, and is only beaten by SVR on the first one. Our model performs especially well on the third case (GE, PM to GM), implying our method works best when the predictivity of values is high (in other words, when the MSE of predictions is low). The HMF D-MTF and HMF S-MF methods perform slightly worse, but are still competitive with the other machine learning methods. Finally, the HMF CP model does not give good predictions in the out-of-matrix setting, barely outperforming the gene average baseline.

5.6 Model choices

We performed several additional experiments on the drug sensitivity and methylation datasets, to explore the advantages and limits of our models. In particular, we looked at the best initialisation approach; the effectiveness of the automatic relevance determination prior; the best values to use for the dataset importances α; and the effects of factorisation types on predictive performance. At the end of each section, we give our usage recommendations for the model.

5.6.1 Initialisation

As discussed in Section 5.3.4, there are several ways to initialise the Gibbs sampling parameter values. Here, we measure the convergence of the HMF D-MF and HMF D-MTF models (nonnegative shared factors, real-valued private factors) on the drug sensitivity datasets. We compare the following initialisation approaches (we always initialise ARD using the expectation):

1. **Exp:** All parameters initialised using expectation.

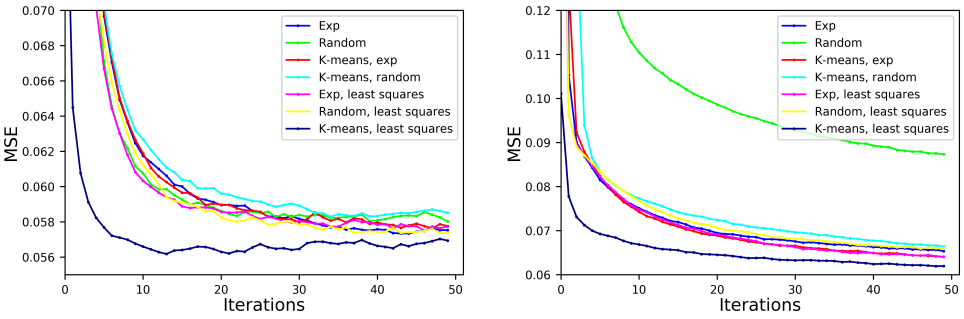
2. **Random:** All parameters initialised using random draws.
3. ***K*-means, exp:** Entity type factor matrices F^t initialised using K-means, other parameters using expectation.
4. ***K*-means, random:** Entity type factor matrices F^t initialised using K-means, other factor matrices using random draws.
5. **Exp, least squares:** Entity type factor matrices F^t initialised using expectation, other factor matrices using least squares.
6. **Random, least squares:** Entity type factor matrices F^t initialised using random draws, other factor matrices using least squares.
7. ***K*-means, least squares:** Entity type factor matrices F^t initialised using K-means, other factor matrices using least squares.

The plots of convergence for HMF D-MF and HMF D-MTF are given in Figure 5.9, averaging the training errors across ten repeats. We can see that the K-means with least squares initialisation strategy (dark blue) provides the fastest convergence on half of the datasets, often significantly faster. Random and least squares (yellow) also performs well. Other strategies sometimes provide faster convergence, but none of them do so consistently.

Recommendation: The fastest convergence is generally provided by combining K-means initialisation for $\boldsymbol{F}^t$ with least squares for the other factor matrices. Random initialisation for $\boldsymbol{F}^t$ with least squares also works well.

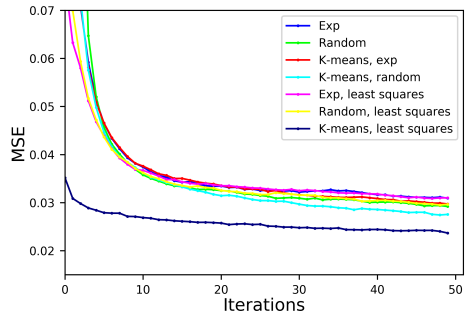
5.6.2 Model selection

In this section we measured how effective the automatic relevance determination is at performing automatic model selection. We performed a similar experiment in Chapter 3 on a single dataset—here we show that it is similarly very effective on multiple datasets. We repeat the in-matrix cross-validation experiments on the four drug sensitivity experiments, for the HMF D-MF (multiple matrix factorisation) and HMF D-MTF (multiple matrix tri-factorisation) models, and vary the values for K_t. Results are given in Figure 5.10, where we clearly see that adding ARD to the model consistently reduces overfitting on all four drug sensitivity datasets. ARD is not perfect, and we can still see that the curve goes up as the values for K_t increase, but this overfitting is significantly less severe than the models without ARD.

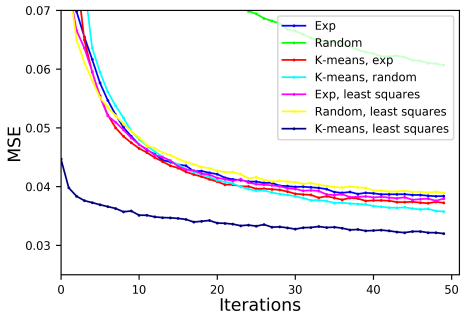


(a) HMF D-MF, GDSC IC_{50}

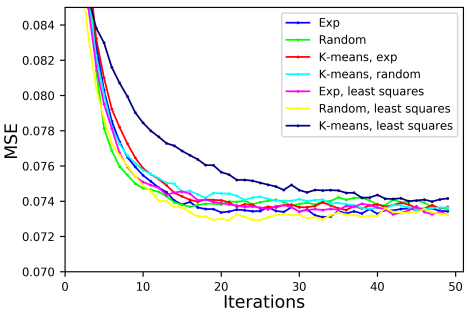
(b) HMF D-MTF, GDSC IC_{50}



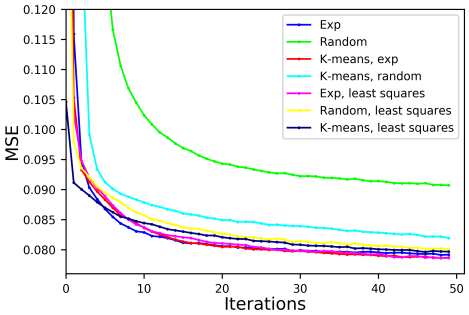
(c) HMF D-MF, CCLE IC_{50}



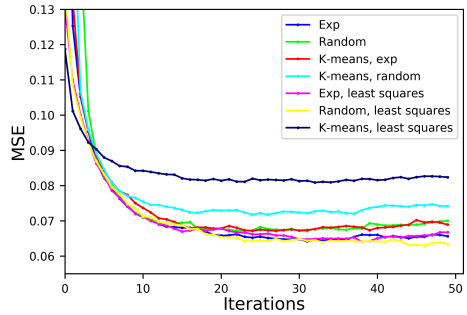
(d) HMF D-MTF, CCLE IC_{50}



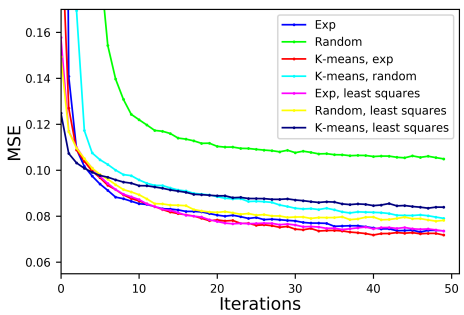
(e) HMF D-MF, CTRP EC_{50}



(f) HMF D-MTF, CTRP EC_{50}

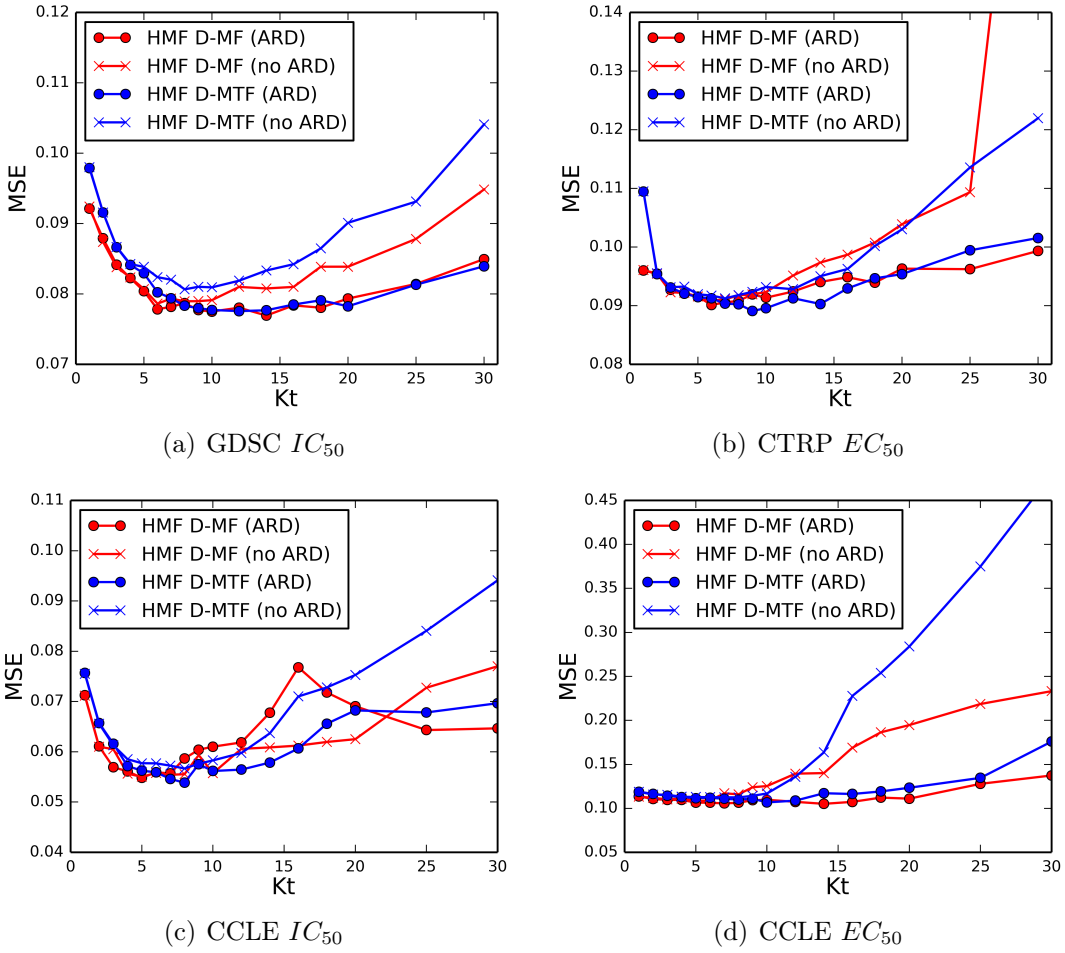


(g) HMF D-MF, CCLE EC_{50}



(h) HMF D-MTF, CCLE EC_{50}

Figure 5.9 Graphs showing the convergence of the HMF D-MF (top two rows) and HMF D-MTF (bottom two rows) models on the four drug sensitivity datasets, for the seven different initialisation approaches.



(a) GDSC IC_{50}

(b) CTRP EC_{50}

(c) CCLE IC_{50}

(d) CCLE EC_{50}

Figure 5.10 Graphs showing the cross-validation performance of in-matrix predictions on the drug sensitivity datasets, where we vary the dimensionality K_t for our HMF models (HMF D-MF in red, HMF D-MTF in blue), both for HMF with ARD (o), and without (x). Adding ARD clearly reduces overfitting as K_t increases.

Recommendation: Always use ARD to reduce overfitting in the model. Even though ARD does not always entirely eliminate the need for model selection (there is still some overfitting as K_t becomes very large), it generally makes it much less critical to try a large range of dimensionalities to find the best one. Instead, trying one or a couple will prove just as effective.

5.6.3 Importance value

We experimented with different values for the importance values α, for the out-of-matrix prediction setting. Specifically, we consider the case where we predict one dataset (either gene expression, promoter-region methylation, or gene body methylation) using

the other two datasets as additional datasets. We vary the value of α for the dataset we are trying to predict (α_0) as well as for the two other datasets we are learning from (α_1, α_2). We use $K_t = 10$ for all experiments, nonnegative factors for the shared factor matrices, and real-valued for the private ones. For initialisation we use K-means and least squares. We perform 10-fold cross-validation, taking out 10% of the samples each time for the dataset we are trying to predict, and then measuring the mean squared error (MSE) of predictions. The average performances can be found in Table 5.4, for both HMF D-MF (multiple matrix factorisation) and HMF D-MTF (multiple matrix tri-factorisation).

For the HMF D-MF model (left column) we see that datasets with low predictivity (such as GE and PM—note the high MSE) have the best parameter values when the importance of the dataset to be predicted (α_0) is low, and the importance of the datasets to learn from (α_1, α_2) is high. This is presumably because higher importance values lead to a better fit to the data, and if there is low predictivity, we should not fit to the data too much (otherwise we might overfit). In contrast, when the predictivity is high (such as GM, bottom row), the importance value for all datasets should not be too low, because this results in a poor fit to the data and hence poor predictions. For the HMF D-MTF model (right column) we see a similar effect, in that if the predictivity is better, the best values for the importance increase. However, for this approach all of the importance values should generally be set low if the datasets are different. For the approach based on similarity kernels (HMF S-MF) we found that the importance value had much less of an impact (results omitted), as long as it is not lower than 1.0.

Recommendation: If the datasets are more similar, use normal importance values (1.0) for all datasets. If the datasets are very dissimilar and have low predictivity, use a low importance value for the main dataset for which we are trying to predict values (like 0.5). For the other datasets use a higher importance value when using HMF D-MF (like 1.5), but for HMF D-MTF use a low one (like 0.5). Finally, when using similarity kernels (HMF S-MF), use the normal importance value for the kernels (1.0).

5.6.4 Factorisation types

Finally, we experimented with the choice of factorisation types. Recall that each dataset can be factorised either using matrix factorisation ($\boldsymbol{D}_l$) or matrix tri-factorisation ($\boldsymbol{R}_n$). For the drug sensitivity dataset, there are therefore a number of hybrid factorisation possibilities: using matrix factorisation for all (as used in the main paper; HMF D-MF),

Table 5.4 Performances of out-of-matrix cross-validation results for HMF D-MF (left column) and D-MTF (right column), where we vary the importance value for the dataset we are trying to predict (α_0), and for the other two datasets we are learning from (α_1, α_2). We have three different datasets (gene expression, GE; gene body methylation, GM; and promoter region methylation, PM). We therefore have three different prediction settings. We have highlighted the most promising parameter value areas in green, and the least promising in red.

HMF D-MF, GM + PM → GE

	GM (α_1), PM (α_2)				
GE (α_0)	0.25	0.5	1.0	1.5	2.0
0.25	0.878	0.843	0.835	0.831	0.832
0.5	0.845	0.849	0.829	0.832	0.829
1.0	0.848	0.916	1.195	1.322	0.831
1.5	0.871	0.948	1.288	1.407	1.470
2.0	0.896	0.966	1.392	1.711	1.782

HMF D-MTF, GM + PM → GE

	GM (α_1), PM (α_2)				
GE (α_0)	0.25	0.5	1.0	1.5	2.0
0.25	0.866	0.906	0.941	0.950	0.954
0.5	0.874	0.870	0.914	0.941	0.945
1.0	0.933	0.985	0.940	0.942	0.961
1.5	0.959	0.958	1.065	1.026	1.000
2.0	0.957	1.080	1.211	1.145	1.147

HMF D-MF, GE + GM → PM

	GE (α_1), GM (α_2)				
PM (α_0)	0.25	0.5	1.0	1.5	2.0
0.25	0.859	0.799	0.795	0.799	0.798
0.5	0.811	0.812	0.783	0.783	0.784
1.0	0.789	0.814	0.987	0.788	0.783
1.5	0.784	0.809	1.070	1.247	0.870
2.0	0.798	0.835	1.111	1.280	1.255

HMF D-MTF, GE + GM → PM

	GE (α_1), GM (α_2)				
PM (α_0)	0.25	0.5	1.0	1.5	2.0
0.25	0.862	0.893	0.943	0.946	0.947
0.5	0.841	0.885	0.898	0.943	0.944
1.0	0.852	0.848	1.012	0.885	0.904
1.5	0.876	0.866	0.900	1.080	0.904
2.0	0.884	0.881	0.900	1.096	1.099

HMF D-MF, GE + PM → GM

	GE (α_1), PM (α_2)				
GM (α_0)	0.25	0.5	1.0	1.5	2.0
0.25	0.790	0.708	0.703	0.697	0.701
0.5	0.724	0.670	0.687	0.688	0.685
1.0	0.698	0.657	0.659	0.670	0.685
1.5	0.698	0.655	0.667	0.657	0.666
2.0	0.689	0.668	0.671	0.676	0.665

HMF D-MTF, GE + PM → GM

	GE (α_1), PM (α_2)				
GM (α_0)	0.25	0.5	1.0	1.5	2.0
0.25	0.773	0.775	0.851	0.854	0.861
0.5	0.761	0.746	0.774	0.818	0.850
1.0	0.782	0.757	0.754	0.786	0.779
1.5	0.832	0.752	0.745	0.783	0.795
2.0	0.836	0.811	0.802	0.791	0.805

Table 5.5 Spearman correlation between values of each of the drug sensitivity (top) and methylation (bottom) dataset pairs.

Drug sensitivity	GDSC IC_{50}	CTRP EC_{50}	CCLE IC_{50}	CCLE EC_{50}
GDSC IC_{50}	-	0.47	0.59	0.39
CTRP EC_{50}	0.47	-	0.44	0.45
CCLE IC_{50}	0.59	0.44	-	0.65
CCLE EC_{50}	0.39	0.45	0.65	-

Methylation	GE	GM	PM
GE	-	-0.07	-0.12
GM	-0.07	-	0.14
PM	-0.12	0.14	-

using matrix tri-factorisation for all (HMF D-MTF), using matrix factorisation on one dataset and tri-factorisation for the other three, or using matrix factorisation on two datasets and tri-factorisation on two as well. Note that applying matrix tri-factorisation on only one dataset is equivalent to using matrix factorisation on all four (since the second factor matrix is not shared with any other dataset). In this section we explore some of these choices.

We computed the Spearman correlation of values in each of the pairs of the drug sensitivity datasets (taking only the overlapping entries per pair), and similarly for the gene expression and methylation datasets. These are given in Table 5.5, where we see that the former are very highly correlated, whereas the latter have little to no correlation. As a result, we are expecting that for the drug sensitivity data, multiple matrix tri-factorisation should often do best, and for the methylation data multiple matrix factorisation. However, sometimes a hybrid approach might do better.

To investigate this, we measured the 10-fold cross-validation performance, similar to Section 5.5.1. We tried all variations of combinations between matrix factorisation and tri-factorisation, and for simplicity used $K_t = 10$, K-means initialisation for the shared factor matrices ($\boldsymbol{F}_t$) and least squares for the private ones ($\boldsymbol{G}_l, \boldsymbol{S}_n$). For the drug sensitivity data we used $\alpha = 1.0$, and for the methylation $\alpha = 0.5$. Whenever we used matrix factorisation we shared the row factors (corresponding to drugs).

The results for the drug sensitivity data are given in Table 5.6. The multiple matrix tri-factorisation approach oftentimes achieves the best performance, as expected.

Table 5.6 Performances of in- and out-of-matrix cross-validation results on the drug sensitivity (top table) and methylation (bottom) datasets, where we vary the factorisation types on each of the datasets. The best performances are highlighted in bold.

Drug sensitivity – Factorisation type				Performance			
GDSC	CTRP	CCLE IC_{50}	CCLE EC_{50}	GDSC	CTRP	CCLE IC_{50}	CCLE EC_{50}
R	R	R	R	0.0767	**0.0889**	**0.0537**	0.1071
D	R	R	R	0.0765	0.0901	0.0546	0.1098
R	D	R	R	**0.0758**	0.0909	0.0543	0.1079
R	R	D	R	0.0765	0.0890	0.0584	**0.1055**
R	R	R	D	0.0768	0.0893	**0.0537**	0.1078
D	D	R	R	0.0763	0.0899	0.0569	0.1079
D	R	D	R	0.0766	0.0901	0.0553	0.1090
D	R	R	D	0.0769	0.0898	0.0554	0.1064
R	D	D	R	0.0765	0.0901	0.0566	0.1060
R	D	R	D	0.0772	0.0906	0.0543	0.1082
R	R	D	D	0.0771	0.0892	0.0542	0.1076
D	D	D	D	0.0776	0.0910	0.0562	0.1064

Methylation – Factorisation type			Performance		
GE	GM	PM	GE	GM	PM
R	R	R	0.876	0.744	0.864
D	R	R	0.877	0.717	0.949
R	D	R	1.128	0.698	0.960
R	R	D	**0.860**	0.692	0.834
D	D	D	0.869	**0.663**	**0.799**

Sometimes a hybrid combination of factorisation can lead to improvements, although we could not determine a rule of thumb for this (for example based on the overlap or correlation between datasets). By trying out multiple candidates a suitable hybridity can be found. For the methylation data the results are also given in Table 5.6. As can be seen, using multiple matrix factorisation ($\boldsymbol{D}$ for all matrices) gives the best performance most of the time, which is unsurprising since the three datasets are so weakly correlated.

Recommendation: For dissimilar datasets, with low correlation (like the methylation data), it is best to use multiple matrix factorisation. When the datasets are very similar, with high correlation (like the drug sensitivity data), matrix tri-factorisation can give better results. These two models will generally give very good performance already. One of the hybrid combinations of matrix factorisation and tri-factorisation

can sometimes lead to even better results. Nested cross-validation can be used to find the best hybrid combination.

5.7 Conclusion

In this chapter we studied how matrix factorisation can be used for jointly decomposing multiple datasets. We have presented a fully Bayesian model for data integration, based on a hybrid of nonnegative, semi-nonnegative, and real-valued matrix factorisation and tri-factorisation models. The general nature of this model allows it to easily integrate many datasets across different entity types, including repeated experiments, similarity matrices, and very sparse datasets.

We demonstrated the model on two biological applications. On four drug sensitivity datasets we obtained significant in-matrix prediction improvements compared to state-of-the-art matrix factorisation and machine learning methods. Our data fusion approach based on multiple matrix tri-factorisation (HMF D-MTF) is particularly powerful, achieving the best performance on three of the four datasets. We also showed that our proposed model can provide consistently better predictions on very sparse datasets, outperforming all other matrix factorisation models. Additionally, we integrated methylation and gene expression data in an out-of-matrix prediction setting. Here the approach based on multiple matrix factorisation (HMF D-MF) proved to be very powerful, beating all state-of-the-art machine learning methods on two of the three datasets. The approaches using multiple matrix tri-factorisation and similarity datasets are also promising. Furthermore, we explored the best initialisation approach, the effectiveness of the automatic relevance determination in the data integration setting, the best importance value settings, and explored the trade-offs between the different factorisation type choices.

The experiments show that Bayesian matrix factorisation is a very powerful paradigm for predicting missing values in both the in- and out-of-matrix settings. The multiple matrix tri-factorisation approach has not been explored much in the literature, but for integrating similar datasets this is a very promising area. More research is necessary to exactly determine what factorisation type is appropriate when, and in particular which hybrid combination of multiple matrix factorisation and tri-factorisation to use. This will be particularly interesting when integrating a large number of datasets, which will be increasingly more common in this age of big data.

Chapter 6

Conclusion

The aim of this thesis is to demonstrate the power of Bayesian matrix factorisation models for predicting missing values in real-world datasets, as well as exploring the trade-offs between the different modelling choices: the inference approach, prior and likelihood choices, and factorisation types. We have achieved the above goals through the development of new models and algorithms, as well as carefully reviewing the literature and providing new comparisons on real-world datasets. We provide three main contributions:

(a) **Trade-offs between inference approaches**. Starting with a Bayesian nonnegative matrix factorisation model, we reviewed three inference approaches based on (non-probabilistic) multiplicative updates, Gibbs sampling, and iterated conditional modes. We added another fully Bayesian approach that uses variational Bayesian inference, which was new for this model. We also extended the model to Bayesian nonnegative matrix tri-factorisation, deriving the four inference algorithms, and added the automatic relevance determination prior for both of these models to help with model selection. We then extensively compared the inference approaches for both models on synthetic and four drug sensitivity datasets, comparing their convergence and runtime speeds, cross-validation performances, robustness to noise and sparsity, and how effective they are at model selection.

These experiments demonstrated that the fully Bayesian approaches are more robust to noise and sparsity and provide better cross-validation performances,

but at the cost of inference speed. Furthermore, the automatic relevance determination prior helps with model selection, but does not provide more robustness to noise and sparsity. The models and experiments were implemented in Python and are publicly available at https://github.com/ThomasBrouwer/BNMTF_ARD.

(b) **Trade-offs between prior and likelihood choices**. Next we considered the Bayesian prior and likelihood distribution choices. We reviewed a total of sixteen models from the literature with different likelihoods, priors, and hierarchical priors; and grouped them into four categories: real-valued, nonnegative, semi-nonnegative, and Poisson models. Considering three different applications (drug sensitivity, movie rating predictions, and methylation profiles) we compared the four groups of methods on the same areas as we did for the inference approaches.

We discovered that on the small datasets considered, Poisson-likelihood approaches are not competitive with the other three groups, which goes against claims in the literature where they are much better on larger datasets. We also found that the nonnegative models are more constrained than the real-valued ones, which can help at high sparsity levels—although this difference can be bridged by using the right hierarchical prior. The semi-nonnegative models were simply equivalent to the real-valued ones. Finally, some hierarchical models made no difference at all. The Python code and experiments can be found at https://github.com/ThomasBrouwer/BMF_Priors.

(c) **Data integration using multiple matrix factorisation**. Finally, we introduced a new Bayesian hybrid matrix factorisation model, which can be used for integrating multiple datasets by jointly factorising them. In particular, our model hybridly combines matrix factorisation and tri-factorisations by allowing the user to specify which factor matrices should be shared between the factorisations, as well as specifying for each matrix whether the factor values should be real-valued or nonnegative—hence combining the first three model groups from the priors and likelihood chapter.

We then compared the predictive performance of this new model with that of other non-probabilistic and Bayesian matrix factorisation models, as well as state-of-the-art machine learning approaches—linear regression, random forests, and support vector machines. In both in- and out-of-matrix prediction experiments we showcased improvements and outperformed competing methods. The multiple matrix tri-factorisation approach is especially interesting for datasets that are

very similar, such as repeated experiments. As before, all code, experiments, and datasets are available at https://github.com/ThomasBrouwer/HMF.

6.1 Future work

The work undertaken in this thesis can be extended in several interesting ways.

(a) **Priors and likelihood choices for bigger datasets**. As discussed before, Poisson-likelihood methods are generally claimed to be a more efficient and accurate approach for factorising very large datasets. In the experiments of Chapter 4 we only considered relatively small datasets, where we found the opposite to be true. It would be interesting to see how the four groups of methods perform on datasets of varying sizes, and whether we can draw general lessons about when one approach might be better than the others.

(b) **Factor analysis**. Matrix factorisation models are typically used either to predict missing values, or to find patterns in the datasets by looking at the factor matrices. In this thesis we decided to largely focus only on the former, because this is already a huge research area. However, it would be interesting to see how the inference, prior, and likelihood choices affect our ability to find clusters and other patterns in the data. We briefly considered the effect of the prior and likelihood choice on the distribution of factor values in Chapter 4, and saw that some priors cause the factor values to be more similar, but there is a lot of room for further experimentation in this direction.

(c) **Best hybrid multiple matrix factorisations**. In Chapter 5 we explored whether a hybrid combination of matrix factorisations and tri-factorisations could give better predictions than using the same type for all datasets. As the number of datasets increase, this choice can become more important. We were not able to determine a rule of thumb based on the overlap and correlation between the datasets that could help with this choice, but we hope that future research will shed light on this problem.

(d) **Bicluster analysis using multiple matrix factorisation**. The hybrid matrix factorisation model introduced in Chapter 5 could also be very useful at finding biclusters in datasets. Multiple matrix factorisation approaches only share one factor matrix at a time, but for similar datasets it makes sense to share more of

the factor values, and multiple matrix tri-factorisation allows us to do this. We already saw that this can lead to predictive improvements; we believe that it can also help us better identify biclusters in the datasets. For example for the gene expression and methylation datasets, it could help us identify a subset of patients for which only a subset of genes are differentially expressed. To better identify these clusters, the model would need to be modified with more sparsity-inducing priors for the middle (biclustering) matrix.

References

Adams, C. P. and Brantner, V. V. (2006). Estimating the cost of new drug development: is it really 802 million dollars? *Health affairs (Project Hope)*, 25(2):420–8.

Ammad-ud din, M., Georgii, E., Gönen, M., Laitinen, T., Kallioniemi, O., Wennerberg, K., Poso, A., and Kaski, S. (2014). Integrative and personalized QSAR analysis in cancer by kernelized Bayesian matrix factorization. *Journal of chemical information and modeling*, 54(8):2347–59.

An, S., Yoo, J., and Choi, S. (2010). Manifold-respecting probabilistic matrix tri-factorization. In *Proceedings of the 2010 IEEE International Workshop on Machine Learning for Signal Processing, MLSP 2010*, pages 24–28. IEEE.

Arngren, M., Schmidt, M. N., and Larsen, J. (2011). Unmixing of Hyperspectral images using bayesian non-negative matrix factorization with volume prior. In *Journal of Signal Processing Systems*, volume 65, pages 479–496. IEEE.

Asuncion, A., Welling, M., Smyth, P., and Teh, Y. W. (2009). On smoothing and inference for topic models. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*.

Barretina, J., Caponigro, G., Stransky, N., Venkatesan, K., Margolin, A. A., Kim, S., Wilson, C. J., et al. (2012). The Cancer Cell Line Encyclopedia enables predictive modelling of anticancer drug sensitivity. *Nature*, 483(7391):603–7.

Beal, M. and Ghahramani, Z. (2003). The Variational Bayesian EM Algorithm for Incomplete Data: with Application to Scoring Graphical Model Structures. *Bayesian Statistics 7, Oxford University Press*.

Bennett, J. and Lanning, S. (2007). The netflix prize. In *Proceedings of the KDD Cup Workshop 2007*, pages 3–6. ACM.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2012). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3(4-5):993–1022.

Booth, B. and Zemmel, R. (2004). Prospects for productivity. *Nature reviews. Drug discovery*, 3(5):451–6.

Bouchard, G., Yin, D., and Guo, S. (2013). Convex collective matrix factorization. In *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 31, pages 144–152.

Brunet, J. P., Golub, T. R., Tamayo, P., and Mesirov, J. P. (2004). Metagenes and molecular pattern discovery using matrix factorization. *Proceedings of the National Academy of Sciences*, 101(12):4164–4169.

Chatzis, S. P. (2014). Dynamic Bayesian Probabilistic Matrix Factorization. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, pages 1731–1737.

Chu, W. and Ghahramani, Z. (2009). Probabilistic models for incomplete multi-dimensional arrays. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Costello, J. C., Heiser, L. M., Georgii, E., Gönen, M., Menden, M. P., Wang, N. J., Bansal, M., Ammad-ud din, M., Hintsanen, P., Khan, S. A., et al. (2014). A community effort to assess and improve drug sensitivity prediction algorithms. *Nature Biotechnology*, 32(12):1202–1212.

De Niz, C., Rahman, R., Zhao, X., and Pal, R. (2016). Algorithms for Drug Sensitivity Prediction. *Algorithms*, 9(4):77.

Dimasi, J. A. (2001). New drug development in the United States from 1963 to 1999. *Clinical pharmacology and therapeutics*, 69(5):286–96.

Ding, C., He, X., and Simon, H. D. (2005a). On the equivalence of nonnegative matrix factorization and spectral clustering. In *SIAM International Conference on Data Mining*.

Ding, C., He, X., and Simon, H. D. (2005b). On the Equivalence of Nonnegative Matrix Factorization and Spectral Clustering. In *Proceedings of the fifth SIAM International Conference on Data Mining (SDM)*, pages 606–610.

Ding, C., Li, T., and Jordan, M. I. (2010). Convex and semi-nonnegative matrix factorizations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:45–55.

Ding, C., Li, T., and Peng, W. (2008). On the equivalence between Non-negative Matrix Factorization and Probabilistic Latent Semantic Indexing. *Computational Statistics and Data Analysis*, 52(8):3913–3927.

Ding, C., Li, T., Peng, W., and Park, H. (2006). Orthogonal nonnegative matrix t-factorizations for clustering. In *Proceedings of the 12th ACM SIGKDD*.

Dong, Z., Zhang, N., Li, C., Wang, H., Fang, Y., Wang, J., and Zheng, X. (2015). Anticancer drug sensitivity prediction in cell lines from baseline gene expression through recursive feature selection. *BMC Cancer*, 15(1):489.

Eicher, T. S., Papageorgiou, C., and Raftery, A. E. (2011). Default priors and predictive performance in Bayesian model averaging, with application to growth determinants. *Journal of Applied Econometrics*, 26(1):30–55.

Figueiredo, M. and Jain, A. (2002). Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):381–396.

Gligorijević, V. and Pržulj, N. (2015). Methods for biological data integration: perspectives and challenges. *Journal of the Royal Society, Interface*, 12(112).

Golub, G. H. and Reinsch, C. (1970). Singular value decomposition and least squares solutions. *Numerische Mathematik*, 14(5):403–420.

Gönen, M. (2012). Predicting drug-target interactions from chemical and genomic kernels using Bayesian matrix factorization. *Bioinformatics*, 28(18).

Gönen, M. and Kaski, S. (2014). Kernelized Bayesian Matrix Factorization. *IEEE transactions on pattern analysis and machine intelligence*, 36(10):2047–60.

Gonzalez-Perez, A., Perez-Llamas, C., Deu-Pons, J., Tamborero, D., Schroeder, M. P., Jene-Sanz, A., Santos, A., and Lopez-Bigas, N. (2013). IntOGen-mutations identifies cancer drivers across tumor types. *Nature Methods*, 10(11):1081–1082.

Gopalan, P. and Blei, D. M. (2014). Bayesian Nonparametric Poisson Factorization for Recommendation Systems. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 33, pages 2–4.

Gopalan, P., Hofman, J. M., and Blei, D. M. (2015). Scalable recommendation with hierarchical Poisson factorization. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, pages 326–335. AUAI Press.

Harper, F. M. and Konstan, J. A. (2015). The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems*, 5(4):1–19.

Harshman, R. A. (1970). Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multimodal factor analysis. *UCLA Working Papers in Phonetics*, 16(10):1– 84.

Harshman, R. A. (1972). PARAFAC2: Mathematical and technical notes. *UCLA working papers in phonetics*, 22(10):30–44.

Hayashi, K., Hirayama, J.-i., and Ishii, S. (2009). Dynamic Exponential Family Matrix Factorization. pages 452–462.

Hoff, P. D. (2013). Equivariant and scale-free Tucker decomposition models. *arXiv preprint at arXiv:1312.6397*.

Hofmann, T. (1999). Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on research and development in information retrieval*, pages 50–57, New York, New York, USA. ACM Press.

Hu, C., Rai, P., and Carin, L. (2015). Zero-Truncated Poisson Tensor Factorization for Massive Binary Tensors. In *Uncertainty in Artificial Intelligence (UAI)*.

Hwang, T., Atluri, G., Xie, M., Dey, S., Hong, C., Kumar, V., and Kuang, R. (2012). Co-clustering phenome-genome for phenotype classification and disease gene discovery. *Nucleic Acids Research*, 40(19):e146.

Jang, I. S., Neto, E. C., Guinney, J., Friend, S. H., and Margolin, A. A. (2014). Systematic assessment of analytical methods for drug sensitivity prediction from cancer cell line data. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, pages 63–74.

Jing, L., Wang, P., and Yang, L. (2015). Sparse Probabilistic Matrix Factorization by Laplace Distribution for Collaborative Filtering. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*.

Kaufman, G. M. and Press, S. J. (1973). Bayesian factor analysis.

Khan, S. A., Leppäaho, E., and Kaski, S. (2016). Bayesian multi-tensor factorization. *Machine Learning*, 105(2):233–253.

Kim, Y.-D. and Choi, S. (2007). Nonnegative Tucker Decomposition. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE.

Kim, Y.-d. and Choi, S. (2014). Scalable Variational Bayesian Matrix Factorization with Side Information. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 33.

Klami, A., Bouchard, G., and Tripathi, A. (2014). Group-sparse Embeddings in Collective Matrix Factorization. In *Proceedings of the 2nd International Conference on Learning Representations*.

Klami, A., Virtanen, S., and Kaski, S. (2010). Bayesian exponential family projections for coupled data sources. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*.

Koboldt, D. C., Fulton, R. S., McLellan, M. D., Schmidt, H., Kalicki-Veizer, J., McMichael, J. F., Fulton, L. L., et al. (2012). Comprehensive molecular portraits of human breast tumours. *Nature*, 490(7418):61–70.

Kong, D., Ding, C., and Huang, H. (2011). Robust nonnegative matrix factorization using l21-norm. In *Proceedings of the 20th ACM international conference on Information and knowledge management (CIKM)*, pages 673–682, New York, New York, USA. ACM Press.

Kuang, D., Ding, C., and Park, H. (2012). Symmetric Nonnegative Matrix Factorization for Graph Clustering. In *SIAM International Conference on Data Mining (SDM)*, pages 494–505.

Kundaje, A., Meuleman, W., Ernst, J., Bilenky, M., Yen, A., Heravi-Moussavi, A., Kheradpour, P., et al. (2015). Integrative analysis of 111 reference human epigenomes. *Nature*, 518(7539):317–330.

Lakshminarayanan, B., Bouchard, G., and Archambeau, C. (2011). Robust bayesian matrix factorisation. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 15, pages 425–433.

Lawson, C. L. and Hanson, R. J. (1995). *Solving least squares problems*, volume 15.

Lee, C. M., Mudaliar, M. a. V., Haggart, D. R., Wolf, C. R., Miele, G., Vass, J. K., Higham, D. J., and Crowther, D. (2012). Simultaneous non-negative matrix factorization for multiple large scale gene expression datasets in toxicology. *PloS one*, 7(12):e48238.

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

Lee, D. D. and Seung, H. S. (2000). Algorithms for Non-negative Matrix Factorization. *NIPS, MIT Press*, pages 556–562.

Ley, E. and Steel, M. F. (2009). On the effect of prior assumptions in Bayesian model averaging with applications to growth regression. *Journal of Applied Econometrics*, 24(4):651–674.

Li, S., Hou, X. W., Zhang, H. J., and Cheng, Q. S. (2001). Learning spatially localized, parts-based representation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 1–6. IEEE Comput. Soc.

Li, T. (2010). Bridging Domains with Words : Opinion Analysis with Matrix. In *Proceedings of the 10th SIAM International Conference on Data Mining*, pages 293–302.

Li, T., Zhang, Y., and Sindhwani, V. (2009). A non-negative matrix tri-factorization approach to sentiment classification with lexical prior knowledge. *Proceeding of the 47th Annual Meeting of the Association for Computational Linguistics*.

Lippert, C., Weber, S., and Huang, Y. (2008). Relation prediction in multi-relational domains using matrix factorization. In *NIPS workshop on structured input, structured output*.

Liu, Y., Gu, Q., Hou, J. P., Han, J., and Ma, J. (2014). A network-assisted co-clustering algorithm to discover cancer subtypes based on gene expression. *BMC bioinformatics*, 15(1):37.

Masood, A., Pan, W., and Doshi-Velez, F. (2016). An Empirical Comparison of Sampling Quality Metrics: A Case Study for Bayesian Nonnegative Matrix Factorization. *arXiv preprint at arXiv:1606.6250*.

Mayekawa, S. (1985). Bayesian factor analysis. Technical report, IOWA UNIV IOWA CITY.

Menden, M. P., Iorio, F., Garnett, M., McDermott, U., Benes, C. H., Ballester, P. J., and Saez-Rodriguez, J. (2013). Machine Learning Prediction of Cancer Cell Sensitivity to Drugs Based on Genomic and Chemical Properties. *PLoS ONE*, 8(4):e61318.

Paatero, P. (1997). Least squares formulation of robust non-negative factor analysis. *Chemometrics and Intelligent Laboratory Systems*, 37(1):23–35.

Paatero, P. and Tapper, U. (1994). Positive Matrix Factorization - A Nonnegative Factor Model with Optimal Utilization of Error Estimates of Data Values. *Environmetrics*, 5(2):111–126.

Pal, R., Berlow, N., and Haider, S. (2012). Anticancer drug sensitivity analysis: An integrated approach applied to Erlotinib sensitivity prediction in the CCLE database. In *Proceedings 2012 IEEE International Workshop on Genomic Signal Processing and Statistics (GENSIPS)*, pages 9–12. IEEE.

Paquet, U., Thomson, B., and Winther, O. (2012). A hierarchical model for ordinal matrix factorization. *Statistics and Computing*, 22(4):945–957.

Pauca, V., Shahnaz, F., Berry, M., and Plemmons, R. (2004). Text mining using non-negative matrix factorizations. In *Proceedings SIAM International Conference on Data Mining (SDM)*, pages 452–456.

Pauca, V. P., Piper, J., and Plemmons, R. J. (2006). Nonnegative matrix factorization for spectral data analysis. *Linear Algebra and Its Applications*, 416(1):29–47.

Remes, S., Mononen, T., and Kaski, S. (2015). Classification of weak multi-view signals by sharing factors in a mixture of Bayesian group factor analyzers. *NIPS Workshop on Machine Learning and Interpretation in Neuroimaging (MLINI)*.

Salakhutdinov, R. and Mnih, A. (2008). Bayesian Probabilistic Matrix Factorization using Markov Chain Monte Carlo. In *International Conference on Machine Learning (ICML)*, pages 880–887, New York, New York, USA. ACM Press.

Salimans, T., Kingma, D. P., and Welling, M. (2015). Markov Chain Monte Carlo and Variational Inference: Bridging the Gap. In *Proceedings of the 32nd International Conference on Machine Learning*.

Schmidt, M. N. and Mohamed, S. (2009). Probabilistic non-negative tensor factorization using Markov chain Monte Carlo. In *17th European Signal Processing Conference*.

Schmidt, M. N., Winther, O., and Hansen, L. K. (2009). Bayesian non-negative matrix factorization. In *International Conference on Independent Component Analysis and Signal Separation, Springer Lecture Notes in Computer Science, Vol. 5441*.

Seashore-Ludlow, B., Rees, M. G., Cheah, J. H., Cokol, M., Price, E. V., Coletti, M. E., Jones, V., et al. (2015). Harnessing Connectivity in a Large-Scale Small-Molecule Sensitivity Dataset. *Cancer discovery*, 5(11):1210–23.

Seichepine, N., Essid, S., Févotte, C., and Cappé, O. (2013). Soft nonnegative matrix co-factorization with application to multimodal speaker diarization. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3537–3541. IEEE.

Shen, B. and Si, L. (2010). Nonnegative Matrix Factorization Clustering on Multiple Manifolds. In *AAAI Conference on Artificial Intelligence*, pages 575–580.

Singh, A. P. and Gordon, G. J. (2008). Relational learning via collective matrix factorization. In *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 650, New York, New York, USA. ACM Press.

Tan, V. Y. F. and Févotte, C. (2013). Automatic relevance determination in nonnegative matrix factorization with the (β)-divergence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(7):1592–1605.

Tucker, L. R. (1966). Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311.

Virtanen, S., Klami, A., and Kaski, S. (2011). Bayesian CCA via Group Sparsity. In *Proceedings of the 28th International Conference on Machine Learning*.

Virtanen, S., Klami, A., Khan, S., and Kaski, S. (2012). Bayesian group factor analysis. In *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Žitnik, M. and Zupan, B. (2015). Data Fusion by Matrix Factorization. *IEEE transactions on pattern analysis and machine intelligence*, 37(1):41–53.

Wang, F., Li, T., and Zhang, C. (2008). Semi-supervised clustering via matrix factorization. In *Proceedings of the 2008 SIAM International Conference on Data Mining*.

Wang, H., Huang, H., Ding, C., and Nie, F. (2013). Predicting Protein–Protein Interactions from Multimodal Biological Data Sources via Nonnegative Matrix Tri-Factorization. *Journal of Computational Biology*, 20(4):344–358.

Wang, H.-Q., Zheng, C.-H., and Zhao, X.-M. (2015a). jNMFMA: a joint non-negative matrix factorization meta-analysis of transcriptomics data. *Bioinformatics*, 31(4):572–80.

Wang, L., Li, X., Zhang, L., and Gao, Q. (2017). Improved anticancer drug response prediction in cell lines using matrix factorization with similarity regularization. *BMC Cancer*, 17(1):513.

Wang, Z., Yuan, W., and Montana, G. (2015b). Sparse multi-view matrix factorization: a multivariate approach to multiple tissue comparisons. *Bioinformatics*, 31(19):3163–71.

Welling, M. and Weber, M. (2001). Positive tensor factorization. *Pattern Recognition Letters*, 22(12):1255–1261.

Xu, Z., Yan, F., Yuan, and Qi (2012). Infinite Tucker Decomposition: Nonparametric Bayesian Models for Multiway Data Analysis. In *In Proceedings of the 29th International Conference on Machine Learning (ICML)*.

Yang, W., Soares, J., Greninger, P., Edelman, E. J., Lightfoot, H., Forbes, S., Bindal, N., et al. (2013). Genomics of Drug Sensitivity in Cancer (GDSC): a resource for therapeutic biomarker discovery in cancer cells. *Nucleic acids research*, 41(Database issue):D955–61.

Yang, Y. and Dunson, D. B. (2015). Bayesian Conditional Tensor Factorizations for High-Dimensional Classification. *Journal of the American Statistical Association*.

Yoo, J. and Choi, S. (2009). Probabilistic matrix tri-factorization. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*.

Yılmaz, Y. K. and Cemgil, A. T. (2010). Probabilistic Latent Tensor Factorization. In Vigneron, V., Zarzoso, V., Moreau, E., Gribonval, R., and Vincent, E., editors, *LVA/ICA 2010*, Lecture Notes in Computer Science, Berlin, Heidelberg. Springer Berlin Heidelberg.

Zhang, D. Q., Chen, S. C., and Zhou, Z. H. (2005). Two-dimensional non-negative matrix factorization for face representation and recognition. In *Analysis and Modelling of Faces and Gestures*, volume 3723, pages 350–363.

Zhang, X., Zhao, L., Zong, L., Liu, X., and Yu, H. (2014). Multi-view Clustering via Multi-manifold Regularized Nonnegative Matrix Factorization. In *IEEE International Conference on Data Mining*, pages 1103–1108. IEEE.

Zhang, Y. and Yeung, D.-Y. (2012). Overlapping community detection via bounded nonnegative matrix tri-factorization. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 606, New York, New York, USA. ACM Press.

Zhao, Q., Zhang, L., and Cichocki, A. (2015). Bayesian Sparse Tucker Models for Dimension Reduction and Tensor Completion. *arXiv preprint at arXiv:1505.02343*.

Zhe, S., Xu, Z., Chu, X., Qi, Y., and Park, Y. (2015). Scalable nonparametric multiway data analysis. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Zhu, S., Yu, K., Chi, Y., and Gong, Y. (2007). Combining content and link for classification using matrix factorization. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, page 487, New York, New York, USA. ACM Press.