technische universität
dortmund

Master's thesis

Obtaining trustable results for Probabilistic Boolean Matrix Factorization using SGD/MCMC

Felix Gonsior

März 2019

Supervisors:
Prof. Dr. Katharina Morik
Dr. Nico Piatkowski

Technische Universität Dortmund
Fakultät für Informatik
Lehrstuhl für Künstliche Intelligenz (LS-8)
www-ai.cs.tu-dortmund.de

Contents

Chapter 1

Introduction

1.1 Motivation

The use of machine learning based intelligent methods has seen a surge in the past years in science, in business and in daily life. Due to the increased use of digitalization and the related trend of recording as much data as is available, datasets and models have grown towards sizes where their inner workings cannot be easily understood even though a mathematical, formal description exists. Datasets might be badly curated due to their size and extent and users might be relatively inexperienced.

Nevertheless, even in such situations machine learning based methods are used to derive decisions which are expected to be correct as well as understandable and thus actionable. However with erroneous data and user error, unexpected results arise more often than not. In these cases the first question asked is "Why did the method deliver decisions in the way it did?" Depending on the complexity of data and model as well as the devotion of the user to a full analysis the question might stay unanswered. However, depending on the severity of the misdecision, the trust in the method or machine learning methods in general may be broken.

As more of these cases concern topics of public interest, more of them lead to public outcry, diminishing the general trust in machine learning and related scientific disciplines. Recent notable cases of this phenomenon are the debate about racist A.I., the killer robots debate as well as issues about life or death decisions pertaining to automated decision making in autonomous driving.

While the above is not meant as a critique of the public reporting or the public reaction to such cases of "faulty A.I.", it makes clear that the credible use of machine learning methods needs more than blind trust in data, method and operator. But how can we trust modern machine learning methods in an informed way, if we are not able to gain useful insights either due to the sheer size of the data they operate on or due to a noninformative model formulation as for example in neural networks?

One possible way is the use of probabilistic models. These models assign an uncertainty factor to each output, delivering with each result a self contained diagnostic of their decisions. Based on this diagnostic a trained operator can evaluate the quality of a machine learning model for a given task and often locate the source of issues in decisions or even in the data. Based on these insights, the method can be improved or the data corrected.

In this thesis we will examine Boolean Matrix Factorization (BMF), which together with the related method Nonnegative Matrix Factorization (NMF) has seen widespread use in many different disciplines, as diverse as genetics, data mining, recommender systems, signal analysis and text mining. In all these use cases, BMF and NMF are used to discover hidden structure in a given database. This is achieved by finding a representation of the database which is reduced in size, but from which the original database can be reconstructed.

Even though the basic mathematical description for these models is short and concise, the implied interactions between model variables are nontrivial. A model diagnostic as provided by a probabilistic model could be a helpful tool for understanding the results of BMF or NMF. While there exist a wealth of probabilistic solutions for NMF, BMF is generally still solved with optimization methods and only a few probabilistic approaches exist.

Probabilistic approaches do not only differ in the model formulation but also in the inference method. While inference by optimization has found a wider application in the data mining community, sampling based approximation methods for probability distributions such as Markov-Chain Monte Carlo (MCMC) provide the means necessary to obtain probabilistic results. They have been used extensively in statistics and also in physics, as these disciplines regularly deal with the analysis of probabilistic phenomena.

As such, there is a transfer of knowledge between these disciplines and in recent years connections between physical simulation methods and statistical inference have been researched. Based on this knowledge transfer, very recently a connection between sampling based inference and optimization has been made, connecting the famous Stochastic Gradient Descent optimization method to the Markov Chain Monte Carlo approach. Using this connection a quite direct transformation from optimization problem to probabilistic model can be applied.

We will use this connection to our advantage in this thesis to create a probabilistic model out of the BMF optimization model. Based on this probabilistic model we expect to obtain a deeper insight into the behavior and the uncertainties associated with BMF.

1.2 Structure of this thesis

Besides this introductory chapter this thesis consists of three main chapters. In chapter 2 we deliver the theoretical prerequisites related to the topic. We begin with a review of

relevant literature. Next we introduce boolean databases, Boolean Matrix Factorization and related concepts. The chapter continues with the topics probabilistic modeling and inference, where we first outline Bayesian models and related distributions. Following, we introduce Markov Chain Monte Carlo sampling and discuss recent developments and connections between research in optimization and in sampling methods. The chapter closes with the discussion of a state of the art sampling method based on a physical model of motion.

Chapter 3 first shows how a probabilistic model can be constructed from an optimization model in a simple way. It continues by discussing the specific implementation of a sampling method for the problem at hand. Further we discuss the unusual structure of the problem compared to usual machine learning problems and derive the necessary adaptations in the sampling method. At the end of the chapter we provide details about the evaluation goals and its process as well as the generation of test data together with the used test datasets.

Chapter 4 analyzes and discusses the results of this thesis. In this chapter different perspectives on the behavior of the sampling process are given. Also, issues and pitfalls which we encountered are discussed. We review the influence of prior information on our results as well as the effects of truncation. We examine the role of uncertainty and the existence of different kinds of uncertainty in our model. An attempt is made to reconstruct parameter matrices from the results obtained by creating samples from different model instances and parameterizations.

The thesis closes with Chapter 5 where we review the work that was done in this thesis, as well as the insights gathered and new questions that have emerged.

1.3 Notation

The following notation is used throughout this document with the given interpretation. Special notation that is used in a limited scope is not listed here. It will be introduced in the respective sections in which it is used.

Expression	Interpretation
$\mathbb{B}$	The binary numbers
$\mathbb{N}$	The natural numbers
$\mathbb{N}_0$	The natural numbers including zero
$\mathbb{R}$	The real numbers
$\mathbb{R}_+$	The positive real numbers
S^n	A column vector over the set S with length $n \in \mathbb{N}$
$S^{m \times n}$	A matrix with m rows and n columns $m, n \in \mathbb{N}$.
$\nabla_x f(x, y)$	(Partial) gradient of f wrt. x
$\lvert\cdot\rvert$	1-Norm on vectors. On matrices Frobenius 1-norm
$\lVert\cdot\rVert$	2-Norm on vectors. On matrices Frobenius 2-norm
$\log x$	The natural logarithm of x
$\exp\{x\}, e^x$	The natural exponential function
$\min(a, b)/\max(a, b)$	The smaller/larger value of a and b
$\min_x f(x)$	The minimizer in x of f
$\arg\min_x f(x)$	The values of x for which f is minimal
$x \bmod y$	The integral modulo function
$\int_{s \in S} f(s) ds$	Integral of f over the set S
$\sum_{s \in S} f(s)$	Discrete sum of f over the set S
$\sum_{i=0}^{n} x_i$	Discrete sum over indexed elements x_i
$\lceil\cdot\rceil$	Ceiling function (the next largest integer value)
$\lfloor\cdot\rfloor$	Floor function (the next smallest integer value)
$p(a, b, c)$	The joint probability distribution over random variables a, b, c
$p(a\|b, c)$	The conditional probability distribution of a given b and c
$L_D(a)$	The likelihood function of a for a constant dataset D
$L(a, D)$	The likelihood function of a for a varying dataset D
$a \propto b$	Expression a is proportional to b
$a \sim \text{Law}(\cdot)$	The random variable a is distributed as $\text{Law}(\cdot)$
$p(a, b) \sim \text{Law}(\cdot)$	The joint distribution of a and b is distributed as $\text{Law}(\cdot)$

Chapter 2

Theoretical prerequisites

2.1 Related work

As Boolean Databases are abound, Boolean Matrix Factorization belongs to the more prominent machine learning algorithms. However multiple issues with the BMF approach require a lot of work from users of existing methods to verify that a given solution can be trusted, if that is at all possible. It has been shown early that with current knowledge no efficient algorithms for BMF can be constructed and no efficient approximation schemes can guarantee a bound on its approximation quality[Mie06]. In addition structural ambiguities in the basic definition of BMF may lead to a situation where there is no clear definition of a correct solution to BMF.

These issues have been recognized in the machine learning community and trustworthy algorithms are being sought after. Here "trustworthy" means that a solution guarantees properties which are generally accepted as useful for obtaining an interpretable result. In consequence, the focus of recent work by Hess et al. on the PAL-Tiling framework lies on integrating trustworthiness into BMF by constraining the space of solutions by generally accepted principles. In one instance they are information theoretical criteria[HMP17] which apply Occams razor through the Minimum Description Length principle[MV14] and focus on structurally simple solutions which can still explain the data. In an other instance Hess et al. consider bounding the False Discovery Rate[HPM18] which is a statistical quality measure for statistical estimators. Hess et. al. apply optimization algorithms which reduce the BMF problem to the related problem of Nonnegative Matrix Factorization[PT94] and are augmented with regularization terms to express the constraints described above.

While Hess et. al are not the only ones interpreting BMF as an optimization problem, the focus of this thesis lies on an alternative interpretation as a probabilistic model. We refer the reader to[HMP17] where a good overview of alternative optimization approaches is given.

The results of Hess et al. certainly contribute a lot towards the development of trustworthy BMF, however users of the method are still left with a single solution and no information about possible alternatives and their validity. To handle this issue, probabilistic models are necessary, as they explore the space of all solutions and label each whole solution as well as parts thereof with a probability. In this way the user is able to diagnose strengths and weaknesses of solutions and inform themselves about the validity of the model, different solutions and possibly also the underlying data.

For this reason we are going to explore Bayesian probabilistic models for BMF in this thesis together with assorted inference algorithms. We are not only motivated by the apparent advantages of probabilistic inference but also by the relative lack of of probabilistic approaches to BMF.

For Boolean Matrix Factorization our literature review has yielded only the recently published OrMachine [RHTY17] as well as a contribution by Meeds et al. with an independent development of BMF [MGNR07]. Both approaches use Bayesian Modelling and sampling to obtain solutions. There also is an early result by Streich et al.[SFBB09] who apply BMF to the task of role mining and use an expectation-maximization algorithm together with an annealing schedule to obtain results. Furthermore there is prior work by Hernandez-Lobato et al. [HHG14] who use an approach based on stochastic gradient optimization. In the related field of tensor factorization there exists some work on probabilistic models for boolean tensors. Rukat et al.[RHY18] have generalized their OrMachine to tensors, but there is also earlier work by Rai et al.[RHHC15], by Kolda and Bader[KB09] as well as Hu et al.[HRC15].

To set these nine contributions into perspective, we take a look at the related method of Nonnegative Matrix Factorization (NMF). Inspired by the prominent paper "Bayesian probabilistic Matrix Factorization [...]"[SM08] there are hundreds of other approaches, indeed to many to list here. We refer the reader to the surveys given in Koren et al.[KBV09], Wang and Zhang [WZ13] and also Shi et al.[SZY17] for an overview of optimization approaches as well as probabilistic approaches to NMF. Incorporating NMF here also serves another purpose, as this method will play an important role in the creation of approximate models for BMF later on.

Another motivating factor to delve into probabilistic models for BMF are recent discoveries about connections between stochastic optimization and probabilistic inference. The original idea of stochastic optimization algorithms was introduced by Robbins and Monro [RM51]: optimize a function despite being given only noisy measurements. In the field of machine learning it has found a prominent application in the Stochastic Gradient Descent (SGD) algorithm for learning from huge datasets[Bot10]. SGD handles only small random parts of the dataset at a time and thus introduces noise in the optimization process. However, in turn the needs in memory and processing power per iteration can be bounded. This idea has been carried over from optimization methods to sampling

methods for probabilistic inference by Welling and Teh[WT11] thereby making sampling methods usable on huge datasets. Recently, Mandt. et al.[MHB17] have elaborated on these results by giving an answer to the open question of the optimal noise scale used with these methods. In combination, these results yield an almost cookie cutter way of developing stochastic inference algorithms from optimization problems.

In the following sections we will introduce the reader to the theory behind the topics outlined above. We begin with a definition of Boolean Databases and give an example for their application. Then we introduce Boolean Matrix Factorization as well as Nonnegative Matrix Factorization. We discuss the issues with the BMF model in its basic formulations and how they may lead to distrust in BMF results. Next, we will discuss solution methods. Here we use the PAL-Tiling framework as an example of how to solve BMF with optimization. We move on to Bayesian probabilistic models and inference methods, where we introduce well known MCMC sampling methods. We will close the chapter by taking a closer look at the above mentioned connection between optimization and sampling.

2.2 Boolean- and Nonnegative Matrix Factorization

Boolean Matrix Factorization (BMF) and the related method Nonnegative Matrix Factorization (NMF) are unsupervised learning methods. This means that they are used to discover hidden structure in a dataset when no input-output relation is given as in supervised learning. Both methods have the property of discovering whole-parts relationships which are semantically meaningful[LS99]. An example for this behavior is given in fig 2.1. In part a) of the figure, NMF was used to learn patterns from a dataset of human faces. The learned patterns correspond to semantically meaningful parts like eyes, mouth and nose. Results obtained with the well known PCA/SVD method are shown in part b) of figure 2.1. While this method also is able to reconstruct the face information, the learned representation is made up of weighted arbitrary parts of the whole image.

In general the behavior of BMF/NMF can be interpreted as clustering or learning of topic models, a task which has applications in the real world reaching far beyond image segmentation.

2.2.1 Boolean Databases

Let us assume we were asked by an online music service to explore the listening behaviors of their users. One basic question we could ask is which different tastes of music exist in the user base. We have obtained a dataset listing for each user which songs they have listened to. The dataset is given to us as a matrix where there is one row for each of the users and one column for each of the available song titles. Whenever a user has listened at least once to a song, there is a one entry in the corresponding row and column of the dataset, else there is a zero entry. Such a dataset is called a Boolean Database.

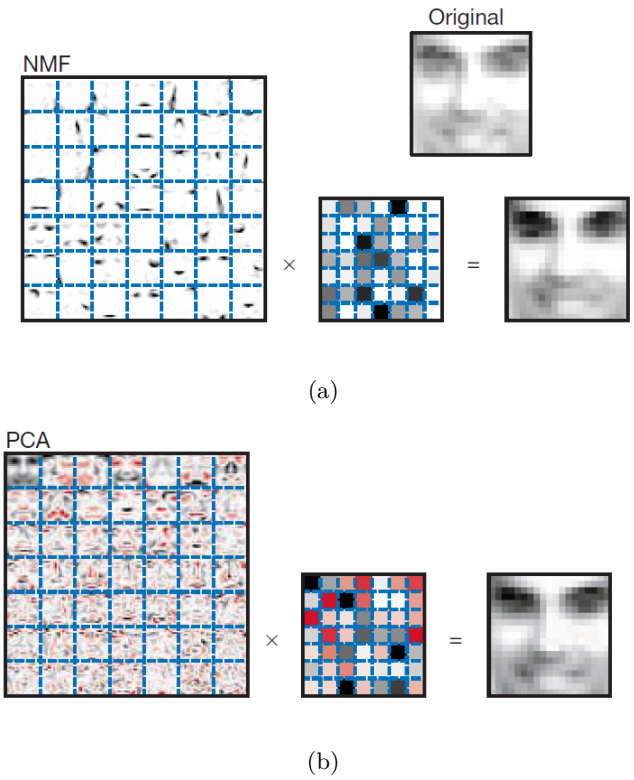

Figure 2.1: a) NMF learns to segment semantically meaningful parts of faces in images. b) PCA learns a weighted representation of the whole image.[LS99]

The rows of this database represent the individual data cases or measurements, the columns represent possible features and the entries record individual presence or absence of features. The image on the left side of figure 2.2 gives an example. There the rows and columns have been ordered in such a way that patterns in the data become visible.

2.2.1 Definition. Boolean Database[HMP17] A boolean database consists of a set of items (features) $\mathcal{I} = \{1, \ldots, n\}$ and a set of transactions (data cases) $\mathcal{T} = \{1, \ldots, m\}$. In matrix form the items are represented by the columns and the transactions are represented by the rows. If an item occurs in a transaction the corresponding entry of the matrix is set to 1. The result is a matrix $D \in \{0,1\}^{m \times n}$. Throughout this thesis we will use the terms *boolean database* or *database* or *data matrix* interchangeably to refer to the definition given.

2.2.2 Definition. Pattern[HMP17] A set of items is called a pattern. If a pattern is a subset of the items occurring in a transaction, the transaction *supports* the pattern.

We might now ask if there are groups of users which have listened to the same songs, or given the definition above, transactions which support the same patterns. If we were able to find such a situation, we could ascribe the same taste of music to a group of users and

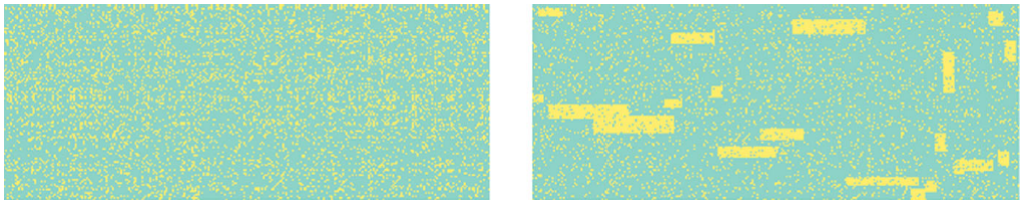

Figure 2.2: An example for a boolean database as given in [HMP17]. On the left the database is in its original visually uninterpretable form. On the right columns and rows have been reordered to show patterns within the data.

as such cluster the set of users by the musical taste they exhibited. Under the assumption that users with a given taste in music mostly listen to similar songs and genres we could also cluster the songs into proposed genres. However from directly looking at the left side of figure 2.2 there is no way to answer those questions. Only after rearranging rows and columns in a certain way, we can see some noisy patterns emerge, as is shown in the image on the right side.

In the example in figure 2.2 it is easy to show a randomized and an ordered version of the database, because the ordering was already known as a result of the generating process. In general however it is not even known if any transactions share patterns. The naive approach is then to enumerate all possible row and column orders. However the exponential runtime of this method is obviously not acceptable for anything other than toy datasets. The key to a solution of this problem lies in transforming the database into a more succinct representation while keeping most of the information of the original. This is exactly the task of Boolean Matrix Factorization.

2.2.2 Boolean Matrix Factorization

Boolean matrix factorization is best envisioned by looking at its generative process. It is assumed that the matrix D has been constructed by the *boolean multiplication* of two factor matrices.

2.2.3 Definition. Boolean Matrix Multiplication[HPM18] Boolean Matrix Multiplication "$\odot$" is defined for matrices over the Boolean semiring $\mathbb{B} = (\{0,1\}, \wedge, \vee)$, i.e. the boolean numbers together with boolean *and* as well as boolean *or*. Given matrices $A \in \mathbb{B}^{m\times r}$ and $B \in \mathbb{B}^{r\times n}$, $r \in \mathbb{N}^+$ we have

$$(A \odot B)_{ij} = \bigvee_{k=1}^{r} a_{ik} \wedge b_{kj} \tag{2.1}$$

2.2.4 Definition. Boolean Matrix Factorization[HPM18] A given data matrix $D \in \mathbb{B}^{m\times n}$ is assumed to consist of factor matrices $X \in \mathbb{B}^{n\times r}$ and $Y \in \mathbb{B}^{m\times r}$. The result of the

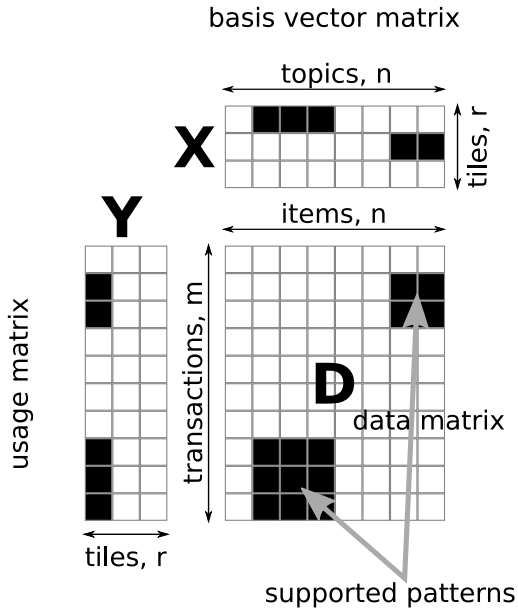

Figure 2.3: The relation of data matrix, usage matrix and basis vector matrix

multiplication is assumed to be distorted by unknown positive and negative noise terms $N_+, N_- \in B^{m \times n}$ as follows (here $\odot$ has precedence over $\vee$)

$$D = (Y \odot X^T \vee N^+) \wedge (\overline{Y \odot X^T} \vee \overline{N^-}) \tag{2.2}$$

The dimension r of the factor matrices is the *factorization rank*. It is expected to be much smaller than the dimensions of D, such that $1 < r \ll min(m, n)$.

The consequence of a small factorization rank is that the amount of matrix entries required to represent both X and Y is much smaller than the total amount of entries in D. However, as it is possible to construct the full matrix D from the smaller matrices, these must contain the same information as D itself. Due to the limited available space in X and Y the information available in D must be encoded in a more efficient form in X and Y. The noise terms in the above definition capture the idea that bits can be set (N_+) or cleared (N_-) in such a way that an otherwise perfect pattern is damaged. One reason could for a damaged pattern could be simply errors in the data. However it is also possible that the pattern was never complete to begin with. When BMF reconstructs a matrix it tries to predict how the noise free matrix would be structured. This behavior can be used to predict missing values in patterns or to repair errors in the database.

Topics, tiles and basis vectors Figure 2.3 shows a graphical representation of the relations of the different matrices used in BMF. Next to to the names for different matrices and their dimensions we show the variable names as an orientation aid. To get a unified view on these matrices and their interpretation we have combined notations for

the matrices from different sources. Miettinnen et al. [Mie06] use the names "basis vector matrix" for the column oriented matrix X and "usage matrix" for the row oriented matrix Y. They view BMF as applied to topic models where the items correspond to words and selected subsets of the words correspond to topics. Hess et al.[HMP17] interpret the matrices in terms of the content of their rows, which they call "tiles". We have also drawn some supported patterns and their effect on the data matrix D.

2.2.3 Nonnegative Matrix Factorization

Nonnegative Matrix Factorization (NMF) is a generalization of BMF to the real numbers. Actually it was proposed way earlier than BMF and popularized the use of this kind of matrix factorization models in many disciplines. In 1994 Paatero and Tapper introduced the method under the name of "Positive Matrix Factorization" in the context of problems in chemistry and physics[PT94]. The task at hand was the detection of patterns in spectral analysis data.

2.2.5 Definition. Nonnegative Matrix Factorization (NMF)[PT94] Given a data matrix $D \in \mathbb{R}_+^{m \times n}$ with m transactions and n items, the goal of NMF is to find a decomposition of D into matrices $X \in \mathbb{R}_+^{n \times r}$ and $Y \in \mathbb{R}_+^{m \times r}$ with $1 \leq r \leq \min(n, m)$ as well as a residual matrix E such that

$$D = YX^T + E \tag{2.3}$$

with the factor matrices being combined by the usual matrix product over $(\mathbb{R}, \cdot, +)$.

$$(AB)_{ij} = \sum_{k=0}^{r} a_{jk} \cdot b_{ki} = \langle A_{j\cdot}, B_{\cdot i} \rangle \tag{2.4}$$

Here we follow the idea that data in D can be represented up to some residual error by linear combinations of basis vectors from X and Y. Solving (2.4) will yield for X a weighting of each item wrt. each topic and for Y a weighting of each transaction wrt. each topic. This results in a soft allocation of topics and can in consequence represent more refined structure in the data matrix than the pure presence/absence information of BMF can. Depending on the application however one or the other method might be a better fit for the needs of the user. When developing solution methods for BMF the relaxation to NMF is helpful because the problem is transferred from a combinatorial parameter space to a continuous parameter space and as such enables the use of methods which are more time efficient than integer programming and related methods of combinatorial search.

2.2.4 (Dis-)trusting Boolean Matrix Factorization

When applying any mathematical method to real world data, it is common sense that one should not blindly trust the results. There are at least two basic reasons why this

is good practice. On the one hand there might be uncertainties about the data itself, about the generating process as well as about errors. On the other hand there might be doubt in the method used to obtain a result, in its correctness and applicability to the problem at hand. While we recognize that data might be faulty and that this might lead to erroneous results[1], a deep treatment of this issue is far outside the reach of this thesis. We are however concerned with the second reason for distrusting a mathematical method, namely issues with the method itself. As we will discuss below, the structural properties as well as the computational properties of BMF must lead to ambiguities in any proposed solution method. Some of these ambiguities can be mitigated or at least signaled by sensibly constructed algorithms while others cannot. Thus it is necessary to keep in mind the assumptions under which the BMF method operates.

Unknown factorization rank When we obtain a data matrix to analyze with BMF, we need to define the factorization rank r of the BMF which we wish to compute. However we do not know how many topics, if any, make up the hidden structure of the data. So we might choose a rank which does not correspond to reality and obtain a factorization which might have at best a higher reconstruction error. At worst it might lead to false inferences and thus false conclusions with respect to the application domain. To mitigate this issue Hess et. al. suggest repeating the calculation for many different ranks[HMP17], in the worst case for all ranks $1 \leq r \leq min(n, m)$ and take the result with the lowest reconstruction error. This is of course very calculation intensive, and many candidate solutions have to be obtained only to be thrown away.

Equivalent solutions due to symmetries Another structural issue concerns symmetries in the topic allocation. Because one entry of the data matrix is reconstructed as the *logical or* of a set of data-topic and topic-feature allocations, any of these allocations which yields a logical one will contribute in the same way to the final result for the data matrix entry. In this way, the allocations are exchangeable as a whole, i.e. columns of the usage matrix Y or rows of the basis vector matrix X can be permuted without changing the outcome of the reconstruction. This leads to many structurally different matrices describing exactly the same topic structure of the data and yielding the same reconstruction error.

Noisy Data and overlapping Patterns The third ambiguity occurs because the data matrix might have errors. Although we are not concerned with how these errors were created, we recognize the fact that they might exist. In this case it is not possible to derive a correct topic structure. As an example take again the right image in fig. 2.2 where the hidden block structure of the data matrix is shown, but the blocks have some of

[1] The "garbage in, garbage out" principle.

their entries flipped from 1 to 0. A BMF algorithm will most likely fill in the zero entries which lie within a block with ones and zero out entries which lie outside of any block. This behavior can be accepted as long as we can be sure that the entries were really flipped due to random error and do not correspond to the ground truth.

An additional issue occurs when the blocks in the data matrix overlap in such a way that they are not separable, i.e. the overlap in each matrix dimension between blocks is too large. The BMF reconstruction will then never be able to recover a correct topic structure and will yield one rather large topic describing one large block instead of many smaller ones.

Preference of the zero solution Finally, there might be issues when the database is very sparse, i.e. there are many more zero entries than one entries. This could be the case with a real world instance of the aforementioned music database, as there are usually way more songs than a user will be able to listen to, so most of the songs (items) will not be selected for any given user (transaction). In such cases, a possible reconstruction with relatively small error is to set X and Y to the zero matrix. This uninformative result is usually not something one wants to obtain.

Computational Complexity Issues Above we have given quite a lot of reasons why BMF in many cases has no exact solution. Let us assume for a moment an ideal situation where we could guarantee a noise free data matrix with the correct way of pattern formation so that it is applicable to BMF. In this scenario we could obtain an exact solution up to symmetry. Even then we would not be able to construct an efficient algorithm to solve BMF. This result is due to Miettinen et. al. who studied the computational complexity of BMF and proved the accompanying decision problem to be NP-complete[Mie06][2].

One could ask now if there is at least an efficient way to approximate solutions to BMF. We are dealing with all kinds of ambiguity issues already and cannot realistically hope for exact solutions. Sadly, the answer to this question is also negative. In the same work where they prove NP-completeness, Mietinnen et al. proved BMF also to be APX-hard, which means that there cannot be an efficient approximation scheme which approximates a solution to within a given constant factor of the optimal solution.

In consequence any reasonably fast algorithm which approximates BMF can only be seen as a best effort solution without hard theoretical guarantees. As such it is an easy target to critique the method. Still, lots of work has been done to construct BMF algorithms, with researchers focusing their effort on improving on the issues outlined above. In the following sections we will look at some of these solution approaches and describe how they help users to put more trust in solutions obtained from BMF models.

[2]Therein called “Discrete Basis Problem”

2.2.5 BMF as an optimization problem

Most of the material introduced in this section is taken from two papers by Hess et al., namely [HMP17] and [HPM18] which we reference here once. Material from additional sources will be referenced where it is used in this section.

The goal of BMF is to find factor matrices X, Y which can be multiplied as $Y \odot X^T$ to obtain the original data matrix D. As we have already laid out, this is not always possible i.e. due to the influence of noise or the lack of structure in the data. However, we can calculate the difference between the reconstruction and the original matrix. By minimizing this *reconstruction error* we obtain factor matrices which admit a best effort reconstruction.

2.2.6 Definition. BMF optimization problem Given a data matrix $D \in \mathbb{B}^{m \times n}$, assign parameters X, Y which minimize the reconstruction error $|D - Y \odot X^T|$. As we expect the parameters to be as sparse as possible, the penalty terms $|X|$ and $|Y|$ are added to the objective, shifting the model preference towards zero values by increasing the objective value by the sum of one values in both factor matrices.

$$\begin{aligned} &\min_{X,Y} |D - Y \odot X^T| + |X| + |Y| \\ &s.t. \quad X \in \mathbb{B}^{n \times r}, Y \in \mathbb{B}^{m \times r} \end{aligned} \tag{2.5}$$

Note that the optimization above is defined over the boolean matrices as required by the BMF problem. In this way we obtain a combinatorial optimization problem. Together with the proven NP-hardness of the BMF decision problem we see that for high dimensional instances solving this combinatorial problem becomes infeasible very quickly.

PAL-Tiling Hess et al. discuss multiple ways of obtaining approximate solutions to the optimization problem as well as specific algorithmic extensions to solve it and model extensions to impart quality guarantees on the solutions. This collection of methods is proposed under the umbrella of the Proximal-Alternating-Linearized Tiling (PAL-Tiling) framework. The name of this framework incorporates the three main principles from which this framework derives its functionality, which are

- **Linearization** The optimization problem is relaxed to the real numbers to allow for efficient solution algorithms.
- **Alternating gradient descent** A modified version of gradient descent with interleaved descent steps for each factor matrix is the local search method chosen by Hess et al.

- **Proximal Operator** The gradient descent scheme is augmented by a special projection operation which bounds the parameter search to the interval $[0, 1]$.

Linearization is the solution chosen by Hess et al. to relax the domain of the optimization problem. By allowing X and Y to take positive real numbers instead of binary values, they represent the BMF problem as an instance of the related Nonnegative Matrix Factorization problem. In this way they obtain an optimization problem for which efficient solvers can be implemented. After a solution over the real values has been computed, this solution is then binarized by using a thresholding function.

Relaxing the Objective We state the optimization problem using the PanPAL objective, as it is the simplest model proposed by Hess et al. Nonetheless, it has yielded quite good empirical results and we expect it to be easier to transform into a probabilistic model later on.

2.2.7 Definition. PanPAL The PanPAL model minimizes the reconstruction error over the real numbers. The binary matrix multiplication has been replaced with ordinary matrix multiplication. We obtain the reconstruction error as $\|D - YX^T\|^2$.

$$F(X, Y, D) = \frac{1}{2}\|D - YX^T\|^2 + \frac{1}{2}(|X| + |Y|) \tag{2.6}$$

Proximal alternating gradient descent The PanPAL optimization problem is solved by using an alternating gradient descent scheme. In gradient descent algorithms the idea is to follow the direction of the negative gradient of the objective function. An illustration of the general idea is given in figure 2.4. Starting from any point in the parameter space, the algorithm takes steps proportional to the negative direction and the magnitude of the gradient, but scaled with a step size μ. Using θ as the place holder for a general set of optimization parameters, the step update for gradient descent is thus computed as

$$\theta_{t+1} = \theta_t - \alpha_t \nabla f(\theta) \tag{2.7}$$

In alternating gradient descent, the set of variables is partitioned into two subsets (in the case of PAL-Tiling) corresponding to the factor matrices X and Y. There is one update formula for each factor matrix and the updates for each factor are interleaved. In addition the results of each update are skewed and restricted to the interval $[0, 1]$ by applying a so called proximal operator to each update. We get the update formulas

$$X_{t+1} = \text{prox}(X_t - \alpha_t \nabla_X F(X_t, Y_t, D)) \tag{2.8}$$

$$Y_{t+1} = \text{prox}(Y_t - \beta_t \nabla_Y F(X_{t+1}, Y_t, D)) \tag{2.9}$$

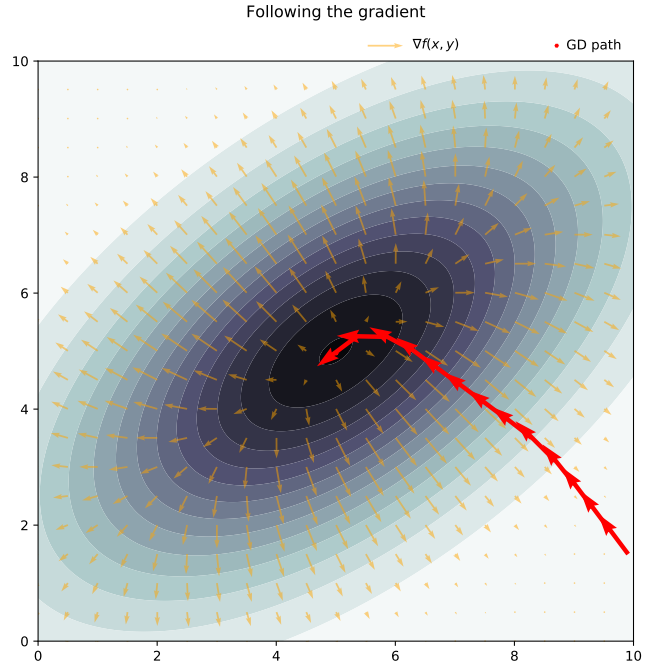

Figure 2.4: By following the negative gradient step by step, the gradient descent method arrives at a local minimum.

with the partial gradients being

$$\nabla_X F(X, Y, D) = (YX^T - D)^T Y + (0.5)_{nr} \tag{2.10}$$

$$\nabla_Y F(X, Y, D) = (YX^T - D)X + (0.5)_{mr} \tag{2.11}$$

A proximal operator is a transformation designed in such a way that it projects each point in the solution space into the the direction of a minimum of the objective function.

2.2.8 Definition. Proximal Operator[PB14] A proximal operator for a given constraint function ϕ implements the following operation wrt. a current point in solution space X and local optima X^*.

$$\text{prox}_\phi(X) \in \arg\min_{X^*} \left\{ \frac{1}{2}\|X - X^*\|^2 + \phi(X^*) \right\} \tag{2.12}$$

This approach has the great advantage that it works even for non differentiable constraints. Points outside the constraints are projected onto the constraint boundaries, while points inside the constraints are pushed slightly towards regions in which the nearest minimum of the function is expected. The illustration in figure 2.5 shows the penalty function Λ used with the PanPAL proximal operator. Solving eq. (2.12) in closed form is possible due to the simple structure of Λ and leads to the following proximal operator for PAL-Tiling. For $\rho > 0$ let

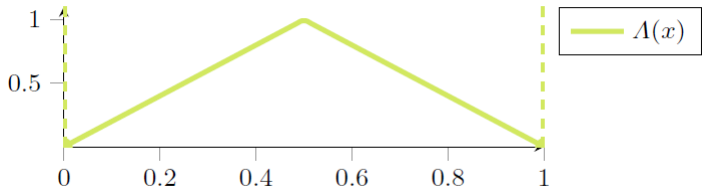

Figure 2.5: The underlying constraint function for the PAL-Tiling proximal operator penalizes the middle of the interval [0,1] slightly and values outside of the boundaries with infinite costs.

$$\text{prox}_{\rho}(x) = \begin{cases} \max\{0, x - 2\rho\} & x \leq 0.5 \\ \min\{1, x + 2\rho\} & x > 0.5 \end{cases} \tag{2.13}$$

This function is defined for scalar values x. When applied to a matrix X it is applied to each element x_{ij} of the matrix.

Lipschitz continuity and step sizes The crucial point to guarantee convergence of gradient descent with the relaxed BMF objective are the step sizes α_t and β_t in the alternating gradient descent scheme. Choosing the right magnitude and decay schedule for these step sizes is necessary, so that the gradient descent method can arrive at an optimal point in the solution space. In general choosing a step size that is too large might lead to overshooting an optimum and finding only suboptimal solutions in its vicinity, so its magnitude is decayed during the optimization run. Choosing a step size that is too small or decays too fast might lead to a very long optimization process and to early stopping also at a suboptimal point.[Jor06]

However it is known from optimization theory that for Lipschitz continuous functions making the step size any multiple of the Lipschitz constant suffices to lead gradient descent to convergence at an optimal point.

2.2.9 Definition. Lipschitz continuity[Jor06] The maximal rate of change of the gradient of a Lipschitz continuous function f is bounded by a constant M, called Lipschitz modulus.

$$\|\nabla f(x) - \nabla f(x')\| \leq M\|x - x'\| \tag{2.14}$$

Hess et al. have calculated the Lipschitz Moduli for the PanPAL objective, thus the PanPAL Objective is Lipschitz continuous. Note that the Moduli are constant wrt. the partial gradient to which they apply, however they depend on the respective other variable.

$$M_{\nabla_X F}(Y) = \|YY^T\| \qquad M_{\nabla_Y F}(X) = \|XX^T\| \tag{2.15}$$

The step sizes are calculated proportional to the inverse of the Lipschitz moduli where $\gamma > 0$ is a scaling factor.

$$\alpha_t = \frac{1}{\gamma M_{\nabla_X F}(Y_t)} \qquad \beta_t = \frac{1}{\gamma M_{\nabla_Y F}(X_{t+1})} \tag{2.16}$$

Besides the convergence guarantees given by the use of Lipschitz moduli, they are also connecting to results within the theory of Ordinary Differential Equations where the existence of Lipschitz moduli and their use to calculate the integration step size is related to the convergence of the numerical integration process. We note, that the calculations appearing in gradient descent schemes are related to those appearing in integration schemes. Also, later on in this work we will visit the field of Stochastic Differential Equations when we introduce stochastic gradient sampling methods. In this field there exist also some convergence theorems which are connected to the Lipschitz continuity of the underlying model functions. In this sense, Lipschitz continuous models are especially well behaved from the standpoint of solvability, whatever that means within the respective field.

Binarization by thresholding When we have computed optimal factor matrices X^*, Y^* in the relaxed problem, the corresponding binary matrices related to the original problem need to be determined. In the simplest case this is done by applying a thresholding function

$$\theta_\tau(x) = \begin{cases} 0 & x \leq \tau \\ 1 & x > \tau \end{cases} \tag{2.17}$$

Again, this function is defined for scalar valued x but if applied to a matrix X it is applied to each element X_{ij}. The issue with this method lies in choosing the right threshold τ. Letting $\tau = \frac{1}{2}$ comes to mind, but as we will see later on this is often not the right choice, as sparsity constraints and also the relaxation itself bias the location of the optima away from integer solutions. In [HPM18] Hess et al. apply a grid search scheme, testing pairs of threshold candidates τ_X, τ_Y with values out of the set $\{0, 0.05, 0.1, 0.15, \ldots, 1\}$. They choose a pair of thresholds which minimizes an error related to the selection of the factorization rank. However, as we will not be searching for the optimal factorization rank, one could also imagine choosing thresholds which minimize the reconstruction error in the original problem.

2.3 Probabilistic modeling and inference

In a probabilistic model we have observed data, free variables and a mathematical model of their dependencies. However instead of postulating one optimal solution to the problem at

hand we assume that any solution is valid but some are more likely. Especially in Bayesian modeling the likelihood of a solution is connected to our specific intention with a given model.[3]

2.3.1 Bayesian models

The Bayesian way of modeling and inference revolves around Bayes Rule, also called the rule of probability reversal. Given a set of observations D we are looking for parameters θ to calibrate our probabilistic model in such a way that it delivers the best explanation for the observed data.

2.3.1 Definition. Bayes' Rule[GCD+13]

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)} \tag{2.18}$$

In the above expression we have $p(D|\theta)$ as the so called *likelihood*, the probability of observing the data given a specific parameter θ. The term $p(\theta)$ is called the *prior* probability. This term is used to bias the inference result towards certain values of θ. The prior is generally used to include assumptions which are assumed to hold before observing the data. However the prior does not have a fixed influence, so that when more data is observed the data governs the outcomes of the probabilistic model. The result of the inference $p(\theta|D)$ on the left hand side is called the posterior parameter distribution or *posterior*. The term $p(D)$ is called the "evidence".

2.3.2 Definition. Evidence The evidence is the probability of observing D averaged over all possible values of θ.

$$p(D) = \int_{\theta\in\Theta} p(D|\theta)p(\theta)d\theta \tag{2.19}$$

There are two things worth noting here. First, the domain of integration is the whole parameter space. This integral can be calculated analytically or numerically in finite time for special cases, but in general it is intractable as the parameter space is infinite. Second, the evidence is constant for a given set of observations D. It follows that $p(\theta|D) \propto p(D|\theta)p(\theta)$. This means that we only need to calculate likelihood and prior when we want to make relative judgments for points in the parameter space. The calculations for these terms can become computationally intensive with large models or large datasets but they are often finite. When we need to make absolute judgments of the parameter probability however, $p(D)$ serves as a normalization constant which ensures that we are obtaining a probability distribution. As this is our usual goal with Bayesian inference, we need a method to estimate the evidence, i.e. estimate the integral in the denominator of Bayes

[3] In frequentist models we would ask for a prediction of the most frequent outcomes assuming that we could repeatedly observe the modeled process in reality.

rule. The part of the remaining chapter concerning sampling methods deals with exactly this task. In these methods our model (the expression in the numerator of Bayes rule) is described by a helpful function, called the log-likelihood function.

2.3.3 Definition. Log-likelihood function The log-likelihood function for a fixed dataset D is defined to be a function that is only proportional to the log of the model terms.

$$LL_D(\theta) \propto \log(p(D|\theta)p(\theta)) \tag{2.20}$$

The logarithm in the formulation is an aid in the calculation of the model as it usually involves a lot of multiplications which could lead to numerical issues on computers when using numeric types with a finite precision. By applying the log, all multiplications are transformed to additions so that the resulting terms are easier to handle.

2.3.2 Exponential families and the Gibbs distribution

Use of the logarithm in the log-likelihood function also eases computation in another way. Many of the probability distributions in frequent use belong to the so called *exponential family* of distributions. In these distributions taking the log eliminates calculation of exponentials which leads to well behaved and faster computations.

2.3.4 Definition. Exponential family[WJ08] A probability density function belongs to the exponential family if it can be constructed as follows. Let $x = (x_1, \ldots, x_m)$ be a vector of random variables that takes values from the product space $\mathcal{X} = \oplus_{i=1}^{m} \mathcal{X}_i$. Let $\phi = (\phi_1, \ldots, \phi_n)$ be a collection of functions, each mapping $\mathcal{X}^m \mapsto \mathbb{R}$. The functions $\phi_i(\cdot)$ are *sufficient statistics* or *potential functions*. Additionally let $\vartheta \in \mathbb{R}^n$ be a constant vector of so called canonical parameters. Then we have

$$p_\vartheta(x) = \frac{1}{Z(\vartheta)} \exp\left\{\langle \vartheta, \phi(x) \rangle\right\} \tag{2.21}$$

where $Z(\vartheta)$ is called the *partition function*. It has the same role as the evidence from Bayes Rule (2.19) as it represents also a normalizing constant and also requires integration over the whole parameter space.

$$Z(\vartheta) = \int_{x \in \mathcal{X}^m} \exp\left\{\langle \vartheta, \phi(x) \rangle\right\} \nu(dx) \tag{2.22}$$

Probabilistic models can be structured in such a way, that not all variables interact with each other. In this case so called potential functions describe direct interactions within subsets of the model variables while indirect interactions are modeled by sharing variables between different potential functions. Probability distributions with this structure are known as Probabilistic Graphical Models (PGM). Whenever a probability distribution has a PGM structure and its potentials are from an exponential family, then the distribution can be written down in the form of a Gibbs distribution as follows.

2.3.5 Definition. Gibbs Distribution The definition given here is based on [WJ08] but specialized to the case that the PGM contains only exponential families. For a set of random variables $X = \{x_1, \ldots, x_n\}$ that are exponential families and associated canonical parameters $\vartheta \in \mathbb{R}^m$ the Gibbs distribution is an exponential family. In this case the sufficient statistics $\Psi(X)$ is a sum over sufficient statistics of the potential functions $\psi(X_i)$ occurring in the PGM, where $X_i \subseteq X$.

- Probability density function and partition function

$$p_\vartheta(X) = \frac{1}{Z(\vartheta)} e^{\Psi(X,\vartheta)} \tag{2.23}$$

$$Z(\vartheta) = \int_{X_i \subseteq X} e^{\Psi(X,\vartheta)} dX_i \tag{2.24}$$

- Structured potential function

$$\Psi(X) = \sum_{X_i \subseteq X} \psi_i(X_i, \vartheta) \tag{2.25}$$

- Log-likelihood function

$$LL(X, \vartheta) = -\log(Z(\vartheta)) + \Psi(X, \vartheta) \tag{2.26}$$

- θ-gradient of the log-likelihood

$$\nabla_\vartheta LL(X, \vartheta) = \nabla_\vartheta \Psi(X, \vartheta) \tag{2.27}$$

$$= \sum_{X_i \subseteq X} \nabla_\vartheta \psi_i(X_i, \vartheta) \tag{2.28}$$

The Gibbs distribution can be defined for general potential functions but as many important distributions are exponential families, it seems fitting to give the specialized definition. The partition function has been calculated analytically for many of the important exponential families and there are some instances of Probabilistic Graphical Models for which efficient inference methods exist. In most settings however the partition function stays intractable.

2.3.3 Monte Carlo Integration

To approximate the normalization constant/partition function Z for a given probability distribution $p(x)$ where $x \in \mathcal{X}$ and integration over $\mathcal{X}$ may be intractable, it suffices to evaluate the unnormalized density $Zp(x)$ at a finite, albeit large, number of points[AdDJ03]. We can then write $p_N(x)$ the empirical point-mass function as follows.

$$p_N(x) = \frac{1}{N} \sum_{i=1}^{N} \delta_{x^{(i)}}(x) \tag{2.29}$$

Here $\delta_{x^{(i)}}$ is the point mass recorded for x accumulated by all visits to points $x^{(i)}$ within the sample and N is the sample size. This formula makes two important assumptions.

- There is a non vanishing probability to visit any $x \in \mathcal{X}$ more than once.
- The points $x \in \mathcal{X}$ are visited proportional to their density $p(x)$.

The first assumption is satisfied in finite, discrete spaces with increasing sample size. In infinite discrete spaces it is satisfied with infinite sampling size. In continuous spaces it cannot be satisfied unless we partition the continuous space into a countable infinity of regions. In this case the approximation of the integral is then reduced to the calculation of the area under a histogram. The approximation error decreases with increasing sample size. When the sample size goes to infinity, we have the following result[AdDJ03].

$$I_N(Zp(x)) = \frac{1}{N}\sum_{i=1}^{N} Z\delta_{x^{(i)}}(x) \quad \xrightarrow[N \to \infty]{a.s.} \quad \int_{\mathcal{X}} Zp(x)dx = Z \tag{2.30}$$

The limit given above shows, that by taking a sample of fixed size we can approximate the normalization constant.

Rejection sampling To obtain a sample where the points of $\mathcal{X}$ are visited proportional to the density we use a method known as *rejection sampling*[AdDJ03]. We cannot directly calculate $p(x)$, however we can obtain a sample from the unnormalized density $Zp(x) \propto p(x)$. Furthermore, we need to construct a *proposal distribution* $q(x)$ for which we can calculate the probabilities. Often, a normal distribution is used, but depending on the problem at hand, other distributions may be used. For all x the density under q must be greater than the corresponding value of $Zp(x)$. In cases where we have information about the location of the modal region of $p(x)$ the proposal distribution q should be constructed in such a way, that its modal region corresponds with p. We define a sample size N and execute algorithm 2.1 until we have gathered N samples.

Algorithm 2.1 Rejection sampling

1: **for all** $i \in \{1, \ldots, N\}$ **do**
2: $\quad x_i \sim q$
3: $\quad u_i \sim \mathcal{U}(0, 1)$
4: $\quad a = \frac{Zp(x_i)}{Mq(x_i)}$
5: $\quad$ **if** $u_i < a$ **then**
6: $\quad\quad$ **output** x_i

In each iteration we draw a candidate point x_i from our proposal distribution q (line 2) as well as a uniform random number $u \in [0, 1]$ as a decider (line 3). The acceptance

probability is calculated as the ratio of $Zp(x_i)$ and $q(x_i)$ where M is a scaling constant that is set in such a way that $q > p$ for all x is guaranteed (line 4). The sample is accepted (lines 5, 6) when the decider u_i is below the threshold set by the acceptance probability (see fig. 2.6). As a consequence we accept samples relative to the density under $p(x_i)$. The acceptance rate is scaled by the coefficient Z/M but this coefficient is constant for the whole sampling process and as such can only lower the general acceptance rate, but not the relative acceptance rate for different points x.

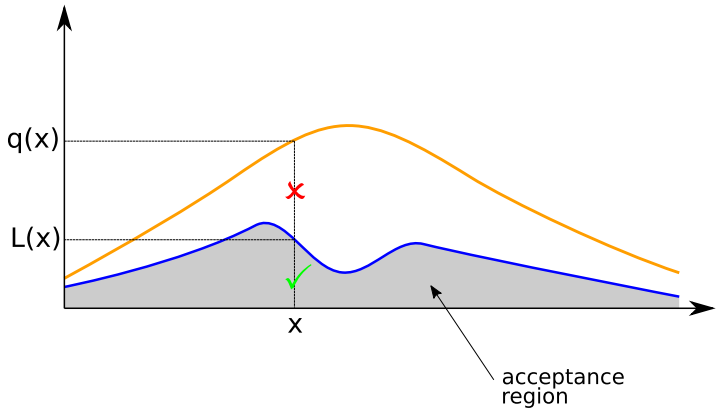

Figure 2.6: Illustration of rejection sampling.

Unfortunately, rejection sampling only works when we can sample a bounded space or have information about the location of the modal regions. Also, it becomes inefficient with very large spaces, as the choice of M might lead to an inefficient sampling process. The next section introduces the use of Markov chains in MCMC to dynamically control the exploration of the sample space.

2.3.4 Markov chains

When we are sampling from a general distribution, we usually have no information where to place the proposal distribution, because we have no information about its modal regions. To obtain samples in such a case, we need to construct a process which guides itself towards the modal region and also collects sample points proportional to the sampled density. This behavior is achieved by constructing a Markov chain, a controlled random walk through the sample space. In the context of Markov chains, the sample space is also called state space as each point in this space is interpreted as a certain state the Markov chain is in.

2.3.6 Definition. Markov chain Given a state space Θ with states $\theta \in \Theta$, a distribution over initial states $p(\theta_0)$ and a transition distribution $T(\theta_{i+1}|\theta_i)$, a Markov chain is the process $\{\theta_0, \theta_1, \theta_2, \dots\}$ obtained by iterated application of T beginning from θ_0.

The transition distribution is the guiding law by which the state of the Markov chain evolves in time. As we can see from the definition of the transition distribution above, at

each point in time the choice of the next state only depends on the current state. This is called the *Markov property* or *memoryless property* and gives Markov Chains their name.

If we let a Markov chain run for an infinite amount of time, we will observe that it has on average visited each state with a fixed probability. This probability is independent of the choice of the initial point. This distribution is called the *stationary distribution* of the chain. The transition distribution must be remain unchanged at each point in time during the execution of the Markov chain. In addition the chain must not have a tendency to get trapped in cycles through parts of the state space. A condition which is sufficient to guarantee this behavior is called detailed balance.

2.3.7 Definition. detailed balance[AdDJ03] A Markov Chain has the *detailed balance* property or *time reversibility* if the following holds.

$$p(\theta_{i+1})T(\theta_i|\theta_{i+1}) = p(\theta_i)T(\theta_{i+1}|\theta_i) \tag{2.31}$$

An implication of this property is that trajectories of the chain that share the same states have the same probability, regardless of the time ordering of the states, so at infinity the stationary distribution can form regardless of the initial point of the chain. If we supply a transition distribution that adds a weighting to the state transitions which is proportional to the relative changes in some density or likelihood function, the stationary distribution of the chain will mimic the distribution belonging to this density. This is exactly what we want, as it implies that the Markov chain has indeed approximated and at infinity computed the normalization constant.

Metropolis-Hastings algorithm The Metropolis-Hastings algorithm[AdDJ03] is the staple method that realizes a Markov chain with the properties just described. It has a long history, and many modifications and improvements have been made. We have given the algorithm in a basic form in 2.2. There we have described one sampling iteration, this iteration can be repeated until the sample size is deemed large enough for the problem at hand.

The algorithm carries only the current state θ_i over between iterations. To impose the structure of the density $p(x)$ an approach comparable to rejection sampling is used. First a candidate point θ_* is drawn from a proposal distribution (line 2). The proposal distribution has been modified, so that it can be conditioned on the current state. For a conditionally Normal distribution this is achieved by moving the mean onto the current state θ_i. In addition a decider is drawn uniformly from $[0, 1]$ (line 3).

The acceptance criterion (line 4) has been modified to accommodate for the fact that we now are comparing neighboring points of the density $p(\theta)$, where the neighborhood is determined by the proposal distribution. If the proposal distribution is symmetric, the q terms in line 4 of the algorithm cancel out, leaving only the ratio between current state

Algorithm 2.2 Metropolis-Hastings algorithm.

Require: number of samples S, proposal distribution $q(\cdot|\cdot)$

1: **for all** $i \in \{1, \ldots, N\}$ **do**
2: $\quad \theta_* \sim q(\theta_*|\theta_i)$
3: $\quad u_i \sim \mathcal{U}(0, 1)$
4: $\quad a = \min\left\{1, \frac{p(\theta_*)q(\theta_i|\theta_*)}{p(\theta_i)q(\theta_*|\theta_i)}\right\}$
5: $\quad$ **if** $u_i < a$ **then**
6: $\quad\quad$ **output** θ_*
7: $\quad\quad \theta_{i+1} \leftarrow \theta_*$
8: $\quad$ **else**
9: $\quad\quad \theta_{i+1} \leftarrow \theta_i$

and the candidate point from the neighborhood. This ratio will be greater or equal to one if the neighboring point has a higher density value, thus driving the process towards high density regions. If the ratio is smaller than one, the algorithm performs rejection sampling by using the current state θ_i as a reference. If the candidate point θ_* is accepted (line 5), it is used as the new state in the next iteration (lines 6,7). If it is not accepted the state is not updated and a new sampling approach is made from the old state (lines 8,9).

While the Metropolis-Hastings algorithm has everything that is needed to sample a probability density, it does not work well with high dimensional problems. The algorithm can become stuck if the acceptance probabilities become unfavorable due to large differences in local density. Alternative approaches have been developed which make use of gradient information to guide the sampling process through difficult high dimensional distributions.

2.3.5 Connecting optimization and sampling

In addition to the already discussed theoretical complications of the BMF problem, it also poses some practical issues which make simple sampling approaches such as Metropolis-Hastings infeasible. The high dimensionality of the problem calls for proposal distributions which are tailored to the problem or better, can adapt to the problem instance. Large data instances may make it necessary to split up the computation of the log-likelihood which needs computation time linear in the size of the datasets due to the addition of the partial likelihoods of each data case.

Recently both optimization and sampling research have seen progress in resolving these issues and there have been indications that the solutions in both disciplines share many of the underlying mathematical principles. Below we give a short history of the important developments in both fields to clarify this connection.

In the construction of samplers performance and convergence issues have been tackled on a problem by problem basis until recently. Although the advantages of using gradient information are well known in the optimization community there was no direct transfer of these results to sampling methods. Instead, the importance of physical models of particle motion in an external field has been recognized[Bet17], thereby incooperating gradient information into the sampling process, thus yielding adaptive proposal distributions.

Resource constrained handling of large datasets has been achieved in optimization due to the development of the Stochastic Gradient Descent algorithm[Bot10], enabling the training of machine learning systems on very large datasets. Instead of handling the whole dataset at once, the algorithm only uses small subsets of the data cases, so called minibatches, during each iteration.

The content of a minibatch changes randomly in each iteration, thereby representing the objective function and its gradient as random variables. This type of optimization on stochastic information is however well known as Robbins&Monro type stochastic optimization[RM51].

Roberts et al.[RT96] and later also Welling et. al.[WT11] recognized the similarity of the Robbins&Monro approach to the Langevin Diffusion, another physical model of particle motion. Both authors popularized different sampling methods using this model. Recent research has led to a unification of Langevin-based approaches[MCF15]. Recently, Mandt. et al. have calculated optimal parameterizations for different Langevin based methods and proposed a sampler which is derived in a direct way from the Stochastic Gradient Descent algorithm[MHB17].

In this way, insights from different fields of research, namely optimization, statistics and physics have been transferred between the fields and into the discipline of machine learning. This transfer enabled new and improved ways to handle todays ever growing problem complexity and workloads with inference methods based on both optimization and sampling.

2.3.6 Sampling with discretized Langevin diffusions

The behavior of the Langevin diffusion is best imagined as the movement of a very light or massless particle, e.g. a particle of dust. The particle is surrounded by another medium which in this case could be air and it is influenced by an air pressure potential field. The movement of the particle will be influenced by the random collisions with the air molecules and thus be chaotic. But it is also guided by the pressure potential and thus the motion will be biased in the direction of the pressure gradient. In this model the general behavior of the particle will be governed by the interplay between random collisions and the strength of the gradient of the external field. If the external field is strong, the particle will follow its gradient, if it is weak, the particle will move around randomly.

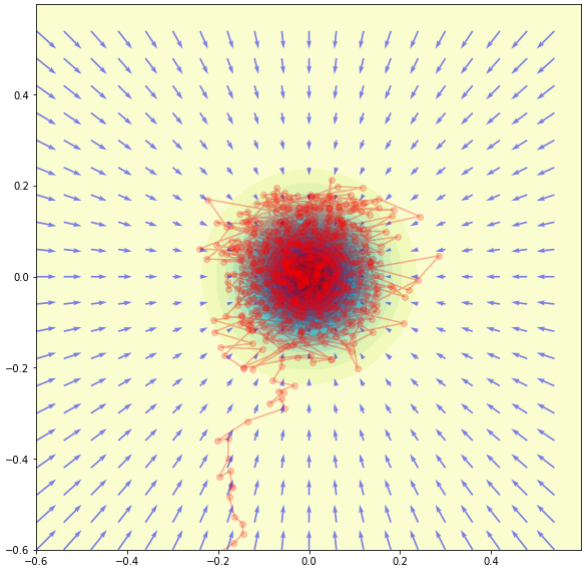

Figure 2.7: Evolution of a Langevin diffusion over the log-likelihood of a Normal distribution. The particle starts at the bottom of the diagram. Red: Sample path, blue arrows: gradient, background: density function.

This behavior corresponds to that of the Metropolis-Hastings MCMC sampler but without the need for an acceptance step. Figure 2.7 illustrates the behavior for a Langevin based sampler that explores the log-likelihood of a Normal distribution. Outside the modal region the particle follows mainly the direction of the gradient. In a modal region of the explored distribution the noise dominates, so that the sampler explores this region. The evolution of the sampling path over time is given by the following differential equation.

2.3.8 Definition. Langevin Diffusion Let x be the position of the sampler, $L(x)$ be a likelihood function and B be Brownian motion, i.e. $\frac{dB}{dt} \sim \mathcal{N}(0, \sigma)$. Then

$$\frac{dx}{dt} = \frac{1}{2}\nabla \log L_D(x) + dB \tag{2.32}$$

Roberts and Tweedie have proven that the Langevin diffusion as given above is a Markov chain which will converge at infinity to the distribution related to the given likelihood[RT96]. However, this insight is not of the greatest practical value. To use this differential equation in practice, a discretized update equation needs to be derived. Iterating this update equation and recording the trajectory of the simulated particle for a fixed number of steps would then produce a sample of the distribution at hand. Roberts and Tweedie show in the same article, that the Unadjusted Langevin Algorithm, a “naive” discretization based on the Euler integration scheme will diverge for most functions.

2.3.9 Definition. Unadjusted Langevin Algorithm Given a likelihood function L and a noise scale $h > 0$ the ULA computes a discretized Langevin diffusion using the following update equation.

$$x_{i+1} = x_i + \frac{1}{2}\nabla \log L_D(x_i) + h\mathcal{N}(0, I) \tag{2.33}$$

The introduction of discrete time requires the use of a scaling parameter which in the above expression is given by the constant noise scale h. In more recent approaches, the scaling parameter can also be a diagonal or a full matrix, thus scaling or decorrelating the noise term in all dimensions. In the multidimensional case the noise scale is called the preconditioner[PT13].

The authors reason that there is no general way to calculate the correct noise scale as it does not only depend on the function class of L but also on the properties of the concrete function instance. They propose the addition of a Metropolis-Hastings step to the ULA algorithm and show that this extension guarantees convergence to the expected distribution. Later research indicates that convergence can be guaranteed without the Metropolis Hastings step when the likelihood function is locally Lipschitz continuous[MSH02]. This is fortunate, as the Lipschitz moduli for the relaxed BMF problem exist.

While researching a suitable Langevin based inference method for the probabilistic BMF problem, the above insights led us to dismiss most of the existing approaches. Determining the correct noise scale would mean repeated lengthy experimentation for each problem instance. As such the larger part of the time available for this thesis would have been spent by tinkering with parameters before any further insight could have been gained. However, a recent approach by Mandt et al. proposes a Langevin based sampling method together with a way to calculate the noise scale or preconditioners. We used their approach to realize the inference method for the probabilistic BMF.

Mandt et. al. propose an algorithm for locally quadratic likelihoods, which validates the use for probabilistic BMF as its log-likelihood is a quadratic polynomial. Furthermore they have derived their algorithm partially from sampling theory and partially from optimization theory, in particular by reinterpreting the Stochastic Gradient Descent[Bot10] algorithm. Their method is presented in algorithm 2.3.

The algorithm starts in a given initial state θ_0. The number of data cases N, the minibatch size S as well as the sample size M are given. At each iteration a randomly selected minibatch $\tilde{D}$ containing a subset of the datacases is supplied to the algorithm. With this minibatch the current state is evolved using the discrete Langevin update (line 3). As the noise term is incorporated in the gradient of the likelihood L due to the random nature of $\tilde{D}$, the preconditioner ϵ is applied directly to the gradient expression.

While each iteration yields a new intermediate state, the algorithm outputs a sample only each T iterations (line 4,6). This sample is produced by averaging the last T intermediate states (line 5). Averaging the highly correlated states in this way leads to

Algorithm 2.3 The Iterate Averaging Stochastic Gradient (IASG) algorithm

Require: number of datacases N, batchsize S, number of samples M, initial state θ_0

1: $T = \frac{N}{S}$
2: **for all** $t \in \{1, \ldots, MT\}$ **do**
3: $\quad \theta_t \leftarrow \theta_{t-1} - \epsilon \nabla \log L(\theta_{t-1}, \tilde{D}_{t-1})$
4: $\quad$ **if** $t \bmod T = 0$ **then**
5: $\quad\quad \mu_t = \frac{1}{T} \sum_{j=t-T}^{t} \theta_j$
6: $\quad\quad$ **output** μ_t

samples which are optimally decorrelated[MHB17]. The size of the averaging window T is important. To output one valid sample, the sampler must have the chance, at least statistically, to observe all N data cases[MHB17]. The averaging window is therefore chosen as $T = N/S$ (line 2). The sample yield of this method can outperform Metropolis-Hastings based approaches. Although the method generates many intermediate states, the algorithm has a guaranteed rate of sample production. The rate of sample production is not guaranteed for Metropolis-Hastings based approaches, as the algorithm can reject arbitrary many samples between outputting new ones.

As mentioned above, Mandt et al. also give an optimal preconditioner ϵ^* for their algorithm. It is optimal under the assumption of a locally quadratic likelihood. Using this preconditioner the algorithm converges to a distribution which minimizes the Kullback-Leibler divergence to the expected distribution.

2.3.10 Definition. Optimal Diagonal Preconditioner for IASG

$$\epsilon^* = \frac{2S}{N(BB^T)_{kk}} \tag{2.34}$$

The preconditioner scales proportional to the relative batch size S/N, in effect matching it to the magnitude of the log-likelihood gradient which increases with the number of data cases in the minibatch. It also scales antiproportional with some noise covariance matrix BB^T, thereby acting as a normalizing transformation on the noise term. This noise term is data dependent and usually determined by calculating the observed Fisher information matrix [GC11].

2.3.11 Definition. Observed Fisher Information

$$BB^T = \tilde{F}(\theta) = \widehat{\mathrm{cov}} \left[\nabla_\theta \log L_D(\theta) \right] \tag{2.35}$$

The Observed Fisher Information $\tilde{F}(\theta)$ captures the variation in magnitude of the the log-likelihood gradient on its diagonal, as well as the cross correlation of these magnitude variations between dimensions of the parameter θ. One can interpret the function of the

preconditioner proposed by Mandt. et al. as a scaling factor which scales the noisy gradient of the log-likelihood on average towards unity and thereby controlling for magnitude differences which would otherwise cause oscillations in the integration process.

In the next chapter we will lay out our implementation of IASG, which has to undergo some changes due to the fact that data and parameter structure in probabilistic BMF differ from "normal" machine learning problems. We will also describe our test datasets and evaluation procedure.

Chapter 3

Implementation and evaluation

3.1 Constructing a probabilistic BMF model

To construct a Bayesian model it is necessary to supply a likelihood and a prior, as we discussed in section 2.3.1. The likelihood $p(D|\theta)$ expresses the probability that the data D has been generated by the given parameter θ. In the relaxed BMF objective for the optimization problem, the reconstruction error F_r has a related role as it expresses the mismatch between the reconstruction obtained using the parameters X, Y and the observed data. The reconstruction error as introduced earlier is defined as follows

$$F_r(X, Y, D) = \frac{1}{2}\|D - YX^T\|^2 \tag{3.1}$$

We can assume that parameter values which minimize the reconstruction error in the optimization model have a higher probability of being the best fitting parameter values within a probabilistic model. This can be achieved by negation of F_r turning it from a minimizer into a maximizer. Next, we make the assumption, that the probabilistic model is an exponential family. This allows us to reinterpret F_r as the sufficient statistic of this model and use it in its unnormalized form, knowing that we have to compute $Z(D)$ the normalization constant later on by sampling. Adding priors for the parameters X, Y leads to the following Bayesian model.

$$p(X, Y|D) = \frac{e^{-F_r(X,Y,D)}}{Z(D)} p(X)p(Y) \tag{3.2}$$

We still need to supply probabilistic models for the prior. In this work we have used the uniform, normal, laplace as well as the beta prior, thereby expressing different assumptions about the properties of the parameters within the probabilistic model. We evaluated the prior choices given below.

$$p(X) \sim \mathcal{U}(0,1)^{m\times k} \quad \text{and} \quad p(Y) \sim \mathcal{U}(0,1)^{n\times k} \tag{3.3}$$
$$p(X) \sim \mathcal{N}(0,1)^{m\times k} \quad \text{and} \quad p(Y) \sim \mathcal{N}(0,1)^{n\times k} \tag{3.4}$$
$$p(X) \sim \text{Laplace}(0,2)^{m\times k} \quad \text{and} \quad p(Y) \sim \text{Laplace}(0,2)^{n\times k} \tag{3.5}$$
$$p(X) \sim \text{Beta}(0.5,0.5)^{m\times k} \quad \text{and} \quad p(Y) \sim \text{Beta}(0.5,0.5)^{n\times k} \tag{3.6}$$

Plots of the prior densities densities in the interval $[0,1]$ are given in figure 3.1. Using the uniform prior assumes that there is no additional preference over parts of the parameter space *other than the preferences which might be expressed by the likelihood alone.* The normal and Laplace priors are priors which enforce sparsity, as they put a strong preference on parameter values which are near zero. The Laplace(0,2) prior corresponds to the regularizer for the PanPAL model given in [HMP17], as the log-likelihood of this prior is proportional to the absolute value function.

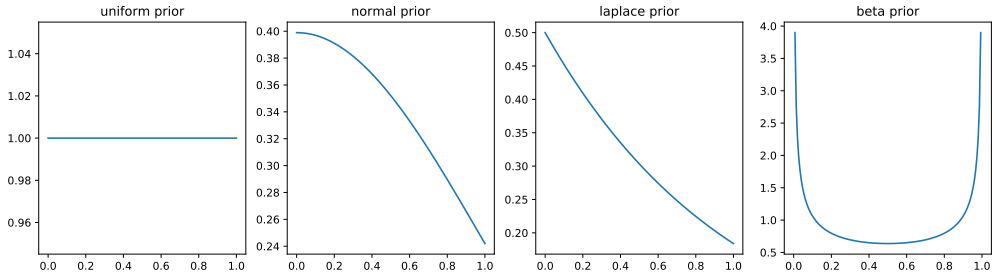

Figure 3.1: Plots of the different prior densities used.

We also used a beta prior in our experiments, as it is the recommended uninformative prior for random variables which are Bernoulli distributed in the posterior parameter distribution[GCD+13]. The beta prior is however not part of our main experiments, as it became clear very early that the strong suppression of nonintegral parameter values leads to convergence issues with the Langevin based sampling approach. We will review this phenomenon later on when discussing our results.

While the construction of the probabilistic model by supplying a sufficient statistic was straightforward, there is another way to justify the exponential family structure. When we expand the polynomial expression for F_r we obtain a sum of functions of the form $\sum_{i,j}\left[D_{ij} - \sum_k x_{ik}y_{jk}\right]$ in the exponent. As this is effectively a potential structure, the distribution can also be interpreted as a Gibbs distribution. Thus it is possible to construct a PGM, whose graph we have shown in figure 3.2.

In this graph the white circles are the random variables for single elements of the parameter matrices X, Y. Blue rectangles represent deterministic operations over the random variables. They make up the larger part of the network. The elements of the data matrix are shown in green as observed random variables. Variables and operators are

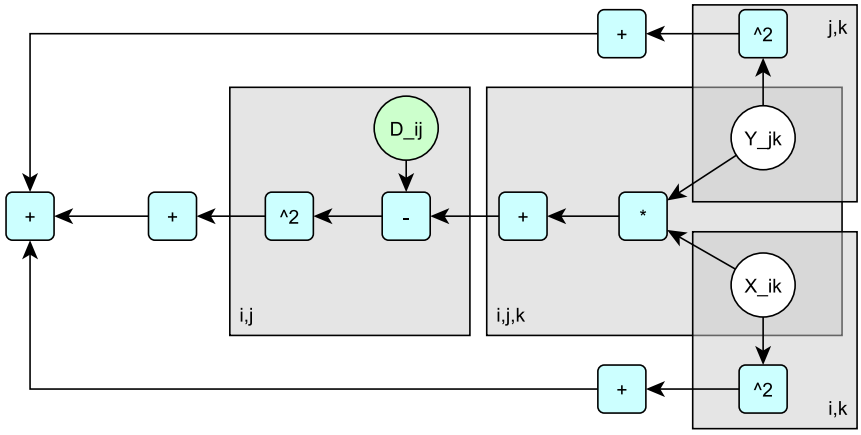

Figure 3.2: The arithmetic structure of F_r, represented as a PGM with mostly deterministic operations (blue) around the random variables X_{jk}, Y_{ik} and the observations D_{ij}. Plate notation (grey) is used to visualize the index sets over which the respective nodes are replicated.

replicated across their respective index sets depending on the indexing of the associated matrix. Operations in the index sets i, j and i, j, k represent the calculation of F_r while the index sets i, k and j, k contain operations that are concerned with calculating a prior.

We have a probabilistic model, but we still need a way to run inference on it. The next section will discuss how a Langevin diffusion sampler can be implemented to handle this model.

3.2 Adaption of IASG to the BMF problem

In the BMF problem the layout of parameters and dataset does not match the usual way in which parameters and data are given in a model. In a general machine learning model there is a parameter vector and a dataset where each row forms a data case. The situation is different with BMF as with the parameter matrices X and Y we have two parameter blocks and the data cases relevant to each parameter are either created out of rows (transactions) of the data matrix D for the parameter X or out of columns (items) of D for the parameter Y. As we have this special structure, the whole sampling process needs to be split into two separate but interacting processes. A related situation arises in the optimization problem where the gradient descent algorithm needs to have an X, Y alternating structure. We have to calculate alternating partial gradients ∇_X and ∇_Y and associated information for each of the parameters. There are also two different ways to draw minibatches, which we introduce next.

3.2.1 Drawing minibatches

A minibatch is a randomly selected subset of all data cases with a predetermined size. At the beginning of each iteration of a minibatch based optimization or sampling algorithm a new minibatch is selected at random. The model update is then performed only on the current minibatch instead of the whole dataset, thus reducing the computations needed to compute the model function.

In the usual machine learning problem format each row of the dataset contains one data case while each column contains feature information. Minibatches are drawn by subsetting the rows of the dataset. With BMF we have transactions in the rows of the data matrix and items in the columns. Due to the way the parameters X, Y relate to transactions and items in the data matrix D, we can draw minibatches from the rows of D only for the parameter X. For parameter Y we need to draw minibatches from the columns of D. The situation is shown in figure 3.3. If we need to calculate the partial gradient for X, the tiles of X depend on all items of D, but we can select transactions independently if we also select from Y the allocations corresponding to the selected transactions. Because Y is held constant when we update X it is interpreted as additional data in this situation. The "dual" situation occurs when we want to calculate the partial gradient for Y. In this case the tiles of Y depend on all transactions but we can select independent items. Depending on the situation we reduced the dimensionality of the problem in either the transaction dimension or the item dimension.

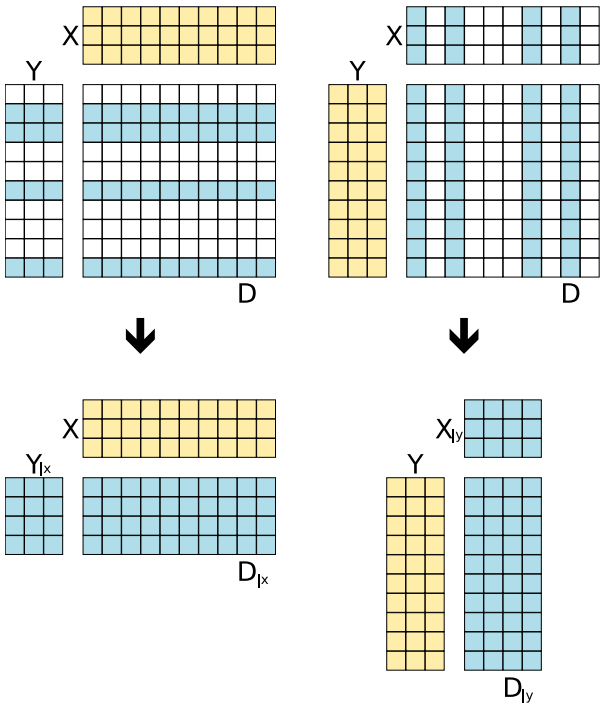

Figure 3.3: Depending on the active parameter (yellow), minibatches (blue) are either taken row wise or column wise.

In the following sections we will use additional notation to distinguish between matrices that are part of the original problem and matrices which belong to a minibatch. We will use $Y_{|x}$,$D_{|x}$ for a minibatch in x-orientation, i.e. a minibatch over the rows of D together with the matrix for the reduced parameter Y. When we have an additonal iteration index i we might also write $Y_{i|x}$. For minibatches in y-orientation we have $X_{|y}$,$D_{|y}$ as a minibatch over the columns of D and the associated subsetted version of X. Under an iteration index we again might write $X_{i|y}$.

3.2.2 Noise scale calibration

In section 2.3.6 we have introduced the Iterate Averaging Stochastic Gradient (IASG) algorithm by Mandt et al. One outstanding feature of this algorithm is existence of an optimal noise scale/preconditioner, so that with a locally quadratic model the algorithm has an asymptotic convergence guarantee. In this way it relieves the user from time consuming experimentation to find the right preconditioner. The preconditioner itself is based on approximating the Fisher information of the parameter space. In 3.1 we have given an algorithm that first approximates the diagonal of the Fisher information by uniform sampling of the parameter space and subsequently also calculates the noise scale. As the whole algorithmic design revolves around separate handling of the parameters X, Y there are two Fisher information matrices as well as two preconditioners, one for each parameter.

Algorithm 3.1 Noise scale calibration

Require: number of calibration samples M_{cal}, batch sizes S_X, S_Y.

for all $i \in \{1, \dots, M_{\text{cal}}\}$ **do**
  $X_i \sim \mathcal{U}(0,1)^{n \times k}$
  $Y_i \sim \mathcal{U}(0,1)^{m \times k}$
  Draw a minibatch $Y_{i|x}, D_{|x}$ in the x-orientation
  Draw a minibatch $X_{i|y}, D_{|y}$ in the y-orientation
  $C_{xi} \leftarrow \nabla_X \log L(X_i, Y_{i|x}, D_{|x})$
  $C_{yi} \leftarrow \nabla_Y \log L(X_{i|y}, Y_i, D_{|y})$
$\epsilon_X = 2\frac{S_X}{m} \frac{1}{\widehat{\text{var}}(C_{x\cdot})}$
$\epsilon_Y = 2\frac{S_Y}{n} \frac{1}{\widehat{\text{var}}(C_{y\cdot})}$
output ϵ_X, ϵ_Y

The number of calibration samples M_{cal} is determined by the user. Different batch sizes S_X, S_Y can be assigned to each of the problem dimensions. For each parameter matrix M uniformly distributed parameter values X_i, Y_i are then drawn and their partial gradients C_{xi}, C_{yi} are calculated on minibatches for each parameter, as the magnitude of the noise scale depends on the number of data cases included in the calculation of the

partial gradient. The Fisher information diagonals are calculated as the variances of the partial gradients. From them the preconditioners for each parameter ϵ_X, ϵ_Y are derived.

3.2.3 Sampling algorithm

After the noise calibration is finished, the main sampling process can begin. The algorithm calculates two interleaved discrete Langevin diffusions, one for each of the parameters X, Y. The diffusions are coupled via their state variables X_t, Y_t, so that each of the partial calculations uses the current state of the respective other calculation. The sampling algorithm is presented in algorithm 3.2. First the averaging time window is determined, as in IASG. As we can have two different batch sizes, one corresponding to each of the parameters X, Y, the larger of the two possible time windows is chosen. In this way the algorithm is able to output samples for which a sufficient amount of data has been observed. Next, the number of iterations of the main loop is set so that M samples can be output.

Algorithm 3.2 Adapted IASG for the relaxed BMF model

Require: minibatch sizes S_X, S_Y, sample size M, initial state X_0, Y_0, data matrix D.

1: $\epsilon_X, \epsilon_Y \leftarrow$ NOISESCALE(D)
2: Let $T = \lceil \max(m/S_X, n/S_Y) \rceil$
3: **for all** $t \in \{1, \ldots, MT\}$ **do**
4: Draw a minibatch $Y_{t-1|x}, D_{|x}$ in the x-orientation
5: $X_t \leftarrow X_{t-1} + \epsilon_X \nabla_X \log L(X_{t-1}, Y_{t-1|x}, D_{|x})$
6: $X_t \leftarrow$ CLAMP(X_t,δ)
7: draw a minibatch $X_{t|y}, D_{|y}$ in the y-orientation
8: $Y_t \leftarrow Y_{t-1} + \epsilon_Y \nabla_Y \log L(X_{t|y}, Y_{t-1}, D_{|y})$
9: $Y_t \leftarrow$ CLAMP(Y_t,δ)
10: **if** $t \bmod T = 0$ **then**
11: $\mu_{X,t} = \frac{1}{T} \sum_{j=t-T}^{t} X_j$
12: $\mu_{Y,t} = \frac{1}{T} \sum_{j=t-T}^{t} Y_j$
13: **output** $\mu_{X,t}, \mu_{Y,t}$

The main loop begins with an update of the X parameter, by drawing a minibatch $Y_{t-1|x}$,$D_{|x}$ in x-orientation based on the Y parameter of the *last* iteration (line 4). The partial gradient in X is calculated using the minibatch and the state X_{t-1} of the *last* iteration. It is multiplied by the preconditioner and added to X_{t-1} yielding the state X_t (line 5). After the update, the parameter X_t is clamped, i.e. hard limited to the range $[0, 1]$ or for $1 \gg \delta > 0$ to a slightly smaller interval (line 6). After the new state X_t is determined, the update proceeds for Y. The minibatch in y-orientation $X_{t|y}$,$D_{|y}$ is drawn using the *current* state X_t of X (line 7). The partial gradient in Y is calculated, multiplied

with the preconditioner and added to the state Y_{t-1} of the *last* iteration to determine the current state Y_t (line 8). This state is then also clamped (line 9). At the passing of each time window (line 10)a new sample consisting of $\mu_{X,t}$ and $\mu_{Y,t}$ is calculated by taking the means of the last T states of X as well as Y and then outputted (lines 11 to 13).

While the algorithm 3.2 outputs samples as its main output, in our implementation it also delivers gradient information as well as other diagnostics. Additionally the computation of means and the variances in the calibration routine can be done using incremental formulas. As these changes do not change the semantics they were left out for reasons of readability.

Validity of the method Clamping of the values to the allowed interval has been implemented as a last resort solution and may distort the sampling process as it invalidates the detailed balance condition. The correct way to limit the parameter space is the implementation of sampling from a truncated distribution. While the issue is known in statistics, there is no general method to sample from truncated distributions. While there are methods to sample from truncated Normal distributions[Rob95] and other simple distributions, it seems that such complex objects as truncated stochastic processes have not been touched.

Leaving the clamping aside for a moment, interleaving two diffusions might not be a valid operation under the assumptions outlined in this thesis, as it implies a nonstationary transition distribution for each chain due to the time dependent change of the respective non-state parameter. We do not provide proof, but some hints why that may not matter. Using the given update formulas and substituting the minibatches by an explicit noise term we have

$$X_t = X_{t-1} + \nabla_X(X_{t-1}, Y_{t-1}, D) + \mathcal{N}(0, \Sigma_X) \tag{3.7}$$

$$Y_t = Y_{t-1} + \nabla_Y(X_t, Y_{t-1}, D) + \mathcal{N}(0, \Sigma_Y) \tag{3.8}$$

Depending on the current regime of the chain we make the following observations. If the chain is gradient controlled, i.e. not in a modal region, the noise term is negligible. In this case the chain behaves like an alternating gradient ascent algorithm. If the chain is dominated by noise, the chain behaves like brownian motion independent of the respective other state space. In both regimes the chain displays a valid behavior, so it can be expected that it performs the expected overall function.

To obtain comprehensive insights into the algorithm as well as the model, we have evaluated the algorithm on different datasets and parameterizations, which we are going to discuss in the following section.

3.3 Evaluation process and goals

To test the inference method we have formulated some questions which will serve as goals around which the configuration of datasets and parameterizations is oriented. The questions concern behavior and convergence of the inference method and the implications of the uncertainty observed in the probabilistic model. We have generated test datasets using a test data generator and devised sets of parameters to execute multiple sampling runs. We aim to collect evidence which allows us to perform an analysis of the behavior of a sampling method and model and give at least qualitative answers to the questions that were posed.

3.3.1 Goals

The focus of our evaluation lies on obtaining qualitative results which enable deeper insights into probabilistic BMF and Langevin based inference methods. For the implemented inference method we have the following questions.

- Is the method able to converge to a distribution around an optimum even in the high-dimensional setting of BMF?
- How does the choice of the batch size parameter influence the result?

Mandt. et al. show results for IASG where the sampler does converge in the mean but not in distribution. In comparison even to small BMF instances their datasets were of quite low dimension but have a large number of data instances. There were namely 8 features/45730 instances for the Protein dataset and three features/245057 instances for the Skin dataset[MHB17]. Asking if the method works at all with really high dimensional models is therefore justified. We specifically ask for convergence around a single optimum. Due to the high degree of symmetry of the model it is not feasible to visit all possible optima even when only drawing a single sample for each one. Our next questions concern the role of uncertainty in BMF.

- Which aspects of the problem are expressed by the uncertainty of a probabilistic solution?
- How do the uncertainties look around a known optimum?
- Is it possible to use the probabilistic information to obtain a better reconstruction method?
- How does a a high noise level in the data show within a probabilistic model?

From a probability theory point of view, the most uncertain parts of a solution are those for which the marginal distributions show the largest variances. Thus, when inputting a known optimum as the initial value of the sampler, we would expect to obtain an evaluation of the given result in terms of model uncertainty and thus be able to draw inferences on further properties of the problem instance. Such inferences could be helpful in creating a reconstruction method which follows a probabilistic argument. In addition to answering these questions with a known optimal solution as a reference, we also would like to understand what happens if there can be no optimal or no feasible solution, i.e. when we use a factorization rank which does not match the one used in creating the dataset or when the dataset is extremely noisy. Especially the first scenario should be often true in real world applications of BMF as the factorization rank is one of the unknowns in the original problem formulation. Increasing the noise level is a straightforward way to further increase the complexity of the task and observe the performance of the algorithm.

3.3.2 Test data generator

To generate our test datasets we have implemented the generation method as described in [HMP17]. In the same article the authors note, that the procedure as given has been commonly used to generate test datasets within related publications. As such it seems to be the most sensible choice to follow that tradition.

The method generates a dataset consisting of the data matrix as well as factor matrices with defined dimensions and a defined pattern density. Additionally the factor matrices contain a column index. The column index opens the possibility to map resulting factor matrices onto or as near as possible to the original matrices, if instead of the original matrix structure a permuted matrix with equal reconstruction error has been discovered by an inference algorithm. We have implemented this routine as shown in algorithm 3.3 and used it to generate test datasets which will be described in the following section.

3.3.3 Test datasets

We chose to generate two datasets, one of which has a relatively low noise percentage and one where the noise percentage is considerable, relative to the pattern density. Both datasets have 512 rows and columns and a factorization rank of 30, leading to a dimension of $2 \cdot 512 \cdot 30 = 30720$ for the parameter space for each problem. Both datasets are sparse with a pattern density q of 0.1. An overview of the parameters is given in table 3.1.

One might now argue that two datasets alone do not suffice to characterize the behavior of a complex algorithm. While there is truth to that argument, during our work we noticed that by choosing different models and different model and algorithm parameters a considerable variation of behavior occurred, which still yielded interesting insights.

Algorithm 3.3 GenerateTestData

1. **Given** dimensions n, m and factorization rank r^*, pattern density q, positive noise rate p_+ and negative noise rate p_-.
2. Let $k = \lceil \frac{n}{100} \rceil$ and $l = \lceil \frac{m}{100} \rceil$
3. Generate index matrices $X_{\text{idx}} \in \mathbb{B}^{kr \times r}, Y_{\text{idx}} \in \mathbb{B}^{lr \times r}$ by diagonally stacking 1 vectors of length k resp. l.

$$X_{\text{idx}} = \begin{pmatrix} 1_k & 0 \\ 0 & 1_k \end{pmatrix} \qquad Y_{\text{idx}} = \begin{pmatrix} 1_l & 0 \\ 0 & 1_l \end{pmatrix} \tag{3.9}$$

4. Generate pattern matrices $X_{\text{pat}} \in \mathbb{B}^{(n-kr) \times r}, Y_{\text{pat}} \in \mathbb{B}^{(m-lr) \times r}$ which are initially filled with zero.
   - For each column of X_{pat} set $\lfloor (n - lr) \cdot q \rfloor$ of entries uniformly random to 1.
   - For each column of Y_{pat} set $\lfloor (m - kr) \cdot q \rfloor$ of entries uniformly random to 1.
5. Construct factor matrices X^*, Y^* by stacking the respective index and pattern matrix.

$$X^* = \begin{pmatrix} X_{\text{idx}} \\ X_{\text{pat}} \end{pmatrix} \qquad Y^* = \begin{pmatrix} Y_{\text{idx}} \\ Y_{\text{pat}} \end{pmatrix} \tag{3.10}$$

6. Let $D = Y^* X^{*T}$.
7. Create the noise term N as follows and add it to D.
   - If $D_{ji} = 0$ then $N_{ji} = 1$ with probability p_+
   - If $D_{ji} = 1$ then $N_{ji} = -1$ with probability p_-

Dataset	n	m	r	p_+	p_-	q
clean	512	512	30	0.01	0.05	0.1
noisy	512	512	30	0.05	0.15	0.1

Table 3.1: Parameters for the clean and the noisy dataset.

Therefore we chose to work with two datasets considering the "clean" dataset to be the easy case and the "noisy" dataset to be the hard case.

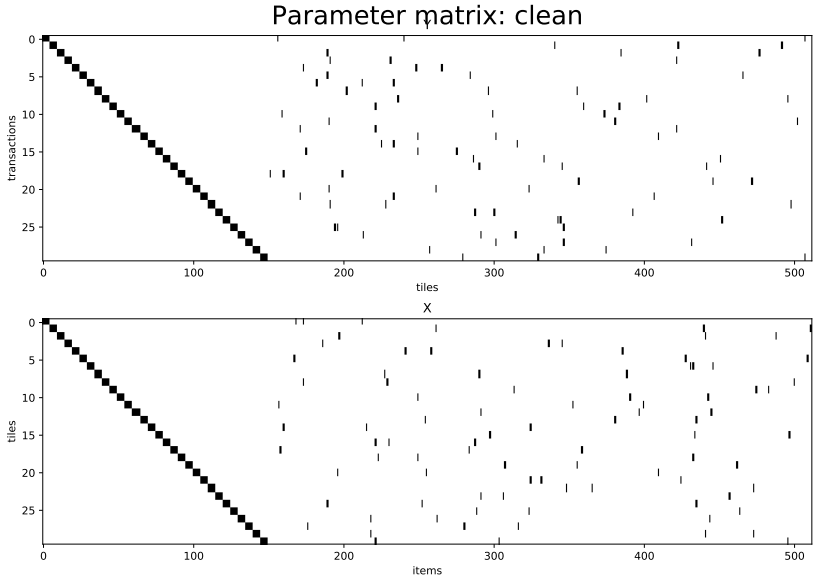

Figure 3.4: Parameters X and Y of the clean dataset. Note that due to the reduced aspect ratio of the drawing the columns appear to be widened.

In figure 3.4 the parameter matrices X, Y for the "clean" dataset are shown. Note that when plotting these matrices, a reduced aspect ratio was used to allow for better visibility of the matrix contents. Thus each single column appears to be multiple columns lines wide. The indexing matrix and pattern matrix for each parameter can be clearly distinguished. Figure 3.5 shows the data matrix with and without noise as well as the involved positive and negative noise matrices. The matrix C is the result of the matrix multiplication. Adding noise yields the matrix D, the actual output of the data generator routine. N and P are the negative and positive noise parts of the noise matrix. The matrix D is what is given as input to the BMF algorithm. To calculate the reconstruction error wrt. the original, it is necessary to keep matrix C around, as we expect the BMF reconstruction to be a reconstruction of the noiseless original C when given the noisy matrix D.

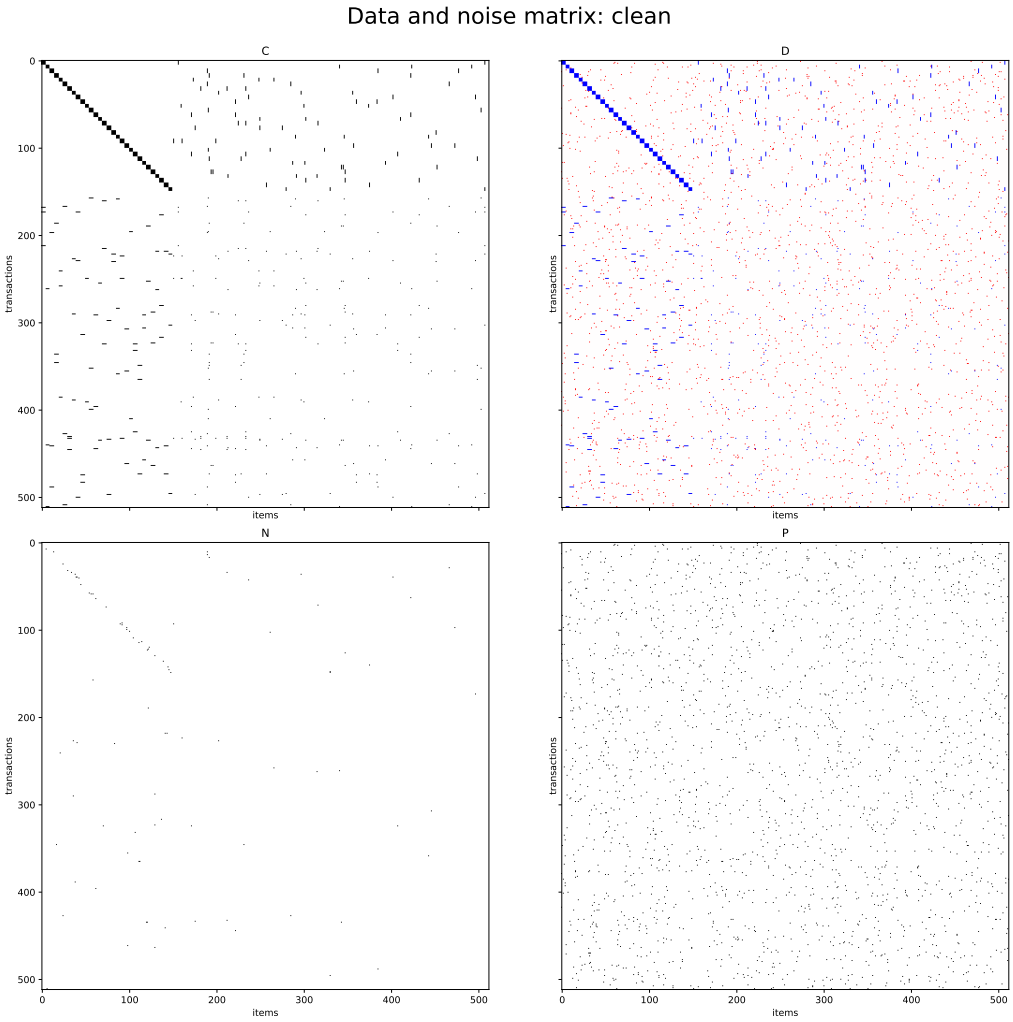

Figure 3.5: Data and noise matrices of the clean dataset. C: clean data matrix, D: noisy C with information of C in blue and noise in red. N: negative noise matrix. P: positive noise matrix.

3.3.4 Sample creation and data recording

We have implemented our adapted version of the algorithm in the Python programming language[Ros95]. The numeric code was implemented using the Python language bindings to the ArrayFire[YAM+15] matrix and tensor computation library. We chose this combination out of familiarity with the Python language and the transparent support for GPU computing which comes with the ArrayFire library. The choice of libraries enabled us to achieve reasonably good execution performance and fast development times without diving too deep into the intricacies of software development for GPU computing, thereby gaining more time to focus on the task at hand. Source code and the data generated in our experiments can be found on the attached CD-ROM (see appendix in chapter 6).

sample size	dataset	model	start solution	rank	minibatch size
60k	clean, noisy	BMF + flat prior, PanPAL, BMF + normal prior	optimum, random	20, 30, 40	75, 250, 512
600k	clean, noisy	PanPAL BMF + normal prior	optimum, random	20, 30, 40	512
2M	clean, noisy	BMF + normal prior,	random	30	512

Table 3.2: Run parameters and their variations for each sample size.

We calculated multiple runs with varying configurations. For each run the step size was calibrated with the calibration procedure given in section 3.2.2 using 2500 independent samples. To obtain a basic overview of the behavior of different parameterizations and models we started with relatively small sample sizes. The varied parameters were as follows

- Dataset. Clean or noisy.
- Model. Relaxed BMF reconstruction error with varying prior.
- Initial position of the sampler. At a known optimum or at a random point.
- Factorization rank. Smaller (20), larger (40) and equal to (30) the known rank of the dataset.
- Minibatch size. small (75), medium (250), full (512).

As the sampling algorithm works based on the noise that is induced by selecting minibatches, for the full data case batches of size 512 are sampled. By allowing multiple selection of data cases, we can still get a noise term in a full size dataset. We recorded results for all parameter variations for a sample size of 60.000 points, because at this sample size the occurrence of defects or problematic parameter settings could be reliably observed. However at that sample size, no model had converged to a useful distribution. At the given problem dimension this does not come as a big surprise. For this reason we also started some sampling runs at or near a known optimum, so that the path to the high density region is not included, as it could skew the sample distribution. To get a chance at observing convergence when starting at a random point, we did some extended runs with 600.000 samples and also with two million samples. For the larger sample sizes we only chose parameter variations where we expected an improvement in the reconstruction

error. An overview of the varied parameters is given in table 3.2. For each run we recorded the following measurements

- Log-likelihood trace for each sample.
- Mean and variance of the X, Y parameter matrices over all samples.
- Mean and variance of the X, Y gradient matrices over all samples.
- Marginal histograms for each element $x \in X, y \in Y$ in the range $[0, 1]$ and with 50 buckets each.

We made these specific measurements because each of them summarizes the contents of the sample from another point of view. The log-likelihood indicates the general direction of movement through the distribution. Means and variances of the parameters summarize the mean marginal results of the process as well as their marginal probabilistic uncertainties. Histograms allow drilling down into the distributions of specific marginal parameters to inspect the concrete distribution of the assigned probability mass. Recording the gradient mean and variance for all enables judgements about the convergence of the process and also about the influence of the choice of model and parameters that are observed by the sampler on its trajectory. It turned out that each of the measurements contributed an important part to our understanding of the observed behavior, as we will see in the following analyses.

Chapter 4

Results

4.1 Observation of the model log-likelihood

In the theory part of this thesis (see chapter 2) we have described conceptual similarities between sampling and optimization methods. In optimization methods the trajectory of the objective function value is an important indicator for convergence as well as for performance. The conceptual analog in probabilistic models is the log-likelihood function, however its usefulness as an indicator is limited. In a probabilistic model, convergence is usually measured wrt. the time evolution of the state of the Markov chain[GCD+13]. Nevertheless, analyzing the trajectory of the log-likelihood can yield first pointers to defects in a given run.

4.1.1 Log-likelihood at the start of the sampling run

A sampler is usually either initialized to begin its trajectory at a random point or at a known but non optimal point in the parameter space. At this stage, the position update is influenced more by the gradient than by the noise term. We expect the algorithm to move proportional to the gradient towards the nearest maximum in the parameter space. From a certain point on a transition occurs, as the gradient gets smaller the closer one moves to the maximum. At this phase, the noise term in the update equations takes over and leads to exploration of the region around the given maximum.

In runs with a random initial position, this behavior becomes visible as a steep initial drop in the log-likelihood, as shown in the top part of figure 4.1. In this figure we give an example from the clean dataset but comparable behavior also occurred on the noisy dataset.

The algorithm behaved in an unexpected way in runs where it was started from a known point which is optimal in the non relaxed BMF problem. We observed a considerable initial drop from the integer optimum to a related non integer point which is optimal in the relaxed problem for all models. This behavior is shown in the bottom part of figure

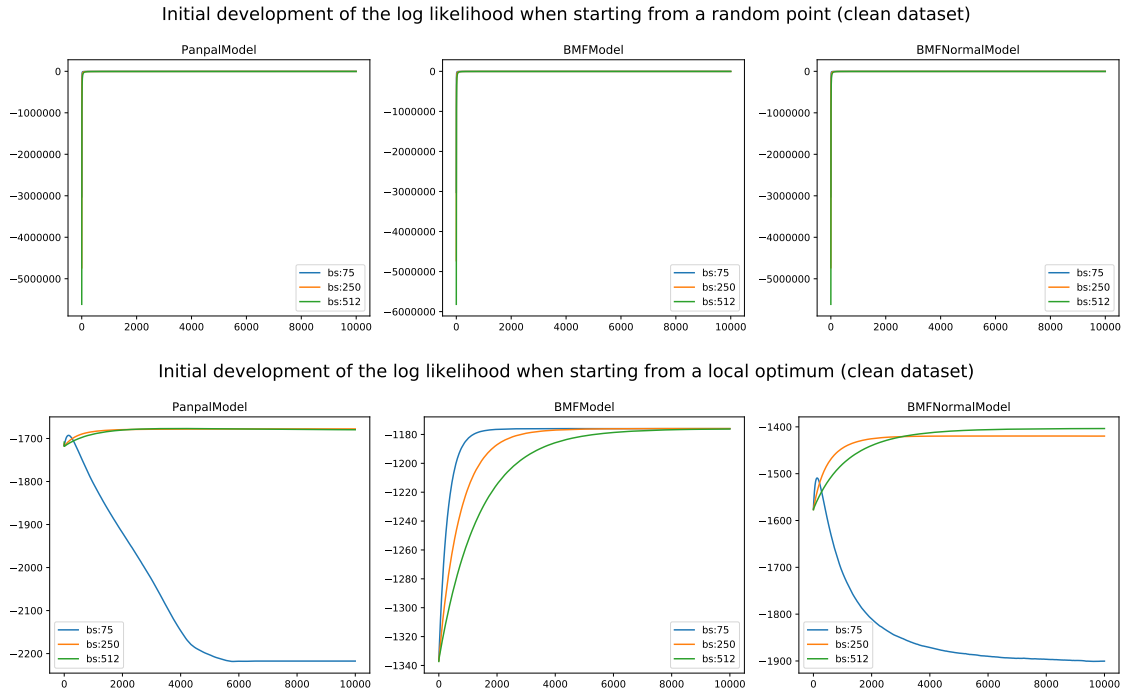

Figure 4.1: Initial log-likelihood (sample 0 to 10000) for different models and batch sizes. Top: initialized at random point. Bottom: initialized at know integer optimum. Correct factorization rank $r = 30$. Clean dataset.

4.1. For batch sizes 512 and 250 there is a drop from the integer optimum to a related non integer point.

Interestingly, for batch size 75 the algorithm diverges to some lower likelihood value. In 4.1 this is clearly visible due to the scaling of the graphic only for the cases where the sampling was started at a known optimum. The effect however occurs also for other runs with the same batch size. Due to the fact that it happens only when there is a non-flat prior, we can already guess that there might be not enough data left in the model with this small batch size. As discussed in section 2.3.1, in the Bayesian setting the prior dominates the posterior distribution when there is only a small number of data cases.

As we are simulating a stochastic process, we should expect to see some noise in the log-likelihood. However it is not visible in figure 4.1, as the noise term on the log-likelihood is small in comparison to its absolute value. This can be explained by recalling that MCMC methods take samples from the typical set[Bet17] for a given sample size. All points belonging to this set have a probability that is related to the entropy of the sampled distribution. While trajectories like the one shown in 4.1 would indicate convergence to an optimum in the case of an optimization problem, for the sampling problem they only show that the sampler is exploring a region of high and roughly constant probability density. So, for the sampling case no convergence observations can be made from observing the log-likelihood alone.

4.1.2 Exploration of a low dimensional BMF analog

To understand why the sampler moves from the integer optimum to another point in the relaxed BMF model, we constructed low dimensional analogs of the BMF log-likelihood as examples. They share the same polynomial structure as the relaxed BMF objective in Hess et al.[HMP17], but have each only three variables $x, y, z \in [0, 1]$ and three constants $a, b, c \in \{0, 1\}$. The variables correspond to single elements in the matrices X, Y while the constants correspond to elements in the matrix D. We have one low dimensional BMF model with flat prior (LL_1), and one with normal prior (LL_2), defined as follows

$$LL_1(x, y, z) = -\frac{1}{2}\left[(xy - a)^2 - (xz - b)^2 - (yz - c)^2\right] \tag{4.1}$$

$$LL_2(x, y, z) = -\frac{1}{2}\left[(xy - a)^2 - (xz - b)^2 - (yz - c)^2\right] - \frac{1}{2}(x^2 + y^2 + z^2) \tag{4.2}$$

In figures 4.2 and 4.3 we have produced grid plots for each of the above functions which serve to give an impression of the structure of local maxima in a low dimensional setting. Each subplot shows a contour plot of the xy plane for a given assignment of a, b, c and z. In the columns of the grid plot we enumerated each of the eight possible states of a, b, c. For each row the z-cuts were taken at $z \in \{0, 0.25, 0.5, 0.75, 1.0\}$. Maximal points in each subplot have been marked with a red dot.

Figure 4.2 cuts through the function LL_1 where only the BMF reconstruction error without any prior is used to calculate the log-likelihood. One would expect that in this case the behavior is nearest to the integer valued problem setting. However, at first glance we can see that in more than one case the real valued maximum does not correspond to the integer maximum. First, note the maxima and extent of the high density regions for the cases in the top left of the figure. The maxima extend along each axis, i.e. there is total confusion as to the most probable value for the variables. As the first column of the diagram belongs to the data case (000) the variables x, y are now in effect unconstrained. When we start to constrain the model with nonzero observations in columns 2 to 4, the confusion situation stays in effect for the cases where $z = 0$ although the likelihood value is higher in the other cases and highest for $z = 1$. From an optimization standpoint there are no issues with this situation, as an optimization algorithm would go straight towards the $z = 1$ case and yield the result. However, with a sampling method the whole high density region around an optimum counts towards the information that is carried in the sample and thus will influence the sample means.

Another situation which could spell trouble for a sampling method arises in column four of figure 4.2. The optimum for this situation is reached at $z = 1$ in row 5, however for the intermediate assignments to z the optimum splits into two likely assignments for x and y. In the given case these optima arise because (011) is an inconsistent data case wrt. the given formula. While the zero forces x, y to be zero, both ones force at least

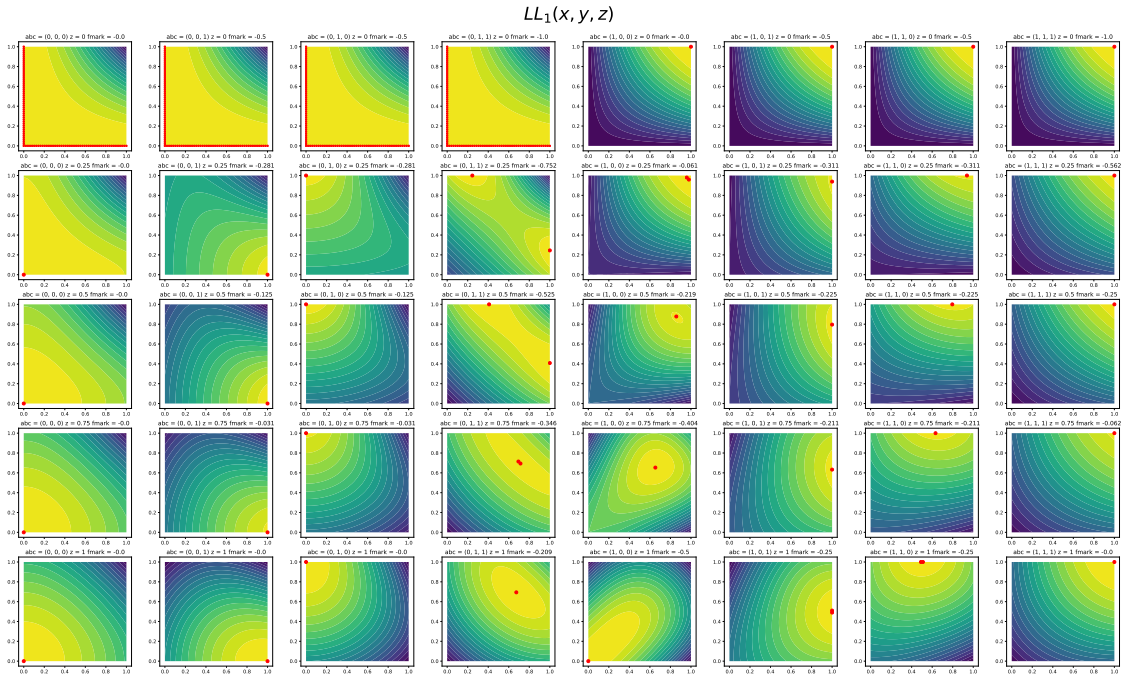

Figure 4.2: A low dimensional analog to the BMF model with flat prior. Values for all $x, y \in [0, 1]$ are shown at differing values for z (rows) and for differing values of $a, b, c \in 0, 1$ (columns). The red dots mark the maximum at the given z.

two variables out of the three x, y, z to be one. This is a situation which occurs when no perfect reconstruction is possible due to noise influence. However looking at the mass distribution for all z-cases in column 4 the convergence to the correct mean could take quite a long time[1], as there is a lot of probability mass in the region of total confusion (row one).

The columns five through seven also describe situations where a perfect reconstruction is not possible due to a data case which is inconsistent wrt. the given model formula. Observe, that for all these solutions the probability mass is concentrated around real values which lie somewhere in $[0, 1]$, but depending on the value of z the mass concentration might span large parts of the parameter space.

In figure 4.3 a plot for the function LL_2 is shown, which adds the Normal prior as a sparsity assumption to the model. In an optimization model this would correspond to a quadratic regularizer. The differences to the situation in figure 4.2 become immediately visible. The region of confusion is eliminated in columns one to four. In column one, the density is pulled towards (000) which is the correct solution. In column two however the solution is also pulled towards (000) while the high density region moves from the region around zero to the region around 0.5 for x when z goes towards 1. In this case we would

[1] if this situation occurs a few hundred times for different sets of marginals in a high dimensional setting

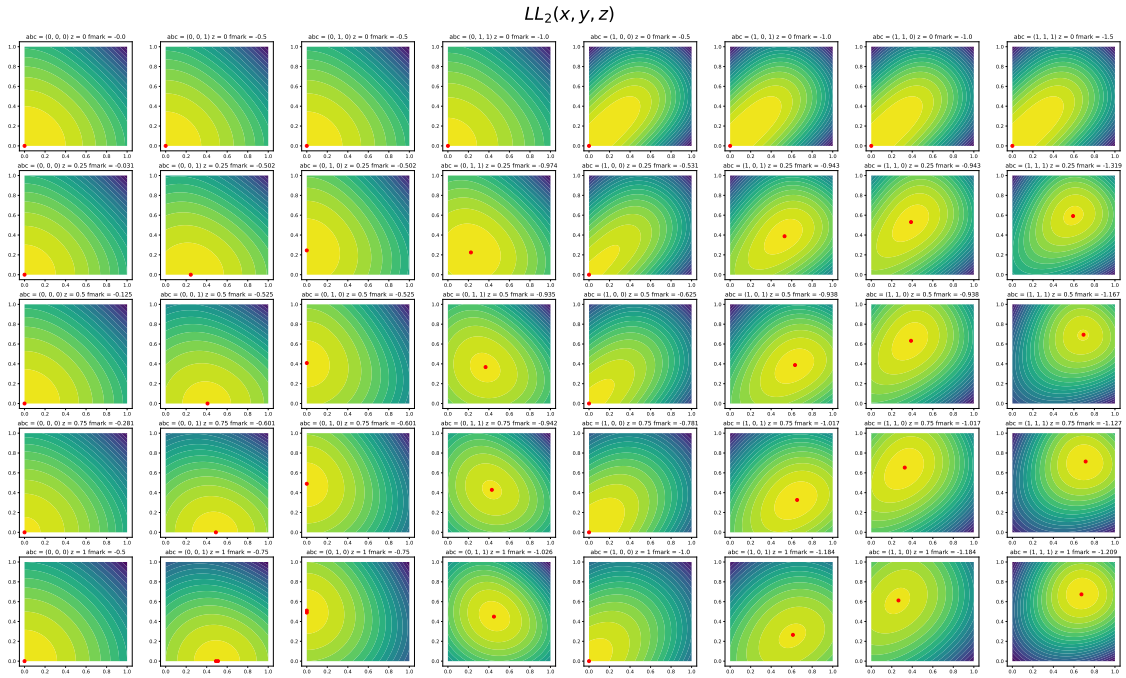

Figure 4.3: A low dimensional analog to the BMF model with Normal prior. Values for all $x, y \in [0, 1]$ are shown at differing values for z (rows) and for differing values of $a, b, c \in 0, 1$ (columns). The red dots mark the maximum at the given z.

infer $z = 0$ for sure because the least part of the probability mass is located in the vicinity of $z = 1$. Instead the model puts probability mass on x but this is mostly confined to $x \leq 0.5$. In this way the model went from the inference $z = 1$ and x, y likely to be 1 (to restore consistency) figure 4.2 to the inference $y = 0$ and $x = 0, z = 0$ with having high uncertainty on the latter assignments. The model effectively switched to the wrong but sparser solution in the face of an inconsistent data case, in effect preferring the assumption of positive noise.

We need to mention here, that the regularized model pulls all of the given cases to the inference (000) which is a consequence of having only one data case. This is obviously not enough data to overrule the prior. In effect, we have shown two extreme cases, namely no prior influence in figure 4.2 and full prior influence in figure 4.3. The geometry of the density of a real model with many data cases will lie somewhere in between.

The examples have yielded two important hints why we have observed a drop in the likelihood when starting the sampler from an integer optimum in section 4.1. On the one hand, the typical set in high dimensions spans regions which lie relatively far from the density maximum, so we should observe a likelihood trajectory where the likelihood values are smaller than the value at the integer optimum. On the other hand, real valued maxima in the log-likelihood arise from ambiguities or inconsistencies due to noise in the data.

The influence of a sparsity prior might move the expected values of the uncertain variables towards zero. This has the effect of skewing marginal distributions towards zero. It also changes the logical propositions associated with the resulting inferences if the prior influence is too large. As we will see in the following sections, the larger issue with a sparsity prior is the effect of skewing the sampled distribution, which also persists when we use all available data and causes considerable issues with the matrix reconstruction.

4.2 Inspection of parameter means and variances

After this excursion into lower dimensional territory we now come back to the probabilistic BMF model and present further results. We will see that some of the presented characteristics of the solution can be linked to the insights we obtained from analyzing a low dimensional case first. Initially, let us take a look at the sample means and variances of the parameters.

4.2.1 Structure of the mean parameter matrices

Figure 4.4 shows images of the parameter mean matrices of a sampling run are shown. We used the BMF model with Normal prior and let the sampler run for 60,000 iterations. The coloring in the images is chosen so that by the coloring one can clearly distinguish parameter values which are near to zero (red), near to one (blue) or near to 0.5 (yellow). These images give a qualitative impression of the sample mean after the given number of iterations. Note that in the the images, the parameter matrices are again drawn transposed with an increased aspect ratio for the columns.

The images on the top show the situation when the starting point of the sampler was set to the optimal integer solutions. The indexing pattern is clearly visible, it has the correct column ordering and its values are tending towards one. In the mean the algorithm has explored only around the optimal initial solution, which is the expected behavior for the algorithm.

Starting the algorithm from a random position yields the results shown in the bottom images of figure 4.4. The indexing region contains an unordered pattern, so the algorithm most possibly found a permutation of the original solution.

Comparing the X and Y matrices one gets the impression, that the Y matrix is noisier when the algorithm is started from a random initial position. Taking the histogram of the parameter values as shown in figure 4.5 confirms this impression. The histograms on the bottom show considerable differences in the distribution of parameter values between X and Y.

Note that the histograms are cutting out values which are smaller than 0.05. In this way the tail of the distribution becomes visible. The number of near zero values suppressed

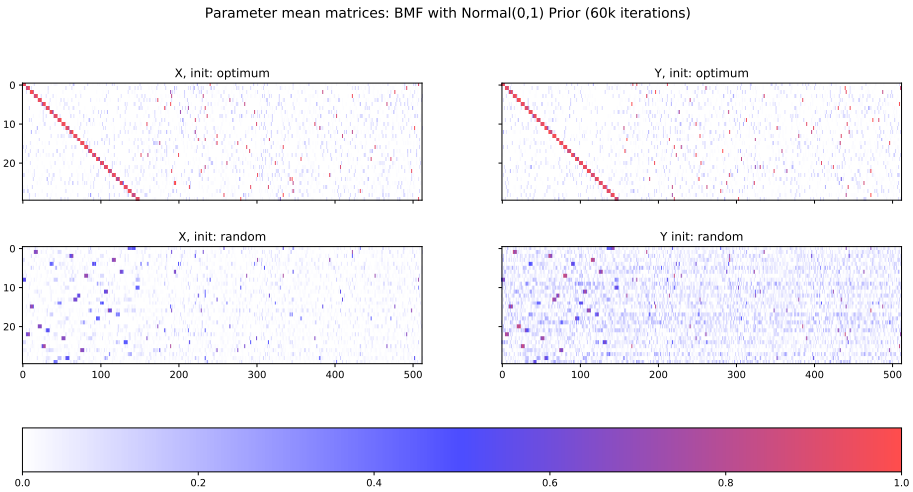

Figure 4.4: Images of the parameter matrices X, Y for a BMF Model with normal prior after 60k iterations. Values range from zero (red) to one (blue). Clean dataset, correct factorization rank (30) and maximum batch size (512). The matrices are drawn transposed, and with a higher aspect ratio.

is displayed in the title of each histogram. The number of near zeros deviates only for the Y matrix when the sampler was started from a random position. We observed the behavior discussed above for each combination of parameters, except when we set the batch size of the sampler to 75. In this case the algorithm converged to a zero solution, most probably because there was not enough data to overrule the prior. In the BMF model with flat prior the noise in the Y matrix was even more pronounced. Unfortunately, such an imbalanced state cannot lead to a correct reconstruction for our case, as we know that the fill rate and structure of both matrices is similar.

For models with sparsity prior the imbalance straightens out in the long run. Figure 4.6 shows the situation after 600k iterations. We take this as an indication, that a sample with 60k points is not enough to explore the area around a density maximum to the degree necessary to yield correct parameter mean values for this BMF instance.

When sampling the BMF model with flat prior we observed that the algorithm would cycle between imbalances, one time having too much noise in X and at other times too much noise in Y. When we sampled using the model for some low dimensional test instances, the algorithm would even cycle through permutations and eventually run into states where one parameter matrix was noisy enough such that the other parameter matrix was forced to be zero. We took these results as an indication that the BMF model with flat prior will not always yield usable means when sampled.

Based on this insight and the fact that the BMF model with Normal prior as well as the PanPAL model (BMF plus Laplace Prior) yielded comparable results, we have focused

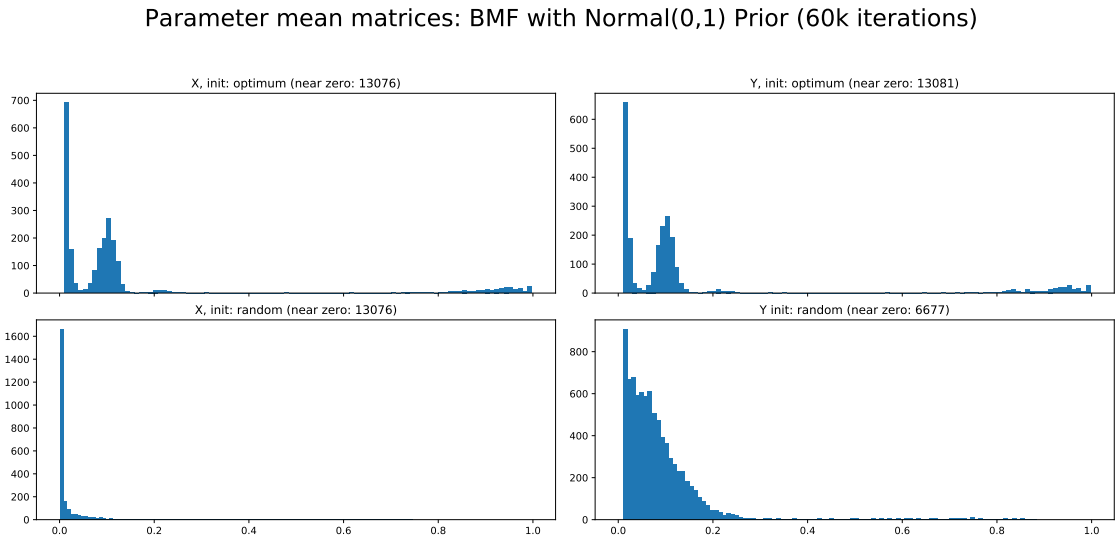

Figure 4.5: Histogram of the parameter matrices X, Y for a BMF Model with normal prior after 60k iterations. Values range from zero (red) to one (blue). Clean dataset, correct factorization rank (30) and maximum batch size (512). Data points near zero have been left out of the histogram, but their count is given in the titles.

our investigations on the BMF model with Normal prior. In the following section we are looking at the results in terms of the variances, i.e. the uncertainty attached to the means.

4.2.2 Parameter means vs. standard deviations

In the last section we found indications that a sample size of 60k does not suffice to approximate the posterior mean for the given problem instance. We have plotted elementwise parameter means against their elementwise standard deviations for the BMF model with Normal prior in figure 4.7. Means are given on the x-axis, standard deviations on the y-axis. When the sampler is started at a random position (top row of the figure), we observe that after 60k iterations many of the mean values that are near zero are assigned a lower uncertainty by the model than values which are far greater than zero. In the diagrams we observe that the maximum mean value is around 0.8 for the 60k sample size and that there seems to be a cluster of variables for which the model assigns a rather high uncertainty. Reasons for this behavior are most likely the sparsity of the whole dataset in combination with a sparsity prior, so that in overall the zero value is more likely than the one value.

At 600k iterations more evidence has been collected and the means of some parameters have shifted towards one. While there is still a high uncertainty on some parameters, the cluster near 0.8 has shifted towards standard deviations of around 0.1. After two million iterations, the sample asserts the respective mean values with even more certainty. Starting the sampling from a known optimum yields a totally different result. In the lower part of

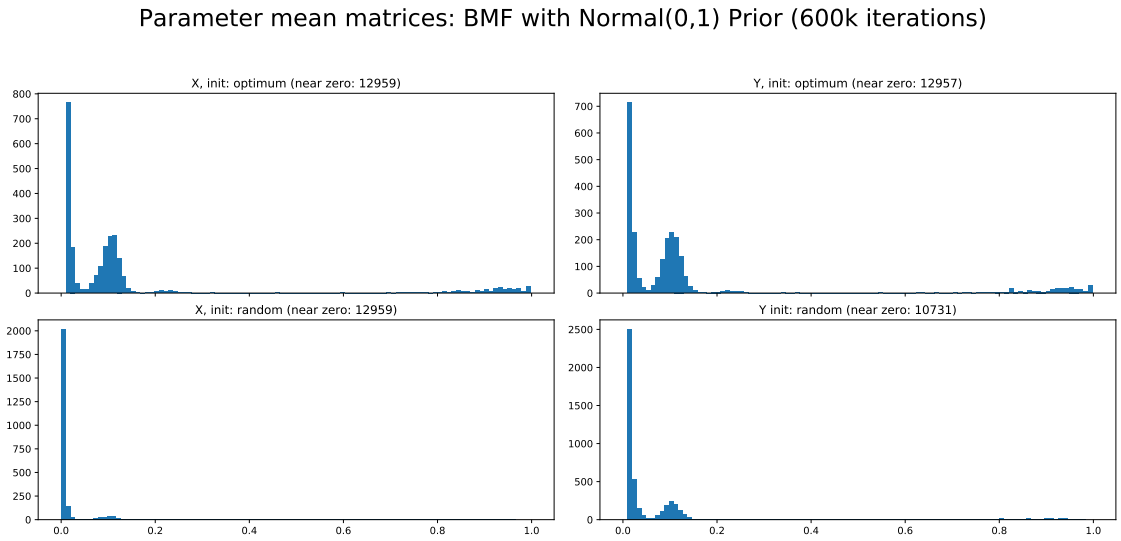

Figure 4.6: Histogram of the parameter matrices X, Y for a BMF Model with normal prior after 600k iterations. Values range from zero (red) to one (blue). Clean dataset, correct factorization rank (30) and maximum batch size (512). Data points near zero have been left out of the histogram, but their count is given in the titles.

figure 4.7 we have shown the sample with 60k points and 600k points. We have not created a sample with two million points for this situation as we did not expect any significant improvement or change in the resulting sample. In this sample every parameter value is quite certain even after only 60k iterations. A linear relationship has formed between the parameter mean values at the low end and their uncertainties, as well as the parameter mean values at the high end and their uncertainties. This linear relationship flattens with increasing sample size. While we have looked at these results from a probabilistic standpoint and talked about the Bayesian uncertainty, there is actually more to it. If the model is quite certain about assigning a mean value of e.g. 0.8 to a given element of a parameter matrix, it doesn't actually mean that it meant to assign a value of one. We as observers know that we are working with a relaxed version of the boolean model. So we could interpret it that way, as we know that there is a binarization step involved after taking the sample. The model however strictly infers that it is quite certain about the value 0.8. As we have seen in our inspection of low dimensional models, real valued inferences occur most likely due to ambiguities or inconsistencies in the data cases.

4.2.3 Differing batch sizes and the choice of prior

The batch size parameter controls the size of the subset of data cases which is randomly selected for each iteration of the sampling algorithm. In addition to the controlling resource usage, the noise introduced by the use of minibatches is the heart of the algorithmic principle behind the IASG sampler.

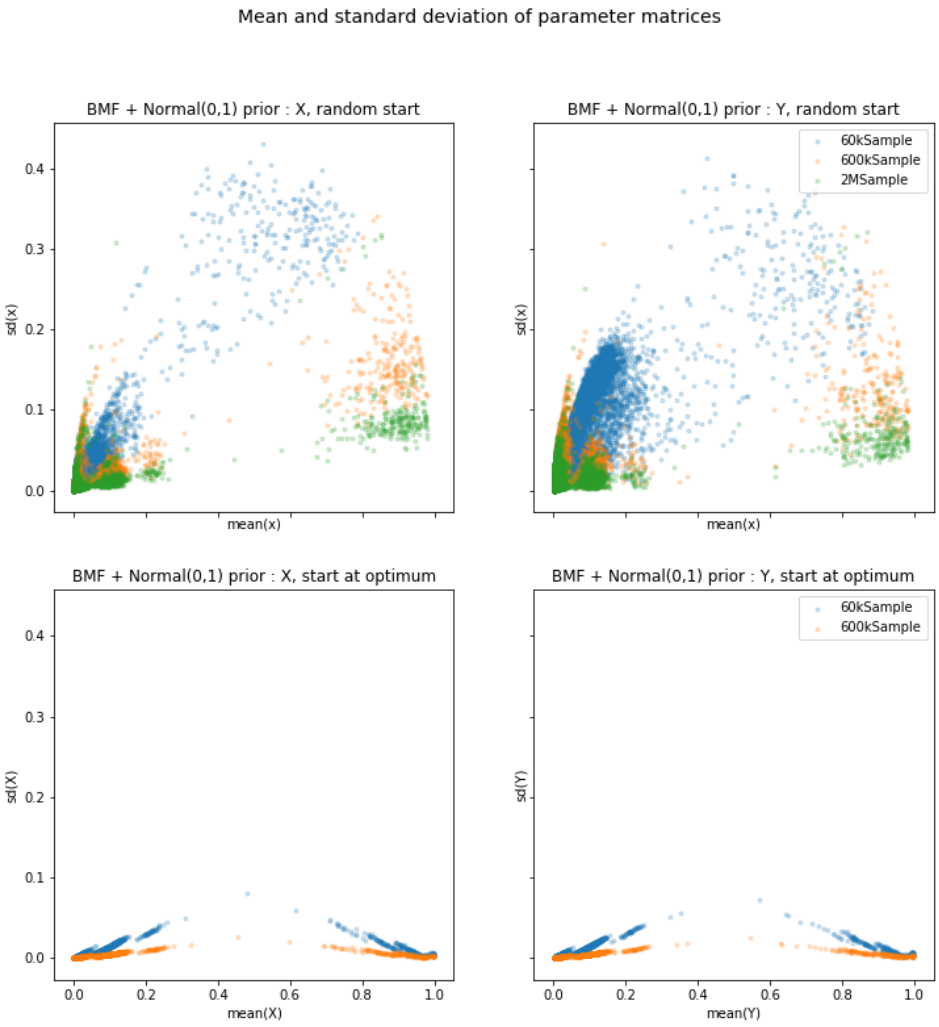

Figure 4.7: Scatter plot of the mean values vs. the standard deviations for parameter matrices X, Y for a BMF Model with Normal(0,1) prior. Clean dataset, correct factorization rank (30) and maximum batch size (512).

We found that the effects of the batch size setting are nontrivial in the presence of priors or regularizers. In figure 4.8 we have plotted again mean versus standard deviation of the parameter mean matrices, this time for different models, i.e. BMF with different priors and different batch sizes. The top row contains plots for the situation with the smallest batch size (75) while the situation with the full batch size (512) is plotted in the bottom row. The size of the given samples is 60k points.

Note the behavior when the algorithm was started at a random point. From the three models only BMF without regularization/with flat prior shows any sign of development towards a usable solution, although the solution is very unbalanced between X and Y,

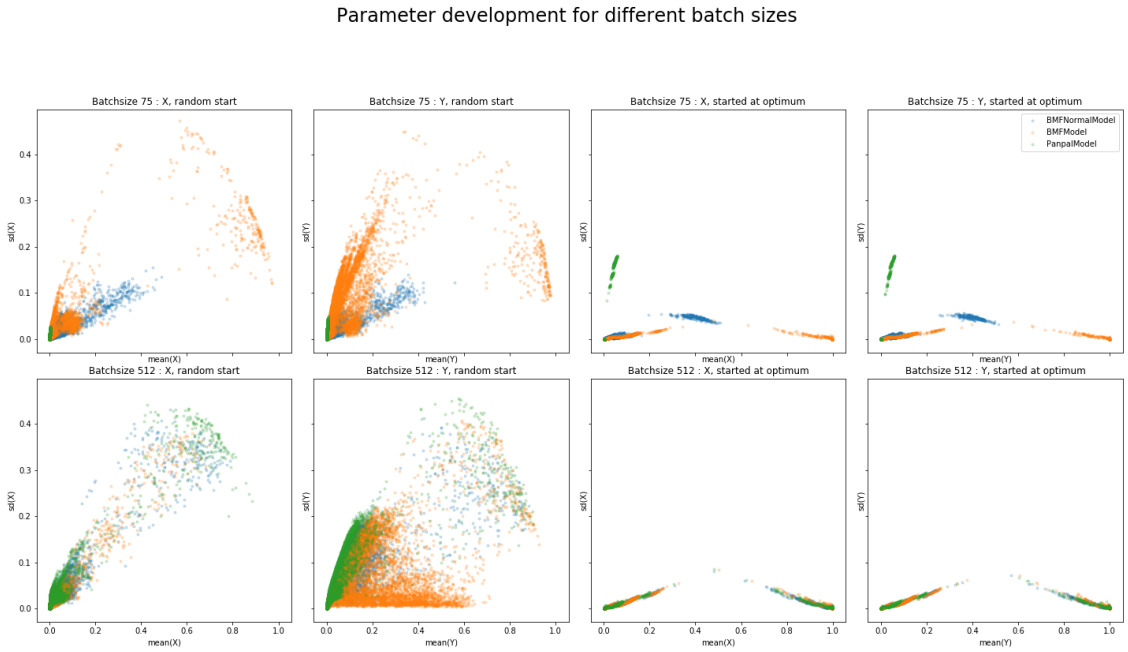

Figure 4.8: Scatter plot of the mean values vs. the standard deviations for parameter matrices X, Y for BMF with different priors. Clean dataset, correct factorization rank (30).

an effect which we already observed. The means for PanPAL (BMF with Laplace prior) have completely contracted towards 0. The means for BMF with Normal prior show some development. With full batch size, the means develop differently and an early stage clustering of likely zeros and ones is already recognizable, even though the sampler is not converged yet.

When started from an integer optimum and with small batch size the means of the models with non-flat prior shift towards zero. They stay near the initial solution when using the full batch size. As we already discussed above, staying near the optimum when started from one is the expected behavior.

Figure 4.9 shows a plot of the parameter means vs their mean gradients over the whole sampling process. In the lower row we have again plotted the situation for the full batch size. There we can see that most of the mean gradients are distributed around zero. This distribution is narrower for the cases where the sampler was started at an integer optimum than for those where the sampler was started randomly. The gradient means were taken over *all* samples, meaning that the gradient means include information about the early stages of the sampling process where the algorithm moved towards a high density region and experienced steep gradients. When started from the integer optimum the gradients were relatively flat to begin with and as such the whole distribution narrows down.

Looking at the gradient mean plots for the smallest batch size however reveals a surprise. Only the distribution for the mean gradient of BMF with flat prior has developed

towards the expected form. The gradients for BMF with Laplace prior have contracted into a small interval around 0.5 while the gradients for BMF with Normal prior show a linear correlation with the parameter mean. This effect occurred most likely because there is not enough data in these models. It seems, that 75 data cases is not enough to override the prior influence in a meaningful way in the BMF model with prior. Similar results occurred independent of the chosen starting position, while in the full batch size case the algorithm stayed in very close proximity to the integer optimum when started there. Usually the prior assumptions are constructed in such a way that they support the likelihood when there are not enough data cases. In the case we just discussed, the prior does not support the likelihood as the results with little data and with a lot of data are totally different.

In theory, good priors for the (relaxed) BMF likelihood would have to be U-shaped, i.e. Beta with $\alpha, \beta < 1$, as they put the major part of their probability mass on zero and one, thereby enforcing a boolean result. However, in combination with the BMF likelihood the mean sampling result was in any case a set of zero matrices. This behavior occurs likely because the relaxed BMF model needs to compute inferences in the face of ambiguous data data as parameter assignments between zero and one. Taking most of the probability mass off this part of the solution space suppresses the capability of the BMF model to handle ambiguous situations. The next best choice is always to set a parameter to zero or a small value if the input data was sparse. In addition a large additional amount of probability mass on zero is added by the prior.

As the U-shaped priors did not yield meaningful results at all, we used the next best assumption, namely sparsity priors. However, as we have seen from the results with small batch sizes, sparsity priors in the given form as Normal or Laplace distributions do not support the BMF assumptions. They support sparsity of the parameter matrices, but that is a consequence of the data in the given model. The assumption that many near zero values in the parameter matrices are a reasonable result is clearly not the meaning of "sparsity" implied by the user.

4.2.4 Mismatched factorization rank

Up until now we have only discussed situations where a nearly perfect reconstruction (up to the noise) term was possible as the factorization rank of the model matched the one used when generating the dataset. We have seen, that in this case the model converges towards the correct solution or a permutation thereof and the probabilistic uncertainty reduces with increasing sample size. For a real dataset we usually do not know the factorization rank. Even though we could search for it by enumeration of ranks as done by Hess et al.[HMP17], it it interesting to see how the rank mismatches are expressed in the probabilistic uncertainty of the sample.

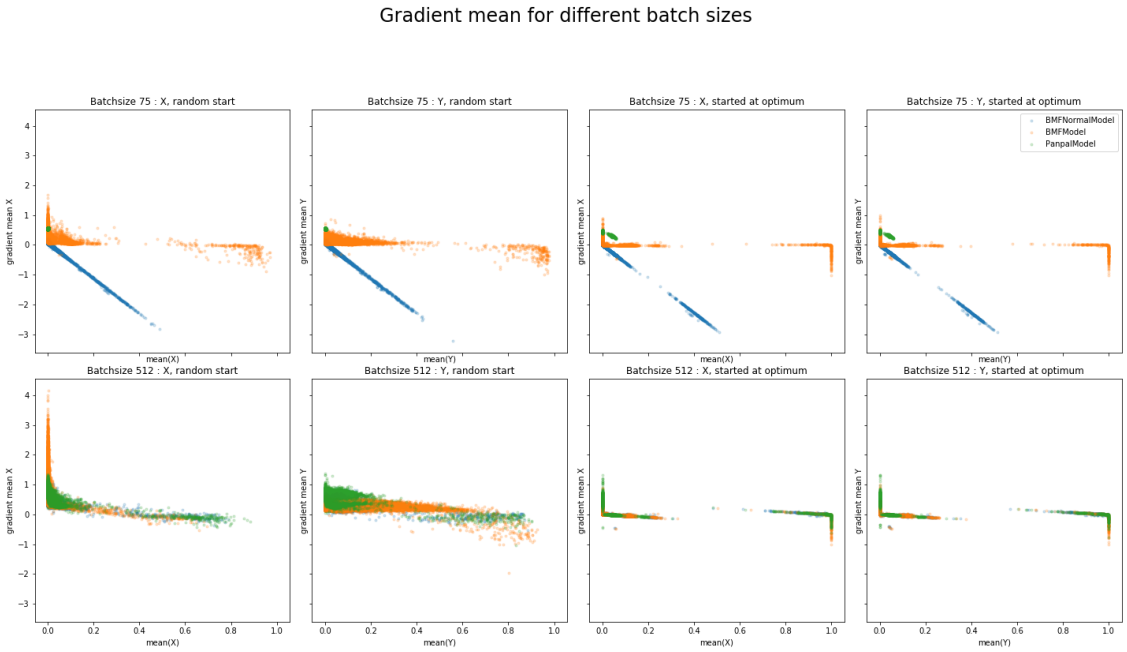

Figure 4.9: Scatter plot of the mean values vs. the mean gradient for parameter matrices X, Y for BMF with different priors. Clean dataset, correct factorization rank (30).

In figure 4.10 we have shown the results of a run on the "clean" dataset using BMF with a Normal prior. This diagram shows scatter plots of means vs. standard deviations just as in figure 4.7. However we have changed the factorization rank of the model to 20, whereas the factorization rank of the data is 30. The plots in the top row again show the situation when the sampler was started from a random location. The plots in the bottom row show the situation when the sampler was started by using one of the known optimal matrices but cutting off 10 columns to match the reduced factorization rank. In the upper row of figure 4.10 we observe a situation which does not seem to be too different from the situation in figure 4.7 where the factorization rank matched the dataset. In both plots we see a cluster around zero with low to medium uncertainty and values with very high uncertainty which detach and form a cluster around 0.8 to one with increasing sample size. The lower row of figure 4.10 shows significant differences compared to figure 4.7. The probabilistic uncertainties have not settled at low values. The largest uncertainties exist around 0.2 but they stay higher over the whole interval even after 600k iterations

The situation with a factorization rank that is too large is shown in figure 4.11. Here the samples scatter plots in the top row were again created by starting the sampler from a random position and increasing the factorization rank of the model to 40. The samples for the plots in the lower row were created by taking the known optimal parameter matrix and extending it by 10 columns of uniformly random values in $[0, 1]$. While the top row again shows cluster formation and the movement of values towards 0.8 to one, the behavior

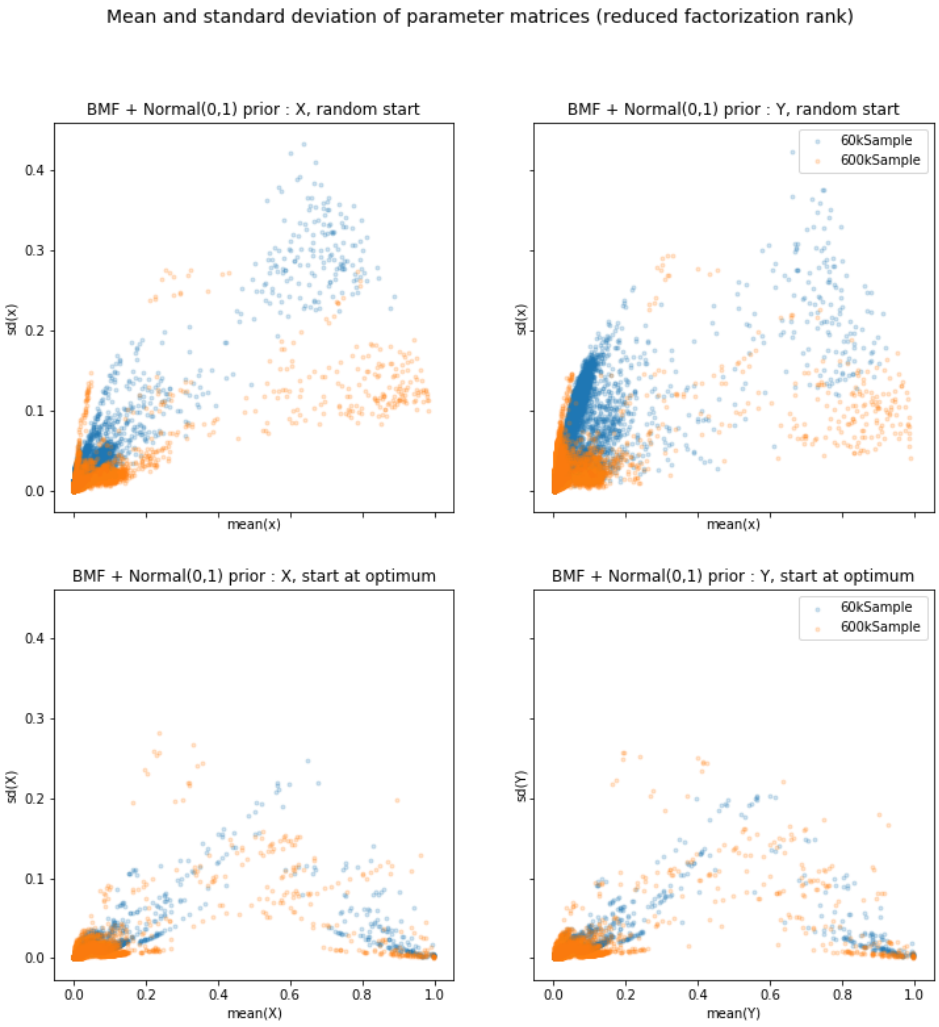

Figure 4.10: Scatter plot of the mean values vs. the standard deviations for parameter matrices X, Y for a BMF Model with Normal(0,1) prior. Clean dataset, reduced factorization rank (20) and maximum batch size (512).

shown in the lower row differs in a significant way from the behavior shown in figure 4.7. For most values near one the forming of a linear relationship between means and uncertainties can be observed. However near zero a cluster has formed which does not resolve even with increasing sample size. Also at higher sample size some values with very high uncertainties in comparison to the rest exist in $[0.2, 0.8]$.

These results show that a mismatched factorization rank finds a distinct expression in the probabilistic uncertainty reported by the model that persists even with increasing sample size. However it also shows again, that a converged sample is necessary to be able

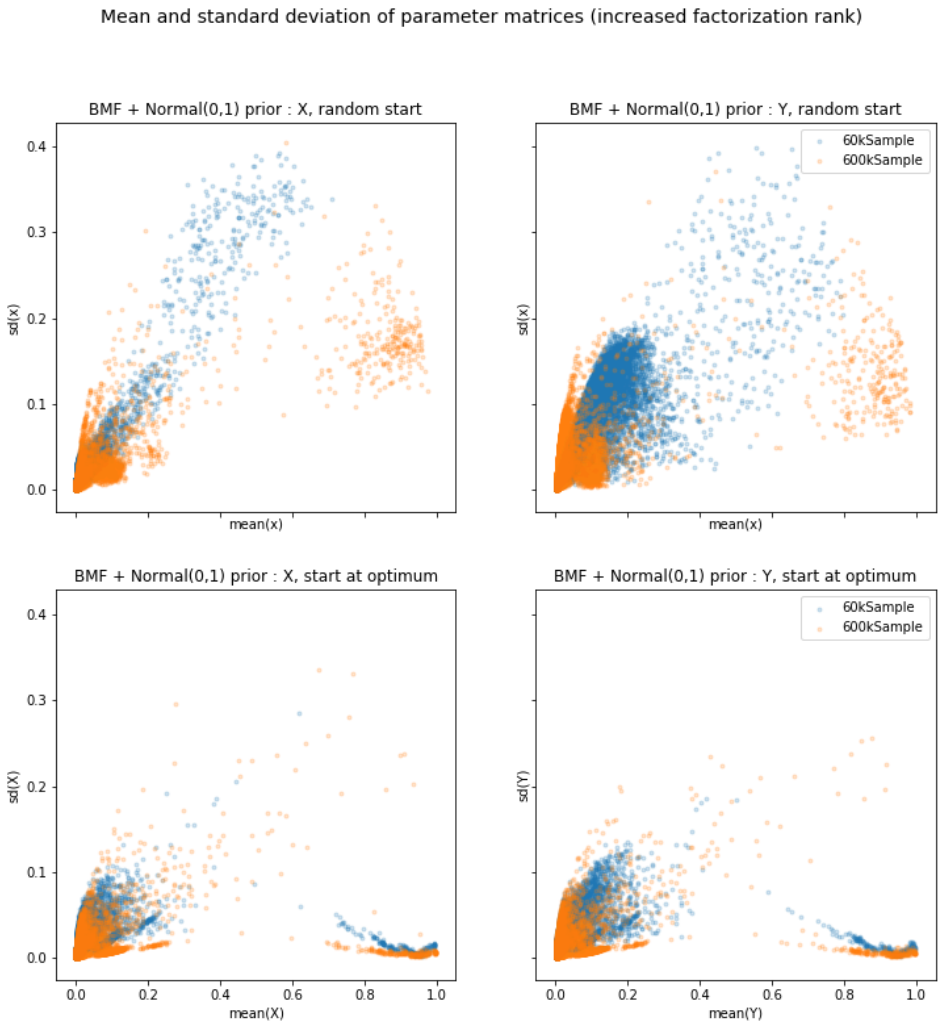

Figure 4.11: Scatter plot of the mean values vs. the standard deviations for parameter matrices X, Y for a BMF Model with Normal(0,1) prior. Clean dataset, increased factorization rank (40) and maximum batch size (512).

to make any reasonable inference about the parameter values or other model properties like the factorization rank.

4.2.5 Reconstruction by grid search and thresholding

In the last few sections we discussed the behavior of the relaxed probabilistic BMF model with respect to model ambiguities, uncertainties and the variation of parameters and models. However, we still need to obtain a result in the original boolean domain. This is achieved by reconstructing boolean matrices from the real valued parameter matrix. A known reconstruction method is the use of a thresholding function to binarize the real

valued matrices [HMP17]. A thresholding parameter that approximately minimizes the reconstruction error can be determined by grid search. However, one of the goals of this thesis is to find an alternative method where we can use the additional information provided by the probabilistic uncertainty to develop an alternative binarization method. While we have not found a method based on the sampling uncertainty, we might have asked the wrong question. We begin this section with an evaluation of the reconstruction errors wrt. the thresholding method for different models and parameterizations. Afterwards we provide an argument why the search for a probabilistic reconstruction method should be continued with the model ambiguities instead of the probabilistic uncertainties.

To find the optimal reconstruction threshold we choose the reconstruction threshold τ out of a set of candidates and use it to binarize the parameter matrices X, Y using the same τ for both matrices. This is in contrast to the method by Hess et al. where the thresholds for both matrices may differ. From our point of view, differing thresholds imply a different definition of the truth values for each factor matrix. Such a difference would only be justified if we knew that the BMF problem itself was biased in the interpretation of truth values. However, while solutions may become biased, the problem itself is not. As such we do not see a reason to apply different definitions of truth and falsehood to each parameter matrix.

Given the noise free data matrix C and parameter mean matrices X, Y we define the thresholded BMF as follows

$$G(\tau) = C - \theta_1 \left[\theta_{\tau_i}(Y)\theta_{\tau_i}(X^T)\right] \tag{4.3}$$

where $\theta_1(0) = 0$ and $\theta_1(n) = 1$ for $n \geq 1$ is the threshold imposed by the boolean $\vee$ operation. Using this thresholding model we have implemented the threshold grid search as shown in algorithm 4.1.

Algorithm 4.1 Binarization with threshold

Require: index set $I = \{1, 2, \cdots, 99\}$

1: for all $i \in I$ let $\tau_i \leftarrow 0.01i$
2: $\epsilon = \min_{i \in I} G(\tau_i)$
3: $i^* = \arg\min_{i \in I} G(\tau_i)$
4: $\tau_{\min} = \tau_i \quad \text{s.t. } i = \min i^*$
5: $\tau_{\max} = \tau_i \quad \text{s.t. } i = \max i^*$
6: $\epsilon_{0.5} = G(0.5)$

The algorithm takes an index set with indices from 1 to 99 that determines the search grid size. First we calculate the threshold related to each index by multiplying the index with 0.01, yielding our parameters for the grid search. We have left out zero as well as 100 as these parameters would force all values of the matrices either to one or to zero and are

thus unrealistic given the problem setting. We calculate the minimal reconstruction error over the grid in line 2 and its index set i^* in line 3 because multiple thresholds might yield the same reconstruction error. From this index set i^* we determine the threshold at the smallest index $\tau_{\min}$ and the one at the largest index $\tau_{\max}$ in lines 4 and 5. For comparison we also calculate the reconstruction error at $\tau = 0.5$ as this is the binarization threshold used by Rukat et al. in their OrMachine[RHTY17].

Applying this routine to our BMF samples with 60k points and factorization rank 30, which matches with the original factorization rank, yielded the results shown in table 4.1 where the sampler was started at a random position and table 4.2 where the sampler was started at an integer optimum. Note that independent of the way the algorithm was started and configured, most of the resulting threshold intervals do not include 0.5. This situation is shown in figure 4.12. The larger part of the threshold intervals is located below 0.5. The reconstruction error at $\tau = 0.5$ is almost never the lowest error, which is also confirmed by the given error values ($\epsilon_{0.5}$).

index	dataset	model	batch size	$\epsilon_{0.5}$	ϵ	$\tau_{\min}$	$\tau_{\max}$
0	clean	BMF+flat	75	87.0	72.0	0.41	0.44
1	clean	BMF+flat	250	937.0	928.0	0.46	0.46
2	clean	BMF+flat	512	1431.0	1325.0	0.42	0.42
3	clean	BMF+Normal	75	1919.0	713.0	0.19	0.19
4	clean	BMF+Normal	250	383.0	220.0	0.42	0.42
5	clean	BMF+Normal	512	821.0	552.0	0.32	0.32
6	clean	PanPAL	75	1919.0	1919.0	0.01	0.99
7	clean	PanPAL	250	1919.0	1918.0	0.03	0.03
8	clean	PanPAL	512	551.0	294.0	0.30	0.34
9	noisy	BMF+flat	75	732.0	722.0	0.48	0.48
10	noisy	BMF+flat	250	1594.0	1551.0	0.53	0.53
11	noisy	BMF+flat	512	1938.0	1913.0	0.58	0.60
12	noisy	BMF+Normal	75	1885.0	407.0	0.26	0.26
13	noisy	BMF+Normal	250	1016.0	876.0	0.39	0.39
14	noisy	BMF+Normal	512	1494.0	1469.0	0.47	0.47
15	noisy	PanPAL	75	1919.0	1919.0	0.01	0.99
16	noisy	PanPAL	250	1195.0	1096.0	0.38	0.40
17	noisy	PanPAL	512	1533.0	1497.0	0.48	0.48

Table 4.1: Reconstruction errors after binarization for 0.5 threshold and optimal threshold intervals. Sampler initialized at random point. Sample size 60k, correct rank (30).

Table 4.1 shows further properties of the results. The overall reconstruction error is lower when the model is started from a known integer optimum than when started from a random initial position. Also the reconstruction error for the noisy dataset is generally higher than for the clean dataset. While this does not come as a surprise, there are also some surprising results. Note, that for a batch size of 75 there was not enough data in the model to override the prior influence. As such the parameter mean values appeared skewed towards zero, except for the BMF model with flat prior. Looking at table 4.1 we observe the lowest reconstruction error for the clean dataset for BMF+flat prior with a batch size of 75 (index 0). It is followed by the BMF+Normal model at a batch size of 250 (index 4) and by the PanPAL model with a batch size of 512 (index 8). While the BMF+Normal model has a larger reconstruction error at the batch sizes 75 and 512, the PanPAL model yields a reconstruction error around the number of one values in the data matrix (1918) for both batch sizes 75 and 250. This indicates that the binarized matrices are zero matrices. It follows that the threshold should have been chosen below 0.01 as most of the parameter mean values must lie even below that value. Even though it is unlikely, here we may have missed a better reconstruction. On the positive side however, this case shows the major drawback of grid search: the precision of the method depends on the users idea of reasonable candidate values.

Another indication for the influence of prior and batch size can be found in table 4.2. It stands out that for the clean dataset the grid search procedure could find thresholds such that in all cases the reconstruction error is zero. In the cases with batch size 75 and a nonflat prior (index 3 and 6) the 0.5 threshold again results in the maximum error in ones and as such in a zero matrix. The optimal threshold intervals are far below 0.5 with the PanPAL model (index 6) achieving optimal reconstruction at $\tau = 0.01$. Recall that for this case the parameter mean values were shifted very strongly towards zero, while their standard deviations had been considerably increased. Even though there is a considerable skew in both axes, the general structure of the solution is correct and an optimal binarization can be obtained.

All in all these observations show that the correct structure of the parameter matrices can form very early in the sampling process, even though the marginal distributions do not imply any convergence at all. As we have observed earlier, convergence to the correct distribution will take a very large sample, more than the two million points which we sampled for some of the models.

4.2.6 Defects due to imposed boundary conditions

As we discussed earlier, we constructed the sampler in such a way that the results are clamped to the boundary values [0,1]. We need to address this topic here again, because we know and accepted that clipping will lead to defects in the sample. The probabilistic

index	dataset	model	batch size	$\epsilon_{0.5}$	ϵ	$\tau_{\min}$	$\tau_{\max}$
0	clean	BMF+flat	75	8.0	0.0	0.40	0.47
1	clean	BMF+flat	250	8.0	0.0	0.39	0.48
2	clean	BMF+flat	512	0.0	0.0	0.38	0.50
3	clean	BMF+Normal	75	1918.0	0.0	0.17	0.19
4	clean	BMF+Normal	250	8.0	0.0	0.34	0.43
5	clean	BMF+Normal	512	8.0	0.0	0.36	0.48
6	clean	PanPAL	75	1919.0	0.0	0.01	0.01
7	clean	PanPAL	250	8.0	0.0	0.29	0.42
8	clean	PanPAL	512	8.0	0.0	0.34	0.48
9	noisy	BMF+flat	75	125.0	104.0	0.46	0.47
10	noisy	BMF+flat	250	72.0	72.0	0.48	0.50
11	noisy	BMF+flat	512	56.0	40.0	0.53	0.54
12	noisy	BMF+Normal	75	1549.0	111.0	0.29	0.29
13	noisy	BMF+Normal	250	63.0	63.0	0.50	0.50
14	noisy	BMF+Normal	512	32.0	32.0	0.50	0.50
15	noisy	PanPAL	75	1919.0	0.0	0.01	0.01
16	noisy	PanPAL	250	55.0	48.0	0.46	0.47
17	noisy	PanPAL	512	48.0	32.0	0.53	0.53

Table 4.2: Reconstruction errors after binarization for 0.5 threshold and optimal threshold intervals. Sampler initialized at known optimal point. Sample size 60k, correct factorization rank (30).

interpretation of clipping is to assign a zero probability to every MCMC state outside $[0, 1]$ but mapping the whole probability mass outside of $[0, 1]$ onto the boundaries, thereby skewing the boundary regions of the probability distribution.

Our reason for this rather desperate measure is as follows. The BMF problem is defined for the parameter interval $[0, 1]$ and the high density regions lie around zero and around one, with high density regions on and outside these boundaries in the relaxed BMF problem, forcing the sampler to overstep the boundary with some of the marginal parameters on each sample.[2] As a consequence doing a Metropolis-Hastings rejection Test will have a 100% reject rate, so we cannot apply this method here although it would be correct in theory[RT96]. Also, we cannot simply continue the sampling process from the point outside the boundary because chances are high that the gradient points away from the boundary and our distance to the boundary is far more than 1σ of our noise term so

[2] Already with a 100 dimensional standard Normal distribution that is truncated through its maximum, around 10% of the marginals go out of bounds with each sample.

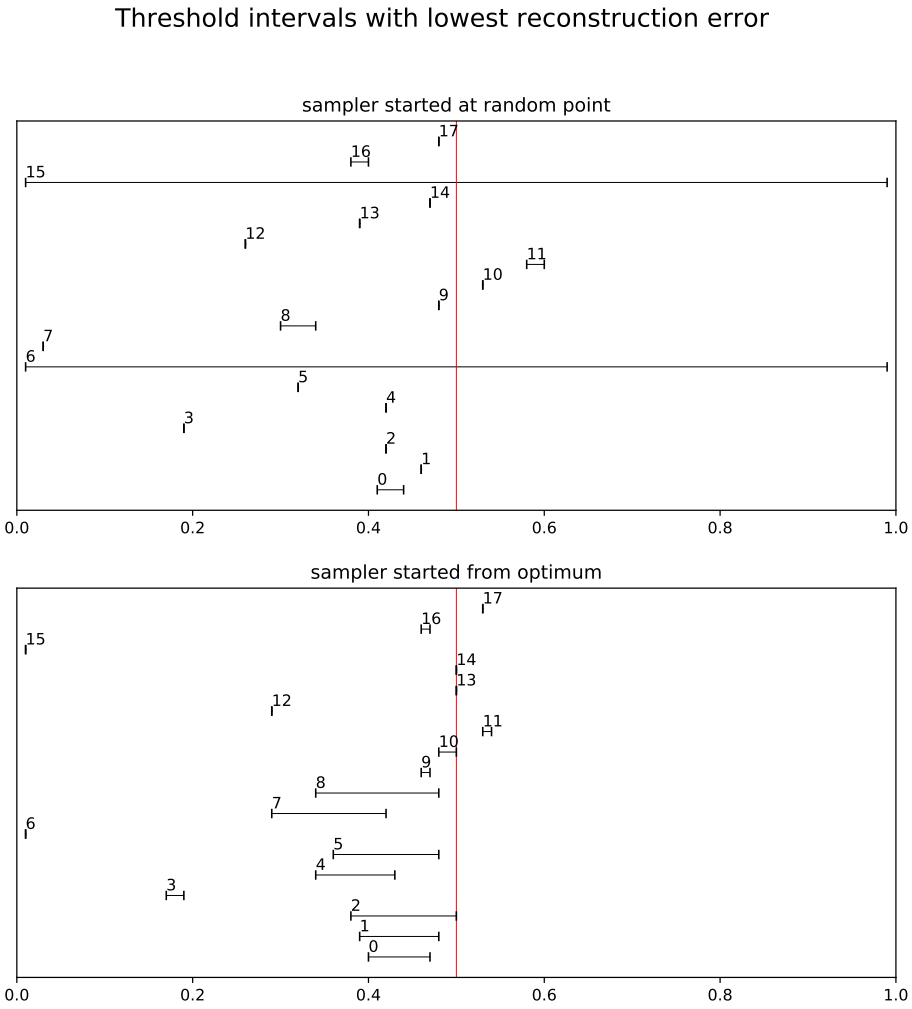

Figure 4.12: Overview of the minimal reconstruction threshold intervals. The red line marks the 0.5 threshold. Sample size 60k, correct factorization rank (30). Intervals are numbered by their indices in tables 4.1 and 4.2.

the chance to randomly jump back is very small. At this point the sampler has become stuck.

Possible alternatives are some reparameterization techniques, unfortunately they do also not work. Reparameterization of the whole problem to [-1,1] solves the out of bounds problem for the zero boundary, but not for the boundary at one. A reparameterization to the reals by the use of a sigmoid transformation as implemented by Rukat et al.[RHTY17] is a reasonable solution for the case described in their paper, because they are still using an integral state space in combination with Gibbs sampling. Applying the same reparameterization in a continuous state space with the IASG leads to a sampling trajectory that

diverges into the almost zero gradient regions of the sigmoid function. We are not inclined to do Gibbs sampling either, as it would undermine the larger part of the idea behind this thesis.

The necessity to use clipping is unfortunate, however we still obtained results that have lead to some interesting insights. In the next section we will examine if we can obtain good reconstructions of the boolean parameter matrices from these results.

4.2.7 Ambiguity vs. probabilistic uncertainty

In section 4.1.2 we showed cases where the relaxed BMF model reacts to conflicting as well as non conflicting information in the data cases. Non conflicting information are assignments to the values of the data matrix for which a perfect reconstruction is possible, i.e. the reconstruction error is zero. However even in the low dimensional example a large part of the data configurations had conflicts, so that the reconstruction error could not be brought to zero using only integral assignments to the variables. In these cases the high density regions of the log-likelihood moved to non integral values for some or all variables, taking advantage of the nonlinearity of the expression and yielding a further reduction in the reconstruction error. While this situation might arise by the construction of the data matrix alone (e.g. by overlapping tilings) it is guaranteed by adding a noise term. In datasets of unknown internal structure we can think of conflicting information as being the norm rather than the exception.

We interpreted this choice of intermediate values upon conflict as the model signaling ambiguities in the data which it does not know to resolve. Elaborating on this thought, we can further interpret these ambiguities as a specific form of model uncertainty. The other form of uncertainty which concerns us in this thesis is probabilistic uncertainty. Probabilistic uncertainty can in many cases be used as a diagnostic for issues with the model[3] and by further processing it also for issues with the data [4].

For the following argumentation, please understand ambiguity and probabilistic uncertainty as two different dimensions of uncertainty. With this interpretation in mind, let us again take a look at figure 4.7. Note, that the x-axis on all plots shows the mean values of the elements of the parameter matrices for different samples, while the y-axis shows the standard deviations. We can interpret these diagrams now as plots of ambiguity vs. probabilistic uncertainty.

In the upper row of the figure we show the development of the uncertainties for different sample sizes, i.e. when the sampler slowly converges in distribution starting from a random configuration. For each sample we can infer a linear relationship between both

[3] e.g. in Bayesian model selection

[4] e.g. in active learning, when uncertain model outputs are used to determine weakly represented regions of the feature space

uncertainties, where the probabilistic uncertainty is reduced over time due to the collection of more evidence within the larger sample.

In the lower row we show the situation when the sampler is started from an integer optimum, i.e. the situation where sure convergence within the correct high density region has been anticipated. We expect that the samples shown in the top row, when extended by some million more samples would converge to a situation not unlike these shown in the bottom row. There the linear relationships between ambiguity and probabilistic uncertainty are even more pronounced. This implies that there is no more information in the probabilistic uncertainty than in the ambiguities expressed by the model. Furthermore the probabilistic uncertainty diminishes with increasing sample size.

Thus we have two situations where we cannot use the probabilistic uncertainty. In the early stages of taking a sample the probabilistic uncertainty values are invalid as the sampler is not yet starting to converge. With increasing sample size the probabilistic uncertainty goes towards zero and the ambiguities contain all the information.

All this leads to the conclusion, that the ambiguities provided by the BMF to NMF relaxation could be the better measure of uncertainty for probabilistic BMF. This is true however only for cases where the factorization rank is matched. In cases where it is not matched we have shown evidence in figures 4.10 and 4.11 that the probabilistic uncertainty still carries information which is at least partially distinct from the ambiguities, even with increasing sample size.

Chapter 5

Recapitulation

5.1 Conclusion

We began the work on this thesis with the idea that trust in machine learning methods cannot only be blind trust. Methods should provide means to support the user to build trust in the results when using the method. In this way some of the burden to perform time consuming analyses on data and results could be lifted from the user.

One method which provides a built in diagnostic in terms of uncertainties is probabilistic modeling, so we opted to build a probabilistic model and explore its uncertainties. We chose Boolean Matrix Factorization as the machine learning model, as it is one of the more popular models. However, as we have discussed, there are reasons to have little trust in it. In addition relatively little research exists in which it is cast as a probabilistic model, while the bulk of the research focuses on optimization based inference.

In addition to trusting the model, a user must also have trust in the inference method. This can become difficult with many inference methods as most of them have parameters related to learning rates and other internal settings. These are often not well understood and it is not clear how to set them for specific problem instances, requiring lots of tinkering by the user. With the IASG we used a probabilistic inference method that is based on Langevin diffusions and can tackle the challenges of large and high dimensional datasets but which comes without most of the usual settings. Up to the specific modifications for BMF it worked out of the box. With these tools in hand we constructed a probabilistic inference method for BMF by reinterpreting the related relaxation of BMF to NMF. We explored recently found insights about the connection between inference via optimization and inference via sampling to affirm for ourselves that the combination of the model and inference method is sound.

After constructing the model and the algorithm we analyzed their behavior by generating multiple samples with varying parameters and plotting as well as discussing their properties. We gained the impression that that we have only scratched the surface and

that there are many more ways in which the model could be further examined. Yet we demonstrated the amount of work that is involved in understanding and gaining trust in such a method.

While we knew that we could negatively influence our results by using clipping to sample from a truncated distribution, we saw no other possibility as we have not found anything that comes close to a viable alternative for our concrete situation. We believe however, that deeper research into physical models of motion could have brought up better answers.

Some of the other results of our analysis were unexpected. The strong influence of the prior and thus the slow convergence of the sampler together with the resulting issues for binarization were surprising. It was also surprising that the sparsity assumption works as a regularizer for the relaxed BMF optimization problem, while it represents model assumptions that are invalid within a probabilistic context. The Beta prior, which is theoretically the right choice, interfered with the behavior of the sampling process. Within our choice of priors the flat prior seems to be the most unintrusive choice, but we also observed situations where the model could not converge to a sensible solution. We regret that we did not look into the more advanced regularizers provided by Hess et al. but in the context of IASG they would most likely have found limited usability, as they are not constructed from quadratic polynomials.

Another main goal was to use the uncertainties provided by the model and create a binarization method that uses probabilistic reasoning to infer the optimal threshold. This task held the next surprise. The skew induced by sparsity priors led to slow convergence, and even after millions of samples had been drawn some of the sampled instances were still in the convergence process. We noticed not only the issues with convergence, but also that the prior worked against the intention of the model in cases where the sampler was run with a low batch size. The effect of the prior with a low amount of data cases is known and usually appreciated, as it supports the model assumptions. In this case the sparsity priors did not support the model assumptions and thus led to false inferences in the case of low batch sizes. This leads to the suspicion, that sparsity priors might not in general be the right kind of prior for BMF models.

Although we did not create a new binarization method and thus did not deliver on our intention to lift some of the trust burden from the users of BMF, we feel that the exploration of the topic itself was worthwhile, as it demonstrates some of the work involved and the pitfalls encountered when developing, using, understanding and trusting new machine learning methods.

Still we believe that the overall outcome of this thesis is a positive one, rather than a negative one. It enabled us to apply state of the art sampling methods for inference, test them on very high dimensional models of BMF and gain new insights about these models

related to the role of uncertainty. This in itself was an educational endeavor which has led to several new questions.

5.2 Open questions

While the search for the existence of a probabilistic binarization method is still not finished, we take away the following new questions.

The first question relates to the calculation of the optimal preconditioner of IASG, which was calculated by Mandt. et al. under the assumption of locally quadratic functions. Can an optimal preconditioner be determined for locally Lipschitz functions? The extension to this function class would on the one hand enable IASG as an inference method for the whole PAL-Tiling framework and on the other hand broaden its general usability.

The second question relates to the problem of using a Langevin diffusion to sample a truncated distribution. The simulation of physical models with boundary conditions is well known, however the boundary conditions in these models are meant to influence the processes within, as they do in the real world. Can boundary conditions be constructed, which correct for their effects and do not skew the sampled distribution? We believe that further research into physical models of motion as well statistical physics could yield insights about the feasibility of this idea.

The third question regards the probabilistic model for BMF which we have constructed assuming that the rank is fixed throughout the sampling process. To create a model where the rank in itself is a random variable calls for the use of more sophisticated sampling methods, however one could probably obtain more insights about the relation of factorization rank and probabilistic uncertainty. How would model and inference process to be constructed in this case?

Our last question concerns the relation of probabilistic uncertainty and ambiguity. How do they relate? We have shown evidence that they must be partially related but that probabilistic uncertainty carries different information in the case of a mismatched factorization rank. Determining the relationship of these types of uncertainty could probably open up deeper insights into the determination of threshold values and how to diagnose a factorization rank mismatch for BMF. As we have used the BMF to NMF relaxation, we see the possibility that insights from the NMF research literature will yield answers to this question.

Chapter 6

Appendix

6.1 CD-ROM with code and experiment data

List of Figures

List of Tables

List of Algorithms

Bibliography

[AdDJ03] Christophe Andrieu, Nando de Freitas, Arnaud Doucet, and Michael I Jordan. An Introduction to MCMC for Machine Learning. *Machine Learning*, 50(1):5–43, January 2003.

[Bet17] Michael Betancourt. A Conceptual Introduction to Hamiltonian Monte Carlo. *arXiv:1701.02434 [stat]*, January 2017.

[Bot10] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT 2010 - 19th International Conference on Computational Statistics, Keynote, Invited and Contributed Papers*, pages 177–186, 2010.

[GC11] Mark Girolami and Ben Calderhead. Riemann manifold Langevin and Hamiltonian Monte Carlo methods: Riemann Manifold Langevin and Hamiltonian Monte Carlo Methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, March 2011.

[GCD+13] Andrew Gelman, John B Carlin, Hal S Sternand David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian Data Analysis*. CRC Press, 2013.

[HHG14] Jose Miguel Hernandez-Lobato, Neil Houlsby, and Zoubin Ghahramani. Stochastic Inference for Scalable Probabilistic Modeling of Binary Matrices. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 379–387. PMLR, June 2014.

[HMP17] Sibylle Hess, Katharina Morik, and Nico Piatkowski. The PRIMPING routine—Tiling through proximal alternating linearized minimization". *Data Mining and Knowledge Discovery*, 31(4):1090–1131, July 2017.

[HPM18] Sibylle Hess, Nico Piatkowski, and Katharina Morik. The Trustworthy Pal: Controlling the False Discovery Rate in Boolean Matrix Factorization. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, pages 405–413, 2018.

[HRC15] Changwei Hu, Piyush Rai, and Lawrence Carin. Zero-Truncated Poisson Tensor Factorization for Massive Binary Tensors. In *UAI*, 2015.

[Jor06] Jorge Nocedal, Stephen J. Wright. *Numerical Optimization*. Springer, New York, 2 edition, 2006.

[KB09] T. Kolda and B. Bader. Tensor Decompositions and Applications. *SIAM Review*, 51(3):455–500, 2009.

[KBV09] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix Factorization Techniques for Recommender Systems. *Computer*, 42(8):30–37, August 2009. Place: Los Alamitos, CA, USA.

[LS99] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788, October 1999.

[MCF15] Yi-An Ma, Tianqi Chen, and Emily B. Fox. A Complete Recipe for Stochastic Gradient MCMC. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pages 2917–2925. MIT Press, 2015.

[MGNR07] Edward Meeds, Zoubin Ghahramani, Radford M. Neal, and Sam T. Roweis. Modeling Dyadic Data with Binary Latent Factors. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 977–984. MIT Press, 2007.

[MHB17] Stephan Mandt, Matthew D Hoffman, and David M Blei. Stochastic Gradient Descent As Approximate Bayesian Inference. *J. Mach. Learn. Res.*, 18(1):4873–4907, January 2017.

[Mie06] Miettinen, Pauli and Mielikäinen, Taneli and Gionis, Aristides and Das, Gautam and Mannila, Heikki. The Discrete Basis Problem. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *Knowledge Discovery in Databases: PKDD 2006*, pages 335–346, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[MSH02] J.C. Mattingly, A.M. Stuart, and D.J. Higham. Ergodicity for SDEs and approximations: Locally Lipschitz vector fields and degenerate noise. *Stochastic Processes and their Applications*, 101(2):185–232, October 2002.

[MV14] Pauli Miettinen and Jilles Vreeken. MDL4BMF: Minimum Description Length for Boolean Matrix Factorization. *ACM Trans. Knowl. Discov. Data*, 8(4):18:1–18:31, October 2014.

[PB14] Neal Parikh and Stephen Boyd. Proximal Algorithms. *Found. Trends Optim.*, 1(3):127–239, 2014. Place: Hanover, MA, USA.

[PT94] Pentti Paatero and Unto Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, June 1994.

[PT13] Sam Patterson and Yee Whye Teh. Stochastic Gradient Riemannian Langevin Dynamics on the Probability Simplex. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3102–3110. Curran Associates, Inc., 2013.

[RHHC15] Piyush Rai, Changwei Hu, Matthew Harding, and Lawrence Carin. Scalable Probabilistic Tensor Factorization for Binary and Count Data. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 3770–3776, Buenos Aires, Argentina, 2015. AAAI Press.

[RHTY17] Tammo Rukat, Chris C. Holmes, Michalis K. Titsias, and Christopher Yau. Bayesian Boolean Matrix Factorisation. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2969–2978, International Convention Centre, Sydney, Australia, August 2017. PMLR.

[RHY18] Tammo Rukat, Chris Holmes, and Christopher Yau. Probabilistic Boolean Tensor Decomposition. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4413–4422. PMLR, July 2018.

[RM51] Herbert Robbins and Sutton Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[Rob95] Christian P. Robert. Simulation of truncated normal variables. *Statistics and Computing*, 5(2):121–125, June 1995.

[Ros95] Guido Rossum. Python Reference Manual. Technical report, CWI (Centre for Mathematics and Computer Science), Amsterdam, The Netherlands, The Netherlands, 1995.

[RT96] Gareth O. Roberts and Richard L. Tweedie. Exponential Convergence of Langevin Distributions and Their Discrete Approximations. *Bernoulli*, 2(4):341, December 1996.

[SFBB09] Andreas P. Streich, Mario Frank, David Basin, and Joachim M. Buhmann. Multi-assignment clustering for Boolean data. In *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*, pages 1–8, Montreal, Quebec, Canada, 2009. ACM Press.

[SM08] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the 25th International Conference on Machine Learning - ICML '08*, pages 880–887, Helsinki, Finland, 2008. ACM Press.

[SZY17] Jiarong Shi, Xiuyun Zheng, and Wei Yang. Survey on Probabilistic Models of Low-Rank Matrix Factorizations. *Entropy*, 19(8), 2017.

[WJ08] Martin J Wainwright and Michael I Jordan. Graphical Models, Exponential Families, and Variational Inference. *Foundations and Trends in Machine Learning*, 1, 2008.

[WT11] Max Welling and Yee Whye Teh. Bayesian Learning via Stochastic Gradient Langevin Dynamics. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 681–688, New York, NY, USA, 2011. ACM. Series Title: ICML '11.

[WZ13] Yu-Xiong Wang and Yu-Jin Zhang. Nonnegative Matrix Factorization: A Comprehensive Review. *IEEE Trans. on Knowl. and Data Eng.*, 25(6):1336–1353, 2013. Place: Piscataway, NJ, USA.

[YAM+15] Pavan Yalamanchili, Umar Arshad, Zakiuddin Mohammed, Pradeep Garigipati, Peter Entschev, Brian Kloppenborg, James Malcolm, and John Melonakos. ArrayFire - A high performance software library for parallel computing with an easy-to-use API. 2015.