Applications of Approximate Learning and Inference for Probabilistic Models

THÈSE N^{O} 7637 (2017)

PRÉSENTÉE LE 2 MAI 2017
À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS
LABORATOIRE POUR LES COMMUNICATIONS INFORMATIQUES ET LEURS APPLICATIONS 4
PROGRAMME DOCTORAL EN INFORMATIQUE ET COMMUNICATIONS

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Young Jun KO

acceptée sur proposition du jury:

Prof. B. Rimoldi, président du jury
Prof. M. Grossglauser, Dr M. Seeger, directeurs de thèse
Prof. T. Hofmann, rapporteur
Prof. J. Hensman, rapporteur
Prof. M. Jaggi, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2017

To my family

Acknowledgements

First of all, I would like to thank my two advisors, Matthias Seeger and Matthias Grossglauser. I am grateful to Matthias Seeger for giving me the opportunity to come to EPFL and join his lab. I greatly benefited from the breadth and depth of his knowledge, and much of this thesis is inspired by his prior work. I am especially grateful to Matthias Grossglauser, whose generosity allowed me to complete my dissertation. He welcomed me into his lab in the most unbureaucratic way. Thank you for your guidance, trust and support.

Many thanks to James Hensman, Thomas Hofmann, Martin Jaggi, and Bixio Rimoldi for instantly agreeing to join my thesis committee and to review the manuscript in spite of their extremely busy schedules.

I would like to thank the IC doctoral school for a great program and for supporting my transition. I am especially grateful to Corinne Degotte for patiently answering all my questions and removing all obstacles on the road to my defense.

I was very fortunate to have been part of two groups, both outstanding in terms of atmosphere and working conditions: IPG, comprising the labs headed by Michael Gastpar, Bixio Rimoldi, Emre Telatar, and Rüdiger Urbanke, and LCA "3.5", consisting of the labs headed by Matthias Grossglauser and Patrick Thiran. I would like to express my gratitude to the IPG family, to which the lab in which I started belonged as well. Thank you very much for allowing me to stick around and making me feel at home, even after I had to leave. A very special thanks to the IPG senior researchers Olivier Lévêque and Nicolas Macris for their kindness and to the permanent staff of both groups, particularly Muriel Bardet, Françoise Behn, Angela Devenoge, Patricia Hjelt and Damir Laurenzi, for all the help over the years and for making even the most mundane administrative matters, software updates, or travel arrangements a pleasant and hassle-free experience.

The great thing about this place is that the halls are populated with extraordinary people, some of which I was fortunate to have as colleagues. Thank you, Akshat, Alberto, Alla, Baker, Dan, Ehsan, Elie, Emti, Eren, Farid, Farnood, Hamed, Jean, Julien, Mani, Mathias, Mine, Mohamed Kafsi, Mohamed Karzand, Nick, Nicu, Nikos, Pablo, Runwei, Saeid, Serj, Sébastien, Vahid, Vassilis, Vincent, and William, for your company not only at work, but also on group outings, during coffee breaks, and dramatic foosball matches.

Special thanks to Christina for the good spirit and the Greek lessons, Marc D. for taking us climbing and ski touring, Marc V. for organizing an unforgettable trip to Zermatt, Marco for the bike trips and great dinners, Rafah for all the laughs, Rajai for enlightening lectures on various topics and for developing our theoretical carving skills, and Victor for saving the planet.

I am greatly indebted to Lucas Maystre for countless helpful discussions, joint projects, and for proofreading this thesis, among other things. To behold his commitment, perseverance, attention to detail, and kindness was a constant source of inspiration.

Due to a very dear group of people I am now looking back on a wealth of great memories of barbecues, bike trips, board-game/fondue/movie nights, hikes, skiing weekends, and travels.

Thank you, Camila and Karol, for all your help and fun in and out of the office and the best Caipirinhas.

Thank you, Genya and Stefano, for the artistic and musical element and your curiosity and enthusiasm for finding new ways to have a great time together.

Thank you, Lyusya, Ivan, and Sasha, for all the joy, berry picking, dinners, and Russian treats.

Thank you, Nastya and Adrian, for having me over so many times, picking my spirit up with highly intellectual movies, Romanian specialties of various consistency, and home-made sushi.

Thank you, Andrei, for being so generous with your time, knowledge, and help, no matter the situation.

Thank you, Brunella, for your patience, encouragement, organizational skills, and for seeing the good in people and the beauty in the world.

This work is dedicated to my family: my parents and my siblings Young-Eun, Young-In, and Mark. Thank you for your unconditional love.

Lausanne, 06 Mars 2017 Y.-J. Ko

Abstract

We develop approximate inference and learning methods for facilitating the use of probabilistic modeling techniques motivated by applications in two different areas. First, we consider the ill-posed inverse problem of recovering an image from an underdetermined system of linear measurements corrupted by noise. Second, we consider the problem of inferring user preferences for items from counts, pairwise comparisons and user activity logs, instances of implicit feedback.

Plausible models for images and the noise, incurred when recording them, render posterior inference intractable, while the scale of the inference problem makes sampling based approximations ineffective. Therefore, we develop deterministic approximate inference algorithms for two different augmentations of a typical sparse linear model: first, for the rectified-linear Poisson likelihood, and second, for tree-structured super-Gaussian mixture models.

The rectified-linear Poisson likelihood is an alternative noise model, applicable in astronomical and biomedical imaging applications, that operate in intensity regimes in which quantum effects lead to observations that are best described by counts of particles arriving at a sensor, as well as in general Poisson regression problems arising in various fields. In this context we show, that the model-specific computations for Expectation Propagation can be robustly solved by a simple dynamic program.

Next, we develop a scalable approximate inference algorithm for structured mixture models, that uses a discrete graphical model to represent dependencies between the latent mixture components of a collection of mixture models. Specifically, we use tree-structured mixtures of super-Gaussians to model the persistence across scales of large coefficients of the Wavelet transform of an image for improved reconstruction.

In the second part on models of user preference, we consider two settings: the global static and the contextual dynamic setting.

In the global static setting, we represent user-item preferences by a latent low-rank matrix. Instead of using numeric ratings we develop methods to infer this latent representation for two types of implicit feedback: aggregate counts of users interacting with a service and the binary outcomes of pairwise comparisons.

We model count data using a latent Gaussian bilinear model with Poisson likelihoods. For this model, we show that the Variational Gaussian approximation can be further relaxed to be available in closed-form by adding additional constraints, leading to an efficient inference algorithm.

In the second implicit feedback scenario, we infer the latent preference matrix from pairwise preference statements. We combine a low-rank bilinear model with non-parameteric item-feature regression and develop a novel approximate variational Expectation Maximization algorithm that mitigates the computational challenges due to latent couplings induced by the pairwise comparisons.

Finally, in the contextual dynamic setting, we model sequences of user activity at the granularity of single interaction events instead of aggregate counts. Routinely gathered in the background at a large scale in many applications, such sequences can reveal temporal and contextual aspects of user behavior through recurrent patterns. To describe such data, we propose a generic collaborative sequence model based on recurrent neural networks, that combines ideas from collaborative filtering and language modeling.

Key words: Probabilistic models, generalized linear and bilinear models, recurrent neural networks, approximate inference, image reconstruction, implicit feedback

Zusammenfassung

Wir entwickeln approximative Inference- und Lernmethoden, um die Anwendung probabilistischer Modellierungstechniken zu ermöglichen. Unsere Arbeit ist hierbei motiviert durch Problemstellungen aus zwei verschiedenen Anwendungsgebieten.

Im ersten Teil beschäftigen wir uns mit dem inversen Problem der Bildrekonstruktion aus unterdeterminierten und verrauschten linearen Messungen. Im zweiten Teil befassen wir uns damit, die Präferenzen von Nutzern für Gegenstände aus einem unübersichtlichen Angebot zu lernen, auf Grundlage von implizitem Feedback, wie etwa aggregierten sowie detaillierten Interaktionsprofilen und Vergleichsergebnissen zwischen zwei Gegenständen.

Plausible Bild- und Akquisitionsmodelle machen exakte probabilistische Inferenzen in der Regel unmöglich. Stichprobenbasierte Verfahren sind wegen der hohen Dimensionalität ineffektiv. Wir entwickeln daher deterministische approximative Inferenzalgorithmen für zwei Erweiterungen von dünnbesetzten linearen Modellen: erstens für eine sog. rektifiziert-lineare Poisson Likelihood und zweitens für baumartig strukturierte Super-Gauss'sche Mischverteilungen.

Die rektifiziert-lineare Poisson Likelihood ist ein alternatives Rauschmodell, sowohl relevant für Bildgebungsverfahren in der Astronomie und Biomedizin, die in Intensitätsbereichen operieren, in denen Quanteneffekte zu Beobachtungen führen, die durch die Anzahl von Partikeln, die auf einen Sensor treffen, bestimmt sind, als auch für allgemeine Poisson-Regressionsprobleme, die in verschiedenen Bereichen auftreten. Wir zeigen, dass die modellspezifischen Berechnungen für den Expectation Propagation Algorithmus effizient durch ein einfaches dynamisches Programm gelöst werden können.

Danach beschreiben wir einen skalierbaren approximativen Inferenzalgorithmus für strukturierte Mischverteilungen, die ein diskretes grafisches Modell verwenden um Abhängigkeiten zwischen latenten Mischkomponenten einer Menge von Mischverteilungen zu repräsentieren. Wir betrachten insbesondere baumartige Super-Gauss'sche Mischverteilungen, die wir dazu verwenden, die Persistenzeigenschaft grosser Koeffizienten der Wavelet-Transformation über Skalen hinweg zu berücksichtigen.

Im zweiten Teil, über Präferenzmodelle für Nutzer, betrachten wir zwei Szenarien: das global-statische und das kontextuell-dynamische Szenario.

Im global-statischen Szenario repräsentieren wir Nutzerpräferenzen numerisch durch eine niedrig-rang Matrix. Anstatt auf expliziten Bewertungen basieren unsere Methoden auf zwei Arten von implizitem Feedback der Nutzer: Aggregierte Statistiken über Nutzerinteraktionen und binäre Vergleichsergebnisse.

Für die aggregierten Statistiken verwenden wir bilineare Modelle mit Poisson Likelihood. Für dieses Modell schwächen wir die Gauss'sche Variationsapproximationsschranke durch zusätzliche Bedingungen kann und zeigen, dass sie dadurch eine geschlossene Form erhält, die effizient berechnet werden kann.
Im zweiten Fall, dem global-statischen Szenario, nutzen wir Vergleichsergebnisse zwischen zwei Gegenständen um auf die Vorlieben der Nutzer zu schließen. We kombinieren ein bilineares Modell niedrigen Rangs mit einer nicht-parametrischen Regressionskomponente und entwickeln einen neuen approximativen Expectation Maximization Algorithmus, der die Schwierigkeiten, verursacht durch die Abhängigkeiten zwischen den latenten Variablen, abmildert.
Im kontextuell-dynamischen Szenario modellieren wir Benutzeraktivitätssequenzen. Solche Sequenzen werden in vielen Applikationen routinemäßig im Hintergrund aufgezeichnet und können zeitliche und kontextuelle Aspekte des Nutzerverhaltens durch wiederkehrende Muster offenbaren. Wir präsentieren ein generisches kollaboratives Sequenzmodell, basierend auf rekurrenten neuralen Netzwerken, das Aspekte des kollaborativen Filtern mit denen von Sprachmodellen vereint.

Stichwörter: Probabilistische Modelle, generalisierte lineare und bilineare Modelle, rekurrente neurale Netzwerke, approximative Inferenz, Bildrekonstruktion, implizites Feedback

Contents

Acknowledgements **i**

Abstract (English/Deutsch) **iii**

Notation **1**

1 Introduction **3**

- 1.1 Outline and Contributions 6
- 1.2 Models and Methods 8
  - 1.2.1 Generalized Linear and Bilinear Models. 8
    - 1.2.1.1 Sparse Linear Models 10
    - 1.2.1.2 Latent Gaussian Models 11
  - 1.2.2 Sequence Modeling 12
  - 1.2.3 Methods of Variational Inference 13
    - 1.2.3.1 The Variational Gaussian Approximation 16
    - 1.2.3.2 Convexity-based Direct Site Bounding 17
    - 1.2.3.3 Expectation Propagation 18

2 Inference for Generalized Linear Models **21**

- 2.1 Rectified-Linear Poisson Regression 23
  - 2.1.1 Overview and Related Work 23
  - 2.1.2 Inference for the Poisson Likelihood Model 25
    - 2.1.2.1 Expectation Propagation 28
  - 2.1.3 Tractable EP Updates for the Rectified-Linear Function 30
    - 2.1.3.1 Implementation Details 34
  - 2.1.4 Experiments 35
    - 2.1.4.1 Synthetic Data 35
    - 2.1.4.2 Cox Processes: Coal Mining Disaster Data 35
    - 2.1.4.3 Sparse Linear Models 37
  - 2.1.5 Discussion 38
- 2.2 Tree-Structured Scale Mixtures 40
  - 2.2.1 Overview and Related Work 40
  - 2.2.2 Approximate Inference for Tree-Structured Scale Mixtures 42
    - 2.2.2.1 Preliminaries 42

2.2.2.2 Algorithm of Seeger and Nickisch [2011a] 46
2.2.2.3 Inference Algorithm for Structured Scale Mixtures 50
2.2.2.4 Parameter Learning. 54
2.2.3 Experiments 56
2.2.3.1 Denoising 57
2.2.3.2 Inpainting 58
2.2.4 Discussion 59

3 Models of Implicit Feedback **63**
3.1 Poisson Matrix Factorization 66
3.1.1 Overview and Related Work 66
3.1.2 Variational Gaussian Inference 67
3.1.3 Experiments 72
3.1.3.1 Methods 72
3.1.3.2 Results 73
3.1.4 Discussion 75
3.2 Pairwise Preferences 78
3.2.1 Overview and Related Work 78
3.2.2 Inference and Learning for Collaborative Preference Learning . . . 80
3.2.2.1 Preliminaries 80
3.2.2.2 PPCA for Coupled Likelihoods 81
3.2.2.3 Our Approach: EM Algorithm under Reparameterization 84
3.2.2.4 M-Step 86
3.2.2.5 E-Step 89
3.2.2.6 Implementation Details 91
3.2.3 Experiments 92
3.2.3.1 Preliminaries 92
3.2.3.2 Comparison to Alternative Utility Models 93
3.2.3.3 Movielens Datasets 95
3.2.3.4 Visualization of Results 97
3.2.4 Discussion 98
3.3 Collaborative Recurrent Neural Networks 100
3.3.1 Overview and Related Work 100
3.3.2 Preliminaries 102
3.3.3 Baseline Models 103
3.3.4 The Collaborative Recurrent Neural Network Model 105
3.3.4.1 Processing a Single Sequence 106
3.3.4.2 The Collaborative Recurrent Neural Network (C-RNN) . 109
3.3.4.3 Variants 110
3.3.4.4 Implementation Details 111
3.3.5 Empirical Evaluation 111
3.3.5.1 Datasets 111

3.3.5.2 Comparison of RNN Variants 112
3.3.5.3 Baseline Comparison 113
3.3.5.4 Characterization of Error 113
3.3.6 Discussion 114

4 Concluding Remarks 117

Bibliography 119

Curriculum Vitae 131

Notation

u, y	Lowercase letters denote scalars
$\boldsymbol{y}, \boldsymbol{u}$	Bold lowercase letters denote column vectors
$\boldsymbol{X}, \boldsymbol{U}$	Bold uppercase letters denote matrices
y_j	Scalar entry of corresponding vector, 1-based indices
u_{ij}	Scalar entry of corresponding matrix, 1-based indices
$\mathcal{Y}$	Data domain, e.g. $\mathbb{R}$
$\mathbb{O}, \mathbb{U}, \mathbb{I}$	Sets of indices
$\boldsymbol{I}, \boldsymbol{I}_M, \boldsymbol{I}_{\mathbb{O}}$	Identity matrix, of dimension M, restricted to rows indexed by $\mathbb{O}$
$\mathbf{1}$	Constant one vector
$[N]$	Range of integers $\{1, \ldots, N\}$
$\{x_i\}_{i \in \mathbb{O}}$	Set formation from indexed elements
$[x_i]_{i \in \mathbb{O}}$	Vector formation from indexed elements
$\mathbb{1}(x)$	Indicator function $\mathbb{1}(x) = 1$ if x is true, else 0
$\boldsymbol{\delta}_k$	One-hot/indicator vector, $[\mathbb{1}(j = k)]_{j \in [N]}$
$\operatorname{vec}_r(\boldsymbol{V})$	Row-major vectorization of $\boldsymbol{V}$
$\operatorname{vec}_c(\boldsymbol{V})$	Column-major vectorization of $\boldsymbol{V}$
$\operatorname{diag}(\boldsymbol{V})$	Diagonal vector of $\boldsymbol{V}$
$\operatorname{diag}(\boldsymbol{v})$	Diagonal matrix with diagonal $\boldsymbol{v}$
$[\boldsymbol{v}, \boldsymbol{u}]$	Horizontal concatenation of vectors/matrices
$[\boldsymbol{v}; \boldsymbol{u}]$	Vertical concatenation of vectors/matrices
$\lvert\boldsymbol{v}\rvert$	Length of a vector
$\lvert\boldsymbol{X}\rvert$	Determinant of a matrix

$\boldsymbol{A} \succ 0$	$\boldsymbol{A}$ is positive-definite
$\otimes$	Kronecker product
$\circ$	Element-wise product
$*$	Convolution
$\doteq$	Equality up to a constant
$\propto$	Proportionality
$\log$	Natural logarithm
$\sim$	Distributed as, drawn from
$\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}), \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	Gaussian density. The random variable may be dropped
$P(\cdot)$	Probability mass or density function
$Q(\cdot)$	Approximate posterior
$\boldsymbol{\mu}, \boldsymbol{\Sigma}$	Typically prior mean and covariance
$\boldsymbol{\xi}, \boldsymbol{\Xi}$	(Approximate) posterior mean and covariance
$D_{\mathrm{KL}}\left[P \| Q\right]$	Kullback-Leibler Divergence

1 Introduction

The language of probability is exceptionally well suited for the task of modeling the corrupted, incomplete datasets routinely encountered in real-world applications. Not only does it let us specify the visible variables we can observe, latent variables we cannot observe, and the relationship between them. It also lets us represent and precisely quantify our uncertainty about the unknown variables using (conditional) probability distributions. Crucially, it equips us with conceptually simple rules to perform such probabilistic inferences about the unknowns conditioned on the data. Modeling data probabilistically, i.e. declaratively specifying a process that offers an explanation of how the observed data might have been generated, and then assessing its plausibility in the light of our observations by quantifying the uncertainty of the model about the data and, consequently, of the predictions and decisions we derive from the model, is a powerful formalism underlying many approaches to robustly address challenging problems in machine learning and related fields [MacKay, 1992, Ghahramani, 2015]. In fact, the language of probability is so expressive that it offers a unified perspective on many methods for analyzing data developed over the years [Bishop, 2006, Murphy, 2012, Barber, 2012]. Using probabilistic models in practice, however, can be challenging due to the asymmetry in the relative ease of model specification compared to tractably performing inference. We are therefore faced with a trade-off situation to achieve computational tractability in practice: Since models necessarily are abstractions, simplifications and reflections of the modeler's biases, one may be willing to make compromises by imposing additional assumptions to facilitate inference and learning. Alternatively, one may try to impose fewer restrictions on the model and deal with the complexity of the resulting inference problem directly. The majority of work presented in this dissertation is of the latter type, where we are in particular interested in approximating posterior inference in high-dimensional, non-conjugate latent variable models. We use the former approach only in the last part on time-series, where we sacrifice a possibly beneficial probabilistic description of a complex model for the ability to tractably learn variable-length dependencies on large time-series datasets.

Throughout this dissertation, we consider parametric probabilistic models, for which we will adopt notation inspired by [Gelman et al., 1995]. The dataset is the particular realization of observable variables, to which we have access, and is denoted by $\boldsymbol{y}$. We denote by $P(\cdot)$ probability densities or mass functions that we will generally refer to as distributions. We focus on parametric models, i.e. joint distributions of the variables involved that are parameterized by a fixed-size parameter vector $\boldsymbol{\theta}$. When it is clear from the context, the explicit dependence of the model on $\boldsymbol{\theta}$ may be dropped. *Learning* a model then becomes the task of finding particular values of the parameters that explain an observed dataset $\boldsymbol{y}$ well. We therefore approach the learning problem as a *maximum likelihood* (ML) problem

$$\max_{\boldsymbol{\theta}} P(\boldsymbol{y} \mid \boldsymbol{\theta}). \tag{1.1}$$

Introducing latent variables greatly enhances our ability to describe complex distributions of the data by specifying the generative process of the data in a hierarchical manner. Due to this expressiveness, however, learning latent variable models can be challenging in practice: In many interesting applications, and particularly in the ones considered in this dissertation, we are dealing with a high-dimensional real-valued latent variable vector $\boldsymbol{f}$. We specify the model by the joint distribution and can decompose it using the product rule into *likelihood* and *prior*:

$$P(\boldsymbol{y}, \boldsymbol{f} \mid \boldsymbol{\theta}) = P(\boldsymbol{y} \mid \boldsymbol{f}, \boldsymbol{\theta}) P(\boldsymbol{f} \mid \boldsymbol{\theta}) \tag{1.2}$$

We obtain the marginal likelihood, the objective in (1.1), often referred to as the partition function Z, from (1.2) by the sum rule

$$P(\boldsymbol{y} \mid \boldsymbol{\theta}) = \int P(\boldsymbol{y}, \boldsymbol{f} \mid \boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{f} \tag{1.3}$$

and the posterior distribution using Bayes' rule by normalizing (1.2)

$$P(\boldsymbol{f} \mid \boldsymbol{y}, \boldsymbol{\theta}) = \frac{P(\boldsymbol{y}, \boldsymbol{f} \mid \boldsymbol{\theta})}{P(\boldsymbol{y} \mid \boldsymbol{\theta})} \tag{1.4}$$

The posterior concisely summarizes the current state of our knowledge about $\boldsymbol{f}$ including the remaining uncertainty. Probabilistic inference amounts to computing posterior expectations. Its first two moments are useful for estimating a plausible $\boldsymbol{f}$, and for quantifying uncertainties in and correlations between variables, respectively. Assessing predictive distributions for supervised learning requires posterior expectations of data likelihoods, thereby guarding against overfitting. For estimating $\boldsymbol{\theta}$, algorithms for solving (1.1) may require posterior expectations of the log joint model [Dempster et al., 1977].

Note, that in a consistently Bayesian approach, $\boldsymbol{\theta}$ would be considered a random variable as well. In contrast, we pursue a more pragmatic, *empirical* Bayesian strategy [MacKay, 1992, Seeger, 2000] by assuming that $\boldsymbol{\theta}$ is readily estimated from the available data after integrating out the high-dimensional latent variable $\boldsymbol{f}$. Unfortunately, even in this simplified setting, the integral in (1.3) is rarely available in closed form. Restricting the model to use conjugate priors might fail to capture the characteristics of interest exhibited by the data and does not resolve this issue in cases where, e.g., complex dependencies between latent variables are the cause for intractability. Therefore, methods for *approximate* inference are central for using probabilistic models in practice. Though most accurate, sampling-based approximations are difficult to scale to high-dimensional problems and large datasets and are not considered here. Instead, we focus on deterministic approximations, in particular variational methods [Saul et al., 1996, Jordan et al., 1997, Attias, 2000], developed in the context of probabilistic graphical models. Variational methods cast the inference problem into an optimization problem and subsequently simplify the problem by making assumptions on the form of the posterior in (1.4), e.g. by imposing factorizations or by restricting it to lie within a tractable subset of distributions, such as Gaussians, resulting in a much more compact and efficient representation of the posterior than with samples. The models considered here, typically do not exhibit useful graphical model structure amenable for approximation algorithms based on message passing. Instead, we rely on alternating optimization schemes similar to Expectation Maximization, continuous function optimization and certain fixed point algorithms. Such methods typically behave much more predictably in terms of running time and are easier to diagnose in terms of convergence compared to sampling-based methods and often reach a satisfactory solution orders of magnitude faster than sophisticated sampling methods [Kucukelbir et al., 2015], at the same time being generally more suitable to be scaled to very large datasets [Hoffman et al., 2013, Broderick et al., 2013], an attractive feature in the light of the exponentially growing rate of data collection we are witnessing.

Although a separation of concerns between modeling and inference algorithms is highly desirable, finding an efficient variational approximation algorithm is intimately tied to the characteristics of the model. Even methods for automatic variational inference, an active area of research dating back to Bishop et al. [2002], are rather trying to find the right level of abstraction that makes them applicable for a wide variety of problems. In doing so they have to make compromises as concessions to the variety of models that we would like to express: They often require sampling or numerical integration as a sub-routine, albeit typically only in dimensions lower than the original latent dimensionality [Ranganath et al., 2014], require differentiability with respect to all latent variables [Kucukelbir et al., 2015], a property violated in several models of interest here, and often employ a mean-field assumption that neglects posterior covariance.

The work on approximate inference, that we present in this dissertation acknowledges these ties and seeks to contribute both in devising new models accompanied by efficient inference and learning algorithms as well as present novel algorithms for existing models. While in general the algorithms we develop are necessarily iterative in nature, we strive to find closed form updates or sub-problems that can be computed using well-understood numerical primitives.

The work presented here is motivated by and organized according to two applications: image reconstruction in Chapter 2 and models of user preference, useful for recommender systems in Chapter 3. Apart from these running examples, the models are applicable also in other domains which we point out in the corresponding chapters.

The remainder of this chapter consists of an outline of the dissertation in Section 1.1, summarizing the contributions presented in each part together with the publications in which they appeared. This is followed in Section 1.2 by an overview of the different types of models and methods used in the remaining chapters, categorized by application.

1.1 Outline and Contributions

This dissertation is organized in two chapters according to the applications motivating our work.

In Chapter 2, we consider the ill-posed inverse problem of recovering an image from an underdetermined system of linear measurements that are corrupted by noise. This part consists of the following sections:

- In Section 2.1 we consider a Poisson likelihood model with the rectified-linear function as non-linearity. This model can be used as an alternative to the standard additive white Gaussian noise model, e.g. describing physically plausibly the stochastic arrival process of photons or other particles at a detector. To address the intractable inference problem for such models, we present an efficient and robust Expectation Propagation algorithm entirely based on analytically tractable computations operating reliably in regimes where quadrature based implementations can fail. Furthermore, we discuss the rectified-linear function in the context of Poisson regression problems where it can serve as a robust alternative to other common non-linearities. This work was published in [Ko and Seeger, 2015].

- In Section 2.2 we model the hierarchical correlations between the coefficients of an orthonormal multi-scale Wavelet transform using tree-structured mixtures of super-Gaussians. We use this model as an image prior for denoising and inpainting tasks. We derive a scalable inference algorithm to approximately compute the posterior over the latent image as well as the discrete mixture components, that maintains a full covariance representation between the pixels. We provide empirical evidence for the benefits of using this augmented prior over a model that ignores this structure in highly ill-posed problems. This work was published in [Ko and Seeger, 2012].

In Chapter 3, we consider the problem of inferring user preferences for items from various types of implicit feedback. We consider two settings: the global static and the contextual dynamic setting. In the global static setting, we represent preferences of users for items by a latent real-valued matrix with low-rank structure. Instead of using numeric ratings we develop methods to infer this latent representation given two types of implicit feedback: aggregate counts of users interacting with a service and the binary outcomes of pairwise comparisons.

- In Section 3.1 we model count data using a latent Gaussian bi-linear model with Poisson likelihood. For this model, we develop an approximate inference algorithm based on the variational Gaussian lower-bounding technique and derive a closed-form variational objective under additional constraints. We show that the objective is bi-concave, not only for Poisson but for any log-concave likelihood, which suggests an alternating optimization scheme. We compare different approximating families and empirically find that a mean-field approximation achieves a good balance between running time and predictive performance. This work was published in [Ko and Khan, 2014].
- In Section 3.2, we seek to infer the user preferences from the binary outcomes of pairwise preference statements. We combine a low-rank bi-linear model with non-parameteric item-feature regression to address the cold-start problem. We develop a novel approximate variational Expectation Maximization algorithm that mitigates the computational challenges due to couplings between entries in the latent preference matrix introduced by the pairwise comparisons. This work was published in [Khan, Ko, and Seeger, 2014].

In the contextual dynamic setting, we consider a different form of implicit feedback data by taking into account the dynamics of user behavior. Methodologically, we embark on a tangent by using recurrent neural networks. Although it is a generative sequence model and can be trained using maximum likelihood, neither the network weights nor the hidden states are considered latent random variables. Therefore, this model does not explain noise or represent uncertainty, but rather relies on abundantly available data as well as other forms of regularization, as is typical for neural-network-based methods.

- In Section 3.3, we model sequences of user activity at the granularity of single interaction events in contrast to the aggregate perspective assumed in our work on count data. Such activity logs are already routinely gathered in the background at a large scale in many applications. Taking into account temporal and contextual aspects of user behavior revealed by recurrent patterns could potentially lead to more accurate user models. By combining ideas from latent factor models for collaborative filtering and language models, we propose a collaborative sequence model based on recurrent neural networks. The model is designed to capture a user's contextual state as a personalized hidden vector by summarizing cues from a variable number of past time steps, while representing items by a global, real-valued embedding. We demonstrate the versatility of our model by applying it to two different tasks: music recommendation and mobility prediction. We show empirically that our model efficiently exploits the inherent structure of the data and consistently outperforms static and non-collaborative baseline methods. This work was published in [Ko, Maystre, and Grossglauser, 2016].

1.2 Models and Methods

In this part we introduce the models and methods that are the foundation upon which the work in this dissertation is built. We begin by introducing generalized linear and bilinear models in Section 1.2.1. In Section 1.2.2 we introduce recurrent neural networks to model sequential data. Finally, in Section 1.2.3 we introduce the different methods for approximate inference, that we will encounter in this dissertation.

1.2.1 Generalized Linear and Bilinear Models.

The models that we consider in Sections 2.1 and 2.2, are related to generalized linear models [McCullach and Nelder, 1983], while the models in Sections 3.1 and 3.2 are generalized bi-linear models [Lee and Seung, 1999, Collins et al., 2002, Mohamed et al., 2008, Salakhutdinov and Mnih, 2008a], both widely used in machine learning, signal processing and statistics. The linear and bilinear models of interest are latent variable models of the following form:

Observations are denoted by a vector $\boldsymbol{y} \in \mathcal{Y}^M$, where $\mathcal{Y} \in \{\mathbb{R}, \{0,1\}, \mathbb{N}\}$, depending on the application. We assume that the elements of $\boldsymbol{y}$, denoted by y_j, are independent samples of a likelihood given the entries f_j of a latent vector $\boldsymbol{f} \in \mathbb{R}^M$:

$$P(\boldsymbol{y} \mid \boldsymbol{f}) = \prod_{j=1}^{M} P(y_j \mid f_j). \tag{1.5}$$

This formulation allows us to conveniently generalize the model to different types of data or noise by choosing appropriate likelihoods.

The latent vector $\boldsymbol{f}$ has the following representation:

$$\boldsymbol{f} = \boldsymbol{X}\boldsymbol{u} \tag{1.6}$$

where $\boldsymbol{X} \in \mathbb{R}^{M \times N}$ and $\boldsymbol{u} \in \mathbb{R}^N$. In cases where Equation (1.6) can be interpreted as a Matrix-Matrix multiplication, i.e. when $N = UD$ and $\boldsymbol{X} = \boldsymbol{I}_U \otimes \boldsymbol{V}$ for some $\boldsymbol{V} \in \mathbb{R}^{I \times D}$, we may, for ease of indexing, rewrite the likelihood (1.5) in terms of a latent *matrix*

$$\boldsymbol{F} = \boldsymbol{V}\boldsymbol{U}^T \tag{1.7}$$

where $\boldsymbol{U} \in \mathbb{R}^{U \times D}$ is the row-major matrix representation of $\boldsymbol{u}$, and consequently $\boldsymbol{F} \in \mathbb{R}^{I \times U}$. Then, (column) vectors $\boldsymbol{v}_i$ and $\boldsymbol{u}_u$ represent rows of $\boldsymbol{V}$ and $\boldsymbol{U}$, respectively, so that $f_{iu} = \boldsymbol{v}_i^T \boldsymbol{u}_u$.

In this setting, we define *linear* models as distributions

$$P(\boldsymbol{y}, \boldsymbol{u} \mid \boldsymbol{X}), \tag{1.8}$$

i.e. we assume $\boldsymbol{X}$ to be known. While in linear models for supervised learning, i.e. regression and classification problems, $\boldsymbol{X}$ is typically a tall matrix of input features, we predominantly focus on the overcomplete or underdetermined case, commonly encountered in inverse problems in image or signal processing. Examples for reconstruction problems that can be represented in this framework together with the corresponding linear operator $\boldsymbol{X}$ include the following applications:

- Image deconvolution [e.g. Levin et al., 2011]: the linear operator $\boldsymbol{X}$ is a convolution with a known blur kernel $\boldsymbol{k}$, i.e. $\boldsymbol{X} = (\boldsymbol{k} \, *)$.
- Image inpainting [Bertalmio et al., 2000]: $\boldsymbol{X} = \boldsymbol{I}_{\mathbb{O}}$, consists of rows of the identity matrix indexed by the set of observed pixels $\mathbb{O}$.
- Cartesian MRI reconstruction [e.g. Seeger and Nickisch, 2011a]: $\boldsymbol{X} = \boldsymbol{I}_{\mathbb{O}}\boldsymbol{F}$ consists of rows of the discrete Fourier transform.

On the other hand, we define *bilinear* models as

$$P(\boldsymbol{y}, \boldsymbol{u}, \boldsymbol{X}), \tag{1.9}$$

i.e. $\boldsymbol{X}$ is considered unknown. Such models are typical for unsupervised learning or dimensionality reduction and use the representation in Equation (1.7) while assuming $\boldsymbol{F}$ be of low rank, i.e. $D \ll \min\{I, U\}$.

There are two ways to interpret the unknown $\boldsymbol{X}$ in bilinear models: The first variant is similar to probabilistic PCA (PPCA) [Tipping and Bishop, 1998] in that $\boldsymbol{X}$ is treated as a parameter, that is learned by maximizing the marginal likelihood $P(\boldsymbol{y} \mid \boldsymbol{X})$, which is why learning in this setting is closely related to inference in linear models. The second variant treats $\boldsymbol{X}$ as a random variable as well and the goal is to infer a joint posterior over $\boldsymbol{X}$ and $\boldsymbol{u}$. For our purposes, we refer to this variant as probabilistic matrix factorization (PMF) [Salakhutdinov and Mnih, 2008a].

We encounter generalized bilinear models in many areas:

- In image and signal processing, e.g. in blind deconvolution [Levin et al., 2011], where both the unknown image $\boldsymbol{u}$ as well as the blur kernel $\boldsymbol{k}$ are unknown, or for sparse coding [Olshausen and Field, 1997], where $\boldsymbol{X}$ represents an overcomplete dictionary of low-level features from which a corpus of images $\boldsymbol{y}$ can be reconstructed by linear combinations of dictionary elements with weights from a sparse matrix $\boldsymbol{u}$.

- In topic modeling, $\boldsymbol{y}$ is a bag-of-words representations of a corpus of documents. Latent Dirichlet Allocation (LDA) [Blei et al., 2003] can be written as a bi-linear model where $\boldsymbol{X}$ are the word distributions per topic and $\boldsymbol{u}$ the topic distribution per document by summing over the latent topic variables [Buntine, 2002]. Blei et al. [2003] propose both, a version that treats $\boldsymbol{X}$ as a parameters as well as a smoothed version that treats $\boldsymbol{X}$ as a random variable.

- Finally, in the area of recommender systems, bi-linear models are successfully used to impute entries in the matrix of ratings of items given by users by decomposing it into a product of latent factors [Koren et al., 2009, Salakhutdinov and Mnih, 2008a, Ilin and Raiko, 2010].

To complete the specification of the joint model of data and unknowns in Equations (1.8) and (1.9), we consider sparse linear and latent Gaussian models for the latent variables.

1.2.1.1 Sparse Linear Models

For linear models we are primarily interested in heavy-tailed priors, known to better describe statistics of natural images [Seeger and Nickisch, 2011a]. Reconstructing $\boldsymbol{u}$ in (1.6) from observations drawn according to (1.5) is typically considered ill-posed: without additional assumptions we cannot hope to meaningfully determine $\boldsymbol{u}$ from $\boldsymbol{y}$. One such assumption, particularly fruitful due to its wide applicability, is that the image can be represented as a linear combination of a few atoms from an appropriately chosen basis or dictionary, meaning that the actual effective number of degrees of freedom is much lower than it is apparent from a canonical signal representation such that the information contained in the measurements are sufficient for a good estimation of $\boldsymbol{u}$ [Tibshirani, 1996, Olshausen and Field, 1996, Donoho, 2006, Candès et al., 2006]. For natural images, modeling statistically the occurrence of edges as their most salient features, can lead to such sparse descriptions, e.g. by using high-pass filter responses [Portilla et al., 2003]. Let $\boldsymbol{s} = \boldsymbol{B}\boldsymbol{u}$ be coefficients in a appropriately chosen linear transform domain, in which they are (approximately) sparse. Examples for $\boldsymbol{B}$ are derivative filters (Section 2.1) or the Wavelet transform (Section 2.2). The distributions of these transform coefficients are then described by non-Gaussian, heavy-tailed distributions, combined as follows:

$$P(\boldsymbol{u}) \propto \prod_j t(s_j) \tag{1.10}$$

Examples include the popular Laplace prior, for which $t(s) = e^{-\tau|s|}$. From a regularized estimation point of view this prior can be seen as a convex surrogate of the ℓ_0 pseudo-norm. Point estimation under this prior for a Gaussian likelihood is well understood and can be theoretically characterized in terms of properties of $\boldsymbol{X}$ and $\boldsymbol{B}$ [Donoho, 2006, Candès et al., 2006]. While from a probabilistic point of view, to model exact sparsity, a distribution that assigns non-zero probability to the event $s = 0$, such as spike and slab, would be more appropriate [Titsias and Lázaro-Gredilla, 2011], Laplace priors are often used for its mathematical properties [Girolami, 2001, Seeger and Nickisch, 2011a]. We will revisit sparse linear models in Section 2.1 and Section 2.2.

1.2.1.2 Latent Gaussian Models

In our setup, we can use Gaussian priors in various ways. For bilinear models $\boldsymbol{F} = \boldsymbol{V}\boldsymbol{U}^T$, the probabilistic matrix factorization model of Salakhutdinov and Mnih [2008a] places Gaussian priors on the rows of $\boldsymbol{V}$ and $\boldsymbol{U}$. We will revisit this setup in Section 3.1. A probabilistic PCA prior [Tipping and Bishop, 1999] places an isotropic Gaussian on the rows of $\boldsymbol{U}$, while $\boldsymbol{V}$ is considered a parameter. We will encounter this formulation in Section 3.1 and Section 3.2.

A particularly versatile approach to specify a latent Gaussian model in a supervised setting, is to view the linear model (1.5) with (1.6) in terms of a function instead of weights and to generalize it by describing the function non-parametrically using Gaussian processes (GP) [Rasmussen and Williams, 2006]. Assuming a regression perspective, we can consider rows of $\boldsymbol{X}$ as feature vectors $\boldsymbol{x}_j$ and $\boldsymbol{u}$ a weight vector. The components of $\boldsymbol{f}$ can therefore be interpreted as a (real-valued) function $f(\boldsymbol{x})$ evaluated at $\boldsymbol{x}_j$, in this case linear, i.e. $f_j = f(\boldsymbol{x}_j) = \boldsymbol{x}_j^T\boldsymbol{u}$. A Gaussian process prior can be used to model $f(\boldsymbol{x})$ in more general, non-linear terms. Importantly, $f(\boldsymbol{x})$ is viewed as a random quantity. Rasmussen and Williams [2006] introduce Gaussian processes as collections of, possibly uncountably many, real random variables f_j, any subset of which is jointly Gaussian. Therefore, we can represent the co-domain of a function by this collection. To complete the specification, we need to define how function values at different input locations $\boldsymbol{x}_i$, $\boldsymbol{x}_j$ correlate, which effectively specifies our smoothness assumptions on $f(\boldsymbol{x})$. We provide this assumption through a positive definite kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ [Schölkopf and Smola, 2002]. Additionally, we need to provide a mean function $\mu(\boldsymbol{x})$. From a probabilistic perspective, the kernel function determines the covariance of the function values $f(\boldsymbol{x})$. Specifically, the kernel matrix $\boldsymbol{K}$ with entries $\boldsymbol{K}_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is positive definite and serves as the covariance matrix of the vector $\boldsymbol{f}$. We write $f \sim \mathrm{GP}(\mu(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}'))$ to denote a function distributed as specified by a GP. We will revisit GPs as priors in Section 2.1 and Section 3.2.

1.2.2 Sequence Modeling

In Section 3.3 we will encounter a scenario where the observations consist of sequences of events. Recurrent neural networks [RNN, Rumelhart et al., 1988, Werbos, 1990] are conceptually simple, yet powerful and versatile sequence models. Although RNNs can be used to specify the likelihood of a sequence, resulting in a generative sequence model, typically neither the network weights nor the hidden states are considered random. This simplifies training, which is reduced to maximum likelihood estimation. Inference for neural networks is an active research topic, but still prohibitively expensive in practice requiring sampling approximations for which back-propagation has to be run several times more often to estimate gradients [Graves, 2011]. Therefore, other regularization techniques are preferred in practice, some of which can be viewed as approximating model averaging [Srivastava et al., 2014, Pham et al., 2014, Gal and Ghahramani, 2016].

For our purposes, we draw analogies to neural language models [Mikolov et al., 2010], where a sequence $\boldsymbol{y}$ of length T with elements from a fixed alphabet or vocabulary of I items $\mathbb{I} = [I]$ is observed. Given the ordering imposed by time, it is reasonable to choose a factorization of the joint distribution over a sequence that respects that order:

$$P(\boldsymbol{y}) = \prod_{t=1}^{T} P(y_t \mid \boldsymbol{y}_{<t}), \tag{1.11}$$

where $\boldsymbol{y}_{<t}$ denotes all elements of $\boldsymbol{y}$ before index t. In order to capture long-range dependencies, language models such as the RNN Language Model [RNNLM, Mikolov et al., 2010] use a RNN to specify $P(y_t \mid \boldsymbol{y}_{<t})$.

Generally, RNNs compute a mapping from the input sequence to a corresponding sequence of real-valued hidden state vectors of dimension D:

$$\mathrm{RNN}\left([y_1, \ldots, y_T]\right) \; = \; [\boldsymbol{h}_1, \ldots, \boldsymbol{h}_T] \;, \quad \boldsymbol{h}_t \in \mathbb{R}^D. \tag{1.12}$$

The hidden state $\boldsymbol{h}_t$ is a flexible representation, meant to summarize the sequence, seen up to time t, that enables us to use the network for different tasks by defining an appropriate output layer. RNNs can be seen as non-linear dynamic systems that define the computation of the hidden state recursively. In graphical terms, they generalize the directed, acyclic graph structure of feed-forward networks by allowing cycles to represent dependencies on the past state of the network. The RNNLM uses a simple network architecture that goes back to Elman [1990] and expresses the dependency on the past through the following recursive definition of the hidden state vectors:

$$\begin{aligned} \boldsymbol{a}_t &= \boldsymbol{W}_h \boldsymbol{h}_{t-1} + \boldsymbol{W}_{in} \boldsymbol{\delta}_{y_t}, \\ \boldsymbol{h}_t &= \sigma(\boldsymbol{a}_t). \end{aligned} \tag{1.13}$$

The matrices $\boldsymbol{W}_h \in \mathbb{R}^{D \times D}$ and $\boldsymbol{W}_{in} \in \mathbb{R}^{D \times I}$ are parameters of the RNN and $\sigma(\cdot)$ is a non-linear function that is applied element-wise to its input, such as the logistic sigmoid, the hyperbolic tangent or, more recently, the rectified linear function. The input is presented to the network as one-hot encoded vectors denoted by $\boldsymbol{\delta}_{y_t}$, in which case the corresponding matrix-vector product $\boldsymbol{W}_{in}\boldsymbol{\delta}_{y_t}$ reduces to projecting out the y_t-th column of $\boldsymbol{W}_{in}$. Note, that the network can be trivially extended to accept arbitrary side information that characterizes the input at time t. The recurrence is initialized by a constant vector $\boldsymbol{h}_0 = \epsilon \mathbf{1}$ with small $\epsilon \geq 0$. In practice, more sophisticated architectures implementing (1.12) are in use [Hochreiter and Schmidhuber, 1997, Cho et al., 2014b], some of which we will encounter in Section 3.3.

To obtain a distribution over the next item, we can linearly map the hidden state to $\mathbb{R}^I$ using a matrix $\boldsymbol{W}_{out} \in \mathbb{R}^{I \times D}$ and pass the output vector $\boldsymbol{z}_t \in \mathbb{R}^I$ through the softmax function $\sigma_m(\boldsymbol{z}, j) = \frac{\exp(z_j)}{\sum_{k \in [I]} \exp(z_k)}$ to obtain the probability for the next item to be j:

$$\begin{aligned} \boldsymbol{z}_t &= \boldsymbol{W}_{out} \boldsymbol{h}_t, \\ P(y_t \mid \boldsymbol{y}_{<t}) &= \sigma_m(\boldsymbol{z}_t, y_t) \end{aligned} \tag{1.14}$$

This likelihood together with the recursion in (1.13) enables us to sample sequences, by drawing from the multinomial (1.14) and presenting the sample as input to the network for the next timestep.

The network is parameterized by $\boldsymbol{\theta} = \{\boldsymbol{W}_{in}, \boldsymbol{W}_h, \boldsymbol{W}_{out}\}$ and can be trained using maximum likelihood. Gradients are approximated using backpropagation through time [BPTT, Williams and Zipser, 1995]. BPTT computes the gradient by unrolling the RNN in time and by treating it as a multi-layer feed-forward neural network with parameters tied across every layer and error signals and inputs at every layer. For computational reasons, the sequence unrolling is truncated to a fixed size. In Section 3.3, we extend RNNs to sequence modeling in a collaborative setting.

1.2.3 Methods of Variational Inference

Most parts of this dissertation develop or use approximations for intractable posterior inference in latent variable models. Here, we focus on Gaussian approximations, that can be obtained using different methods. In Section 2.1 and Section 3.2, we respectively derive and use Expectation Propagation algorithms [EP, Minka, 2001b, Opper and Winther, 2005]. In Section 2.2 and Section 3.1, we use variational bounding techniques to approximate the log partition function [Opper and Archambeau, 2009, Girolami, 2001]. See Nickisch and Rasmussen [2008] for a comprehensive comparative study of Gaussian approximation techniques for Gaussian process classification. This section provides a brief introduction to the approximations we use.

For the purpose of this brief, conceptual introduction, consider the intractable posterior distribution in a latent variable model

$$P(\boldsymbol{f} \mid \boldsymbol{y}) = \frac{P(\boldsymbol{y} \mid \boldsymbol{f})P(\boldsymbol{f})}{P(\boldsymbol{y})} \tag{1.15}$$

To proceed, we first rewrite the posterior in (1.15) in a slightly more convenient way by abstracting from the notions of prior and likelihood. We assume that our model can be written as a product of positive, scalar functions $t_j(f_j)$, that we will refer to as *potentials*, and an optional coupled Gaussian factor $t_0(\boldsymbol{f}) = \mathcal{N}(\boldsymbol{f} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$ [Nickisch, 2010, Rasmussen and Williams, 2006]:

$$P(\boldsymbol{f}) = Z^{-1} \prod_{j=1}^{M} t_j(f_j)\; t_0(\boldsymbol{f}) \tag{1.16}$$

With this formulation we can express both models with non-Gaussian likelihoods, e.g. in supervised classification tasks, and non-Gaussian priors, e.g. in sparse linear models for images (1.10). Of particular interest are *log-concave* potentials, i.e. potentials $t(f)$ such that $\log t(f)$ is a concave function. Consequently, for a log concave model, MAP estimation can be expressed as a convex minimization problem, which in turn implies uni-modality of the posterior: a scenario in which a uni-modal Gaussian posterior approximation seems appropriate. Furthermore, the convexity properties of log-concave models carry over, to an extent, to variational approximate inference methods [Seeger and Nickisch, 2011a, Challis and Barber, 2011] and lead to stable Expectation Propagation algorithms [Seeger, 2008]. The normalization constant Z corresponds to the marginal likelihood or partition function Eq. (1.1). Assuming it exists, it is the central quantity for probabilistic inference, computed implicitely or explicitely. We seek to approximate $P(\boldsymbol{f})$ by a multivariate Gaussian

$$Q(\boldsymbol{f}) = \mathcal{N}(\boldsymbol{f} \mid \boldsymbol{\xi}, \boldsymbol{\Xi}). \tag{1.17}$$

Gaussian approximations are often chosen for mathematical and algorithmic convenience, but are especially appropriate for log-concave potentials, since the posterior then is unimodal, as mentioned above.

Approximating $P(\boldsymbol{f})$ in (1.16) raises the question of how to define proximity. A global measure of dissimilarity between two distributions is the Kullback-Leibler divergence, which is non-negative and zero for identical distributions $P(\boldsymbol{f}) = Q(\boldsymbol{f})$:

$$D_{\mathrm{KL}}\left[Q(\boldsymbol{f})\|P(\boldsymbol{f})\right] = -\int Q(\boldsymbol{f}) \log \frac{P(\boldsymbol{f})}{Q(\boldsymbol{f})} \,\mathrm{d}\boldsymbol{f} \tag{1.18}$$

Its asymmetry with respect to the order of arguments suggests two criteria to minimize:

$$D_{\mathrm{KL}}\left[P(\boldsymbol{f})\|Q(\boldsymbol{f})\right] \neq D_{\mathrm{KL}}\left[Q(\boldsymbol{f})\|P(\boldsymbol{f})\right] \tag{1.19}$$

Minka [2005] introduces both as special cases of the family of α-Divergence measures. Since we minimize with respect to Q, $D_{\mathrm{KL}}[Q\|P]$ forces Q to be zero where P is zero to cancel the otherwise infinite divergence. On the other hand, $D_{\mathrm{KL}}[P\|Q]$ forces Q to be non-zero where P is non-zero with the same reasoning. Although, $D_{\mathrm{KL}}[Q\|P]$ might give more meaningful results in multi-modal settings, methods motivated by minimizing $D_{\mathrm{KL}}[P\|Q]$ are considered to be more accurate in uni-modal settings by better capturing posterior mass, thereby not underestimating covariances, and producing far more accurate approximations of the partition function [Nickisch and Rasmussen, 2008]. In practice, however, $D_{\mathrm{KL}}[Q\|P]$ is an attractive method for several reasons. Minimizing it is equivalent to maximizing a lower bound on $\log Z$, which is conceptionally intuitive. The objective can be considered easier to handle due to the expectation with respect to Q instead of the, by assumption, intractable P. The resulting optimization problem can therefore typically be solved by standard (gradient-based) algorithms. To see the difficulty of minimizing $D_{\mathrm{KL}}[P\|Q]$, consider the approximating Gaussian in its exponential family representation $Q(\boldsymbol{f}) = Z(\boldsymbol{\lambda})^{-1} \exp\left(\boldsymbol{\lambda}^T \phi(\boldsymbol{f})\right)$, where $\boldsymbol{\lambda}$ are the natural parameters and $\phi(\boldsymbol{f})$ the (vectorized) sufficient statistics. At a minimum of $D_{\mathrm{KL}}[P\|Q]$, we have $\nabla_{\boldsymbol{\lambda}} = 0$:

$$\begin{aligned} -\nabla_{\boldsymbol{\lambda}} D_{\mathrm{KL}}[P\|Q] &\doteq \nabla_{\boldsymbol{\lambda}} \int P(\boldsymbol{f}) \left(\boldsymbol{\lambda}^T \phi(\boldsymbol{f})\right) \,\mathrm{d}\boldsymbol{f} - \nabla_{\boldsymbol{\lambda}} \log Z(\boldsymbol{\lambda}) \\ &= \int P(\boldsymbol{f}) \phi(\boldsymbol{f}) \,\mathrm{d}\boldsymbol{f} - \mathrm{E}_Q\left[\phi(\boldsymbol{f})\right] \\ &= \mathrm{E}_P\left[\phi(\boldsymbol{f})\right] - \mathrm{E}_Q\left[\phi(\boldsymbol{f})\right] \end{aligned} \tag{1.20}$$

The log partition function is convex [Wainwright and Jordan, 2008]. We therefore require moment matching at the minimum:

$$\mathrm{E}_Q\left[\phi(\boldsymbol{f})\right] = \mathrm{E}_P\left[\phi(\boldsymbol{f})\right] \tag{1.21}$$

Except in special cases, moment matching cannot be achieved directly. After all computing P's moments are intractable.

The methods we consider are Expectation Propagation (EP) [Minka, 2001b, Opper and Winther, 2005], which is based on the idea of moment matching, the variational Gaussian (VG) approximation [Seeger, 2000, Opper and Archambeau, 2009, Challis and Barber, 2011], minimizing $D_{\mathrm{KL}}[Q\|P]$ and convexity based direct site bounding (DSB) techniques [Jaakkola and Jordan, 1998, Girolami, 2001, Palmer et al., 2006, Seeger, 2008], an alternative bounding technique.

VG and DSB methods approximate $\log Z$ by maximizing a lower bound with respect to certain variational parameters, while the EP approximation is not a bound. For the case of a class of linear models, Challis and Barber [2011] show that the two bounding methods are related, but the VG bound can be tighter. In comparison, Nickisch and Rasmussen [2008] show in a large empirical study that the EP approximation can be superior in the case of Gaussian-process classification.

These different approximations of the marginal likelihood come with different algorithmic strategies to optimize the variational parameters. A common feature and the major challenge is the cubic scaling in the number of latent variables if a full covariance matrix is to be represented.

1.2.3.1 The Variational Gaussian Approximation

The most common way to derive a variational approximation resembles the derivation of the EM algorithm. We introduce a variational distribution Q and bound $\log Z$ using Jensen's inequality:

$$\begin{aligned}\log Z &= \log \int \prod_{j=1}^{M} t_j(f_j)\; t_0(\boldsymbol{f})\, \mathrm{d}\boldsymbol{f} \\ &= \log \int Q(\boldsymbol{f}) \frac{\prod_{j=1}^{M} t_j(f_j)\; t_0(\boldsymbol{f})}{Q(\boldsymbol{f})}\, \mathrm{d}\boldsymbol{f} \\ &\geq \int Q(\boldsymbol{f}) \log \frac{\prod_{j=1}^{M} t_j(f_j)\; t_0(\boldsymbol{f})}{Q(\boldsymbol{f})}\, \mathrm{d}\boldsymbol{f} \\ &= \mathrm{E}_{Q(\boldsymbol{f})}\left[\log \frac{\prod_{j=1}^{M} t_j(f_j)\; t_0(\boldsymbol{f})}{Q(\boldsymbol{f})}\right] \\ &= \sum_j \mathrm{E}_{Q(f_j)}\left[\log t_j(f_j)\right] - D_{\mathrm{KL}}\left[Q(\boldsymbol{f}) \| t_0(\boldsymbol{f})\right] \\ &=: \mathcal{L}_{\mathrm{VG}}(Q)\end{aligned} \tag{1.22}$$

While the inequality would be tight at the intractable true posterior $Q(\boldsymbol{f}) = P(\boldsymbol{f} \mid \boldsymbol{y})$, we approximate the problem by restricting Q to lie within a tractable family of distributions $\mathcal{F}$. For the Variational Gaussian (VG) approximation, $\mathcal{F} = \{\mathcal{N}(\boldsymbol{\xi}, \boldsymbol{\Xi})\}$. We can further simplify the approximation by restricting the covariance, e.g. to be diagonal $\boldsymbol{\Xi} = \mathrm{diag}(\boldsymbol{\gamma})$. Maximizing the lower bound $\mathcal{L}_{\mathrm{VG}}(Q)$ with respect to the variational parameters $\boldsymbol{\xi}$ and $\boldsymbol{\Xi}$ yields an approximation to the evidence as well as an approximate posterior Q that minimizes $D_{\mathrm{KL}}\left[Q(\boldsymbol{f}) \| P(\boldsymbol{f} \mid \boldsymbol{y})\right]$.

This approximation results in straight-forward optimization problems, amenable for off-the shelf solvers, if the expectations in (1.22) can be evaluated efficiently. Furthermore, it is easy to impose further restrictions on the family of distributions, i.e. the structure of the approximate covariance. Challis and Barber [2011] show, that for log concave potentials the VG approximation results in a convex optimization problem. However, this result depends on particular parameterization and does not hold, e.g. for the compact reparameterization of Opper and Archambeau [2009]. We use this approximation for latent Gaussian bilinear models of count data in Section 3.1.

1.2.3.2 Convexity-based Direct Site Bounding

Another way to arrive at a useful bound on $\log Z$ is to approximate each non-Gaussian potential locally [Bishop, 2006]. A method to achieve this relies on a variational representation of the class of super-Gaussian potentials [Jaakkola, 1997, Girolami, 2001, Palmer et al., 2006]. As their name suggests, these potentials admit a representation as a pointwise maximum over width-parameterized Gaussian functions. Using the notation of Seeger and Nickisch [2011a], the representation is given by:

$$t_j(f_j) = \max_{\gamma_j} e^{-f_j^2/(2\gamma_j)} e^{-h_j(\gamma_j)/2}. \tag{1.23}$$

The condition, a potential $t_j(f_j)$ has to fulfill to admit this representation, is log-convexity in f_j^2, i.e. $g(x_j) := \log t_j(\sqrt{x_j})$ is convex, with $x_j \geq 0$. Then (1.23) follows from convex conjugacy [Rockafellar, 1970]. For a concise, characterization, see e.g. Palmer et al. [2006], where also a connection to the class of Gaussian scale-mixture distributions is shown. We obtain a parameterized Gaussian approximation to (2.6) by replacing each non-Gaussian potential $t_j(f_j)$ with the Gaussian function from (1.23). To set the variational parameters $\boldsymbol{\gamma}$, we optimize the resulting bound on $\log Z$:

$$\begin{aligned}
\log Z &= \log \int \prod_{j=1}^{M} t_j(f_j)\ t_0(\boldsymbol{f})\,\mathrm{d}\boldsymbol{f} \\
&\geq -\frac{1}{2}\sum_{j=1}^{M} h(\gamma_j) + \log \int e^{-\frac{1}{2}\boldsymbol{f}^T \operatorname{diag}(\gamma)^{-1}\boldsymbol{f}}\ t_0(\boldsymbol{f})\,\mathrm{d}\boldsymbol{f} \\
&=: \mathcal{L}_{\mathrm{DSB}}(\boldsymbol{\gamma})
\end{aligned} \tag{1.24}$$

For sparse linear models, [Seeger and Nickisch, 2011a] study this approximation in great detail and make several contributions: they conduct a thorough convexity analysis and find that the optimization problem (1.24) is convex if and only if the underlying MAP estimation problem is convex, i.e. for super-Gaussian potentials that are additionally log-concave. They remark that direct optimization of (1.24) even with first-order line-search based methods is prohibitively expensive, with cubic cost per evaluation of the objective function as well as the gradient. Instead, they reformulate the Gaussian integral in (1.24) as an optimization and then use a convergent algorithm akin to the Convex Concave Procedure [Yuille and Rangarajan, 2003] for optimizing (1.24) in a double-loop manner, where the cubic operation is isolated in the outer loop and is therefore required to be performed only a few times. Additionally, the expensive outer loop can be run approximately based on numerical methods that can exploit additional structure typically present in imaging applications. The resulting method can be scaled to problem sizes appropriate for imaging applications by approximately maintaining an implicit representation of the full posterior covariance matrix. In Section 2.2 we will extend the algorithm of Seeger and Nickisch [2011a] to structured mixture models.

1.2.3.3 Expectation Propagation

For this introduction we adopt the perspective and the notation of and refer to the book by Rasmussen and Williams [2006] for a more detailed introduction. EP [Minka, 2001b, Opper and Winther, 2005] approximates $P(\boldsymbol{f})$ in Eq. (2.6) by approximating each non-Gaussian potential $t_j(f_j)$ using unnormalized Gaussians $\tilde{t}_j(f_j) = \tilde{Z}_j \, \mathcal{N}(f_j|\tilde{\mu}_j, \tilde{\sigma}_j^2)$ to form a Gaussian approximation $Q(\boldsymbol{f})$ following the same factorization:

$$Q(\boldsymbol{f}) = Z_{\mathrm{EP}}^{-1} \prod_{j=1}^{M} \tilde{t}_j(f_j) \, t_0(\boldsymbol{f}) \tag{1.25}$$

Correspondingly, the EP-approximation to the marginal likelihood is given by:

$$Z_{\mathrm{EP}} = \prod_{j=1}^{M} \tilde{Z}_j \int \prod_{j=1}^{M} \mathcal{N}(f_j|\tilde{\mu}_j, \tilde{\sigma}_j^2) \, t_0(\boldsymbol{f}) \, \mathrm{d}\boldsymbol{f} \tag{1.26}$$

EP was devised to address the shortcomings of the assumed density filtering (ADF) method and can be motivated by, and in special cases shown, to minimize the KL-divergence $\mathrm{D_{KL}}\left[P(\boldsymbol{f}) \,||\, Q(\boldsymbol{f})\right]$ [Minka, 2001b]. Note the order of the arguments in contrast to Eq. (1.22). Since this quantity is generally intractable, EP employs the following strategy to determine the variational parameters $\tilde{\mu}_j, \tilde{\sigma}_j^2$.

We define the i-th marginal *cavity* distribution[1] by removing the i-th *approximate* potential $\tilde{t}_i(f_i)$ from $Q(\boldsymbol{f})$ and marginalizing over $\{f_j : j \neq i\}$, denoted as $\boldsymbol{f}_{\setminus i}$:

$$Q_{-i}(f_i) = \mathcal{N}\left(f_i|\mu_{-i}, \sigma_{-i}^2\right) \propto \int \prod_{j \neq i} \tilde{t}_j(f_j) \, t_0(\boldsymbol{f}) \, \mathrm{d}\boldsymbol{f}_{\setminus i} \tag{1.27}$$

The so called *tilted* distribution replaces the approximate potential $\tilde{t}_i(f_i)$ in $Q(\boldsymbol{f})$ by the

---

[1] EP is much more general than we describe here. E.g. it applies to models with multi-variate potentials or other exponential-family approximations.

true non-Gaussian potential $t_i(f_i)$ by multiplying it with the cavity marginal:

$$\hat{P}(f_i) = \hat{Z}_i^{-1}\, t_i(f_i)\, Q_{-i}(f_i) \qquad \text{where} \quad \hat{Z}_i = \int t_i(f_i)\, Q_{-i}(f_i)\, \mathrm{d}f_i \tag{1.28}$$

The criterion to minimize in order to update the parameters of $\tilde{t}_i$ is the KL-divergence between the tilted- and the variational distribution $\mathrm{D}_{\mathrm{KL}}\left[\hat{P}(f_j)\,||\,Q(f_j)\right]$. As we have seen in (1.21), we can solve this sub-problem by moment matching:

$$\begin{aligned} \mathrm{E}_Q\left[f_i\right] &= \mathrm{E}_{\hat{P}}\left[f_i\right] \\ \mathrm{Var}_Q\left[f_i\right] &= \mathrm{Var}_{\hat{P}}\left[f_i\right] \end{aligned} \tag{1.29}$$

This time, however, we are dealing with moments of a one-dimensional marginal distribution that can be seen as a Gaussian expectation of a potential. This computation is often analytically tractable, and if not, is amenable for numerical integration. The constant $\tilde{Z}_i$ is chosen such that the normalization constants of $\hat{P}(f_i)$ and $Q(f_i)$ match, i.e. we solve:

$$\tilde{Z}_i \int \mathcal{N}(f_i|\tilde{\mu}_i, \tilde{\sigma}_i^2)\, Q_{-}(f_i)\, \mathrm{d}f_i = \hat{Z}_i \tag{1.30}$$

The EP update therefore consists of determining the first two moments and the normalization constant of the tilted distribution.

Once the parameters of a single $\tilde{t}_i$ are changed, we can update the representation of the full approximation $Q(\boldsymbol{f})$, which consists of recomputing $\boldsymbol{\xi}$ and $\mathrm{Var}_Q\left[\boldsymbol{f}\right] = \mathrm{diag}\left(\boldsymbol{\Xi}\right)$. This process is repeated until convergence, i.e. until a fixed point in terms of tilted and approximate moments is reached.

The update of $Q(\boldsymbol{f})$, in particular obtaining $\mathrm{Var}_Q[\boldsymbol{f}]$, dominates the algorithm computationally, due to cubic scaling in the latent dimensionality. The cost can be somewhat reduced by doing a pass over all potentials before updating Q. This variant is referred to as parallel EP [Gerven et al., 2010]. Convergence is not guaranteed in either case [Seeger and Nickisch, 2011a]. But for log-concave models EP updates where shown to remain valid, such that the algorithm converges reliably in practice [Seeger, 2008]. Although there have been efforts to scale EP to larger problem sizes [Seeger and Nickisch, 2011b], scalability remains an open problem, not least due to its sensitivity to inaccurate variance estimates.

Opper and Winther [2005] derive EP from a different, more general perspective not discussed here, that allows for more accurate, structured approximations. Furthermore, they provide a more expensive algorithm for which convergence can be established.

2 Inference for Generalized Linear Models

Probabilistic inference for generalized linear models is useful on several levels [MacKay, 1992]: in ill-posed, data-scarce and noisy scenarios, probabilistic methods can not only deliver more robust estimates, but also quantify the uncertainty around them. This is not only crucial for learning meaningful model parameters [Levin et al., 2011], but also for making decisions, e.g. adaptive sampling and design optimization [Seeger and Nickisch, 2011a]. Figure 2.1 depicts a generalized sparse linear model, as introduced in Section 1.2.1, where the left plate constitutes the conditionally independent likelihood given projections of the latent weights $f_j = \boldsymbol{x}_j^T \boldsymbol{u}$. The right plate represents the prior, e.g. enforcing sparsity of $s_k = \boldsymbol{b}_k^T \boldsymbol{u}$ in an appropriate domain.

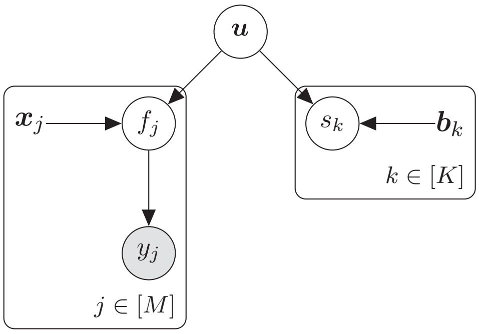


Figure 2.1 – Graphical model illustrating a Sparse Linear Model: The latent weights or image $\boldsymbol{u}$ govern the generation of observations y_j and are a priori assumed to result in sparse coefficients s_k when projected onto $\boldsymbol{b}_k$.

This chapter comprises two sections. In Section 2.1, we are concerned with modifications of the likelihood (left) part of Figure 2.1, whereas Section 2.2 deals with extensions of the prior (right) part of Figure 2.1.

In particular, in Section 2.1, we study a model of count data, which we refer to as rectified-linear Poisson regression. The focus lies on studying the role of non-linear functions used in doubly-stochastic inhomogeneous Poisson processes for mapping a real-valued latent variable to the positive reals to serve as the intensity of the Poisson process. Our main result is a simple dynamic program to robustly compute exact moments of tilted distributions (1.29) required for Expectation Propagation. Since only the likelihood in Figure 2.1 is modified, this result extends immediately to non-parametric regression with a Gaussian process prior, depicted in Figure 2.2, inspired by the illustration in Nickisch and Rasmussen [2008]. Each data point is associated to a new latent variable that are all correlated, symbolized by the bold line, governed by the kernel and its parameters as introduced in Section 1.2.1.2.

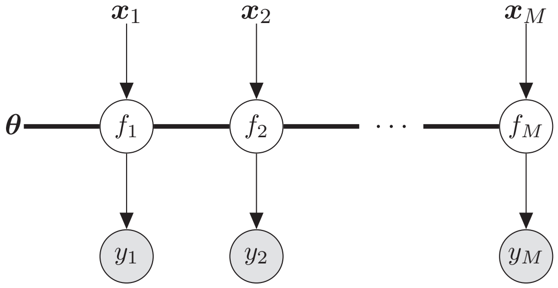


Figure 2.2 – Non-parametric Gaussian process regression: a latent function evaluated at locations $\boldsymbol{x}_j$ with corresponding observations y_j. The function is modeled by a GP, so that function values f_j are jointly Gaussian, indicated by the thick line connecting them.

In Section 2.2, we extend the prior in Figure 2.1 to be structured as illustrated in Figure 2.3: We augment the model with additional discrete latent variables to model the sparse coefficients as mixtures of super-Gaussians and impose tree-structured dependencies between mixture components. This model is useful to represent persistence of non-zero patterns across scales of hierarchical transforms such as Wavelets. For scalable posterior inference in this model, we extend the approximate inference algorithm of Seeger and Nickisch [2011a].

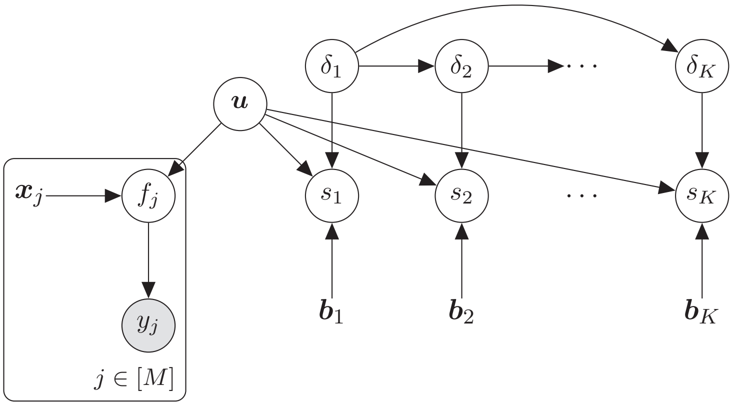


Figure 2.3 – Structured priors: The prior in Figure 2.1 is augmented by a latent discrete graphical model intended to model dependencies in the sparsity pattern of the s_k.

2.1 Rectified-Linear Poisson Regression

2.1.1 Overview and Related Work

In this part, we are concerned with approximating intractable posterior inference for sparse linear and latent Gaussian Poisson-regression models with the following likelihood. We observe counts $\boldsymbol{y} = [y_j]_{j\in[M]} \in \mathbb{N}_0^M$, that are conditionally iid. samples given the intensity or rate parameters $\boldsymbol{\lambda} = [\lambda_j]_{j\in[M]} \in \mathbb{R}_{\geq 0}^M$. Hence, we write the likelihood of the data as

$$P(\boldsymbol{y} \mid \boldsymbol{\lambda}) = \prod_{i=1}^{M} \frac{1}{y_j!} \lambda_j^{y_j} e^{-\lambda_j} \tag{2.1}$$

We model the intensities as $\lambda_j = g(f_j)$, where the $f_j \in \mathbb{R}$ are real-valued latent variables and $g(f)$ is a non-linear function that ensures the non-negativity of λ_j. Thus, we write the likelihood factors as potentials

$$t_y(f) := P(y \mid \lambda(f)) = P(y \mid f) \tag{2.2}$$

We denote by $\boldsymbol{\lambda} = g(\boldsymbol{f}) = [g(f_j)]_{j\in[M]}$ the point-wise application of $g(f)$ to the components of $\boldsymbol{f}$. Such models are used to describe counts of random phenomena in various contexts. The likelihood in (2.1) arises, for example, when discretizing inhomogeneous Poisson processes with stochastic intensities, also known as a Cox process, often used to map the rate at which certain social, economical or ecological events occur in space and/or time [Vanhatalo et al., 2010, Diggle et al., 2013]. In neuroscience, they are referred to as the Linear-Nonlinear-Poisson cascade model, widely used to describe neural responses to external stimuli [Pillow, 2007, Gerwinn et al., 2010, Park and Pillow, 2013, Park et al., 2014]. Moreover, Poisson models have been applied to collaborative filtering tasks to understand user preferences from implicit feedback [Seeger and Bouchard, 2012]. We will revisit such models in Section 3.1. In image processing, the noise process in photon-limited acquisition scenarios, typical for astronomical- and biomedical imaging, is Poisson [Starck and Murtagh, 2002, Dupé et al., 2008, Carlavan and Blanc-Féraud, 2012], an observation that e.g. the Richardson-Lucy model for deconvolution is based on [Richardson, 1972, Lucy, 1974]. Finally, in particle physics, Poisson models are used to infer the spectra of elementary particles generated in a particle accelerator [Kuusela and Panaretos, 2015].

In this work, we are primarily interested in the role of the non-linearity $g(f)$ on posterior inference. In Table 2.1, we list several alternatives for $g(f)$ commonly found in the literature [Pillow, 2007]. While clearly related, we avoid the terminology used for generalized linear models [McCullach and Nelder, 1983] where the inverse mapping $\lambda = g^{-1}(f)$ is referred to as the *link* function, because the RL function is not invertible. Instead we refer to $g(f)$ simply as non-linearity, similar to terminology used in neural networks, where rectified-linear units as a replacement for sigmoidal non-linearities has sparked recent interest [Glorot et al., 2011, Zeiler et al., 2013, Maas et al., 2013]. In particular, we focus on the rectified-linear (RL) function $g(f) = \max(0, f)$, motivated by two reasons. First, RL is asymptotically equivalent to softplus $g(f) = \log\left(1 + e^f\right)$ (SP) and can therefore serve as an alternative in scenarios in which SP is preferred over exponential $g(f) = e^f$ (EXP) [Seeger and Bouchard, 2012, Park et al., 2014]. Second, RL is essential for physically plausible models of particles (photons and others) arriving at a sensor [Starck and Murtagh, 2002, Dupé et al., 2008, Carlavan and Blanc-Féraud, 2012, Kuusela and Panaretos, 2015], making a broad range of methods developed for Gaussian noise available. Here, the SP non-linearity might lead to instabilities: in order to achieve a very low intensity λ, e.g. for dark pixels, we would require $f \to -\infty$.

Although image reconstruction problems are typically addressed by point estimation, there are compelling arguments for full posterior inference: Apart from benefits such as uncertainty quantification and a principled way for hyper-parameter learning, a practical inference method may be necessary to tackle difficult high-level problems in this context, such as blind deconvolution. For this severely ill-posed problem, where neither the blur kernel nor the original image are known, Levin et al. [2011] show that joint MAP estimation tends to lead to degenerate solutions which can be avoided by using the marginal likelihood for learning.

A major issue in tackling approximate inference is the non-differentiability of the log-posterior when using the RL function as well as due to the use of common image priors [Seeger, 2008, Seeger and Nickisch, 2011a], making many popular gradient-based methods such as Laplace's method or Variational Bayes inconvenient or impossible to apply [Gerwinn et al., 2008]. We therefore chose the Expectation Propagation algorithm [Minka, 2001b, Opper and Winther, 2005], known to gracefully deal with a much larger variety of models while delivering practical accuracy and performance [Kuss and Rasmussen, 2005, Nickisch and Rasmussen, 2008]. Its greater generality however can come at the cost of a numerically more challenging implementation. Meeting these challenges is at the heart of our contributions presented here, which we summarize as follows.

Our main result is to provide a simple dynamic program to compute the moments of a distribution

$$\hat{P}(f) \propto t_y(f)\mathcal{N}(f \mid \mu, \sigma^2) \tag{2.3}$$

with the RL non-linearity, a necessary step to run Expectation Propagation approximate inference. We demonstrate, that in comparison to a generic quadrature-based implementation, our formulation (a) is more efficient to compute and (b) can operate in regimes where quadrature experiences numerical instabilities. We conduct a series of experiments that corroborate the utility of using this model: On the mining-disaster data set we show that compared to the RL function, using the exponential function can be harmful in terms of generalization performance. On a deconvolution problem of natural images with

	Exponential	Softplus	Rectified-Linear
$g(f)$	$\exp(f)$	$\log(1+\exp(f))$	$\max(0, f)$

Table 2.1 – Typical non-linearities in the context of Poisson likelihoods (see text).

Poisson noise, where the use of other non-linearities led to numerical instabilities and convergence issues, we show that taking into account the correct noise model significantly reduces the reconstruction error compared a Gaussian likelihood.

Related Work. Posterior inference of the latent variable $\boldsymbol{f}$ in our setting is intractable and requires approximations due to the use of non-conjugate priors. Approximate inference with RL has not been extensively studied, in contrast to its differentiable alternatives, especially EXP [Vanhatalo et al., 2010, Gerwinn et al., 2010, Diggle et al., 2013, Ko and Khan, 2014]. In their work on actively learning a model for neural spike trains from stimuli, Park et al. [2014] use both SP and EXP and observe that EXP is much less sensitive at lower rates. We qualitatively illustrate their observation, by showing effect of the different non-linearities on the posterior mean intensity $\mathrm{E}[\boldsymbol{\lambda}|\boldsymbol{y}]$ in Figure (2.4), where we fitted Model (2.1) with a Gaussian-process prior to the mining-disaster dataset [Jarrett, 1979][1]. The dataset consists of records of accidents over time, each of which is represented by a black line. Most notable is the different behavior in the high density area on the left, to which the exponential non-linearity responds strongest. For their final method, Park et al. [2014] use SP for its superior predictive performance. Seeger and Bouchard [2012] use a Poisson likelihood with SP for a generalized probabilistic PCA model. They chose SP over EXP to be less sensitive to outliers and to fulfill the technical conditions to apply their approximation method. Kuusela and Panaretos [2015] use RL in their forward model of a particle detector, but tackle the inference using MCMC.

2.1.2 Inference for the Poisson Likelihood Model

Before proceeding to discuss inference methods for the Poisson likelihood model, we briefly revisit the relevant priors for the latent variable $\boldsymbol{f}$. Here, we consider two classes of prior distributions, that are commonly encountered in practice: Gaussian process (GP) priors [Rasmussen and Williams, 2006] and sparse linear models (SLM) [Seeger, 2008, Seeger and Nickisch, 2011a].

Gaussian Processes. GP priors prominently feature in applications of spatio-temporal statistics to social or ecological questions [Vanhatalo et al., 2010, Diggle et al., 2013] or

---

[1] We took this example from [Vanhatalo et al., 2013], using the same setup, i.e. isotropic squared exponential kernel function and constant mean. Inference is done using Expectation Propagation. Hyperparameters for mean and covariance function are learned. More information on the data can be found in Section (2.1.4.2).

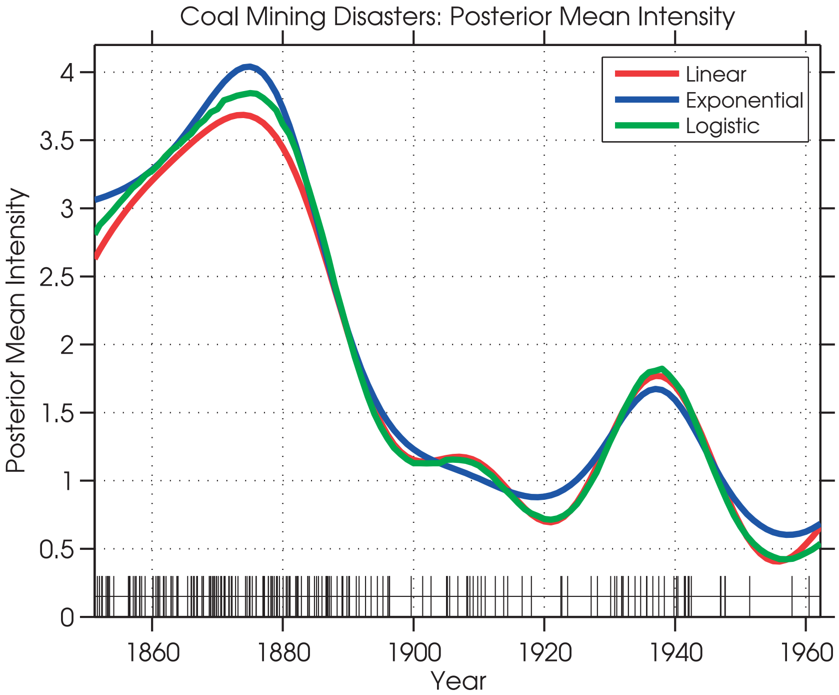


Figure 2.4 – Coal mining disaster data: posterior means of latent functions $\mathrm{E}[g(\boldsymbol{f})|\boldsymbol{y}]$. We recognize the stronger peaking behavior of the exponential non-linearity in high-density regions, while the other non-linearities are more sensitive in low density regions.

to the analysis of neural spike counts [Pillow, 2007, Park and Pillow, 2013, Park et al., 2014] as the multivariate normal is well suited to represent dependencies and dynamics in the input domain. We use the following notation: Let $f : \mathcal{X} \mapsto \mathbb{R}$ be a latent function distributed as $f \sim \mathrm{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}'))$, where $\mu(\boldsymbol{x})$ and $k(\boldsymbol{x}, \boldsymbol{x}')$ denote the mean- and covariance functions. For M inputs $\{\boldsymbol{x}_j \in \mathcal{X}\}_{j=[M]}$ corresponding to the observations $\boldsymbol{y}$, the prior over $\boldsymbol{f}$ can be written as a multi-variate normal distribution:

$$P(\boldsymbol{f}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{K}) \tag{2.4}$$

with mean vector $\mu_j = \mu(\boldsymbol{x}_j)$ and covariance- or kernel matrix $\boldsymbol{K}_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j), \forall i, j \in [M]$.

Sparse Linear Models. In SLMs $\boldsymbol{f}$ itself is defined as a linear function $\boldsymbol{f} = \boldsymbol{X}\boldsymbol{u}$ of a latent vector $\boldsymbol{u}$, where $\boldsymbol{u}$ exposes non-Gaussian, heavy-tailed statistics in an appropriately chosen linear transform domain $\boldsymbol{s} = \boldsymbol{B}\boldsymbol{u}$. SLMs are often encountered in the context of inverse problems, e.g. in image processing, where the prior belief that image gradients or Wavelet coefficients of natural images are sparse [Simoncelli, 1999], has become a popular strategy to regularize ill-posed reconstruction problems. For example, for the deconvolution problem we define the linear operator $\boldsymbol{X}$ such that multiplying it with a vectorized image $\boldsymbol{u}$ amounts to convolving the image with a blur kernel $\boldsymbol{k}$, i.e. $\boldsymbol{f} = \boldsymbol{k} * \boldsymbol{u} = \boldsymbol{X}\boldsymbol{u}$. Assuming that $\boldsymbol{u}$ is well described by piecewise constant functions,

$g(f)$	Prior	Laplace	VB	EP
$\exp(f)$	GP	Tract.	Tract.	Approx.
	SLM	N/A		
$\log(1 + \exp(f))$	GP	Tract.	Approx.	Approx.
	SLM	N/A		
$\max(0, f)$	GP	N/A.	N/A	Tract. (**New**)
	SLM	N/A		

Table 2.2 – Variational Inference methods for different non-linearities. We use the following abbreviations: *Tract.*: Computations are analytically *Tractable* (i.e. gradients/updates available in closed form). *Approx.*: Computations require additional *Approximations*, such as bounding techniques or numerical integration.

one could be interested in penalizing the total variation of $\boldsymbol{u}$, such that $\boldsymbol{B} = [\nabla_x; \nabla_y]$ consists of the horizontal and vertical gradient operators. We model sparsity for $\boldsymbol{s}$ independently for each transform coefficient:

$$P(\boldsymbol{u}) \propto \prod_{j}^{M} t_s(s_j) \tag{2.5}$$

For simplicity we consider the Laplace potential $t_s(s_j) = e^{-\tau|s_j|}$ [Gerwinn et al., 2008, Seeger, 2008, Seeger and Nickisch, 2011a].

Before we begin the discussion of methods for approximate inference, we unify our notation. We would like to approximate an intractable distribution of the following form:

$$P(\boldsymbol{f}) = Z^{-1} \prod_{j=1}^{M} t_j(f_j)\; t_0(\boldsymbol{f}) \tag{2.6}$$

For GPs the optional coupled potential $t_0(\boldsymbol{f})$ is the prior defined in (2.4), and we have a product of the $M = N$ likelihood potentials $t_j(f_j) = P(y_j \mid \lambda_j)$. For SLMs $t_0(\boldsymbol{f}) = 1$, and we redefine $\boldsymbol{f} = [\boldsymbol{X};\ \boldsymbol{B}]\boldsymbol{u}$. The potentials are $t_j(f_j) = P(y_j \mid f_j) = t_{y_j}(f_j)$ for $j \le M$ and $t_j(f_j) = t_s(f_j)$ for $j > M$. We denote the approximation to $P(\boldsymbol{f})$ by $Q(\boldsymbol{f})$ and choose it to be a multivariate normal distribution $Q(\boldsymbol{f}) = \mathcal{N}(\boldsymbol{f} \mid \boldsymbol{\xi}, \boldsymbol{\Xi})$. This is justified by the fact that the likelihood for all non-linearities in Table 2.1 as well as the priors mentioned here are log-concave in $\boldsymbol{f}$, as well as the priors considered here. Therefore, the posterior is unimodal [Paninski, 2004].

In Table 2.2, we list common approximate inference techniques to find the parameters of the approximation. While Laplace's method is often used in the GP setting [Diggle et al., 2013, Park and Pillow, 2013, Park et al., 2014], it cannot be applied to SLMs, because by design we expect many transform coefficients to be zero, where the Laplace potential is not differentiable. Another popular variational Bayesian (VB) technique is referred to as Variational Gaussian approximation [Opper and Archambeau, 2009] or KL method [Nickisch and Rasmussen, 2008, Challis and Barber, 2011]. It is analytically tractable for the exponential function [Ko and Khan, 2014], whereas the softplus function requires approximations, e.g. quadrature or bounding techniques, as shown in [Seeger and Bouchard, 2012]. For the RL function, however, this method is not even defined. This can be seen by examining the VB objective which is the following Kullback-Leibler divergence:

$$\min_{\boldsymbol{\xi},\boldsymbol{\Xi}} D_{\mathrm{KL}}\left[Q(\boldsymbol{f})\|P(\boldsymbol{f})\right] \tag{2.7}$$

Expanding it, here in the GP case, gives

$$D_{\mathrm{KL}}\left[Q(\boldsymbol{f})\|P(\boldsymbol{f})\right] = \mathrm{E}_Q\left[\log \frac{Q(\boldsymbol{f})}{t_0(\boldsymbol{f})}\right] - \sum_{j=1}^{M} \mathrm{E}_Q\left[\log t_{y_j}(f_j)\right] \tag{2.8}$$

This reveals that the logarithm of Eq. (2.1) needs to be integrated over the real line, which is infinite in case of the RL function:

$$\mathrm{E}_Q\left[\log P(y \mid f)\right] \doteq y \int Q(f) \log\left(\max\left(0, f\right)\right) \mathrm{d}f = -\infty \tag{2.9}$$

A simple fix would be to slightly modify the RL function to be non-zero $\max(\epsilon, f)$ for $\epsilon > 0$. As we will see next, Expectation Propagation approximate inference does not require such modifications and deals much more gracefully with non-differentiability.

2.1.2.1 Expectation Propagation

We briefly recapitulate our exposition of EP from Section 1.2.3.3. EP [Minka, 2001a, Opper and Winther, 2005] approximates $P(\boldsymbol{f})$ in Eq. (2.6) by approximating each non-Gaussian potential $t_j(f_j)$ using unnormalized Gaussians $\tilde{t}_j(f_j) = \tilde{Z}_j \mathcal{N}(f_j|\tilde{\mu}_j, \tilde{\sigma}_j^2)$ to form a Gaussian approximation $Q(\boldsymbol{f})$ following the same factorization:

$$Q(\boldsymbol{f}) = Z_{\mathrm{EP}}^{-1} \prod_{j=1}^{M} \tilde{t}_j(f_j)\, t_0(\boldsymbol{f}) \tag{2.10}$$

The EP-approximation to the marginal likelihood is given by:

$$Z_{\mathrm{EP}} = \prod_{j=1}^{M} \tilde{Z}_j \int \prod_{j=1}^{M} \mathcal{N}(f_j|\tilde{\mu}_j, \tilde{\sigma}_j^2)\, t_0(\boldsymbol{f})\, \mathrm{d}\boldsymbol{f} \tag{2.11}$$

EP employs the following strategy to determine the variational parameters $\tilde{\mu}_j, \tilde{\sigma}_j^2$.

We define the i-th marginal *cavity* distribution by removing the i-th *approximate* potential

$\tilde{t}_i(f_i)$ from $Q(\boldsymbol{f})$ and marginalizing over $\boldsymbol{f}_{\setminus i} := \{f_j : j \neq i\}$, denoted as :

$$Q_{-i}(f_i) = \mathcal{N}\left(f_i|\mu_{-i}, \sigma^2_{-i}\right) \propto \int \prod_{j \neq i} \tilde{t}_j(f_j)\, t_0(\boldsymbol{f})\, \mathrm{d}\boldsymbol{f}_{\setminus i} \tag{2.12}$$

The so called *tilted* distribution replaces the approximate potential $\tilde{t}_i(f_i)$ in $Q(\boldsymbol{f})$ by the true non-Gaussian potential $t_i(f_i)$ by multiplying it with the cavity marginal:

$$\hat{P}(f_i) = \hat{Z}_i^{-1}\, t_i(f_i)\, Q_{-i}(f_i) \qquad \text{where} \quad \hat{Z}_i = \int t_i(f_i)\, Q_{-i}(f_i)\, \mathrm{d}f_i \tag{2.13}$$

The criterion to minimize in order to update the parameters of $\tilde{t}_i$ is the KL-divergence between the tilted- and the variational distribution $D_{\mathrm{KL}}\left[\hat{P}(f_i) \| Q(f_i)\right]$ and can be solved using moment matching:

$$\begin{aligned} \mathrm{E}_Q\left[f_i\right] &= \mathrm{E}_{\hat{P}}\left[f_i\right] \\ \mathrm{Var}_Q\left[f_i\right] &= \mathrm{Var}_{\hat{P}}\left[f_i\right] \end{aligned} \tag{2.14}$$

The constant $\tilde{Z}_i$ is chosen such that the normalization constants of $\hat{P}(f_i)$ and $Q(f_i)$ match, i.e. we solve:

$$\tilde{Z}_i \int \mathcal{N}(f_i|\tilde{\mu}_i, \tilde{\sigma}_i^2)\, Q_{-}(f_i)\, \mathrm{d}f_i = \hat{Z}_i \tag{2.15}$$

The EP update therefore consists of determining the first two moments and the normalization constant of the tilted distribution.

Once the parameters of a single $\tilde{t}_i$ are changed, we can update the representation of the full approximation $Q(\boldsymbol{f})$, which consists of recomputing $\boldsymbol{\xi}$ and $\mathrm{Var}_Q\left[\boldsymbol{f}\right] = \mathrm{diag}\left(\boldsymbol{\Xi}\right)$. This process is repeated until convergence, i.e. until a fixed point in terms of tilted and approximate moments is reached.

To the best of our knowledge, tilted moments for the exponential- and softplus functions are not available in closed form. Implementations based on quadrature are commonly found in the context of Gaussian processes [Rasmussen and Nickisch, 2010, Vanhatalo et al., 2013].

Computing tilted marginals is not a trivial task. E.g. plugging (3.1) into (2.13) shows that this quantity depends exponentially both on y and f. In Section 2.1.4 we illustrate that evaluating this expression directly using quadrature can lead to numerical problems.

So far, we have seen that popular methods, such as Laplace and VB approximations, are not particularly suitable for the RL function in contrast to EP, which in turn depends on the tractability of tilted moments. Next, we show that for the RL function these computations are indeed analytically tractable.

2.1.3 Tractable EP Updates for the Rectified-Linear Function

We drop indices and consider the update of a single approximate potential $\tilde{t}(s) = \tilde{Z}\,\mathcal{N}\left(\tilde{\mu}, \tilde{\sigma}^2\right)$. To obtain the first and second moments, it suffices to compute $\log \hat{Z}$, $\alpha := \frac{\partial}{\partial \mu_-} \log \hat{Z}$ and $\beta := -\frac{d^2}{d\mu_-^2} \log \hat{Z}$ since $\log \hat{Z}$ is related to the moment generating function. From these quantities, we can directly update the parameters of the approximate potential. Rasmussen and Williams [2006] show that the parameters of the approximate potential depend on α and β in the following way:

$$\tilde{\sigma}^2 = \frac{1 - \beta\sigma_-^2}{\beta} \qquad\qquad \tilde{\mu} = \tilde{\sigma}^2 \frac{\alpha + \beta\mu_-}{1 - \beta\sigma_-^2} \tag{2.16}$$

Our main result constructively shows how to compute $\hat{Z}$.

First, we rewrite the rectified-linear Poisson potential in order to explicitly take into account the edge case resulting from the latent variable taking on a negative value. Consider the likelihood potential $t_y(f) = \frac{1}{y!}\lambda^y e^{-\lambda}$ for $\lambda = \max\{0, f\}$. Then, for $f \leq 0$, $\lambda = 0$, in which case the Poisson distribution degenerates in the sense that all mass is put on $y = 0$. For convenience, we rewrite $t_y(f)$ as follows:

$$\begin{aligned}
t_y(f) &= \underbrace{(y!)^{-1}\, f^y e^{-f}\, \mathbb{1}(f > 0)}_{=:t_y^+(f)} + \underbrace{\mathbb{1}(y = 0)\mathbb{1}(f \leq 0)}_{=:t_y^-(f)} && (2.17)\\
&= t_y^+(f) + t_y^-(f) && (2.18)
\end{aligned}$$

Thus, by linearity, we can decompose the tilted partition function into two parts:

$$\begin{aligned}
\hat{Z} &= \int t_y(f)\mathcal{N}(f \mid \mu_-, \sigma_-^2)\,\mathrm{d}f && (2.19)\\
&= \int \left(t_y^+(f) + t_y^-(f)\right)\mathcal{N}(f \mid \mu_-, \sigma_-^2)\,\mathrm{d}f && (2.20)\\
&= \underbrace{\int t_y^+(f)\mathcal{N}(f \mid \mu_-, \sigma_-^2)\,\mathrm{d}f}_{=:\hat{Z}_+} + \underbrace{\int t_y^-(f)\mathcal{N}(f \mid \mu_-, \sigma_-^2)\,\mathrm{d}f}_{=:\hat{Z}_-} && (2.21)\\
&= \hat{Z}_+ + \hat{Z}_- && (2.22)
\end{aligned}$$

Our goal will be to compute $\log \hat{Z}_+$ and, if $y = 0$, $\log \hat{Z}_-$, in which case we can obtain $\log \hat{Z}$ as $\log \hat{Z} = \log \hat{Z}_+ + \log\left(1 + \exp\left(\log \hat{Z}_- - \log \hat{Z}_+\right)\right)$ and derivatives

$$\frac{\partial}{\partial \mu_-} \log \hat{Z} = \hat{Z}^{-1}\left(\frac{\partial}{\partial \mu_-}\hat{Z}_+ + \frac{\partial}{\partial \mu_-}\hat{Z}_-\right) \tag{2.23}$$

$$\frac{\partial^2}{\partial {\mu_-}^2} \log \hat{Z} = -\hat{Z}^{-2}\left(\frac{\partial}{\partial \mu_-}\hat{Z}_+ + \frac{\partial}{\partial \mu_-}\hat{Z}_-\right)^2 + \hat{Z}^{-1}\left(\frac{\partial^2}{\partial {\mu_-}^2}\hat{Z}_+ + \frac{\partial^2}{\partial {\mu_-}^2}\hat{Z}_-\right) \tag{2.24}$$

We obtain $\frac{\partial}{\partial \mu_-}\hat{Z}_+$ from $\frac{\partial}{\partial \mu_-}\log \hat{Z}_+$ by using

$$\frac{\partial}{\partial \mu_-}\log \hat{Z}_+ = \hat{Z}_+^{-1}\frac{\partial}{\partial \mu_-}\hat{Z}_+ \Rightarrow \frac{\partial}{\partial \mu_-}\hat{Z}_+ = \hat{Z}_+\frac{\partial}{\partial \mu_-}\log \hat{Z}_+ \tag{2.25}$$

and $\frac{\partial^2}{\partial {\mu_-}^2}\hat{Z}_+$ from $\frac{\partial^2}{\partial {\mu_-}^2}\log \hat{Z}_+$

$$\begin{aligned}\frac{\partial^2}{\partial {\mu_-}^2}\log \hat{Z}_+ &= -\hat{Z}_+^{-2}\left(\frac{\partial}{\partial \mu_-}\hat{Z}_+\right)^2 + \hat{Z}_+^{-1}\frac{\partial^2}{\partial {\mu_-}^2}\hat{Z}_+ \\ \Rightarrow \quad \frac{\partial^2}{\partial {\mu_-}^2}\hat{Z}_+ &= \hat{Z}_+^{-1}\left(\frac{\partial}{\partial \mu_-}\hat{Z}_+\right)^2 + \hat{Z}_+\left(\frac{\partial}{\partial \mu_-}\log \hat{Z}_+\right)^2\end{aligned} \tag{2.26}$$

The term $\hat{Z}_-$ is straight-forward to deal with. Assuming $y = 0$, we have

$$\begin{aligned}\hat{Z}_- &= \int \mathbb{1}(f \leq 0)\mathcal{N}\left(f \mid \mu_-, \sigma_-^2\right)\,\mathrm{d}f && (2.27)\\ &= \int_{-\infty}^{0} \mathcal{N}\left(f \mid \mu_-, \sigma_-^2\right)\,\mathrm{d}f && (2.28)\\ &= \Phi(z), z := -\frac{\mu_-}{\sigma_-} && (2.29)\end{aligned}$$

and derivatives

$$\begin{aligned}\frac{\partial}{\partial \mu_-}\log \hat{Z}_- &= -\frac{\phi(z)}{\sigma_-\hat{Z}_-} && (2.30)\\ \frac{\partial^2}{\partial {\mu_-}^2}\log \hat{Z}_- &= -\left(\frac{\phi(z)}{\sigma_-\hat{Z}_-}\right)^2 + \frac{z\phi(z)}{\sigma_-^2\hat{Z}_-} && (2.31)\end{aligned}$$

The term $\hat{Z}_+$ requires some more work. The following proposition states our main result, which we show constructively by developing a simple dynamic program to compute $\hat{Z}_+$ in time linear in y.

Proposition 1. *The tilted partition function $\hat{Z}_+$ can be computed in $O(y)$.*

Proof. We can write the partition function of the tilted distribution as

$$\hat{Z}_+ = \frac{1}{y!}\int_0^\infty f^y e^{-f}\mathcal{N}(f|\mu_-, \sigma_-^2)\,\mathrm{d}f \tag{2.32}$$

The exponential term results in a shift of the cavity mean and a constant factor:

$$\hat{Z}_+ = \frac{1}{y!}e^{\frac{1}{2}\sigma_-^2-\mu_-}\int_0^\infty f^y\mathcal{N}(f|\mu_- - \sigma_-^2, \sigma_-^2)\,\mathrm{d}f \tag{2.33}$$

Thus, computing $\hat{Z}_+$ boils down to computing the y-th moment of a truncated Gaussian. Define $m = \mu_- - \sigma_-^2$, $v = \sigma_-^2$, and $\kappa = -\frac{m}{\sqrt{v}}$. Let $I_y = \int_0^\infty f^y \mathcal{N}(f|m, v)\,\mathrm{d}f$. For $y \in \{0, 1\}$ the integral I_y is readily evaluated as

$$I_0 = 1 - \Phi(\kappa) \qquad I_1 = mI_0 + \sqrt{v}\phi(\kappa) \tag{2.34}$$

where $\phi(x)$ is the standard normal density and $\Phi(x)$ its CDF. For $y > 1$ the application of integration by parts results in a recursion over y :

$$I_y = \int_0^\infty f^{y-1} f \mathcal{N}\left(f|m, v\right)\,\mathrm{d}f = mI_{y-1} + v(y-1)I_{y-2} \tag{2.35}$$

where we have used that

$$f\mathcal{N}(f|m, v) = m\mathcal{N}(f|m, v) - v\frac{\partial}{\partial f}\mathcal{N}(f|m, v) \tag{2.36}$$

For a potentially more robust implementation, we we express $\hat{Z}_+$ in terms of $L_y := \frac{\partial}{\partial m}\log I_y$, which can be written recursively as well. We begin by writing $L_y = \frac{\partial}{\partial m}\log I_y = I_y^{-1}\frac{\partial}{\partial m}I_y$. By using symmetry, we note that $\frac{\partial}{\partial m}\mathcal{N}\left(f|m, v\right) = -\frac{\partial}{\partial f}\mathcal{N}\left(f|m, v\right)$. Thus, for $y > 0$, we get

$$\frac{\partial}{\partial m}I_y = \int_0^\infty f^y \frac{\partial}{\partial m}\mathcal{N}\left(f|m, v\right)\,\mathrm{d}f \tag{2.37}$$

$$= -\int_0^\infty f^y \frac{\partial}{\partial f}\mathcal{N}\left(f|m, v\right)\,\mathrm{d}f \tag{2.38}$$

$$= \left[f^y \mathcal{N}\left(f|m, v\right)\right]_0^\infty + y\int_0^\infty f^{y-1}\mathcal{N}\left(f|m, v\right)\,\mathrm{d}f \tag{2.39}$$

$$= yI_{y-1} \tag{2.40}$$

where we have used integration by parts and $L_y = \frac{yI_{y-1}}{I_y}$. Using this and (2.35), we can write

$$L_y = \frac{yI_{y-1}}{I_y} = \frac{yI_{y-1}}{mI_{y-1} + vI_{y-2}} = \frac{y}{m + vL_{y-1}} \tag{2.41}$$

where the base cases are

$$L_0 = \phi(\kappa)/(\sigma_-(1 - \Phi(\kappa))) \qquad L_1 = I_0/I_1 \tag{2.42}$$

Then, we can accumulate I_y recursively in the log-domain:

$$\log I_y = -\sum_{r=1}^{y}\log L_r + \log I_0 + \log(y!) \tag{2.43}$$

such that finally

$$\log \hat{Z}_+ = \log I_y - \log(y!) + \frac{1}{2}\sigma_-^2 - \mu_- \tag{2.44}$$

□

Since $\frac{\partial}{\partial m} \log \hat{Z}_+ = \frac{\partial}{\partial \mu_-} \log \hat{Z}_+$, we arrive at $\alpha = L_y - 1$.

Similarly, we can show that $\beta = L_y(L_y - L_{y-1})$. We have $\beta = -\frac{\partial}{\partial m}\alpha = -\frac{\partial}{\partial m}L_y$. For convenience, we denote $\frac{\partial}{\partial m}\cdot$ by $(\cdot)'$. Then,

$$\begin{aligned}
L_y' &= \left(\frac{I_y'}{I_y}\right)' && (2.45)\\
&= \frac{I_y''}{I_y} - \left(\frac{I_y'}{I_y}\right)^2 && (2.46)\\
&= \frac{y(y-1)I_{y-2}}{I_y} - (L_y)^2 && (2.47)\\
&= \frac{y(y-1)I_{y-2}}{mI_{y-1} + v(y-1)I_{y-2}} - (L_y)^2 && (2.48)\\
&= \frac{yL_{y-1}}{m + vL_{y-1}} - (L_y)^2 && (2.49)\\
&= L_yL_{y-1} - (L_y)^2 && (2.50)\\
&= -L_y\left(L_y - L_{y-1}\right) && (2.51)
\end{aligned}$$

In (2.48) we used the recursion for I_y (2.35) and in (2.50) the recursion for L_y (2.41). Therefore, we have $\beta = L_y(L_y - L_{y-1})$, such that all relevant quantities can be computed based on L_y.

An alternative way to characterize moments of $\hat{P}$, which allows us to compute them to higher order is the following

Corrolary 1. *The first k moments of $\hat{P}(f)$ can be computed in $O(y + k)$.*

Proof. The k-th moment can be written as follows, where we index the partition functions by y for clarity

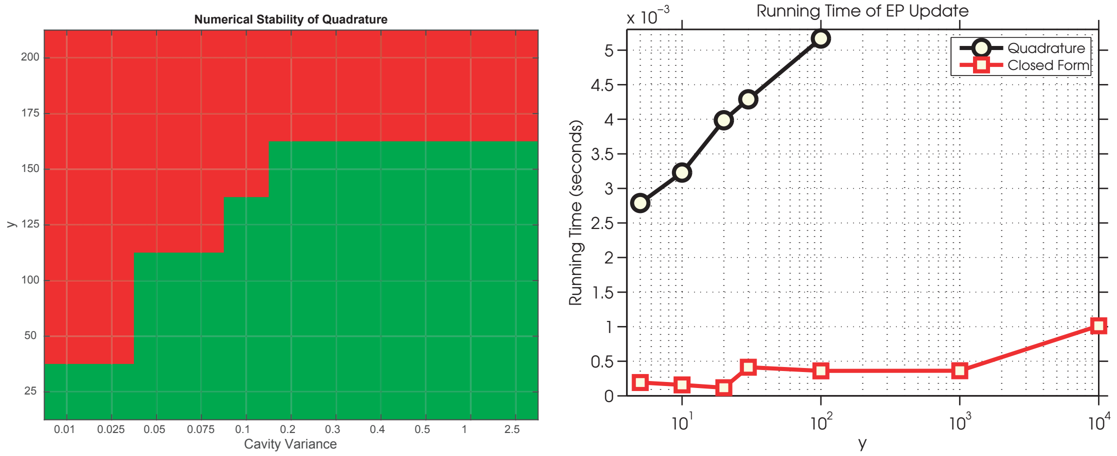
$$\begin{aligned}
\mathrm{E}_{\hat{P}}\left[f^k\right] &= \frac{1}{\hat{Z}(y)} \int f^k t_y(f) Q_-(f)\,\mathrm{d}f && (2.52)\\
&= \frac{1}{\hat{Z}(y)} \left(\int_0^\infty \frac{1}{y!} f^{y+k} e^{-f} Q_-(f)\,\mathrm{d}f + \mathbb{1}(y=0)\int_{-\infty}^0 f^k Q_-(f)\,\mathrm{d}f\right) && (2.53)\\
&= \frac{1}{\hat{Z}(y)} \left(\frac{(y+k)!}{y!}\hat{Z}_+(y+k) + \mathbb{1}(y=0)\int_{-\infty}^0 f^k Q_-(f)\,\mathrm{d}f\right) && (2.54)
\end{aligned}$$

Thus, $\hat{Z}_{+}(y+k)$ can be computed using our previous result by running the recursion for $y+k$ steps. To compute the remaining integral in case of $y=0$, we again need to compute moments of a truncated normal, which can be done using a similar recursion as in (2.35) in $O(k)$ steps. ☐

Having access to all moments allows us to compute higher-order cumulants as well. Thus, for GP priors the techniques to correct the EP approximation described in Opper et al. [2013] directly apply to this likelihood.

2.1.3.1 Implementation Details

We implemented the EP updates in C/C++ and used the GPML MATLAB toolbox for experiments [Rasmussen and Nickisch, 2010]. We use parallel updating EP with the option of fractional updates [Seeger, 2008], which turned out to be unnecessary as EP converged reliably within 15 to 20 iterations. We used MATLAB's Parallel Computing Toolbox to compute variance for the SLM experiments with up to 2^{14} variables on NVIDIA Tesla C2070 GPUs with 6 GB device memory built into a workstation equipped with dual Intel Xeon X5670 CPUs (2.93 GHz), and 128 GB Memory. These computations were performed in single precision resulting in a large speedup without a negative impact on the convergence of EP, which can be quite sensitive to inaccurately approximated variances [Papandreou and Yuille, 2011]. Thus, with minimal effort and without further optimizations, we could run a single iteration of EP in about 30 seconds.



(a) Instability of Quadrature

(b) Running Time

Figure 2.5 – **Left: Transition diagram of numerical stability of EP update using quadrature.** Red shading indicates failure due to numerical instability at a setting. Quadrature cannot handle large counts and is sensitive to small cavity variances. Our formulation works reliably in this regime and beyond. **Right: Running time.** We compare the scaling behavior of the running time of a single EP update using our formulation vs. adaptive quadrature as a function of the count y. As shown in Figure (2.5a), quadrature up to the same counts as our recursion.

2.1.4 Experiments

2.1.4.1 Synthetic Data

In the experiments on synthetic data, we investigated the following two aspects: computational performance and numerical stability of our formulation in contrast to quadrature.

First, we investigate the numerical stability of quadrature by examining the behavior for a single EP update for the RL function[2]. We evaluate the unnormalized tilted density as $\hat{Z}\,\hat{P}(f) = e^{\log t(f) + \log Q_{-}(f)}$ and compute $\hat{Z}$ for different values of y and different cavity parameters. Since we can expect the cavity mean to be close to the observation, we set $\mu_{-} = y$ and vary the cavity variance. The outcome is shown in Figure (2.5a), where red shading denotes failure of quadrature resulting in an output which is infinite or not a number. In the green area the output matches our formulation. Our formulation works reliably in all of these cases. Next, we compare the time to compute a single EP update, i.e. $\hat{Z}$ and the moments of $\hat{P}(f)$, using our recursion versus adaptive quadrature. Quadrature needs to be called three times to compute $\hat{Z}$ and the first and second moments of $\hat{P}(f)$, whereas we need to run our recursion only once. Quadrature can certainly be further optimized. But due to its complexity, this can be expected to be an error-prone undertaking.

Since our recursion scales linearly in the count y, we plot the time against y in Figure (2.5b). We see that our recursion is very efficient and runs robustly up to very large counts. As seen before, for quadrature, the computations cannot be run for counts beyond the order of 100. For the comparison, we performed 50 warm-up runs for both methods before averaging the running time over 150 calls to the respective implementations of the EP updates.

2.1.4.2 Cox Processes: Coal Mining Disaster Data

In this experiment we present a case where the use of exponentials hurts generalization performance. We return to the introductory example of the coal mining disaster dataset and setup a prediction task using 10-fold cross validation. The dataset consists of 191 accidents in the period between 1851 and 1962, which we discretized into 100 equidistant bins. We compare inference for the three different non-linearities, using EP for all of them, where the updates for the softplus and exponential are implemented using quadrature[3]. As error measure, we report the average of the negative log-predictive probabilities of the samples in the test fold, where the predictive probability for an unseen observation y^*

---

[2] We use MATLAB's `integral` routine.

[3] The Laplace approximation for the softplus and exponential yielded similar results, so that we do not report them here.

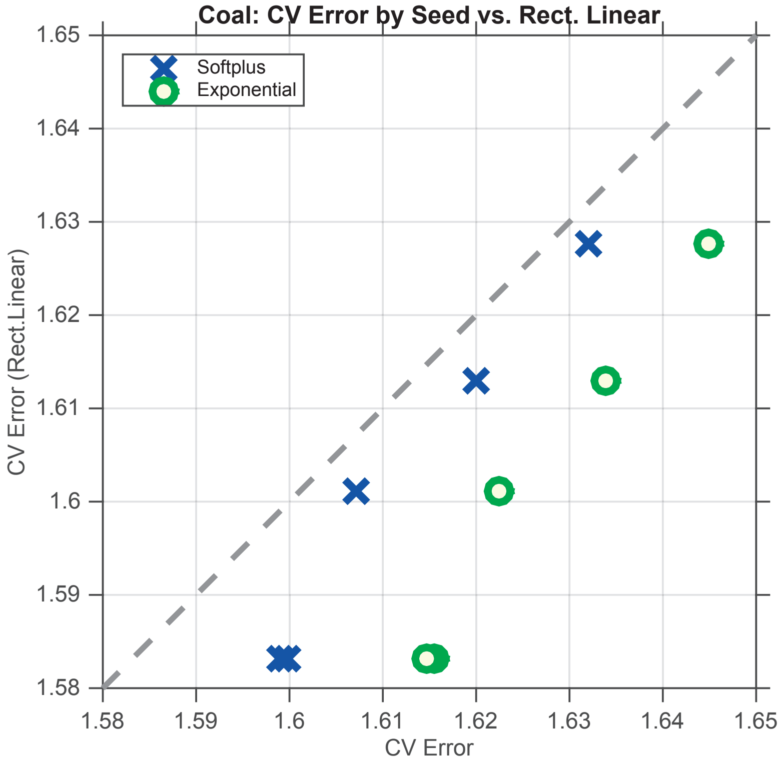


Figure 2.6 – **Coal Mining Disaster Data:** Cross validation errors for different draws of folds. We show errors of Softplus vs. RL and Exponential vs. RL.

given training data $\boldsymbol{y}$ is approximated as:

$$P(y^* \mid \boldsymbol{y}) \approx \int P(y^* \mid g(f^*))Q(f^*, \boldsymbol{f} \mid \boldsymbol{y})\,\mathrm{d}\boldsymbol{f}\,\mathrm{d}f^* \tag{2.55}$$

This quantity can be computed approximately as described in Rasmussen and Williams [2006] and computationally amounts to evaluating $\hat{Z}$.

As in the demo by Vanhatalo et al. [2013], we use a GP prior with a isotropic squared-exponential covariance function and a constant mean. We learn the kernel parameters as well as the mean by maximizing the marginal likelihood on the training fold. We repeatedly ran this experiment for 5 draws of the test folds. We report the cross validation errors in Table 2.3. This is an example where (asymptotically) linear behaviour seems to lead to (slightly) better predictive performance. Softplus- and RL functions perform very similarly in this example, but better than exponential, consistently across different draws

	Exponential	Softplus	Rect. Linear
CV Error	1.63(±0.01)	1.61(±0.01)	**1.60**(±0.02)

Table 2.3 – Coal Mining Disaster Data: Cross Validation Results

of the cross validation folds (Figure 2.6).

2.1.4.3 Sparse Linear Models

In this experiment, we consider a deconvolution problem of natural images under Poisson noise as described in Section 2.1.2. We generate noisy versions of the input as follows: We rescale the maximum intensity of the input image $\boldsymbol{u}$ to a value $u_{\max} \in \{10, 20, 30\}$. We apply Gaussian blur to the image using a 3×3 blur kernel $\boldsymbol{h}$ with standard deviation 0.3 to obtain $\boldsymbol{f} = \boldsymbol{h} * \boldsymbol{u}$ and draw observations from $P(\boldsymbol{y} \mid g(\boldsymbol{f}))$ for 5 different initializations of the random number generator. For reconstruction, we use a total-variation prior with $\boldsymbol{B} = [\nabla_x; \nabla_y]$ and Laplace potentials $t_s(s) = e^{-\tau|s|}$.

Here, we focus on comparing the correct Poisson noise model against a Gaussian noise assumption, which is often chosen based on convenience and familiarity. We use parallel EP for both models to infer the posterior mean as reconstruction. We used grid search to determine hyper-parameters using the marginal likelihood as criterion. We also tried to apply the SP and EXP non-linearities, but experienced numerical instability and convergence issues for a wide range of hyper-parameters.

We report relative ℓ_1 errors [4] of the reconstructions $\hat{\boldsymbol{u}} = \mathrm{E}_Q[\boldsymbol{u} \mid \boldsymbol{y}]$ in Table 2.4. At lower intensities, the signal is much weaker leading generally to a higher error. It is this regime, where the correct likelihood yields the greatest improvements. As the intensity and thus the photon counts increase, the noise is better approximated by a Gaussian, such that both models perform similarly, as expected.

Image	$u_{\max}$	Gauss	Poisson RL
Face 32×32	10	0.488(±0.005)	**0.317**(±0.007)
	20	0.282(±0.008)	**0.248**(±0.026)
	30	0.245(±0.007)	**0.207**(±0.011)
C-Man 128×128	10	0.182(±0.002)	**0.124**(±0.001)
	20	0.113(±0.001)	**0.094**(±0.001)
	30	0.092(±0.001)	**0.084**(±0.001)
Lena 128×128	10	0.224(±0.002)	**0.154**(±0.003)
	20	0.128(±0.001)	**0.111**(±0.001)
	30	0.103(±0.001)	**0.095**(±0.001)

Table 2.4 – Relative ℓ_1 errors for deconvolution with different likelihoods.

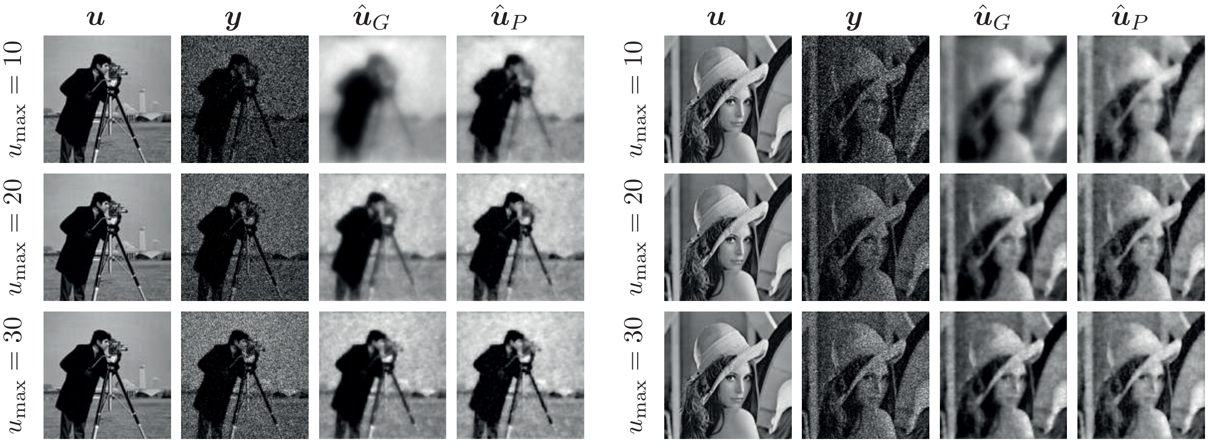
Apart from mere reconstruction errors, it is instructive to visually inspect the reconstructions for both models. We present exemplary reconstructions of the different input images in Figure 2.7 and Figure 2.8. In Figure 2.7 each row corresponds to a different intensity

---

[4] The relative ℓ_1 error of a reconstruction $\hat{\boldsymbol{u}}$ of an image $\boldsymbol{u}$ is defined as $\ell_{\boldsymbol{u}}(\hat{\boldsymbol{u}}) = \|\hat{\boldsymbol{u}} - \boldsymbol{u}\|_1 / \|\boldsymbol{u}\|_1$

level. We denote the reconstruction by $\hat{\boldsymbol{u}}_G$ and $\hat{\boldsymbol{u}}_P$, where a subscript “G” denotes the Gaussian likelihood and “P” the Poisson likelihood.

Poisson noise is difficult to deal with, especially for natural images such as in Figure 2.7b, since fine details become very hard to distinguish from noise. The Gaussian noise model explains the data at very low intensities by an overly smooth image. Noise is removed, but so is also much of the high-frequency content which is crucial for recognizing details. Thus, fine structures tend to be blurred and contrast diminished. Using the Poisson likelihood instead captures edges much better. We illustrate this effect by showing a cross section of Figure 2.8a in Figure 2.9 and magnified sub-images in Figure 2.8b.



(a) (b)

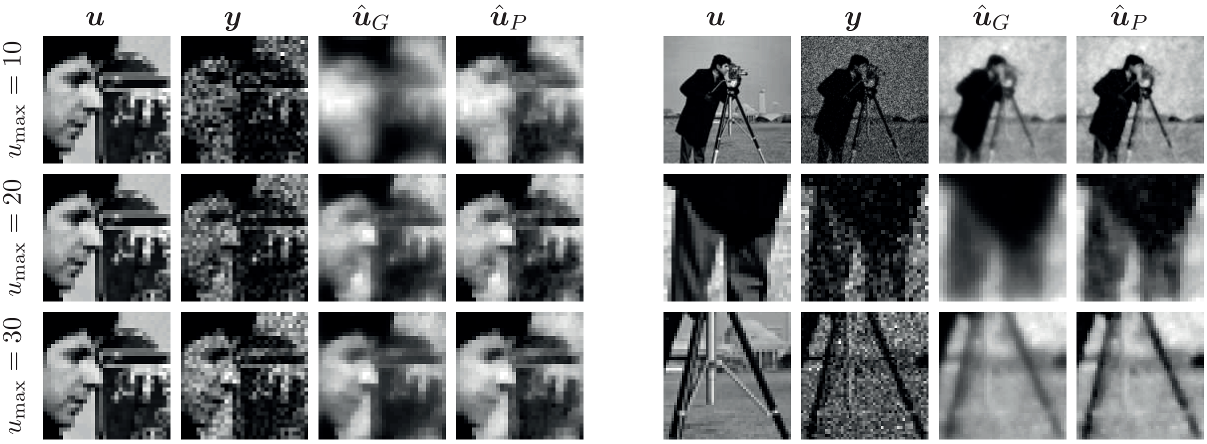
Figure 2.7 – Denoising results at different maximum intensity levels. $\boldsymbol{u}$: input image. $\boldsymbol{y}$: noisy image. $\hat{\boldsymbol{u}}_G$: Gaussian likelihood. $\hat{\boldsymbol{u}}_P$: Poisson likelihood. **Left:** Cameraman 128×128. **Right:** Lena 128×128.

2.1.5 Discussion

We studied inference in Poisson models using the rectified-linear function as non-linearity. This function imposes a hard positivity constraint on the underlying latent variable. This function is the natural and physically plausible choice for models of Poisson noise in image processing, but is challenging to deal with in practice.

Here, we derived an analytically tractable Expectation Propagation algorithm for approximate inference in Poisson likelihood models using the RL function. We showed that in contrast to quadrature, computations required by our formulation are more efficient and numerically stable.

Equipped with this method, we demonstrated that the RL function is useful as an alternative to other non-linearities and that taking into account non-Gaussian noise statistics in a Poisson deconvolution problem leads to superior performance at no extra



(a) (b)

Figure 2.8 – **Left:** Denoising results on high-resolution 32×32 sub-image at different maximum intensity levels. **Right:** Zoom-in comparison at $u_{\max} = 20$ for Cameraman 128×128. $\boldsymbol{u}$: input image. $\boldsymbol{y}$: noisy image. $\hat{\boldsymbol{u}}_G$: Gaussian likelihood. $\hat{\boldsymbol{u}}_P$: Poisson likelihood. The correct noise model helps to recover contrast and distinguish image features from noise.

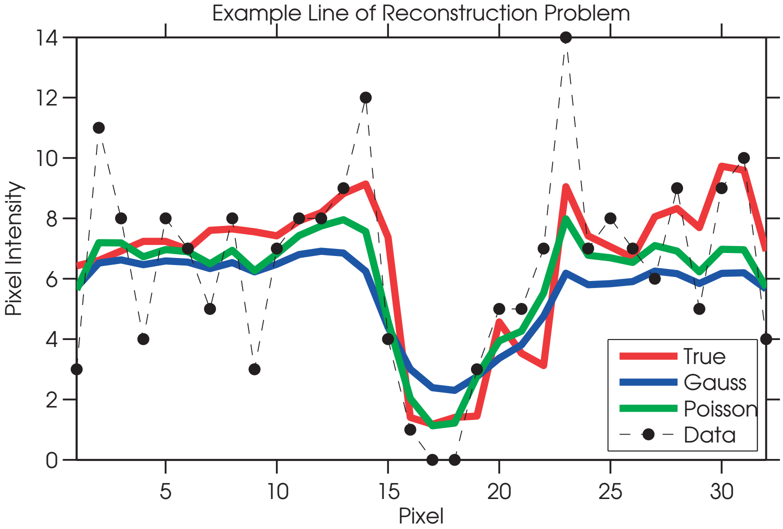


Figure 2.9 – Cameraman Face: Example cross section of $\boldsymbol{u}, \boldsymbol{y}, \hat{\boldsymbol{u}}_G$ and $\hat{\boldsymbol{u}}_P$. We see that modeling the Poisson noise correctly helps recovering contrast and edges, which is crucial for image quality.

cost.

Scalability of the method, however, is limited. Although in our experiments, parallel EP converged within a few iterations, the cost of each iteration scales cubically in the latent dimensionality. Another shortcoming of this study is the lack of a real application, since all experiments were synthetic. An interesting experiment would be to compare an EM algorithm with RL EP with the algorithm of Seeger and Bouchard [2012].

2.2 Tree-Structured Scale Mixtures

2.2.1 Overview and Related Work

We consider structured mixture models, i.e. mixture models of continuous variables, the discrete latent mixture components of which are interrelated by a given graphical model [Koller and Friedman, 2009]. In particular, the motivating application are ill-posed linear inverse problems, common in image restoration, such as denoising, deblurring [Levin et al., 2009] and inpainting [Bertalmio et al., 2000], or compressive sensing tasks [Donoho, 2006, Candès et al., 2006], using sparsity in the Wavelet domain [Portilla et al., 2003] as prior information.

Consider a linear model, where we observe M linear measurements $\boldsymbol{y} \in \mathbb{R}^M$ of an unknown image $\boldsymbol{u} \in \mathbb{R}^N$ with $M \leq N$ corrupted by white Gaussian noise

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{u} + \boldsymbol{\varepsilon}, \qquad \boldsymbol{\varepsilon} \sim \mathcal{N}\left(0, \sigma^2 \boldsymbol{I}\right) \tag{2.56}$$

Recovering $\boldsymbol{u}$ from $\boldsymbol{y}$ is not possible without additional assumptions. One such assumption, particularly fruitful due to its wide applicability, is that the image can be represented as a linear combination of a few atoms from an appropriately chosen basis or dictionary, meaning that the actual effective number of degrees of freedom is much lower than it is apparent from the canonical signal representation such that the information contained in the measurements are sufficient for recovery, or at least a good estimate, of $\boldsymbol{u}$ [Tibshirani, 1996, Olshausen and Field, 1996, Donoho, 2006, Candès et al., 2006]. For natural images, modeling statistically the occurrence of edges as their most salient features, can lead to such sparse descriptions by modeling high-pass filter responses [Portilla et al., 2003]. The Wavelet transform can be seen as applying such edge detection filters with different orientations on multiple rescaled versions of the image leading to a hierarchical decomposition [Mallat, 1999]. Computationally, one step of the discrete Wavelet transform (DWT) consists of filtering with all combinations of 1D low-pass and high-pass filters in all dimensions followed by decimating every second coefficient in all dimensions. The outputs of the different combinations of filters are grouped together according to the orientation of the filter. In 2D the combinations of 1D filters therefore are low-low, low-high, high-low, high-high. Then, the transform recurses on the low-low band, thus proceeding from finer to coarser scales. See Figure 2.10 for an example.

Here, we consider orthonormal Wavelet transforms, that we denote by the linear operator $\boldsymbol{W}$. Importantly, for many types of Wavelets there exist efficient algorithms that can apply $\boldsymbol{W}$ and $\boldsymbol{W}^T$ in $O(N)$. Let $\boldsymbol{s} = \boldsymbol{W}\boldsymbol{u}$ denote Wavelet coefficients. Then, we can impose sparsity on $\boldsymbol{s}$ by using a prior

$$P(\boldsymbol{u}) \propto \prod_{j=1}^{N} t(s_j) \tag{2.57}$$

where $t(\cdot)$ is a sparsity-promoting distribution. This product form, however, neglects structure in the sparsity pattern of the Wavelet coefficients stemming from the hierarchical nature of the operator apparent from its recursive description given above. Figure 2.10 shows, that larger coefficients can persist across scales. Moreover, the relationship of the location of the large coefficients between scales is tree-like, owing to the operations involved in the transform as described above. Specifically, due to the elimination step, it is sensible to relate a coefficient to four children on the next finer scale in the same band. Thus, the persistence structure of non-zeros can be conveniently modeled by a forest of quad trees. Each node in the tree corresponds to a coefficient and is a discrete variable indicating whether the corresponding coefficient has a value close to zero or not. We refer to these states as low (close to zero) and high. An indicator's state depends on the parent's: the children's state should be more likely aligned with the parent's. These notions are the basis for the hidden Markov tree (HMT) model, which was first proposed by Crouse et al. [1998]. We introduce this model in more detail in Section 2.2.2.1.

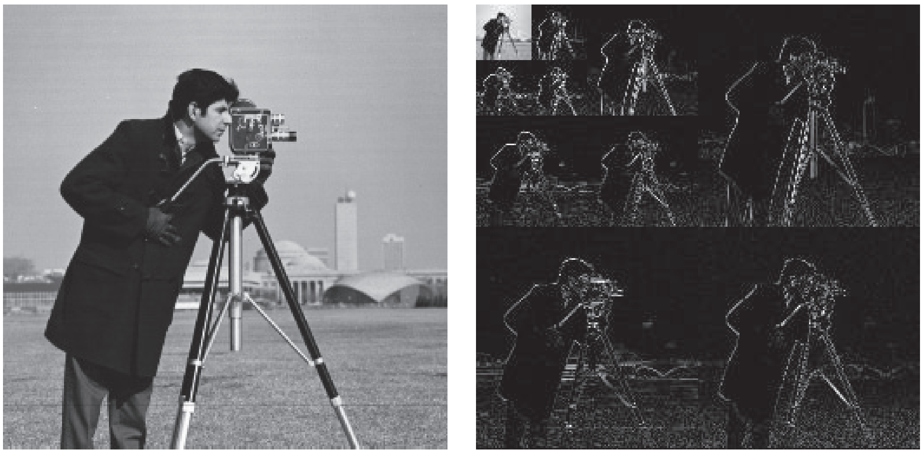
In the following, we describe our work on developing a scalable variational inference algorithm for this image model. Our algorithm combines the double loop algorithm of Seeger and Nickisch [2011a] with the sum-product belief propagation algorithm for trees [Pearl, 1988]. The resulting method applies to models using Gaussian scale-mixture potentials, relevant examples of which are shown in Table 2.5, as the algorithm of Seeger and Nickisch [2011a] optimizes the direct site bounding approximation introduced in Section 1.2.3. Thus, we benefit from the scalability properties of the algorithm by Seeger and Nickisch [2011a], which the authors show to have the same complexity as the MAP estimation problem in the same model, while maintaining a representation of the full posterior covariance. The cubic cost in N due to inverting the natural parameterization of the covariance is mitigated by low rank approximations or, in our case, a Monte Carlo estimate using exact posterior sampling that exploits the special structure of the linear operators involved [Papandreou and Yuille, 2011].

	Density $t(s)$		Comments
Laplace	$\frac{1}{2}\tau e^{-\tau \lvert s \rvert}$	$\tau > 0$	Log-concave, not differentiable at $s = 0$.
Student's T	$C(\nu)(1 + \frac{s^2}{\nu})^{-\frac{\nu+1}{2}}$	$\nu > 0$	Not log-concave. $C(\nu) = \Gamma(\frac{\nu+1}{2})/(\sqrt{\pi\nu}\Gamma(\frac{\nu}{2}))$

Table 2.5 – Two examples of super-Gaussian potentials, used in sparse priors.

Related work. Since its first description for modeling Wavelet coefficients by Crouse et al. [1998], the HMT model has been used for various applications including denoising [Portilla et al., 2003], inpainting [Papandreou et al., 2008], MRI reconstruction [Chen and Huang, 2012] and compressed sensing [Duarte et al., 2008, He and Carin, 2009, He et al., 2010, Som and Schniter, 2012].

Methodological most relevant to our work are the methods developed in the context of



(a) Image domain (b) Wavelet domain

Figure 2.10 – Wavelet transform of an image, demonstrating persistence of non-zeros across scales. Sparsity in the Wavelet domain is structured. Coarser scales are less sparse.

compressed sensing. There, it was established empirically that probabilistic methods integrating over both, the HMT as well as the latent image can lead to superior performance [He and Carin, 2009, He et al., 2010, Som and Schniter, 2012]. He et al. [2010] show that a mean-field variational Bayesian approximation is an order of magnitude faster than their Gibbs sampler with comparable reconstruction error. More recently, Som and Schniter [2012] develop an approximate inference scheme based on Approximate Message Passing [Donoho et al., 2009], an algorithm that can be seen as an approximation of Belief Propagation and Expectation Propagation [Meng et al., 2015]. The resulting algorithm is both much faster than VB of He et al. [2010] and more accurate than the sampling approximation of He and Carin [2009]. In contrast to those methods, that use mixtures of Gaussians or spike and slab priors, our inference scheme generalizes to arbitrary mixtures of Gaussian scale-mixture distributions [Nickisch, 2010]. Furthermore, our method stands alone in supporting full posterior covariances over the image, that could unlock a range of applications beyond reconstruction.

2.2.2 Approximate Inference for Tree-Structured Scale Mixtures

2.2.2.1 Preliminaries

Super-Gaussian distributions. Gaussian linear models, while analytically tractable, are not suitable to model natural image statistics, which are much better characterized by heavy-tailed or leptokurtic distributions, such as the Laplace[5]. In particular, we are interested in symmetric distributions, centered at 0 that admit the following super-

---

[5] MAP estimation under Laplace prior is equivalent to ℓ_1 regularization.

Gaussian representation:

$$t(s) = \max_{\gamma > 0} e^{-\frac{s^2}{2\gamma}} e^{-\frac{1}{2}h(\gamma)}, \tag{2.58}$$

i.e. they can be represented as a point-wise maximum over Gaussian functions. In the context of variational methods, this is particularly convenient, as we obtain a bound in terms of Gaussians with variational parameters γ to adjust [Jaakkola, 1997, Girolami, 2001, Palmer et al., 2006]. Palmer et al. [2006] show that all Gaussian scale-mixtures admit such a representation, which includes Laplace and Student's T. See Nickisch [2010] for more examples.

We use the notation of Seeger and Nickisch [2011a]. The super-Gaussian representation is a consequence of the convexity of $g(x) := \log t(\sqrt{x}), x > 0$, and can be seen by using convex conjugation $g(x) = \max_z zx - g^*(z)$ [Rockafellar, 1970]. Seeger and Nickisch [2011a] show that (2.58) can be obtained by substituting $\gamma = -\frac{1}{2z} > 0$ and $h(\gamma) = 2g^*(-\frac{1}{2\gamma})$. The maximum in (2.58) can therefore be obtained as

$$z^* = \partial_x g(x) = \partial_x \log t\left(\sqrt{x}\right) \Rightarrow \gamma^* = -\left(2\partial_x \log t\left(\sqrt{x}\right)\right)^{-1} \tag{2.59}$$

Hidden Markov Trees. Next, we introduce the HMT model [Crouse et al., 1998] in more detail and the necessary notation. Let $\boldsymbol{u} \in \mathbb{R}^N$ be the column-major vectorization of an image with R rows and C columns. Let

$$\begin{aligned} &\omega_R : [N] \mapsto \{0, \ldots, R-1\} \times \{0, \ldots, C-1\} \\ &\omega_R(j) = (\operatorname{div}(j-1, R), \operatorname{mod}(j-1, R)) \end{aligned} \tag{2.60}$$

be the function that maps a linear index j back to coordinates in the image and

$$\begin{aligned} &\omega_R^{-1} : \{0, \ldots, R-1\} \times \{0, \ldots, C-1\} \mapsto [N] \\ &\omega_R^{-1}(r, c) = cR + r + 1 \end{aligned} \tag{2.61}$$

the mapping from image coordinates to linear index. Here, we index pixels and corresponding transform coefficients starting in the upper left corner from $(0, 0)$ to $\left(2^L - 1, 2^L - 1\right)$ in the lower right corner. For simplicity, we assume $\boldsymbol{u}$ to be a square image with width and height equal to $R = C = 2^L$ and hence, $N = 2^{2L}$. This allows us to compute the DWT for L levels in both dimensions[6]. After l steps, we are left with $N_S := 2^{2(L-l)}$ coefficients in the LL band (upper left corner in Figure 2.10b), called scaling or approximation coefficients [He and Carin, 2009, Som and Schniter, 2012], which are the inputs to the next step of the Wavelet transform. We will refer to the $N_W := N - N_S$ coefficients in the other bands as the Wavelet coefficients. We pair each transform coefficient s_j with a

---

[6] It is typically sufficient to use an incomplete transformation with much less levels [Som and Schniter, 2012]. Moreover, in some of the literature, the problem is simplified by assuming access to the coarse-level coefficients, which simplifies the problem [He and Carin, 2009].

latent binary indicator variable δ_j. Modeling s_j consists of two components. First, s_j is drawn from a mixture $P(s_j \mid \delta_j)$ controlled by δ_j. Second, the δ_j corresponding to Wavelet coefficients are organized in a forest of quad-trees (Figure 2.11). The roots of these quad-trees are the Wavelet coefficients on the coarsest scale. After l steps of the DWT, there are $3 \cdot 2^{2(L-l)}$ such quad-trees of height $l-1$. We denote by $\rho(j)$ the parent of the indicator of a Wavelet coefficient j and by $\zeta(j,c)$ its c-th child coefficient, if either exist. By $\ell(j)$ we denote j's level in the tree where a leaf node's level is 1. Let K denote the actual number of steps to which we restrict the DWT.

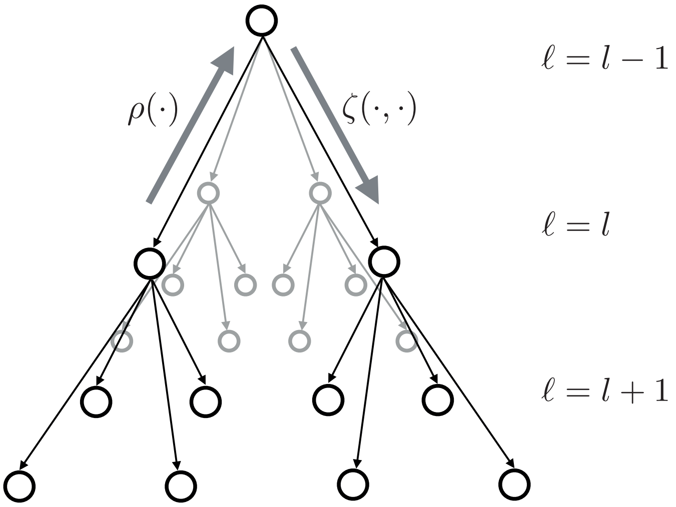


Figure 2.11 – Latent quad-tree-structure of indicator variables. Functions $\rho(j)$ and $\zeta(j,c)$ return the indices of the parent and the c-th child of node j. Furthermore, $\ell(j)$ returns the level of node j.

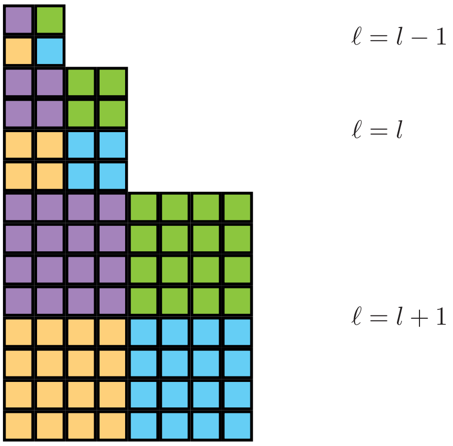
In Figure 2.12a we show as an example the coefficients of a band. Using colors, we distinguish between four quad-trees rooted at the top four coefficients, the members of which appear in the same color. This illustration reveals the heap-like organization of the coefficients, which allows us to efficiently implement the operations $\rho(\cdot)$ and $\zeta(\cdot,\cdot)$:

$$\rho(j) = \omega_R^{-1}(\lfloor \omega_R(j)/2 \rfloor) \tag{2.62}$$

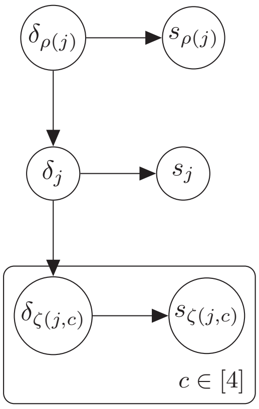
$$\zeta(j,c) = \omega_R^{-1}(2\omega_R(j) + \omega_2(c)) \tag{2.63}$$

where operations on and between tuples are performed element-wise and the four children identified by $c \in [4]$ are mapped to tuples in $\{0,1\} \times \{0,1\}$ by $\omega_2(\cdot)$.

Figure 2.12b illustrates in a graphical model the relationship of an indicator node δ_j with its neighbors: parent $\rho\,(j)$ children $\zeta(j,c)$ and corresponding transform coefficient s_j.



(a) Visualization of quad-tree membership, indicated by color, of coefficients of an example band.



(b) Graphical model from the perspective of node j.

Next, we denote the set indexing the scaling coefficients after l transform steps[7] as

$$\mathbb{S}_l := \left\{ j \in [N] : x < 2^{L-l} \wedge y < 2^{L-l},\ (x, y) = \omega_R(j) \right\} \tag{2.64}$$

and the set of wavelet coefficients after l transform steps as

$$\mathbb{W}_l := \left\{ j \in [N] : x < 2^{L-l+1} \wedge y < 2^{L-l+1} \wedge j \notin \mathbb{S}_l,\ (x, y) = \omega_R(j) \right\} \tag{2.65}$$

Therefore, the indices of the roots of the quad-trees, Wavelet coefficients on the coarsest scale, are $\mathbb{W}_K$.

Using this notation, we can write the joint distribution of $P(\boldsymbol{s}, \boldsymbol{\delta})$ as follows:

$$\begin{aligned} P(\boldsymbol{s}, \boldsymbol{\delta}) &= P(\boldsymbol{s} \mid \boldsymbol{\delta}) P(\boldsymbol{\delta}) \\ &= \prod_{j=1}^{N} P(s_j \mid \delta_j) \prod_{i \notin \mathbb{S}_K \cup \mathbb{W}_K} P(\delta_i \mid \delta_{\rho(i)}) \prod_{k \in \mathbb{S}_K \cup \mathbb{W}_K} P(\delta_k) \end{aligned} \tag{2.66}$$

The distributions emitting the coefficients s_j are two-component mixtures:

$$P(s_j \mid \delta_j) = t_0(s_j)^{1-\delta_j} t_1(s_j)^{\delta_j} =: \tilde{t}(s_j \mid \delta_j) \tag{2.67}$$

In a slight abuse of notation, we will interpret $\tilde{t}(s \mid \delta)$ as a function with a relaxed domain that accepts as δ a value from the real interval[8] $[0, 1]$: $\tilde{t} : \mathbb{R} \times [0, 1] \mapsto \mathbb{R}$. Thus, as indicated in Figure 2.12b, the joint distribution decomposes into the following types of factors:

---

[7] If the transform continues beyond l steps, $\mathbb{S}_l$ indexes coefficients on the coarser scales.

[8] We will later need to evaluate $\tilde{t}(s \mid \mathrm{E}[\delta_j])$.

- Single node factors $P(\delta_k)$ with one parameter $P(\delta_k = 1)$. We use two parameters θ_s for $k \in \mathbb{S}_K$ and θ_r for $k \in \mathbb{W}_K$.
- Discrete two-node factors $P(\delta_i \mid \delta_{\rho(i)})$ with two parameters $P(\delta_i \mid \delta_{\rho(i)} = 0)$ and $P(\delta_i \mid \delta_{\rho(i)} = 1)$. We use the same parameters per level: $P(\delta_i \mid \delta_{\rho(i)} = d) = \theta_{d,l}, \forall i : \ell(i) = l$.
- Mixed two-node factors $P(s_j \mid \delta_j)$ parameterized by the parameters of t_0 and t_1. Again we tie them together over levels: $\boldsymbol{\theta}_{t,l} = \{\theta_{t_0,l}, \theta_{t_1,l}\}$.

We collect all hyper-parameters in $\boldsymbol{\theta} = \{\theta_s, \theta_r, \theta_{d,l}, \boldsymbol{\theta}_{t,l}\}$ for a total of parameters in $O(\log N)$. For convenience, we drop the dependence on $\boldsymbol{\theta}$ in $P(\boldsymbol{s}, \boldsymbol{\delta} \mid \boldsymbol{\theta})$, until we discuss learning them Section 2.2.2.4.

2.2.2.2 Algorithm of Seeger and Nickisch [2011a]

We begin by briefly introducing the scalable inference algorithm of Seeger and Nickisch [2011a] for sparse linear models with factorial super-Gaussian priors, on which our method is based. For the moment, we assume log-concave prior potentials to facilitate the introduction. At the end of this introduction, we will have described the tools to deal with non log-concave potentials and therefore defer the discussion.

The method described here, is a solver for the variational inference relaxation described in Section 1.2.3.2 [Jaakkola, 1997, Girolami, 2001, Palmer et al., 2006]. For SLMs, the bound is given by

$$\begin{aligned}
\log Z &= \log \int P(\boldsymbol{y} \mid \boldsymbol{u}) \prod_{j=1}^{N} t_j(s_j) \, \mathrm{d}\boldsymbol{u} && (2.68)\\
&= \log \int \mathcal{N}\left(\boldsymbol{y} \mid \boldsymbol{X}\boldsymbol{u}, \sigma^2 \boldsymbol{I}\right) \prod_{j=1}^{N} t_j(s_j) \, \mathrm{d}\boldsymbol{u} && (2.69)\\
&\geq \log \int \mathcal{N}\left(\boldsymbol{y} \mid \boldsymbol{X}\boldsymbol{u}, \sigma^2 \boldsymbol{I}\right) e^{-\frac{1}{2}\boldsymbol{s}^T \operatorname{diag}(\boldsymbol{\pi})\boldsymbol{s}} \, \mathrm{d}\boldsymbol{u} - \frac{1}{2}\sum_{j=1}^{N} h(\gamma_j) && (2.70)\\
&=: \mathcal{L}_{\mathrm{DSB}}(\boldsymbol{\gamma}) && (2.71)
\end{aligned}$$

where we used super-Gaussianity $t_j(f_j) \geq e^{-s_j^2/(2\gamma_j)} e^{-h_j(\gamma_j)/2}$ and defined $\boldsymbol{\pi} := \boldsymbol{\gamma}^{-1}$.

The approximate posterior is given by the likelihood and the potential-wise Gaussian approximations and therefore is itself Gaussian:

$$\begin{aligned}
Q(\boldsymbol{u} \mid \boldsymbol{y}) &= Z_Q^{-1} \mathcal{N}\left(\boldsymbol{y} \mid \boldsymbol{X}\boldsymbol{u}, \sigma^2 \boldsymbol{I}\right) e^{-\frac{1}{2}\boldsymbol{s}^T \operatorname{diag}(\boldsymbol{\pi})\boldsymbol{s}} && (2.72)\\
&= \mathcal{N}(\boldsymbol{u} \mid \boldsymbol{\xi}, \boldsymbol{\Xi}) && (2.73)
\end{aligned}$$

The posterior approximation has parameters that are functions of the variational parameters $\boldsymbol{\gamma}$:

$$\boldsymbol{\xi} = \sigma^{-2}\boldsymbol{\Xi}\boldsymbol{X}^T\boldsymbol{y} \tag{2.74}$$

$$\boldsymbol{A} := \boldsymbol{\Xi}^{-1} = \sigma^{-2}\boldsymbol{X}^T\boldsymbol{X} + \boldsymbol{W}^T\operatorname{diag}(\boldsymbol{\pi})\,\boldsymbol{W} \tag{2.75}$$

and normalization constant $Z_Q = \int \mathcal{N}\left(\boldsymbol{y} \mid \boldsymbol{X}\boldsymbol{u}, \sigma^2\boldsymbol{I}\right) e^{-\frac{1}{2}\boldsymbol{s}^T\operatorname{diag}(\boldsymbol{\pi})\boldsymbol{s}}\,\mathrm{d}\boldsymbol{u}$, i.e. the Gaussian integral in (2.71). The authors then use a variational characterization of Z_Q by using that mean and mode of a Gaussian coincide, so that $\max_{\boldsymbol{u}} Q(\boldsymbol{u} \mid \boldsymbol{y}) = Q(\boldsymbol{u} = \boldsymbol{\xi} \mid \boldsymbol{y}) = |2\pi\boldsymbol{\Xi}|^{-\frac{1}{2}}$:

$$\log Z_Q = \log Z_Q \tag{2.76}$$

$$= \log\left(Z_Q\,|2\pi\boldsymbol{\Xi}|^{\frac{1}{2}}\max_{\boldsymbol{u}} Q(\boldsymbol{u} \mid \boldsymbol{y})\right) \tag{2.77}$$

$$= \log\left(|2\pi\boldsymbol{\Xi}|^{\frac{1}{2}}\max_{\boldsymbol{u}} \mathcal{N}\left(\boldsymbol{y} \mid \boldsymbol{X}\boldsymbol{u}, \sigma^2\boldsymbol{I}\right) e^{-\frac{1}{2}\boldsymbol{s}^T\operatorname{diag}(\boldsymbol{\pi})\boldsymbol{s}}\right) \tag{2.78}$$

$$\doteq -\frac{1}{2}\left(\log|\boldsymbol{A}| + M\log\sigma^2 + \min_{\boldsymbol{u}}\sigma^{-2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{u}\|^2 + \boldsymbol{s}^T\operatorname{diag}(\boldsymbol{\pi})\,\boldsymbol{s}\right) \tag{2.79}$$

Plugging this expression into (2.71) yields the following objective to minimize with respect to $\boldsymbol{\gamma}$:

$$\phi(\boldsymbol{\gamma}) = \sum_{j=1}^{N} h(\gamma_j) + \log|\boldsymbol{A}| + \min_{\boldsymbol{u}}\sigma^{-2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{u}\|^2 + \boldsymbol{s}^T\operatorname{diag}(\boldsymbol{\pi})\,\boldsymbol{s} \tag{2.80}$$

$$= \min_{\boldsymbol{u}}\phi(\boldsymbol{\gamma}, \boldsymbol{u}) \tag{2.81}$$

For log-concave potentials, the authors show that $\phi(\boldsymbol{\gamma}, \boldsymbol{u})$ is jointly convex. This insight, however, does not lead to a scalable algorithm as computing gradients of $\log|\boldsymbol{A}|$ as well as evaluating (2.80) several times during line search costs $O(N^3)$: intractable for realistic image sizes. In order to overcome this issue, they employ a different strategy to optimize (2.80). Using another result, namely the concavity of $\boldsymbol{\pi} \mapsto \log|\boldsymbol{A}|$, they apply a fixed-point method inspired by the Convex Concave Procedure [CCCP, Yuille and Rangarajan, 2003], that replaces the concave part by the tight linear upper bound using its concave conjugate [Rockafellar, 1970]

$$\log|\boldsymbol{A}(\boldsymbol{\pi})| \leq \boldsymbol{z}^T\boldsymbol{\pi} - g^*(\boldsymbol{z}) \tag{2.82}$$

which is tight at $\boldsymbol{\pi}$ for the minimizer $\boldsymbol{z}^* = \operatorname{argmin}_{\boldsymbol{z}} \boldsymbol{z}^T\boldsymbol{\pi} - g^*(\boldsymbol{z})$ given by the tangent at that point, i.e. $\boldsymbol{z}^* = \nabla_{\boldsymbol{\pi}}\log|\boldsymbol{A}|$. Plugging this into (2.80) weakens the bound, but, given $\boldsymbol{z}$, simplifies it considerably as the objective decouples additively over the components of

$\boldsymbol{\gamma}$:

$$
\begin{aligned}
\phi(\boldsymbol{\gamma}, \boldsymbol{u}, \boldsymbol{z}) &= \sum_{j=1}^{N} h(\gamma_j) + \boldsymbol{z}^T \boldsymbol{\pi} - g^*(\boldsymbol{z}) + \sigma^{-2} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{u}\|^2 + \boldsymbol{s}^T \operatorname{diag}(\boldsymbol{\pi})\, \boldsymbol{s} && (2.83) \\
&= \sum_{j=1}^{N} \left(h(\gamma_j) + \frac{z_j + s_j^2}{\gamma_j} \right) - g^*(\boldsymbol{z}) + \sigma^{-2} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{u}\|^2 && (2.84) \\
&\geq \phi(\boldsymbol{\gamma}, \boldsymbol{u}) && (2.85) \\
&\geq \phi(\boldsymbol{\gamma}) && (2.86)
\end{aligned}
$$

CCCP prescribes to alternatingly solve for $\boldsymbol{z}$ and $(\boldsymbol{\gamma}, \boldsymbol{u})$, resulting in a method that provably converges to a stationary point, the gist of which is shown in Algorithm 1.

Algorithm 1 Double loop algorithm of Seeger and Nickisch [2011a]

```
repeat
    // Outer Loop
    $\boldsymbol{z} \leftarrow \operatorname{argmin}_{\boldsymbol{z}} \phi(\boldsymbol{z} \mid \boldsymbol{u}, \boldsymbol{\gamma})$
    // Inner Loop
    $\boldsymbol{u}, \boldsymbol{\gamma} \leftarrow \operatorname{argmin}_{\boldsymbol{u}, \boldsymbol{\gamma}} \phi(\boldsymbol{u}, \boldsymbol{\gamma} \mid \boldsymbol{z})$
until Convergence
```

Inner Loop Problem. Due to joint convexity, the order of minimization is interchangeable. This enables us to revert the super-Gaussian representation (1.23) back to the potential form recovering a problem corresponding to a MAP estimation smoothed by the current estimate of posterior variances:

$$
\begin{aligned}
&\min_{\boldsymbol{u}} \min_{\boldsymbol{\gamma}} \sum_{j=1}^{N} \left(h(\gamma_j) + \frac{z_j + s_j^2}{\gamma_j} \right) + \sigma^{-2} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{u}\|^2 && (2.87) \\
&= \min_{\boldsymbol{u}} \min_{\boldsymbol{\gamma}} -2 \sum_{j=1}^{N} \left(-\frac{1}{2} h(\gamma_j) - \frac{1}{2} \frac{z_j + s_j^2}{\gamma_j} \right) + \sigma^{-2} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{u}\|^2 && (2.88) \\
&= \min_{\boldsymbol{u}} -2 \sum_{j=1}^{N} \log t_j \left(\sqrt{z_j + s_j^2} \right) + \sigma^{-2} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{u}\|^2 && (2.89)
\end{aligned}
$$

This penalized-least-squares (PLS) problem can be solved by various first- and second order methods such as non-linear Conjugate Gradients (nCG), L-BFGS or IRLS.

Outer Loop Problem. Using standard results, we see that this operation requires the computation of marginal variances:

$$\boldsymbol{z}^*(\boldsymbol{\pi}) = \nabla_{\boldsymbol{\pi}} \log |\boldsymbol{A}(\boldsymbol{\pi})| = \operatorname{diag}\left(\boldsymbol{W}\Xi(\boldsymbol{\pi})\boldsymbol{W}^T\right) = \operatorname{Var}[\boldsymbol{s} \mid \boldsymbol{y}] \tag{2.90}$$

This is the costly part of the algorithm. Direct computation requires cubic cost to convert natural parameters into moments $\boldsymbol{A} \mapsto \Xi$. Seeger and Nickisch [2011a] use a low-rank approximation of Ξ by a few of its leading eigenvectors, obtained iteratively using the Lanczos method, that requires a matrix-vector multiplication with $\boldsymbol{A}$ per round. Alternatively, Papandreou and Yuille [2010] describe a simple scheme to draw exact posterior samples in Gaussian linear models that can be used to estimate $\operatorname{Var}[\boldsymbol{u} \mid \boldsymbol{y}]$. The method is sketched in Algorithm 2.

Algorithm 2 Perturb and MAP procedure [Papandreou and Yuille, 2010] for marginal variances

```
Require: $\sigma^2, \boldsymbol{X}, \boldsymbol{W}, \boldsymbol{\pi}, N_{\text{Samples}}$
Ensure: Approximate marginal variances $\boldsymbol{z} := \operatorname{Var}_{Q(\boldsymbol{u})}[\boldsymbol{s}]$
$\boldsymbol{z} \leftarrow 0$
for $k = 1, \ldots, N_{\text{Samples}}$ do
    $\boldsymbol{a} \sim \mathcal{N}(0, \boldsymbol{I})$
    $\boldsymbol{b} \sim \mathcal{N}(0, \boldsymbol{I})$
    $\tilde{\boldsymbol{a}} \leftarrow \sigma^{-1}\boldsymbol{X}^T\boldsymbol{a} \Rightarrow \tilde{\boldsymbol{a}} \sim \mathcal{N}\left(0, \sigma^{-2}\boldsymbol{X}^T\boldsymbol{X}\right)$
    $\tilde{\boldsymbol{b}} \leftarrow \boldsymbol{W}^T\left(\boldsymbol{\pi}^{\frac{1}{2}} \circ \boldsymbol{b}\right) \Rightarrow \tilde{\boldsymbol{b}} \sim \mathcal{N}\left(0, \boldsymbol{W}^T \operatorname{diag}(\boldsymbol{\pi})\boldsymbol{W}\right)$
    $\boldsymbol{c} \leftarrow \boldsymbol{A}^{-1}\left(\tilde{\boldsymbol{a}} + \tilde{\boldsymbol{b}}\right) \Rightarrow \boldsymbol{c} \sim \mathcal{N}(0, \Xi)$
    $\boldsymbol{z} \leftarrow \boldsymbol{z} + \boldsymbol{c}^2$
end for
return $\frac{1}{N_{\text{Samples}}}\boldsymbol{z}$
```

Systems with $\boldsymbol{A}$ can be solved iteratively using (linear) Conjugate Gradients (CG). Therefore, both, the low-rank approximation, as well as the sampling approximation, can be reduced to multiplication with $\boldsymbol{A}$ that is typically highly structured. For example, in the imaging applications, we consider here, a multiplication can be performed in linear time.

Non Log-concave Potentials. Super-Gaussian potentials, that are not log-concave, compromise the convexity properties of the algorithm, i.e. the potential-specific term $h(\gamma_j)$ is not convex. To overcome this issue, Seeger and Nickisch [2011a] propose to decompose $h(\gamma_j)$ into a sum of a convex and concave function[9]. The extra term for the concave

---

[9] This is possible for functions with bounded second derivative, which can be seen by adding and subtracting a convex function with a second derivative that dominates the original one [Yuille and

part is dealt with in the same way as with $\log|\boldsymbol{A}|$, by upper-bounding and an additional tightening step in the outer loop. The convex part is used in the inner loop, as before.

2.2.2.3 Inference Algorithm for Structured Scale Mixtures

Extending the method described in Section 2.2.2.2 to the structured prior (2.66) is straightforward, using the following observation about the conditionals in (2.67):

$$\begin{aligned}
\log \tilde{t}(s_j \mid \delta_j) &= (1-\delta_j)\log t_0(s_j) + \delta_j \log t_1(s_j) && (2.91)\\
&\geq (1-\delta_j)\left(-\frac{s_j^2}{2\gamma_{0,j}} - \frac{1}{2}h_0(\gamma_{0,j})\right) + \delta_j\left(-\frac{s_j^2}{2\gamma_{1,j}} - \frac{1}{2}h_1(\gamma_{1,j})\right) && (2.92)\\
&= -\frac{1}{2}s_j^2\tilde{\pi}_j - \frac{1}{2}\tilde{h}_j && (2.93)
\end{aligned}$$

where $\tilde{\pi}_j = (1-\delta_j)\gamma_{0,j}^{-1} + \delta_j\gamma_{1,j}^{-1}$ and $\tilde{h}_j = (1-\delta_j)h_0(\gamma_{0,j}) + \delta_j h_1(\gamma_{1,j})$. We see that the effective potential inherits log-concavity from its constituents and is log-linear in δ_j.

Next, we derive the bound on the log partition function in two steps. First, we use (2.91) as in (2.71) and second, we use a standard variational bound similar to (1.22) introducing the factorizing variational distribution $Q(\boldsymbol{u},\boldsymbol{\delta}) = Q(\boldsymbol{u})Q(\boldsymbol{\delta})$, where $Q(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u} \mid \boldsymbol{\xi}, \boldsymbol{\Xi})$ and $Q(\boldsymbol{\delta})$ will turn out to be a discrete ditribution following the same graphical structure as the prior $P(\boldsymbol{\delta})$. In the following derivations, we will denote by $\langle\cdot\rangle$ the expectation $\mathrm{E}_{Q(\boldsymbol{\delta})}[\cdot]$.

$$\begin{aligned}
\log Z &= \log \sum_{\boldsymbol{\delta}} \int P(\boldsymbol{y} \mid \boldsymbol{u}) P(\boldsymbol{u} \mid \boldsymbol{\delta}) P(\boldsymbol{\delta})\,\mathrm{d}\boldsymbol{u} && (2.94)\\
&= \log \sum_{\boldsymbol{\delta}} \int P(\boldsymbol{y} \mid \boldsymbol{u}) \prod_{j=1}^{N} P(s_j \mid \delta_j) P(\boldsymbol{\delta})\,\mathrm{d}\boldsymbol{u} && (2.95)\\
&\geq \log \sum_{\boldsymbol{\delta}} \int \mathcal{N}\left(\boldsymbol{y} \mid \boldsymbol{X}\boldsymbol{u}, \sigma^2\boldsymbol{I}\right) e^{-\frac{1}{2}\boldsymbol{s}^T \operatorname{diag}(\tilde{\boldsymbol{\pi}})\boldsymbol{s}} P(\boldsymbol{\delta}) e^{-\frac{1}{2}\sum_{j=1}^{N}\tilde{h}_j}\,\mathrm{d}\boldsymbol{u} && (2.96)\\
&\geq \mathrm{E}_Q\left[\log \frac{\mathcal{N}\left(\boldsymbol{y} \mid \boldsymbol{X}\boldsymbol{u}, \sigma^2\boldsymbol{I}\right) e^{-\frac{1}{2}\boldsymbol{s}^T \operatorname{diag}(\tilde{\boldsymbol{\pi}})\boldsymbol{s}} P(\boldsymbol{\delta})}{Q(\boldsymbol{u})Q(\boldsymbol{\delta})}\right] - \frac{1}{2}\sum_{j=1}^{N}\left\langle \tilde{h}_j \right\rangle && (2.97)
\end{aligned}$$

Thus, we need to maximize the bound with respect to three different objects: the variational parameters $\tilde{\boldsymbol{\gamma}} := \{(\gamma_{0,j}, \gamma_{1,j})\}_{j\in[N]}$ and the variational distributions $Q(\boldsymbol{u})$ and $Q(\boldsymbol{\delta})$. Using an alternating structured mean-field algorithm and re-arranging the computations, we will recover an algorithm which is very similar to the one described in Section 2.2.2.2, with an additional step for optimizing with respect to $Q(\boldsymbol{\delta})$ folded in. Crucially, we exploit that the coupling between $\boldsymbol{u}$ and $\boldsymbol{\delta}$ is weak through the quadratic we obtained from the potential bounds. We begin with the update of $\tilde{\boldsymbol{\gamma}}$ and $Q(\boldsymbol{u})$ and

---

Rangarajan, 2003].

determine the statistics we need from an update of $Q(\boldsymbol{\delta})$.

Update of $\tilde{\gamma}$ and $Q(\boldsymbol{u})$. Keeping the other variational parameters fixed, we can read off the optimal $Q(\boldsymbol{u})$ by gathering all terms in the numerator depending on $\boldsymbol{u}$, taking the geometric mean with respect to $Q(\boldsymbol{\delta})$ and normalizing:

$$Q(\boldsymbol{u}) = Z_Q^{-1} \mathcal{N}\left(\boldsymbol{y} \mid \boldsymbol{X}\boldsymbol{u}, \sigma^2 \boldsymbol{I}\right) e^{-\frac{1}{2}\boldsymbol{s}^T \operatorname{diag}(\langle\tilde{\boldsymbol{\pi}}\rangle)\boldsymbol{s}} \tag{2.98}$$

From the definition of $\tilde{\pi}_j$, we see that $\langle\tilde{\pi}_j\rangle = (1 - \langle\delta_j\rangle)\gamma_{0,j}^{-1} + \langle\delta_j\rangle\,\gamma_{1,j}^{-1}$. Thus, the statistics we require from the update of $Q(\boldsymbol{\delta})$ are posterior marginal probabilities, since $\langle\delta_j\rangle = Q(\delta_j = 1)$.

Plugging (2.98) back into the bound (2.94), we obtain

$$\log Z \geq \log Z_Q(\langle\tilde{\boldsymbol{\pi}}\rangle) - \frac{1}{2}\sum_{j=1}^{N}\left\langle \tilde{h}_j \right\rangle - D_{\mathrm{KL}}\left[Q(\boldsymbol{\delta}) \| P(\boldsymbol{\delta})\right]. \tag{2.99}$$

Since the KL-term does not depend on $\tilde{\boldsymbol{\gamma}}$, we recover (2.79). Using the linear bound on the log-determinant for the argument $\langle\tilde{\boldsymbol{\pi}}\rangle$, we obtain the inner loop objective of (2.83).

$$\begin{aligned}\phi(\tilde{\gamma}, \boldsymbol{u}, \boldsymbol{z} \mid Q(\boldsymbol{\delta})) &= \sum_{j=1}^{N}\left(\tilde{h}_j\right) + \boldsymbol{z}^T\langle\tilde{\boldsymbol{\pi}}\rangle - g^*(\boldsymbol{z}) + \sigma^{-2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{u}\|^2 + \boldsymbol{s}^T\operatorname{diag}\left(\langle\tilde{\boldsymbol{\pi}}\rangle\right)\boldsymbol{s}\\ &= \sum_{j=1}^{N}\left(\left\langle\tilde{h}_j\right\rangle + \left(z_j + s_j^2\right)\langle\tilde{\pi}_j\rangle\right) - g^*(\boldsymbol{z}) + \sigma^{-2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{u}\|^2\end{aligned} \tag{2.100}$$

To finally recover the analog of (2.87), we show that the super-Gaussian representation can be reverted individually due to the additive structure of the log-potential. First, we note that the objective remains decoupled for individual coefficients j.

Thus, for a single j, we need to solve

$$\max_{\tilde{\gamma}_j} -\frac{1}{2}\left\langle \tilde{h}_j \right\rangle - \frac{1}{2}\left(z_j + s_j^2\right)\langle \tilde{\pi}_j \rangle \tag{2.101}$$

$$= \max_{\gamma_{0,j}} \max_{\gamma_{1,j}} -\frac{1}{2}\left\langle \tilde{h}_j \right\rangle - \frac{1}{2}\left(z_j + s_j^2\right)\langle \tilde{\pi}_j \rangle \tag{2.102}$$

$$=(1 - \langle \delta_j \rangle) \max_{\gamma_{0,j}} -\frac{1}{2}\left(h_0(\gamma_{0,j}) + \frac{z_j + s_j^2}{\gamma_{0,j}}\right) \tag{2.103}$$

$$+ \langle \delta_j \rangle \max_{\gamma_{1,j}} -\frac{1}{2}\left(h_1(\gamma_{1,j}) + \frac{z_j + s_j^2}{\gamma_{1,j}}\right) \tag{2.104}$$

$$=(1 - \langle \delta_j \rangle) \log t_0(\sqrt{z_j + s_j^2}) + \langle \delta_j \rangle \log t_1(\sqrt{z_j + s_j^2}) \tag{2.105}$$

$$= \log \tilde{t}\left(\sqrt{z_j + s_j^2} \mid \langle \delta_j \rangle\right) \tag{2.106}$$

We recover the penalized least squares inner loop in (2.87) with a not only smoothed but also weighted regularizer, where the uncertainty in the state of the indicator δ_j controls the regularization strength.

Finally, we explicitly recompute $\tilde{\boldsymbol{\gamma}}$ using (2.59)

$$\gamma_{d,j} = -\left(2\partial_x \log t_d\left(\sqrt{z_j + s_j^2}\right)\right)^{-1} \tag{2.107}$$

Update of $Q(\boldsymbol{\delta})$. We can read off the form of $Q(\boldsymbol{\delta})$ in the same way as for $Q(\boldsymbol{u})$ by gathering all terms depending on $\boldsymbol{\delta}$ and taking the geometric mean of the term depending on $\boldsymbol{u}$ with respect to $Q(\boldsymbol{u})$. From (2.94) we obtain

$$Q(\boldsymbol{\delta}) \propto \exp\left(-\frac{1}{2}\mathrm{E}_{Q(\boldsymbol{u})}\left[\boldsymbol{s}^T \operatorname{diag}(\tilde{\boldsymbol{\pi}})\, \boldsymbol{s}\right] - \frac{1}{2}\sum_{j=1}^{N} \tilde{h}_j\right) P(\boldsymbol{\delta}) \tag{2.108}$$

$$= \exp\left(-\frac{1}{2}\mathrm{E}_{Q(\boldsymbol{u})}\left[\operatorname{tr}\left(\boldsymbol{s}\boldsymbol{s}^T \operatorname{diag}(\tilde{\boldsymbol{\pi}})\right)\right] - \frac{1}{2}\sum_{j=1}^{N} \tilde{h}_j\right) P(\boldsymbol{\delta}) \tag{2.109}$$

$$= \exp\left(-\frac{1}{2}\operatorname{tr}\left(\mathrm{E}_{Q(\boldsymbol{u})}\left[\boldsymbol{s}\boldsymbol{s}^T\right] \operatorname{diag}(\tilde{\boldsymbol{\pi}})\right) - \frac{1}{2}\sum_{j=1}^{N} \tilde{h}_j\right) P(\boldsymbol{\delta}) \tag{2.110}$$

$$= \exp\left(-\frac{1}{2}\operatorname{tr}\left(\mathrm{Cov}_{Q(\boldsymbol{u})}\left[\boldsymbol{s}\right] \operatorname{diag}(\tilde{\boldsymbol{\pi}})\right) - \frac{1}{2}\sum_{j=1}^{N} \tilde{h}_j\right) P(\boldsymbol{\delta}) \tag{2.111}$$

$$= \exp\left(-\frac{1}{2}\sum_{j=1}^{N} \left(\mathrm{Var}_{Q(\boldsymbol{s})}\left[s_j\right] + \mathrm{E}_{Q(\boldsymbol{s})}\left[s_j\right]\right)\tilde{\pi}_j + \tilde{h}_j\right) P(\boldsymbol{\delta}) \tag{2.112}$$

where we have used $\boldsymbol{s} = \boldsymbol{W}\boldsymbol{u}$. Tightening the potential bounds by setting $\tilde{\boldsymbol{\gamma}} = \left\{ \left(\partial_{x_j} \log t_0(s_j), \partial_{x_j} \log t_1(s_j) \right) \right\}_{j \in [N]}$, where $x_j := s_j^2$ and the derivatives are evaluated at $x_j = z_j + s_j^2$, we can revert the convex conjugation and plug in the current estimates of the marginal moments of $Q(\boldsymbol{u})$. Hence, we see that the prior tree structure is preserved as only emission probabilities are changed to reflect our current estimate of $Q(\boldsymbol{u})$ and $\tilde{\boldsymbol{\gamma}}$:

$$
\begin{aligned}
Q(\boldsymbol{\delta}) &\propto \exp\left(-\frac{1}{2} \sum_{j=1}^{N} \left(\mathrm{Var}_{Q(\boldsymbol{s})}\left[s_j\right] + \mathrm{E}_{Q(\boldsymbol{s})}\left[s_j\right] \right) \tilde{\pi}_j + \tilde{h}_j \right) P(\boldsymbol{\delta}) && (2.113) \\
&= \exp\left(\sum_{j=1}^{N} \log \tilde{t}\left(\sqrt{z_j + s_j^2} \right) \right) P(\boldsymbol{\delta}) && (2.114) \\
&= \prod_{j=1}^{N} \tilde{t}\left(\sqrt{z_j + s_j^2} \mid \delta_j \right) P(\boldsymbol{\delta}) && (2.115)
\end{aligned}
$$

Therefore, posterior marginals of $Q(\boldsymbol{\delta})$ can be obtained in time linear in N using sum-product belief propagation [Pearl, 1988].

Algorithm 3 Approximate inference algorithm for Tree-structured scale mixtures

```
Require: $\boldsymbol{y}, \boldsymbol{X}$
Ensure: Updated parameters $\boldsymbol{\theta}$
// Initialization
repeat
    // Outer Loop update
    Estimate $\boldsymbol{z}$ using $\langle \tilde{\boldsymbol{\pi}} \rangle$ using Algorithm 2
    // Inner Loop
    for $k$ do
        // Posterior mean
        $\boldsymbol{u} \leftarrow \mathrm{argmin}_{\boldsymbol{u}}\, \sigma^{-2} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{u}\|^2 - 2 \sum_{j=1}^{N} \log \tilde{t}\left( \sqrt{z_j + s_j^2} \mid \langle \delta_j \rangle \right)$
        $\boldsymbol{s} \leftarrow \boldsymbol{W}\boldsymbol{u}$
        // Update node potentials
        $Q(\boldsymbol{\delta}) \propto \prod_{j=1}^{N} \tilde{t}\left( \sqrt{z_j + s_j^2} \mid \delta_j \right) P(\boldsymbol{\delta})$
        // Compute marginals
        $\langle \boldsymbol{\delta} \rangle \leftarrow$ sumProduct($Q(\boldsymbol{\delta})$)
    end for
    Recompute $\tilde{\boldsymbol{\gamma}}$ using Eq. 2.107 with $\boldsymbol{s}$, $\boldsymbol{z}$
    Recompute $\langle \tilde{\boldsymbol{\pi}} \rangle$ using $\tilde{\boldsymbol{\gamma}}$, $\langle \boldsymbol{\delta} \rangle$
until Convergence
```

Final Algorithm. As we have seen, the techniques of Seeger and Nickisch [2011a] immediately apply with only minor modifications. Following their recipe for scalability, we maintain their double loop structure and modify the inner loop to iterate between updates of $(Q(\boldsymbol{u}), \tilde{\gamma})$ and $Q(\boldsymbol{\delta})$ before refitting the log-determinant bound in the outer loop. Algorithm 3 summarizes our algorithm.

Algorithmic Complexity. The dominating computations of Algorithm 3 are

- In the worst case $O(N_{\text{Samples}} N)$ multiplications with $\boldsymbol{A}$ in Algorithm 2:

$$\boldsymbol{A} = \sigma^{-2} \boldsymbol{X}^T \boldsymbol{X} + \boldsymbol{W}^T \operatorname{diag}\left(\langle \tilde{\boldsymbol{\pi}} \rangle\right) \boldsymbol{W} \tag{2.116}$$

- Evaluating gradients of the PLS problem:

$$\frac{1}{2} \nabla_{\boldsymbol{u}} = \sigma^{-2} \left(\boldsymbol{X}^T \boldsymbol{X} \boldsymbol{u} - \boldsymbol{X}^T \boldsymbol{y} \right) - \boldsymbol{W}^T \left[\frac{\tilde{t}' \left(\sqrt{z_j + s_j^2} \mid \langle \delta_j \rangle \right)}{\tilde{t} \left(\sqrt{z_j + s_j^2} \mid \langle \delta_j \rangle \right)} \right]_{j \in [N]} \tag{2.117}$$

Thus, the complexity hinges on the concrete $\boldsymbol{W}$ and $\boldsymbol{X}$. The DWT $\boldsymbol{W}$ and its transpose cost $O(N)$. In denoising and inpainting applications, which we consider in our experiments, $\boldsymbol{X}$ is the (row-deficient) identity matrix, therefore also applicable in linear time.

Algorithmic Variants. There are two orthogonal dimensions along which we can modify our method: MAP estimation versus posterior inference for $\boldsymbol{u}$ and structured versus unstructured prior. Algorithm 3 subsumes all these special cases. As described in Seeger and Nickisch [2011a], the discerning factor between MAP and posterior inference is the outer loop. Disabling it and setting $\boldsymbol{z} = 0$ results in MAP estimation of $\boldsymbol{u}$. Enabling inference over $\boldsymbol{\delta}$ thus amounts to an EM algorithm for estimating $\boldsymbol{u}$ by maximizing the marginal posterior $Q(\boldsymbol{u} \mid \boldsymbol{y}) = \sum_{\boldsymbol{\delta}} Q(\boldsymbol{u}, \boldsymbol{\delta} \mid \boldsymbol{y})$. Fixing $\boldsymbol{\delta} = 1$ and disabling the update of $Q(\boldsymbol{\delta})$ recovers inference and MAP estimation for factorial priors.

2.2.2.4 Parameter Learning.

The prior hyper-parameters reside in the term $P(\boldsymbol{s}, \boldsymbol{\delta} \mid \boldsymbol{\theta})$ (2.66) and consist of the transition probabilities of the HMT and the parameters of the emission probabilities, which we introduced as $\boldsymbol{\theta} = \{\theta_s, \theta_r, \theta_{d,l}, \boldsymbol{\theta}_{t,l}\}$. To compute an M-Step, the relevant part of the bound on $\log Z(\boldsymbol{\theta})$ is given by

$$\begin{aligned} \mathrm{E}_{Q(\boldsymbol{u})Q(\boldsymbol{\delta})} \left[\log P(\boldsymbol{s}, \boldsymbol{\delta}) \right] &= \mathrm{E}_{Q(\boldsymbol{u})Q(\boldsymbol{\delta})} \left[\log P(\boldsymbol{s} \mid \boldsymbol{\delta}) P(\boldsymbol{\delta}) \right] && (2.118) \\ &= \sum_{j=1}^{N} \log \tilde{t} \left(\sqrt{z_j + s_j^2} \mid \langle \delta_j \rangle \right) + \langle \log P(\boldsymbol{\delta}) \rangle && (2.119) \\ & && (2.120) \end{aligned}$$

The first term of (2.119) depends on the $\boldsymbol{\theta}_{t,l}$ and the second part on the rest of $\boldsymbol{\theta}$. We denote the objective to be minimized by

$$\phi(\boldsymbol{\theta}) = -\mathrm{E}_{Q(\boldsymbol{u})Q(\boldsymbol{\delta})}\left[\log P(\boldsymbol{s}, \boldsymbol{\delta} \mid \boldsymbol{\theta})\right] \tag{2.121}$$

Update of Tree Parameters θ_s, θ_r and $\boldsymbol{\theta}_{d,l}$. Using the tree-structured factorization (2.66) and grouping terms on the same level, we obtain

$$\begin{aligned}
\phi(\{\theta_s, \theta_r, \boldsymbol{\theta}_{d,l}\}) &\doteq \langle \log P(\boldsymbol{\delta}) \rangle && (2.122)\\
&= \sum_{j \in \mathbb{W}_K} \left(Q(\delta_j = 1) \log \theta_r + Q(\delta_j = 0) \log(1 - \theta_r) \right) && (2.123)\\
&+ \sum_{j \in \mathbb{S}_K} \left(Q(\delta_j = 1) \log \theta_s + Q(\delta_j = 0) \log(1 - \theta_s) \right) && (2.124)\\
&+ \sum_{l=1}^{K-1} \sum_{d \in \{0,1\}} \sum_{j \in \mathbb{W}_l} \big[Q(\delta_j = 1, \delta_{\rho(j)} = d) \log \theta_{d,l} \\
&\qquad\qquad + Q(\delta_j = 0, \delta_{\rho(j)} = d) \log(1 - \theta_{d,l}) \big] && (2.125)
\end{aligned}$$

Let $q_d(\mathbb{O}) := \sum_{j \in \mathbb{O}} Q(\delta_j = d)$. Then, setting the derivatives with respect to each of the parameters to zero, we obtain update equations

$$\theta_s \leftarrow \frac{q_1(\mathbb{S}_K)}{q_1(\mathbb{S}_K) + q_0(\mathbb{S}_K)} \tag{2.126}$$

$$\theta_r \leftarrow \frac{q_1(\mathbb{W}_K)}{q_1(\mathbb{W}_K) + q_0(\mathbb{W}_K)} \tag{2.127}$$

$$\theta_{d,l} \leftarrow \frac{q_{d,l,1}}{q_{d,l,1} + q_{d,l,0}} \tag{2.128}$$

with

$$q_{d,l,1} = \sum_{j \in \mathbb{W}_l} Q(\delta_j = 1, \delta_{\rho(j)} = d) \qquad q_{d,l,0} = \sum_{j \in \mathbb{W}_l} Q(\delta_j = 0, \delta_{\rho(j)} = d) \tag{2.129}$$

Update of Emission Parameters. Again, we partition the sum to group together terms belonging to a certain level

$$\phi(\{\boldsymbol{\theta}_{t,l}\}) \doteq \sum_{j=1}^{N} \log \tilde{t}\left(\sqrt{z_j + s_j^2} \mid \langle \delta_j \rangle\right) \tag{2.130}$$

$$= \sum_{j \in \mathbb{S}_K} \log \tilde{t}\left(\sqrt{z_j + s_j^2} \mid \langle \delta_j \rangle\right) \tag{2.131}$$

$$+ \sum_{l=1}^{K} \sum_{j \in \mathbb{W}_l} \log \tilde{t}\left(\sqrt{z_j + s_j^2} \mid \langle \delta_j \rangle\right) \tag{2.132}$$

which decomposes additively into terms

$$\phi(\theta_{t_d,-1}) \doteq \sum_{j \in \mathbb{S}_K} Q(\delta_j = d) \log t_d(\sqrt{z_j + s_j^2} \mid \theta_{t_d,-1}) \tag{2.133}$$

$$\phi(\theta_{t_d,l}) \doteq \sum_{j \in \mathbb{W}_l} Q(\delta_j = d) \log t_d(\sqrt{z_j + s_j^2} \mid \theta_{t_d,l}) \tag{2.134}$$

where we denoted the level of the scaling coefficients as $l = -1$. The concrete update depends on the potentials used. E.g. for Laplace potentials, $\log t_d(s) \propto \log \tau_d - \tau_d \, |s|$, we obtain

$$\tau_{d,l} \leftarrow \frac{q_d(\mathbb{W}_l)}{\sum_{j \in \mathbb{W}_l} Q(\delta_j = d) \sqrt{z_j + s_j^2}} \tag{2.135}$$

For Gaussian potentials $\log t_d(s) \propto \log \tau_d - \tau_d s^2$, we obtain essentially the same update:

$$\tau_{d,l} \leftarrow \frac{q_d(\mathbb{W}_l)}{\sum_{j \in \mathbb{W}_l} Q(\delta_j = d) \left(z_j + s_j^2\right)} \tag{2.136}$$

For potentials that are not log-concave, e.g. Student's T, the update needs to be performed at the end of the outer loop, after tightening the bound on the convex-concave decomposition of the potential. Here, we do not have a convenient closed-form update, but need to perform iterative optimization in one-dimension.

2.2.3 Experiments

We present experiments on a range of denoising and inpainting problems, comparing variational inference and MAP estimation for different models. Inpainting problems can be considered to be more challenging, due to severe loss of information. We use peak signal-to-noise ratio (PSNR), a standard measure in the field, to assess performance.

Denoting $\boldsymbol{u}$ as the latent image and $\hat{\boldsymbol{u}}$ its estimate, PSNR is defined as

$$\mathrm{PSNR}(\boldsymbol{u}, \hat{\boldsymbol{u}}) = 10 \cdot \log \frac{(\max(\boldsymbol{u}))^2}{\frac{1}{N}\|\boldsymbol{u} - \hat{\boldsymbol{u}}\|^2} \tag{2.137}$$

Since the intensities of the images we used were already normalized to $[0, 1]$, the maxiumum pixel intensity $\max(\boldsymbol{u}) = 1$, and we measure the negative log of the mean squared reconstruction error. Therefore, higher values are better.

We use a dataset of 77 images from Seeger and Nickisch [2008][10]. All images are greyscale of size 256×256, thus $N = 65536$. Our implementation is based on the `glm-ie` toolbox [11], that implements the double loop algorithm of Seeger and Nickisch [2011a]. We compare eight methods: MAP estimation (MAP) vs. variational inference (VB), factorizing prior (fact) vs. latent tree scale mixture prior (tree), and Laplacian (Lap) vs. Student's t potentials (T). The *Lap-tree* model uses two Laplace potentials $t_{0j}(s_j)$, $t_{1j}(s_j)$ (see Table 2.5) with different hyperparameters $\tau_{0,l}$, $\tau_{1,l}$ per level. The *T-tree* model employs Gaussian $\mathcal{N}(s_j|0, \tau_{0,l}^{-1})$ for the low, Student's t potentials for the high state (Table 2.5), with a pair of hyperparameters $(\tau_{0,l}, \tau_{1,l})$ at each level. We use $2L$ hyperparameters in the *tree*, L (namely, $\{\tau_l\}$) in the *fact* setups. The Student's T shape parameter ν is fixed to 2.1. For each run, we initialize hyperparameters $\boldsymbol{\theta}$ as in Crouse et al. [1998], by maximizing the prior probability of the raw[12] data $\boldsymbol{y}$ (for the *tree* cases, this involves a few steps of expectation maximization), then optimize them by maximizing our approximation to $\log Z$. We update hyperparameters once per outer loop iteration.

We run VB with up to 15 outer loop (OL) iterations. In the outer loop, we run Algorithm 2 for estimating $\boldsymbol{z}$, required for inpainting only, with $N_{\text{Samples}} = 30$ and 70 conjugate gradients (CG) iterations. For denoising ($\boldsymbol{X} = \boldsymbol{I}$), $\boldsymbol{z}$ can be computed exactly in linear time (see below). We did three belief-propagation and PLS calls per outer loop for *tree* setups, PLS ran up to 150 iterations of nonlinear CG. Each iteration of CG requires two matrix-vector multiplications with $\boldsymbol{W}$ and $\boldsymbol{X}$. We did not exhaustively optimize these algorithmic tuning parameters.

2.2.3.1 Denoising

We add Gaussian random noise of variance $\sigma^2 = 0.01$ to each image (with pixel values $u_i \in [0, 1]$). All methods use the correct value of σ^2 in their likelihood. Notice that in this case, Gaussian variances $\boldsymbol{z}$ can be computed exactly at no cost: with $\boldsymbol{A}(\boldsymbol{\pi}) = \sigma^{-2}\boldsymbol{I} + \boldsymbol{W}^T \operatorname{diag}(\boldsymbol{\pi})\, \boldsymbol{W}$ and $\boldsymbol{W}^T\boldsymbol{W} = \boldsymbol{W}\boldsymbol{W}^T = \boldsymbol{I}$ we obtain, using the Woodbury

---

[10] Thanks to H. Nickisch for kindly providing the data consisting of 75 images. We added two images.

[11] www.mloss.org/software/view/269/

[12] For inpainting, the missing pixels are initialized to mean(y_i).

identity:

$$\boldsymbol{z} = \operatorname{diag}\left(\boldsymbol{W}\boldsymbol{A}(\boldsymbol{\pi})^{-1}\boldsymbol{W}^T\right) = (\sigma^{-2}\mathbf{1} + \boldsymbol{\pi})^{-1}.$$

Therefore, approximating variances, the dominating cost for VB in general, is not required: we can optimize the bound in linear time. Results are shown in Table 2.6. For this application, differences between MAP and VB reconstruction are not significant. On the other hand, the structured prior improves PSNR somewhat. Hyperparameter learning improves VB performance, especially when Student's t potentials are used. In contrast, we observe that it does not help[13] (and can even hurt) MAP performance.

	VB		MAP	
MODEL	INIT $\boldsymbol{\theta}$	LEARNED $\boldsymbol{\theta}$	INIT $\boldsymbol{\theta}$	LEARNED $\boldsymbol{\theta}$
LAP-FACT	24.5 ± 0.7	24.7 ± 0.9	24.5 ± 1.4	23.2 ± 1.6
T-FACT	20.8 ± 0.0	23.3 ± 0.5	24.9 ± 1.1	24.7 ± 1.6
LAP-TREE	24.3 ± 0.7	25.0 ± 1.0	25.1 ± 1.3	25.0 ± 1.6
T-TREE	21.1 ± 0.1	$\mathbf{25.2 \pm 1.3}$	24.3 ± 1.0	24.3 ± 1.6

Table 2.6 – Denoising experiments ($\sigma^2 = 0.01$). Shown is PSNR w.r.t. noise-free ground truth (mean and std.dev. over 77 images).

2.2.3.2 Inpainting

We remove 75% of pixels at random, using the same mask $\mathbb{O} \subset \{1, \dots, N\}$ for all images. The design matrix is $\boldsymbol{X} = \boldsymbol{I}_{\mathbb{O}}$, the noise variance was fixed to $\sigma^2 = 10^{-5}$. Results are shown in Table 2.7. As PSNR does not always correlate well with visual quality, we also show example images in Figure 2.13 and Figure 2.14.

	VB		MAP	
MODEL	INIT $\boldsymbol{\theta}$	LEARNED $\boldsymbol{\theta}$	INIT $\boldsymbol{\theta}$	LEARNED $\boldsymbol{\theta}$
LAP-FACT	23.0 ± 2.1	23.3 ± 2.1	15.1 ± 3.8	20.0 ± 2.3
T-FACT	21.0 ± 1.7	20.1 ± 1.7	20.6 ± 2.2	20.0 ± 2.2
LAP-TREE	22.2 ± 2.0	23.5 ± 2.2	19.8 ± 2.5	19.7 ± 2.6
T-TREE	21.2 ± 1.8	$\mathbf{23.6 \pm 2.2}$	19.0 ± 2.3	19.8 ± 2.5

Table 2.7 – Inpainting experiments (75% pixels removed). Shown is PSNR w.r.t. ground truth (mean and std.dev. over 77 images).

In the inpainting scenario, VB posterior mean predictions appear to be superior to MAP

---

[13] There is no justification for maximizing the posterior w.r.t. $\boldsymbol{\theta}$. We include these results only for the fact that "alternating MAP" learning is frequently done in practice.

reconstruction, and VB with non-factorial latent tree prior performs best. While VB with a factorial Laplace prior (Lap-fact) shows similar PSNR values to VB-Lap-tree, the visual appearance of results seem to suffer from less artefacts. The additional runtime compared to MAP estimation, mainly due to the estimation of variances $\boldsymbol{z}$, pays off for these problems.

2.2.4 Discussion

In this part, we extended the approximate inference algorithm of Seeger and Nickisch [2011a] to structured scale-mixture priors for images in the Wavelet domain, resulting in a scalable inference algorithm representing full posterior covariances at an acceptable additional cost over MAP. We experimentally corroborated the robustness of the posterior mean estimator over the posterior mode in highly ill-posed inpainting problems.

The experimental evaluation can be extended along three axes: additional benchmark applications, such as compressed sensing, different combinations of potentials, such as spike and slab and other inference approximations, such as VG and EP. Expectation Propagation might work well in this context as indicated by the results of Som and Schniter [2012], but might be harder to run reliably at this scale [Seeger and Nickisch, 2011a]. Scaling could also be an issue for applying the Variational Gaussian method except for a fully factorized posterior approximation.

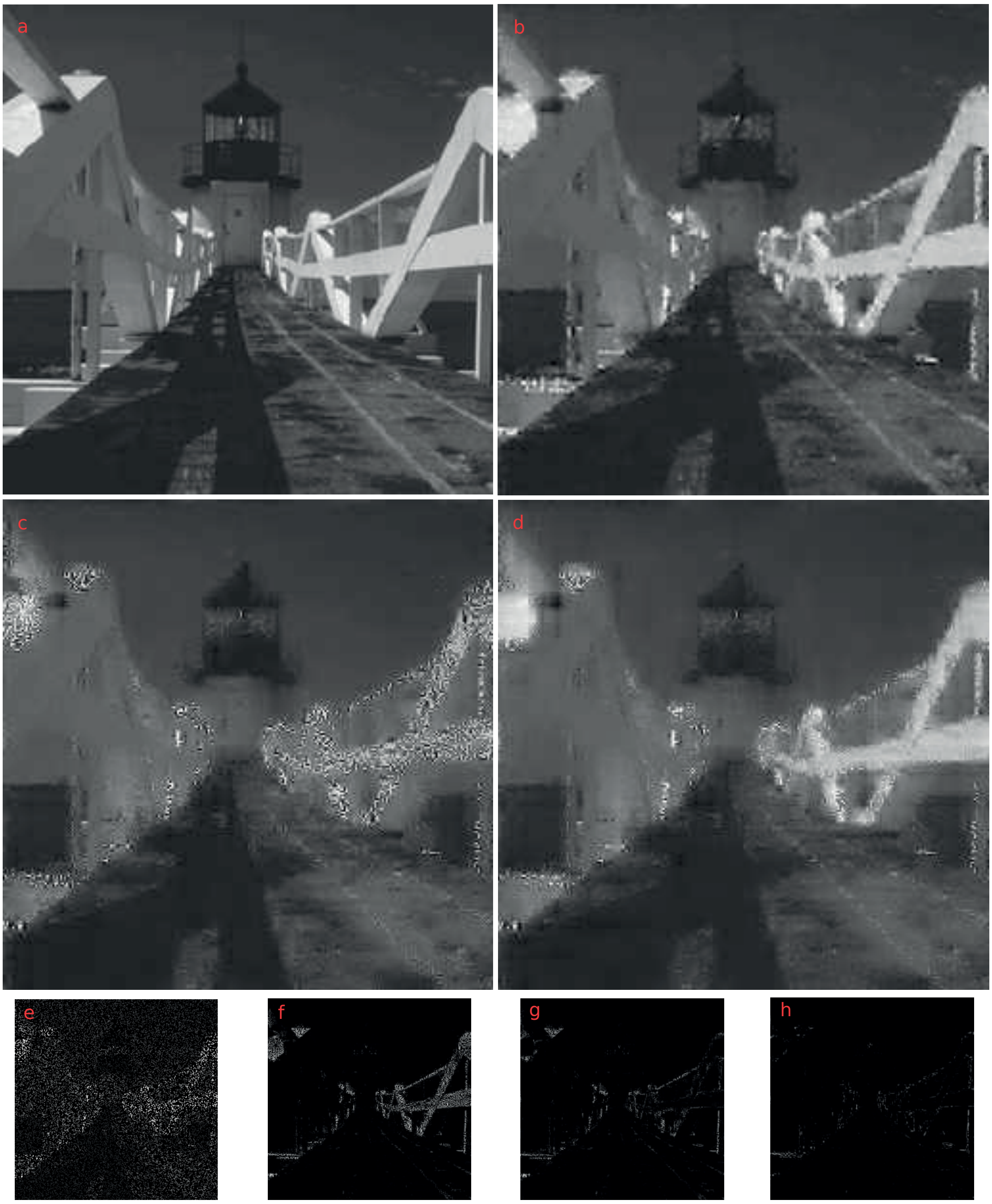


Figure 2.13 – Example results for inpainting "Bridge": a: Truth. **b: VB-Lap-tree**. c: MAP-Lap-tree. d: VB-Lap-fact. e: $\boldsymbol{y}$ (Input: 25% of a). f: $|c-a|$. g: $|d-a|$. **h:** $|b-a|$.

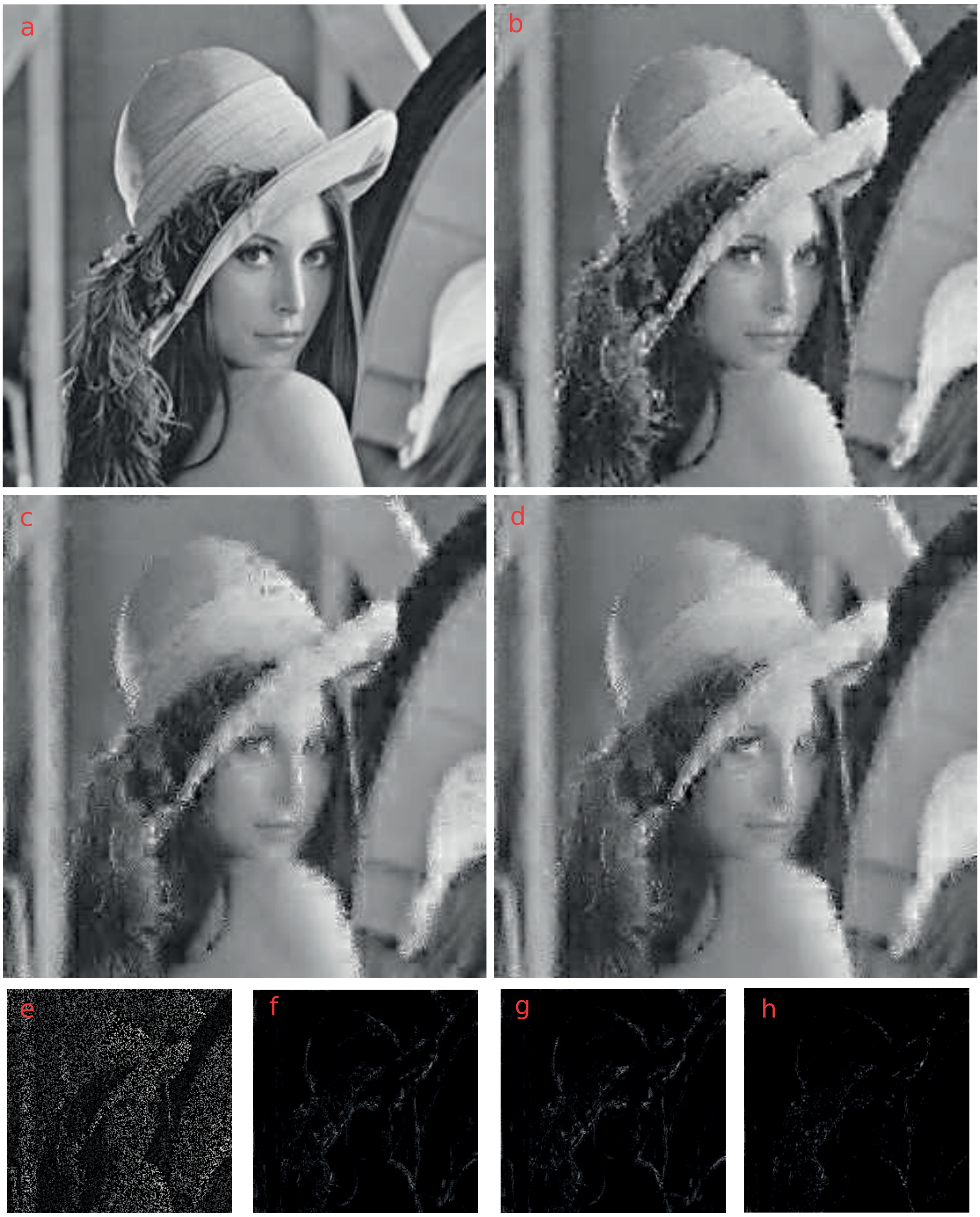


Figure 2.14 – Example results for inpainting "Lena" a: Truth. **b: VB-Lap-tree**. c: MAP-Lap-tree. d: VB-Lap-fact. e: $\boldsymbol{y}$ (Input: 25% of a). f: $|c-a|$. g: $|d-a|$. **h:** $|b-a|$.

3 Models of Implicit Feedback

Recommender systems have become a crucial element for the success of providers of digital goods and services. As product catalogs, audiences and their respective diversities grow, companies are faced with the challenge to keep users engaged by helping them to effectively discover relevant items. Thus, the problem of recommending items to users is one of modeling the tastes or preferences of individuals or groups of individuals for the items on offer, e.g. by inferring personalized rankings over or utilities for the items. The problem of representing and learning such utilities is often posed in terms of a rating prediction problem: For a set of U users indexed by $\mathbb{U} = [U]$ and I items indexed by $\mathbb{I} = [I]$, we target a k-star rating matrix $\boldsymbol{Y} \in \{1, \ldots, k\}^{I \times U}$ where $\boldsymbol{Y}_{iu}$ represents the rating of user u for item i. This matrix is typically only sparsely observed: we only have access to a relatively small set of entries $\mathbb{O} \subset \mathbb{I} \times \mathbb{U}$. A crucial aspect in this setting is therefore to exploit dependencies between users and their rating behavior. A particularly successful approach is to treat the observations as real values and to compute a low-rank decomposition of $\boldsymbol{Y} \approx \boldsymbol{V}\boldsymbol{U}^T$, e.g. by performing an SVD, i.e. minimizing the squared loss. The matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ can be interpreted as latent feature matrices describing users and items respectively such that users with similar features produce a similar rating for the same item [Koren et al., 2009]. While the availability of publicly available benchmark data has facilitated the development and adoption of these methods, it has been recognized that there are downsides to the explicit rating prediction paradigm. For example, Rendle et al. [2009] argue, that a major hurdle is data acquisition. It seems difficult to reliably collect large amounts of high-quality ratings without considerable effort or, most ironically, diminishing user experience. Moreover, depending on the metrics a service provider cares about, rating prediction could even be considered but a proxy problem, in that rating an item might be an atypical event, somewhat disconnected from the actual user behavior. An alternative, potentially much more scalable approach is to leverage sources of *implicit* feedback, i.e. data gathered by passively observing user behavior interacting with the service. Confidently drawing conclusions from implicit feedback, however, is difficult due to the uncalibrated nature of the data: the frequency of interacting with services can vary significantly between users depending on a wide range of circumstantial factors.

Furthermore, there is typically no concept of negative examples: Not choosing an item could be due to unfamiliarity. While the signal contained in this type of feedback is therefore much weaker compared to explicit ratings, it is typically much more abundant and may lead to valuable insights when mined over a large user base in conjunction with appropriate assumptions [Rendle et al., 2009].

A prime example for the transition towards implicit feedback from the private sector is Netflix[1], who is often associated with pioneering the rating prediction paradigm. With the transformation from a DVD rental to an online media streaming service, the company gained access to fine-grained, but implicit, user feedback, quasi in real time, enabling them to implement a much tighter feedback loop and to even discover patterns in plot preferences that they successfully used to produce original content.

A major challenge to deal with implicit feedback is its diversity: user behavior is often recorded as sequences of choices from an item catalog, aggregated as counts or encoded into binary form. Different types of data and the scale of the problems pose immense challenges for developing and deploying probabilistic models in practice. We address such problems arising in two different settings: a global static setting and a contextual dynamic setting.

In the global static setting, we follow the line of work that extends models with bi-linear latent variable structure to observations appropriate to represent implicit feedback. While probabilistic models excel at hierarchically combining discrete and continuous variables, inference in the models of interest here is intractable, even for a Gaussian likelihood, due to the product of Gaussian latent variables [Ilin and Raiko, 2010]. In the context of bi-linear models, Bayesian estimators have proven to be more robust against overfitting compared to point estimates [Blei et al., 2003, Salakhutdinov and Mnih, 2008b, Ilin and Raiko, 2010]. In Section 3.1, we study the posterior inference problem for bi-linear latent Gaussian models for count data. We model the counts using a latent Gaussian bi-linear model with Poisson likelihood. For this model, we develop an approximate inference algorithm based on the variational Gaussian lower-bounding technique and derive a closed-form variational objective under additional constraints. We show that the objective is bi-concave, not only for Poisson but for any log-concave likelihood, suggesting an alternating optimization scheme. In Section 3.2, we devise a method to process pairwise preference statements in a scalable way. There, we combine a low-rank bi-linear model with non-parametric item-feature regression. We develop a novel approximate variational Expectation Maximization algorithm that mitigates the computational challenges due to couplings between entries in the latent preference matrix introduced by the pairwise comparisons of items.

In the contextual dynamic setting, we consider implicit feedback at a different, much finer granularity by looking at logs of individual user activity events. In order to deal with the

---

[1] http://techblog.netflix.com/2012/04/netflix-recommendations-beyond-5-stars.html

abundant availability of such data, in Section 3.3, we turn to recurrent neural networks, as a highly flexible, but non-probabilistic sequence model, able to capture complex, long-range dependencies in sequences. We combine the sequence learning capability of RNNs with ideas from collaborative filtering and present a novel, generic collaborative sequence model.

3.1 Poisson Matrix Factorization

3.1.1 Overview and Related Work

In this part, we consider the posterior inference problem for the generalization of the probabilistic matrix factorization model to count data. Let $\mathbb{U} = [U]$ and $\mathbb{I} = [I]$ denote index sets representing users and items, respectively. We use u as user index and i as item index. Here, we partially observe a matrix $\boldsymbol{Y} \in \mathbb{N}^{I \times U}$ for a subset $\mathbb{O} \subset \mathbb{I} \times \mathbb{U}$. We refer to these observations as $\boldsymbol{Y}_{\mathbb{O}}$ and individual entries as y_{iu}. We assume these entries to be drawn from a Poisson likelihood with intensity $\lambda_{iu} > 0$

$$P(y_{iu} \mid \lambda_{iu}) = \frac{1}{y_{iu}!} \lambda_{iu}^{y_{iu}} e^{-\lambda_{iu}}. \tag{3.1}$$

We parameterize $\lambda_{iu} = e^{f_{iu}}$ and assume the matrix $\boldsymbol{F} = [f_{iu}]_{(i,u) \in \mathbb{I} \times \mathbb{U}}$ to have a low rank representation

$$\boldsymbol{F} = \boldsymbol{V} \boldsymbol{U}^T \tag{3.2}$$

with matrices $\boldsymbol{U} \in \mathbb{R}^{U \times D}$ and $\boldsymbol{V} \in \mathbb{R}^{I \times D}$ and $D \ll \min\{I, U\}$. The vectors $\boldsymbol{u}_u$ and $\boldsymbol{v}_i$ are D row vectors of of $\boldsymbol{V}$ and $\boldsymbol{U}$. We consider the following likelihood:

$$P(\boldsymbol{Y}_{\mathbb{O}} \mid \boldsymbol{F}) = P(\boldsymbol{Y}_{\mathbb{O}} \mid \boldsymbol{V}, \boldsymbol{U}) = \prod_{(i,u) \in \mathbb{O}} \frac{1}{y_{iu}!} \lambda(f_{iu})^{y_{iu}} e^{-\lambda(f_{iu})}. \tag{3.3}$$

The prior over the latent matrices decomposes as $P(\boldsymbol{V}, \boldsymbol{U}) = P(\boldsymbol{V})P(\boldsymbol{U})$ and further factorizes over the rows of the matrices into centered Gaussians:

$$P(\boldsymbol{V}) = \prod_{i \in \mathbb{I}} = \mathcal{N}(\boldsymbol{v}_i \mid 0, \boldsymbol{\Sigma_v}) \tag{3.4}$$

$$P(\boldsymbol{U}) = \prod_{u \in \mathbb{U}} = \mathcal{N}(\boldsymbol{u}_u \mid 0, \boldsymbol{\Sigma_u}) \tag{3.5}$$

We derive a variational Gaussian lower bound on the marginal likelihood and show that under additional constraints the lower bound can be efficiently evaluated in closed form. We show that for a particular parameterization the bound maximization problem is bi-concave. We demonstrate on real-world data that overfitting of point estimation (MAP) and partially Bayesian methods can be avoided by probabilistic inference for prediction. On these datasets, we show that fully factorized approximations are competitive in terms of storage and computation with MAP, while enjoying the robustness and predictive performance common to probabilistic methods. These findings are in line with e.g. [Ilin and Raiko, 2010] for a Gaussian likelihood.

Related Work. There is a rich body of work on models related to generalizing latent bilinear models for different types of data. For a Gaussian likelihood $P(y \mid f) = \mathcal{N}(0, 1)$, Bishop [1999] studies Bayesian inference for PCA and demonstrates that the latent dimensionality can be recovered. In the recommender setting with sparse observations, Lim and Teh [2007] propose a variational inference method while Salakhutdinov and Mnih [2008a] propose a method based on sampling. Ilin and Raiko [2010] give a comprehensive overview of inference and estimation in this model demonstrating the robustness of Bayesian estimators and comparing different algorithmic strategies for variational inference. In a series of publications, the authors of Nakajima et al. [2013] develop the theory of variational inference approximations, characterizing optimal solutions of different methods for complete data.

In a similar spirit to the work in this part are efforts to generalize the model to non-Gaussian likelihoods. Seeger and Bouchard [2012] devise a scalable algorithm using the results of Nakajima et al. [2013] for binary and count data. Their method involves an additional bounding step, but is designed to overcome slow convergence of alternating schemes. Klami [2014] uses auxiliary-variable reformulations to develop new approximate inference schemes for logistic Bernoulli and negative binomial likelihoods.

Apart from the analysis of implicit feedback, count matrices frequently arise in neuroscience [Buesing et al., 2012, Park and Pillow, 2013].

3.1.2 Variational Gaussian Inference

We approximate the intractable posterior $P(\boldsymbol{V}, \boldsymbol{U} \mid \boldsymbol{Y}_{\mathbb{O}})$ by a factorized Gaussian $Q(\boldsymbol{V}, \boldsymbol{U}) = Q(\boldsymbol{V})Q(\boldsymbol{U})$ [Lim and Teh, 2007, Ilin and Raiko, 2010, Seeger and Bouchard, 2012, Nakajima et al., 2013, Klami, 2014]:

$$Q(\boldsymbol{V}) = \prod_{i \in \mathbb{I}} \mathcal{N}(\boldsymbol{v}_i \mid \boldsymbol{\xi}_{\boldsymbol{v}_i}, \boldsymbol{\Xi}_{\boldsymbol{v}_i}) \tag{3.6}$$

$$Q(\boldsymbol{U}) = \prod_{u \in \mathbb{U}} \mathcal{N}(\boldsymbol{u}_u \mid \boldsymbol{\xi}_{\boldsymbol{u}_u}, \boldsymbol{\Xi}_{\boldsymbol{u}_u}) \tag{3.7}$$

The set of variational parameters is thus $\mathcal{V} = \left\{\boldsymbol{\xi}_{\boldsymbol{u}_u}, \boldsymbol{\xi}_{\boldsymbol{v}_i}, \boldsymbol{\Xi}_{\boldsymbol{u}_u}, \boldsymbol{\Xi}_{\boldsymbol{v}_i} : i \in \mathbb{I}, u \in \mathbb{U}\right\}$ The bound on the marginal log-likelihood is

$$\begin{aligned}
\log P(\boldsymbol{Y}_{\mathbb{O}}) &= \log \int_{\boldsymbol{V}} \int_{\boldsymbol{U}} \frac{P(\boldsymbol{Y}_{\mathbb{O}} \mid \boldsymbol{V}, \boldsymbol{U}) P(\boldsymbol{V}) P(\boldsymbol{U})}{Q(\boldsymbol{V}) Q(\boldsymbol{U})} Q(\boldsymbol{V}) Q(\boldsymbol{U}) \; d\boldsymbol{U} d\boldsymbol{V} && (3.8)\\
&\geq \mathrm{E}_{Q(\boldsymbol{V},\boldsymbol{U})} \left[\log \frac{P(\boldsymbol{Y}_{\mathbb{O}} \mid \boldsymbol{V}, \boldsymbol{U}) P(\boldsymbol{V}) P(\boldsymbol{U})}{Q(\boldsymbol{V}) Q(\boldsymbol{U})}\right] && (3.9)\\
&= \sum_{(i,u)\in\mathbb{O}} \mathrm{E}_{Q(\boldsymbol{v}_i)Q(\boldsymbol{u}_u)} \left[\log P(y_{in} \mid f_{in})\right] && (3.10)\\
&\quad - \sum_{i\in\mathbb{I}} D_{\mathrm{KL}} \left[Q(\boldsymbol{v}_i) \| P(\boldsymbol{v}_i)\right] && (3.11)\\
&\quad - \sum_{u\in\mathbb{U}} D_{\mathrm{KL}} \left[Q(\boldsymbol{u}_u) \| P(\boldsymbol{u}_u)\right] && (3.12)\\
&=: \mathcal{L}_{\mathrm{VG}}(\mathcal{V}) && (3.13)
\end{aligned}$$

The variational inference problem is then[2]

$$\begin{aligned}
\max_{\mathcal{V}} \quad & \mathcal{L}_{\mathrm{VG}}(\mathcal{V}) & \\
s.t. \quad & \boldsymbol{\Xi}_{\boldsymbol{u}_u} \succ 0 & \forall u \in \mathbb{U}\\
& \boldsymbol{\Xi}_{\boldsymbol{v}_i} \succ 0 & \forall i \in \mathbb{I}
\end{aligned}$$

In contrast to the KL terms, that are available in closed form, the terms involving the non-Gaussian likelihood cannot be easily evaluated. Dropping indices, a single term reads

$$\mathrm{E}_{Q(\boldsymbol{v})Q(\boldsymbol{u})} \left[\log P(y \mid f)\right] = y \boldsymbol{\xi}_{\boldsymbol{v}}^T \boldsymbol{\xi}_{\boldsymbol{u}} - \mathrm{E}_{Q(\boldsymbol{v})Q(\boldsymbol{u})} \left[e^{\boldsymbol{v}^T \boldsymbol{u}}\right] - \log y! \tag{3.14}$$

In linear models, or if $\boldsymbol{V}$ is a parameter, the likelihood-dependent term can often be written as an expectation over a univariate Gaussian, simplifying the task of computing it.

Our main result is to show that by introducing additional constraints, which effectively loosen the bound, the expectations in (3.14) can be evaluated efficiently. Concretely, we introduce generalized inequalities [Boyd and Vandenberghe, 2004]

$$\boldsymbol{\Xi}_{\boldsymbol{v}_i}^{-1} \succ \boldsymbol{\Xi}_{\boldsymbol{u}_u} \forall (i,u) \in \mathbb{O} \tag{3.15}$$

establishing a link between the variational parameters, where $\boldsymbol{A} \succ \boldsymbol{B}$ means $\boldsymbol{A} - \boldsymbol{B} \succ 0$. The relationship is symmetric due the inherent symmetry in the model. The optimization

---

[2] The constraints can be avoided by reparameterizing the covariance matrices, e.g. $\boldsymbol{\Xi} = \boldsymbol{L}\boldsymbol{L}^T$, which will be done later.

problem we consider is thus the following:

$$\begin{aligned}
\max_{\mathcal{V}} \quad & \mathcal{L}_{\mathrm{VG}}(\mathcal{V}) & \\
s.t. \quad & \boldsymbol{\Xi}_{\boldsymbol{u}_u} \succ 0 & \forall u \in \mathbb{U} \\
& \boldsymbol{\Xi}_{\boldsymbol{v}_i} \succ 0 & \forall i \in \mathbb{I} \\
& \boldsymbol{\Xi}_{\boldsymbol{v}_i}^{-1} \succ \boldsymbol{\Xi}_{\boldsymbol{u}_u} & \forall (i,u) \in \mathbb{O}
\end{aligned}$$

Proposition 2. *Let* $\boldsymbol{v}$ *and* $\boldsymbol{u}$ *be independent and distributed as* $\boldsymbol{v} \sim \mathcal{N}(\boldsymbol{\xi}_{\boldsymbol{v}}, \boldsymbol{\Xi}_{\boldsymbol{v}})$ *and* $\boldsymbol{u} \sim \mathcal{N}(\boldsymbol{\xi}_{\boldsymbol{u}}, \boldsymbol{\Xi}_{\boldsymbol{u}})$*. Then, under the additional constraints (3.15)*

$$\boldsymbol{\Xi}_{\boldsymbol{v}}^{-1} \succ \boldsymbol{\Xi}_{\boldsymbol{u}},$$

the following identity holds for the expectation in (3.14)

$$\mathrm{E}\left[e^{\boldsymbol{v}^T \boldsymbol{u}}\right] = |\boldsymbol{S}|^{-\frac{1}{2}} \exp\left(\frac{1}{2}\boldsymbol{t}^T \boldsymbol{B}^{-1} \boldsymbol{t} - \frac{1}{2}\boldsymbol{\xi}_{\boldsymbol{v}}^T \boldsymbol{\Xi}_{\boldsymbol{v}}^{-1} \boldsymbol{\xi}_{\boldsymbol{v}}\right)$$

where $\boldsymbol{t} := \boldsymbol{\xi}_{\boldsymbol{u}} + \boldsymbol{\Xi}_{\boldsymbol{v}}^{-1}\boldsymbol{\xi}_{\boldsymbol{v}}$*,* $\boldsymbol{S} := \boldsymbol{I} - \boldsymbol{\Xi}_{\boldsymbol{v}}\boldsymbol{\Xi}_{\boldsymbol{u}}$ *and* $\boldsymbol{B} := \boldsymbol{\Xi}_{\boldsymbol{v}}^{-1} - \boldsymbol{\Xi}_{\boldsymbol{u}}$*.*

Proof. We show the identity by first writing

$$\mathrm{E}_{Q(\boldsymbol{v})Q(\boldsymbol{u})}\left[e^{\boldsymbol{v}^T \boldsymbol{u}}\right] = \mathrm{E}_{Q(\boldsymbol{v})}\left[\mathrm{E}_{Q(\boldsymbol{u})}\left[e^{\boldsymbol{v}^T \boldsymbol{u}}\right]\right] \tag{3.16}$$

The inner expectation is readily evaluated defining $f = \boldsymbol{v}^T \boldsymbol{u}$, which, conditioned on $\boldsymbol{v}$, is distributed as $f \sim \mathcal{N}\left(\mu, \sigma^2\right)$, $\mu = \boldsymbol{v}^T \boldsymbol{\xi}_{\boldsymbol{u}}$, $\sigma^2 = \boldsymbol{v}^T \boldsymbol{\Xi}_{\boldsymbol{u}} \boldsymbol{v}$. The resulting uni-variate expectation can be simliarly evaluated as in (2.33) as

$$\mathrm{E}\left[e^f\right] = e^{\frac{1}{2}\sigma^2 + \mu} = \exp\left(\frac{1}{2}\boldsymbol{v}^T \boldsymbol{\Xi}_{\boldsymbol{u}} \boldsymbol{v} + \boldsymbol{v}^T \boldsymbol{\xi}_{\boldsymbol{u}}\right) \tag{3.17}$$

Next, we expand the outer expectation and examine the exponent in

$$\mathrm{E}\left[\exp\left(\frac{1}{2}\boldsymbol{v}^T \boldsymbol{\Xi}_{\boldsymbol{u}} \boldsymbol{v} + \boldsymbol{v}^T \boldsymbol{\xi}_{\boldsymbol{u}}\right)\right] = \int \mathcal{N}\left(\boldsymbol{v} \mid \boldsymbol{\xi}_{\boldsymbol{v}}, \boldsymbol{\Xi}_{\boldsymbol{v}}\right) \exp\left(\frac{1}{2}\boldsymbol{v}^T \boldsymbol{\Xi}_{\boldsymbol{u}} \boldsymbol{v} + \boldsymbol{v}^T \boldsymbol{\xi}_{\boldsymbol{u}}\right) \mathrm{d}\boldsymbol{v} \tag{3.18}$$

which is given and simplified by

$$\begin{aligned}
& -\frac{1}{2}\left(\left(\boldsymbol{v} - \boldsymbol{\xi}_{\boldsymbol{v}}\right)^T \boldsymbol{\Xi}_{\boldsymbol{v}}^{-1}\left(\boldsymbol{v} - \boldsymbol{\xi}_{\boldsymbol{v}}\right) - 2\boldsymbol{\xi}_{\boldsymbol{u}}^T \boldsymbol{v} - \boldsymbol{v}^T \boldsymbol{\Xi}_{\boldsymbol{u}} \boldsymbol{v}\right) && (3.19) \\
= & -\frac{1}{2}\left(\boldsymbol{v}^T\left(\boldsymbol{\Xi}_{\boldsymbol{v}}^{-1} - \boldsymbol{\Xi}_{\boldsymbol{u}}\right)\boldsymbol{v} - 2\left(\boldsymbol{\xi}_{\boldsymbol{u}} + \boldsymbol{\Xi}_{\boldsymbol{v}}^{-1}\boldsymbol{\xi}_{\boldsymbol{v}}\right)^T \boldsymbol{v} + \boldsymbol{\xi}_{\boldsymbol{v}}^T \boldsymbol{\Xi}_{\boldsymbol{v}}^{-1} \boldsymbol{\xi}_{\boldsymbol{v}}\right) && (3.20) \\
= & -\frac{1}{2}\left(\boldsymbol{v}^T \boldsymbol{B} \boldsymbol{v} - 2\boldsymbol{t}^T \boldsymbol{v} + \boldsymbol{\xi}_{\boldsymbol{v}}^T \boldsymbol{\Xi}_{\boldsymbol{v}}^{-1} \boldsymbol{\xi}_{\boldsymbol{v}}\right) && (3.21)
\end{aligned}$$

where we replaced $\boldsymbol{B} = \left(\boldsymbol{\Xi}_{\boldsymbol{v}}^{-1} - \boldsymbol{\Xi}_{\boldsymbol{u}}\right)$ and $\boldsymbol{t} = \boldsymbol{\xi}_{\boldsymbol{u}} + \boldsymbol{\Xi}_{\boldsymbol{v}}^{-1}\boldsymbol{\xi}_{\boldsymbol{v}}$. Defining $\boldsymbol{m} = \boldsymbol{B}^{-1}\boldsymbol{t}$, we complete the square

$$= -\frac{1}{2}\left(\boldsymbol{v}^T\boldsymbol{B}\boldsymbol{v} - 2\boldsymbol{m}^T\boldsymbol{B}\boldsymbol{v} + \boldsymbol{m}^T\boldsymbol{B}\boldsymbol{m} - \boldsymbol{m}^T\boldsymbol{B}\boldsymbol{m} + \boldsymbol{\xi}_{\boldsymbol{v}}^T\boldsymbol{\Xi}_{\boldsymbol{v}}^{-1}\boldsymbol{\xi}_{\boldsymbol{v}}\right) \tag{3.22}$$

$$= -\frac{1}{2}\left(\boldsymbol{v} - \boldsymbol{m}\right)^T\boldsymbol{B}\left(\boldsymbol{v} - \boldsymbol{m}\right) + \frac{1}{2}\boldsymbol{t}^T\boldsymbol{B}^{-1}\boldsymbol{t} - \frac{1}{2}\boldsymbol{\xi}_{\boldsymbol{v}}^T\boldsymbol{\Xi}_{\boldsymbol{v}}^{-1}\boldsymbol{\xi}_{\boldsymbol{v}} \tag{3.23}$$

The additional constraints imply that $\boldsymbol{B}$ is positive definite, such that the quadratic in $\boldsymbol{v}$ is a Gaussian function, and the integral can be evaluated as

$$\int \exp\left(-\frac{1}{2}\left(\boldsymbol{v} - \boldsymbol{m}\right)^T\boldsymbol{B}\left(\boldsymbol{v} - \boldsymbol{m}\right)\right)d\boldsymbol{v} = |2\pi\boldsymbol{B}^{-1}|^{\frac{1}{2}} \tag{3.24}$$

Multiplying this with the the normalizing constant of $Q(\boldsymbol{v})$, $|2\pi\boldsymbol{\Xi}_{\boldsymbol{v}}|^{-\frac{1}{2}}$, gives

$$|2\pi\boldsymbol{\Xi}_{\boldsymbol{v}}|^{-\frac{1}{2}}|2\pi\boldsymbol{B}^{-1}|^{\frac{1}{2}} = |\boldsymbol{\Xi}_{\boldsymbol{v}}\boldsymbol{B}|^{-\frac{1}{2}} = |\boldsymbol{I} - \boldsymbol{\Xi}_{\boldsymbol{v}}\boldsymbol{\Xi}_{\boldsymbol{u}}|^{-\frac{1}{2}} = |\boldsymbol{S}|^{-\frac{1}{2}} \tag{3.25}$$

Combining this with the remaining terms of (3.23) yields the result

$$\mathrm{E}_{Q(\boldsymbol{v})}\left[e^{\boldsymbol{\xi}_{\boldsymbol{u}}^T\boldsymbol{v} + \frac{1}{2}\boldsymbol{v}^T\boldsymbol{\Xi}_{\boldsymbol{u}}\boldsymbol{v}}\right] = |\boldsymbol{S}|^{-\frac{1}{2}}\exp\left(\frac{1}{2}\boldsymbol{t}^T\boldsymbol{B}^{-1}\boldsymbol{t} - \frac{1}{2}\boldsymbol{\xi}_{\boldsymbol{v}}^T\boldsymbol{\Xi}_{\boldsymbol{v}}^{-1}\boldsymbol{\xi}_{\boldsymbol{v}}\right) \tag{3.26}$$

□

Putting everything together, using that KL-divergences between two multivariate Gaussians is given by

$$D_{\mathrm{KL}}\left[\mathcal{N}(\boldsymbol{\xi}, \boldsymbol{\Xi})\|\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})\right] = \frac{1}{2}\left(\mathrm{tr}\left(\boldsymbol{\Sigma}^{-1}\boldsymbol{\Xi}\right) + \left(\boldsymbol{\mu} - \boldsymbol{\xi}\right)^T\boldsymbol{\Sigma}^{-1}\left(\boldsymbol{\mu} - \boldsymbol{\xi}\right) + \log\frac{|\boldsymbol{\Sigma}|}{|\boldsymbol{\Xi}|} - D\right) \tag{3.27}$$

the lower bound up to constants not depending on $\mathcal{V}$ becomes

$$\mathcal{L}_{\mathrm{VG}}(\mathcal{V}) \doteq \sum_{(i,u)\in\mathbb{O}} y_{iu}\boldsymbol{\xi}_{\boldsymbol{v}_i}^T\boldsymbol{\xi}_{\boldsymbol{u}_u} - |\boldsymbol{S}_{iu}|^{-\frac{1}{2}}\exp\left(\frac{1}{2}\boldsymbol{t}_{iu}^T\boldsymbol{B}_{iu}^{-1}\boldsymbol{t}_{iu} - \frac{1}{2}\boldsymbol{\xi}_{\boldsymbol{v}_i}^T\boldsymbol{\Xi}_{\boldsymbol{v}_i}^{-1}\boldsymbol{\xi}_{\boldsymbol{v}_i}\right) \tag{3.28}$$

$$+ \frac{1}{2}\sum_{i=1}^{I}\left[\log|\boldsymbol{\Xi}_{\boldsymbol{v}_i}| - \mathrm{tr}(\boldsymbol{\Sigma}_{\boldsymbol{v}}^{-1}\boldsymbol{\Xi}_{\boldsymbol{v}_i}) - \boldsymbol{\xi}_{\boldsymbol{v}_i}^T\boldsymbol{\Sigma}_{\boldsymbol{v}}^{-1}\boldsymbol{\xi}_{\boldsymbol{v}_i}\right] \tag{3.29}$$

$$+ \frac{1}{2}\sum_{u=1}^{U}\left[\log|\boldsymbol{\Xi}_{\boldsymbol{u}_u}| - \mathrm{tr}(\boldsymbol{\Sigma}_{\boldsymbol{u}}^{-1}\boldsymbol{\Xi}_{\boldsymbol{u}_u}) - \boldsymbol{\xi}_{\boldsymbol{u}_u}^T\boldsymbol{\Sigma}_{\boldsymbol{u}}^{-1}\boldsymbol{\xi}_{\boldsymbol{u}_u}\right] \tag{3.30}$$

Note, that this formulation is symmetric under relabeling of $\boldsymbol{u}$ into $\boldsymbol{v}$ since the order in (3.16) was chosen arbitrarily.

While not jointly concave in all of $\mathcal{V}$, we can easily establish biconcavity of $\mathcal{L}_{\text{VG}}$. First, we reparameterize the covariance parameters by their respective lower-triangular Cholesky factors. We define $\mathcal{V}_L = \left\{\boldsymbol{\xi}_{\boldsymbol{u}_u}, \boldsymbol{\xi}_{\boldsymbol{v}_i}, \boldsymbol{L}_{\boldsymbol{u}_u}, \boldsymbol{L}_{\boldsymbol{v}_i} : i \in \mathbb{I}, u \in \mathbb{U}, \Xi_{\boldsymbol{u}_u} = \boldsymbol{L}_{\boldsymbol{u}_u}\boldsymbol{L}_{\boldsymbol{u}_u}^T, \Xi_{\boldsymbol{v}_i} = \boldsymbol{L}_{\boldsymbol{v}_i}\boldsymbol{L}_{\boldsymbol{v}_i}^T\right\}$. Furthermore let $\mathcal{V}_L(\boldsymbol{v})$ and $\mathcal{V}_L(\boldsymbol{u})$ denote the parameters in $\mathcal{V}_L$ parameterizing to $Q(\boldsymbol{V})$ and $Q(\boldsymbol{U})$, respectively. Finally, let $\mathcal{L}_{\text{VG}}(\mathcal{V}_L(\boldsymbol{u}) \mid \mathcal{V}_L(\boldsymbol{v}))$ denote $\mathcal{L}_{\text{VG}}$ as a function of $\mathcal{V}_L(\boldsymbol{u})$, i.e. $\mathcal{V}_L(\boldsymbol{v})$ fixed.

Proposition 3. *For any log-concave likelihood* $P(y \mid f)$, $\mathcal{L}_{\text{VG}}(\mathcal{V}_L(\boldsymbol{u}) \mid \mathcal{V}_L(\boldsymbol{v}))$ *is jointly concave in* $\mathcal{V}_L(\boldsymbol{u})$ *and* $\mathcal{L}_{\text{VG}}(\mathcal{V}_L(\boldsymbol{v}) \mid \mathcal{V}_L(\boldsymbol{u}))$ *is jointly concave in* $\mathcal{V}_L(\boldsymbol{v})$.

Proof. We show the first part of the statement. The second part follows from symmetry. The lower bound in (3.10) decouples additively over the $\boldsymbol{u}_u$, so that we can focus on all terms belonging to a single index u. We denote by $\mathbb{O}_u$ all items observed by u. The relevant part of $\mathcal{L}_{\text{VG}}$ is therefore

$$\mathcal{L}_{\text{VG}}(\mathcal{V}_L(\boldsymbol{u}_u) \mid \mathcal{V}_L(\boldsymbol{v})) = \sum_{i \in \mathbb{O}_u} \mathrm{E}_{Q(\boldsymbol{v}_i)Q(\boldsymbol{u}_u)}\left[\log P(y_{in} \mid f_{in})\right] - D_{\text{KL}}\left[Q(\boldsymbol{u}_u) \| P(\boldsymbol{u}_u)\right] \tag{3.31}$$

With fixed $\mathcal{V}_L(\boldsymbol{v})$, this has the form of the VG approximation for a linear model. We use the results of [Challis and Barber, 2011], who show that the bound is jointly concave in $\boldsymbol{\xi}_{\boldsymbol{u}_u}$ and $\boldsymbol{L}_{\boldsymbol{u}_u}$ if $\mathrm{E}_{Q(\boldsymbol{v}_i)Q(\boldsymbol{u}_u)}\left[\log P(y_{in} \mid f_{in})\right]$ is. We can easily extend their result that log-concavity of $P(y \mid f)$ in f implies concavity of $\mathrm{E}_{Q(\boldsymbol{u}_u)}\left[\log P(y_{in} \mid f_{in})\right]$ by using that non-negative weighted sums preserve concavity [Boyd and Vandenberghe, 2004]. □

Thus, we can use the standard convergent VB-EM algorithm [e.g. Beal, 2003]

$$\mathcal{V}_L(\boldsymbol{u})^{(t+1)} = \operatorname*{argmax}_{\mathcal{V}_L(\boldsymbol{u})} \mathcal{L}_{\text{VG}}\left(\mathcal{V}_L(\boldsymbol{u}) \mid \mathcal{V}_L(\boldsymbol{v})^{(t)}\right) \qquad \text{s.t. } \Xi_{\boldsymbol{v}_i}^{-1} \succ \Xi_{\boldsymbol{u}_u} \quad \forall (i,u) \in \mathbb{O} \tag{3.32}$$

$$\mathcal{V}_L(\boldsymbol{v})^{(t+1)} = \operatorname*{argmax}_{\mathcal{V}_L(\boldsymbol{v})} \mathcal{L}_{\text{VG}}\left(\mathcal{V}_L(\boldsymbol{v}) \mid \mathcal{V}_L(\boldsymbol{u})^{(t+1)}\right) \quad \text{s.t. } \Xi_{\boldsymbol{v}_i}^{-1} \succ \Xi_{\boldsymbol{u}_u} \quad \forall (i,u) \in \mathbb{O} \tag{3.33}$$

that consists of concave maximzation problems as sub-routines.

Note, by using the Cholesky parameterization, positive (semi-) definiteness constraints are implied. Furthermore, we note that the objective naturally incorporates a barrier against infeasible solutions. Starting from a feasible set of parameters, consider the formulation in Proposition 2 for a sequence of matrices $\boldsymbol{B}^{(t)}$ where the smallest eigenvalue $\lambda_{\min}(\boldsymbol{B}^{(t)}) \to 0$ as $t \to \infty$, thus approaching the boundary of the feasible set. We see that the objective function will go to $-\infty$ exponentially due to the dependence on the inverse of $\boldsymbol{B}$ in the exponent in (3.28). We therefore solve the subproblems in (3.32) and (3.33) using methods for unconstrained problems.

3.1.3 Experiments

3.1.3.1 Methods

In our experimental section, we compare four different methods to investigate the benefits of probabilistic methods in this setting. Our comparison is similar to [Ilin and Raiko, 2010]. All the methods can be seen as variational Gaussian approximations differing in the way the variational distributions (3.6) and (3.7) are defined. Thus, we use the alternating algorithmic template of (3.32) for all of them. In this section we refer to an approximation with full $D \times D$ covariances $\Xi_{\boldsymbol{v}_i}$ and $\Xi_{\boldsymbol{u}_u}$ as the variational Bayesian methods (VB). We refer to a fully factorized mean-field approximation with diagonal covariances as mean-field (MF). Furthermore, we consider a PPCA-like approximation, where one of the matrices is estimated as a parameter using Expectation Maximization (EM) and maximum a-posteriori point estimate of both matrices (MAP). Note, that the E-Step for the EM method is not exact. We employ a variational Gaussian approximation as well, the bound of which can be computed in closed form as is evident from (3.17). The latter two can be thought of as degenerate methods that do not take into account the uncertainty about the parameter(s) that are estimated [Nakajima and Sugiyama, 2014].

For all non-linear optimization problems that arise, we use a quasi Newton L-BFGS method [Bertsekas, 1999].

	MAP	MF	EM	VB
Storage	$O(D(I+U))$	$O(D(I+U))$	$O(DI+D^2U)$	$O(D^2(I+U))$
Computation	$O(DN_{obs})$	$O(DN_{obs})$	$O(D^2N_{obs})$	$O(D^3N_{obs})$

Table 3.1 – Complexity comparison. Complexity increases from left to right. MAP and EM are existing methods, while MF and VB are proposed methods. $N_{obs} = |\mathbb{O}|$ denotes the total number of observations.

Table (3.1) summarizes complexities of memory and computation of these methods. There, the complexity increases from left to right. The storage complexity directly reflects the amount of posterior uncertainty, that is represented. MAP only keeps track of a point estimate of $\boldsymbol{V}, \boldsymbol{U}$. MF additionally represents variances of all variables, effectively doubling the memory requirement. VB represents covariances over all $I+U$ latent factors, requiring $O(D^2)$ memory per factor. EM lies in between MAP and VB, in that covariances are only represented for one set of variables.

As all methods only require gradient information, computational complexity scales with the number of data-likelihood terms N_{obs}, each of which being involved in the accumulation of the gradient. The methods only differ in the cost per term, which is benign due to the assumption that D is relatively small. For MAP, the contribution to the gradient of a single likelihood term is linear in D. For EM, likelihoods contribute terms like (3.17).

Thus, the gradient requires dense matrix-vector multiplication, that scales as $O(D^2)$. The fully Bayesian methods deal with terms of the form presented in Proposition 2. For VB, the dependence on matrix inverses leads to cubic scaling in D, while the mean-field approximation greatly simplifies these expressions, reducing the cost back to linear. Furthermore, the objective function decomposes additively over users and items such that sub-problems can be solved independently in parallel as suggested in Algorithm (4).

Our implementation is written MATLAB, with some computationally demanding gradient-computations in C++ using the Armadillo linear algebra library [Sanderson, 2010]. We used a Linux server equipped with AMD Opteron 6380 CPUs and 512GB RAM.

Algorithm 4 Variational Gaussian Inference Algorithm

```
Initialize randomly $\mathcal{V}_L(\boldsymbol{v})^{(1)}$
$t \leftarrow 1$
repeat
    for $u = 1, \dots, U$ in parallel do
        $\mathcal{V}_L(\boldsymbol{u}_u)^{(t+1)} \leftarrow \text{argmax}_{\mathcal{V}_L(\boldsymbol{u}_u)} \mathcal{L}_{\text{VG}} \left(\mathcal{V}_L(\boldsymbol{u}_u) \mid \mathcal{V}_L(\boldsymbol{v})^{(t)}\right)$
    end for
    for $i = 1, \dots, I$ in parallel do
        $\mathcal{V}_L(\boldsymbol{v}_i)^{(t+1)} \leftarrow \text{argmax}_{\mathcal{V}_L(\boldsymbol{v}_i)} \mathcal{L}_{\text{VG}} \left(\mathcal{V}_L(\boldsymbol{v}_i) \mid \mathcal{V}_L(\boldsymbol{u})^{(t+1)}\right)$
    end for
    $t \leftarrow t + 1$
until Convergence
```

3.1.3.2 Results

We conduct a comparison on several real-world datasets. We focus on count datasets arising in recommendation systems. We use the posterior-predictive probability as our performance measure. Specifically, given a test observation y^*_{iu}, we compute the (negative) logarithm of the following predictive distribution:

$$-\log P(y^*_{iu}|\boldsymbol{Y}) \approx -\log \int P(y^*_{iu}|\boldsymbol{u}_u, \boldsymbol{v}_i) Q(\boldsymbol{u}_u) Q(\boldsymbol{v}_i) \, \mathrm{d}\boldsymbol{u}_u \, \mathrm{d}\boldsymbol{v}_i \tag{3.34}$$

Thus, lower values indicate better performances, i.e. the method, that assigns higher probability to the value at a test location, incurs a lower error. Due to the log transform, very low probabilities assigned to test samples are penalized strongly.

Since the above integral is intractable, we approximate it by a Monte Carlo estimate using 10^5 samples, to ensure a reliable estimate. For MAP, the error measure reduces to a simple plug-in estimates. We report the average of this quantity over all the test examples.

We compare methods on four real-world datasets summarized in Table 3.2. For each data set, we randomly select 8000 observations, of which 25% are held out for testing. The LastFM and Delicious datasets can be downloaded from [Grouplens, 2011]. The LastFM-Tags and Million-Songs datasets can be obtained from [Lamere, 2008] and [Kaggle, 2012], respectively. To avoid numerical instabilities caused by very large counts, we apply the transformation $y \mapsto \lfloor \sqrt{y} + 0.5 \rfloor$ to LastFM and LastFM Tags. We treat zero counts as missing data[3].

Here we use zero-mean, isotropic normal priors as in [Seeger and Bouchard, 2012], i.e. $\boldsymbol{\Sigma_u} = \sigma_u^2 \boldsymbol{I}$ and $\boldsymbol{\Sigma_v} = \sigma_v^2 \boldsymbol{I}$. Diagonal prior covariances can be used for automatic relevance determination, a mechanism for setting D by optimizing the marginal likelihood. We fix $\sigma_u^2 = 1$ and select σ_v^2 and D using a validation set of size 1500 for each dataset. For EM, which breaks the symmetry between $\boldsymbol{V}$ and $\boldsymbol{U}$, we choose the parameter to be the matrix with the smaller size.

Dataset	I	U	N_{obs}	Description
LastFM	17 632	1892	92 834	Listening counts of songs by user
LastFM Tags	20 907	100 784	952 707	Counts of tags assigned to artists
Delicious	38 603	1867	93 210	Counts of webpages bookmarked by user
Million Songs	163 206	110 000	1 450 933	Listening counts of songs by user

Table 3.2 – Datasets used for experiments

We present the effect of σ_v^2 in Fig. 3.1. The MAP estimate overfits, while Bayesian approaches are more robust. These findings are consistent with the results of similar studies in this domain [Salakhutdinov and Mnih, 2008a, Ilin and Raiko, 2010].

Fig. 3.2 shows the speed-accuracy trade-off between the methods. Accuracy is measured using the error defined in Eq. (3.34), while speed is measured by the running time in seconds. For this comparison, we set D to be the same for all the methods but large enough to give a performance similar to the optimal setting determined on the validation set. We show the results for 10 different test-train splits in Fig. 3.2, and summarize them in Table 3.3 for clarity.

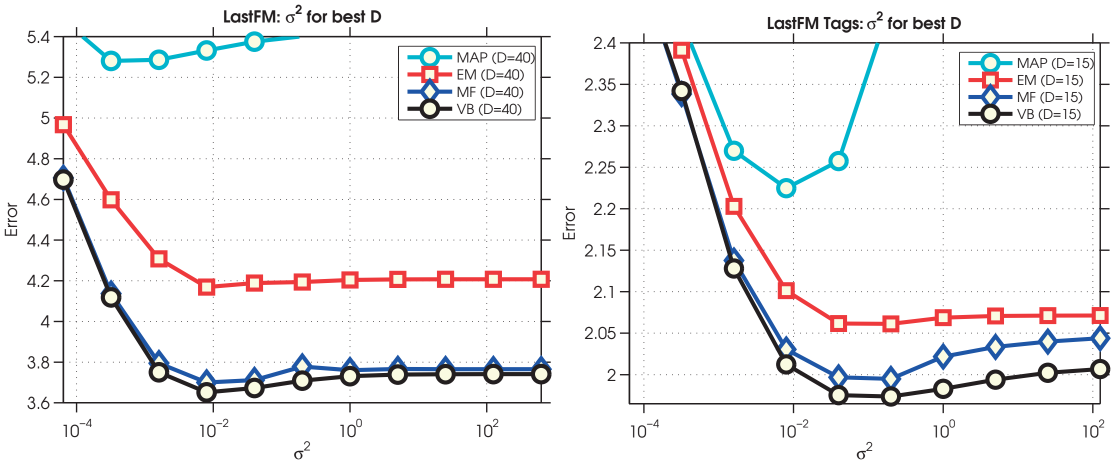
We observe that while VB exhibits the best performance overall, MF is a strong contender due to its speed and competitive performance. It is in the same complexity class as MAP, but slower due to a larger constant factor. Results consistent with ours were obtained for other likelihoods [Ilin and Raiko, 2010, Klami, 2014].

---

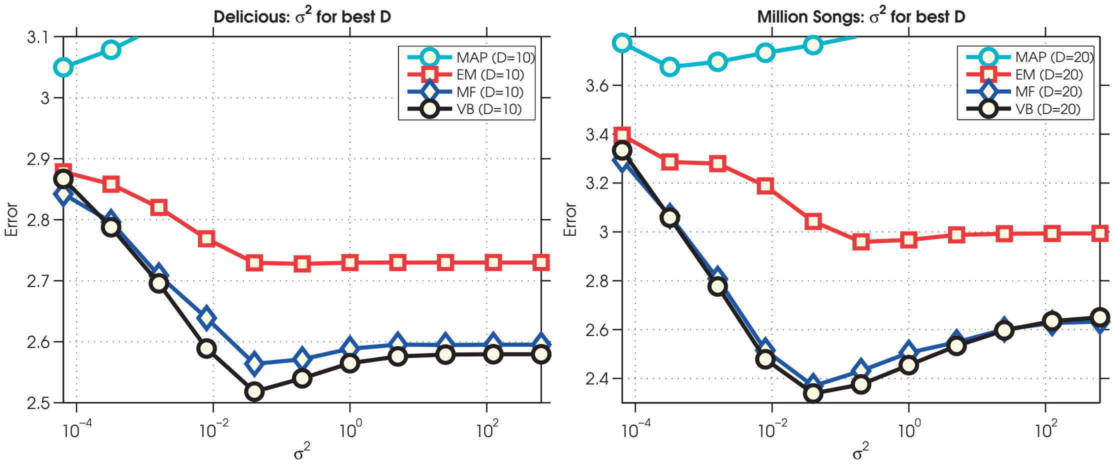
[3] Dealing with unobserved values is a major challenge of dealing with implicit feedback. We cannot know whether a zero count is due to a conscious choice or due to ignorance. We make the common assumption of treating zero counts as missing. We note that this is application specific, as one may actually observe counts of zero, e.g. in neuroscience.

Dataset	MAP	EM	MF	VB
LastFM Tags	2.19 (0.06)	2.02 (0.03)	1.99 (0.02)	1.97 (0.02)
LastFM	5.94 (0.24)	4.65 (0.18)	3.89 (0.08)	3.81 (0.07)
Delicious	3.17 (0.06)	2.62 (0.01)	2.60 (0.01)	2.56 (0.02)
Million Songs	3.54 (0.10)	2.69 (0.07)	2.31 (0.05)	2.28 (0.04)

Table 3.3 – Comparison of methods. We report the error measured by Eq. (3.34). The error is averaged over all test examples for 10 different train-test splits. The standard error is shown inside brackets. We clearly see that both of our methods, MF and VB, achieve the lowest error values.



(a) LastFM (b) LastFM Tags

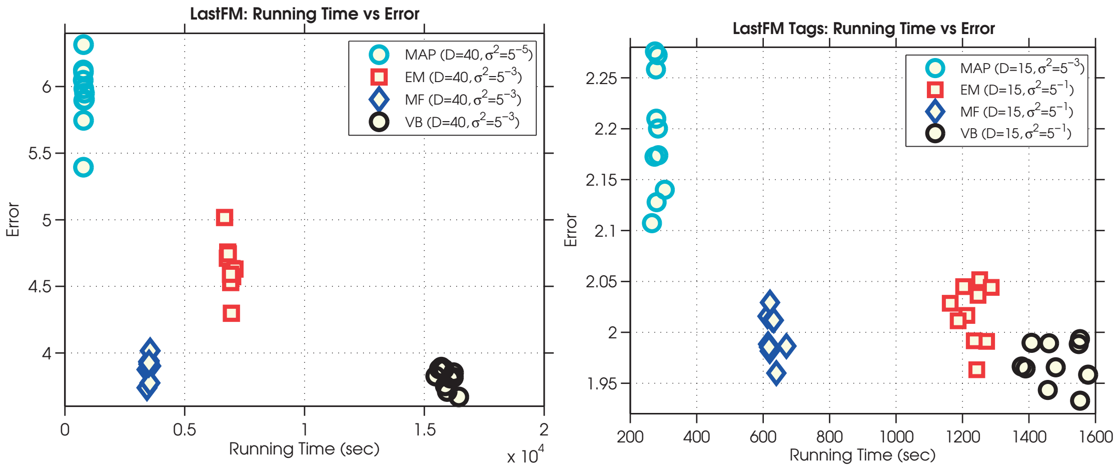


(c) Delicious (d) Million Songs

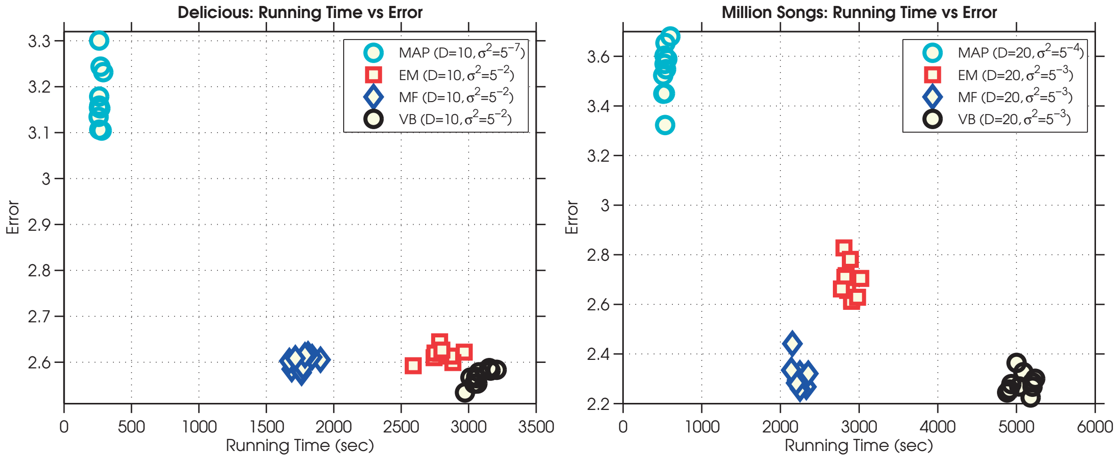
Figure 3.1 – Effect of prior variance: Strong regularization is necessary for MAP. While EM already benefits from the uncertainty regularization, it does not perform on par with the fully Bayesian methods.

3.1.4 Discussion

In this part we studied the variational Gaussian approximation for bi-linear models of count data under standard factorization assumptions. For the sake of efficiency, we



(a) LastFM (b) LastFM Tags



(c) Delicious (d) Million Songs

Figure 3.2 – Running time vs. error for various datasets: We plot running time (X) against error (Y) for different train-test splits. For both, lower is better. We set D to be the same for all methods (see text). The MF approximation seems to strike a good balance between speed and accuracy, which is consistent with the findings of Ilin and Raiko [2010] for Gaussian likelihoods.

introduced additional constraints that relax the original variational problem, enabling us to evaluate the bound in closed form. We demonstrated the benefits of probabilistic inference for robustness against overfitting and showed that even a fully factorized method enjoys this property at the same cost as point estimation.

A few open questions remain however: Algorithmically, we opted for an alternating scheme motivated by convexity structure of the problem. [Ilin and Raiko, 2010] advocate a gradient-based scheme with a different updating schedule as more effective, which, given the similarity of the setup, promises to be applicable for the Poisson model as well.

Another very interesting algorithmic question concerns the work of [Seeger and Bouchard, 2012] that develop a scalable algorithm based the analytic solution of the fully observed Gaussian case. They, too, introduce an additional relaxation by introducing local Gaussian approximations to the likelihoods. Their method, while iterative too, promises to be more effective and less prone to get stuck in poor local minima than alternating schemes. Since they also model count data, a comparison in terms of algorithmic convergence as well as quality of approximation would be interesting. Unfortunately, they do not support the exponential inverse link function, that our formulation uses.

Lastly, it would be interesting to compare a negative-binomial likelihood for overdispersed counts to the Poisson likelihood on this data set, e.g. using the method described in [Klami, 2014].

3.2 Pairwise Preferences

3.2.1 Overview and Related Work

In this part we address the problem of inferring latent preferences of users in $\mathbb{U} = [U]$ over a set of items $\mathbb{I} = [I]$ from the binary outcomes of pairwise comparisons. Such preference data is commonly modeled probabilistically using a framework similar to Gaussian Process classification [Chu and Ghahramani, 2005, Brochu et al., 2008, Birlutiu et al., 2010, Bonilla et al., 2010, Salimans et al., 2012, Houlsby et al., 2012]. Let $i \succ_u j$ denote the outcome of a comparison between items $i, j \in \mathbb{I}$ where user u preferred i over j. For each user $u \in \mathbb{U}$ we observe a list of pairs $\mathbb{O}_u$ where for each pair $(i, j) \in \mathbb{O}_u$ it holds that $i, j \in \mathbb{I}, i \neq j, i \succ_u j$. Thus, we allow for repetitions and make no assumptions about consistently representing a ranking over all items. We postulate a latent preference matrix $\boldsymbol{F} \in \mathbb{R}^{I \times U}$ with a column $\boldsymbol{f}_u$ representing the preferences or utilities of user u for the items in $\mathbb{I}$. Conditioned on the latent preferences, we model the outcome of a comparison as follows:

$$P(i \succ_u j \mid \boldsymbol{f}_u) = \Phi(f_{iu} - f_{ju}) \tag{3.35}$$

Here, $\Phi(\cdot)$ is the standard normal CDF $\Phi(x) = \int_{-\infty}^{x} \mathcal{N}(z \mid 0, 1)\,\mathrm{d}z$, which is the likelihood for probit regression, used e.g. for GP classification [Rasmussen and Williams, 2006].

Our utility model combines matrix factorization in the spirit of PPCA with a non-parametric content-based regression component:

$$\boldsymbol{u}_u \sim \mathcal{N}(0, \boldsymbol{I}) \tag{3.36}$$

$$\boldsymbol{g}_u \sim \mathrm{GP}(0, k(\cdot, \cdot)) \tag{3.37}$$

$$f_{iu} = \boldsymbol{v}_i^T \boldsymbol{u}_u + b_i + \sigma_s g_u(\boldsymbol{x}_i) \tag{3.38}$$

$\boldsymbol{V}\boldsymbol{u}_u$ is the bilinear part, $\boldsymbol{b}$ is a bias term, shared across users, and $g_u(\boldsymbol{x}) \sim \mathrm{GP}(0, k(\cdot, \cdot))$ is a random function drawn per user independently from $\boldsymbol{u}_u$ from a Gaussian Process with shared kernel function $k(\cdot, \cdot)$. We assume to have access to item meta-data in the form of real-valued K dimensional vectors $\boldsymbol{x}_i \in \mathbb{R}^K$ for all $i \in \mathbb{I}$ and denote the vector $\boldsymbol{g}_u = [g_u(\boldsymbol{x}_i)]_{i \in \mathbb{I}}$, the matrix $\boldsymbol{G} = [\boldsymbol{g}_u]_{u \in \mathbb{U}}$ and the kernel matrix $\boldsymbol{K}$ with entries $k_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. We consider $\boldsymbol{\theta} = \{\boldsymbol{V}, \boldsymbol{b}, \sigma_s\}$ parameters of the model[4] where σ_s balances the influence of the two components.

The marginal likelihood of the observations $\mathbb{O} = \{\mathbb{O}_u\}_{u \in \mathbb{U}}$ is

$$P(\mathbb{O} \mid \boldsymbol{\theta}) = \prod_{u \in \mathbb{U}} \int \int \prod_{(i,j) \in \mathbb{O}_u} P(i \succ_u j \mid \boldsymbol{u}_u, \boldsymbol{g}_u, \boldsymbol{\theta}) P(\boldsymbol{u}_u) P(\boldsymbol{g}_u)\,\mathrm{d}\boldsymbol{u}_u\,\mathrm{d}\boldsymbol{g}_u \tag{3.39}$$

---

[4] While kernel parameters can be learned as well, we set them manually using cross validation.

which we would like to maximize with respect to $\boldsymbol{\theta}$.

We proceed by discussing the difficulties of learning the parameters of such a model that arise due to the latent couplings induced by the likelihood. A straightforward EM algorithm expensive due to excessive resource requirements for the M-Step. We mitigate this problem by using ideas from Yu et al. [2009] that lead to a M-Step criterion akin to PPCA allowing us to update the parameters in closed form. This leads to a method that scales to much larger data sizes than most GP-based methods proposed in this area, while delivering strong performance on real-world datasets. We end with a discussion of the limitations of this method.

Related Work. As Brochu et al. [2008] mention, probabilistic models for discrete choice, subsuming pairwise comparisons, have been extensively studied in various areas, including sociology, psychology, economy and sports, as this formalism lends itself to express various problems in those fields. A fundamental problem, that these models address, is to aggregate such observations into global rankings of an underlying set of items. This has immediate applications in the fields mentioned above but also more recently in fields, such as in information retrieval [Liu, 2009]. Here, we see a convergence of methodology, where increasingly techniques developed in the area of machine learning [Burges et al., 2005, Chu and Ghahramani, 2005] are adopted due to the close relationship to classification problems as remarked above, enabling us to use inferences in these models to drive innovative applications [Herbrich et al., 2007, Brochu et al., 2008].

Gaussian processes have been previously used as a flexible model for the latent preferences inferred from pairwise comparisons [Chu and Ghahramani, 2005, Brochu et al., 2008]. Departing from global rankings are efforts to incorporate personalization and to exploit collaboration in the multi-user setting, a category into which our work falls as well. In the collaborative setting, GP based models have also been used: Birlutiu et al. [2010] use a utility model that only incorporates the GP part $\boldsymbol{g}_u$ of our model. They use the dual representation of the latent function parameterized by a vector of length I per user, tied together by a joint hierarchical prior, whose parameters are learned using EM. An E-Step therefore costs $O(I^3U)$. In order to establish the relationship between users and items, Bonilla et al. [2010] model the whole matrix $\boldsymbol{F}$ using a GP using a kernel with kronecker structure that multiplies together a user kernel with an item kernel. The authors employ a Laplace approximation, finding the posterior mode using a Newton method, that requires them to solve a dense $IU \times IU$ system, which costs $O(I^3U^3)$. Ignoring correlations between users, this model can be reduced to Birlutiu et al. [2010]. Houlsby et al. [2012] use an innovative formulation that combines ideas from bilinear models with GPs. Instead of considering GP realizations per user, they target the bilinear term $\boldsymbol{V}\boldsymbol{U}^T$. They directly operate in the space of observed pairs which is of dimension $P = |\cup_{u \in \mathbb{U}} \mathbb{O}_u| \leq \binom{I}{2}$, i.e. they model $\tilde{f}_{iju} = f_{iu} - f_{ju}$. The implied utility over pairs is a linear transformation of the original utility function over items, that in effect decouples the likelihood. Also, a GP prior on the columns of $\boldsymbol{V}$ implies a GP prior on the columns of $\tilde{\boldsymbol{V}} = \boldsymbol{B}\boldsymbol{V}$. Optionally, their method allows to impose GP priors over the columns of $\boldsymbol{U}$. They employ a fully factorized posterior approximation that is computed using a VB/EP hybrid approach. The algorithm is involved and reportedly requires additional measures for convergence while scaling[5] as $O(DP^3)$ and additionally $O(DU^3)$ with user features, which can be reduced by additional approximations. Finally, [Salimans et al., 2012] model utilities only using the matrix factorization term. They treat both matrices as latent variables, but employ a fully factorized posterior approximation and require local approximations of the non-Gaussian likelihoods. Our model can be seen as combining aspects of [Salimans et al., 2012] with [Birlutiu et al., 2010]. As is, our model does not support user features and estimates the matrix $\boldsymbol{V}$. While opting against a mean-field approximation, the complexity of our method is somewhat more benign and given by $O(\sum_{u\in\mathbb{U}} |\mathbb{O}_u|^3)$ which in the worst case is equivalent to $O(UP^3)$. We show, however, that a stochastic version of our algorithm is effective, where we sample a maximum number of pairs per user. Furthermore, the inference in our E-Step converges quickly and reliably using standard parallel EP for GP classification.

3.2.2 Inference and Learning for Collaborative Preference Learning

In this part, we start by introducing learning in our setting as a variant of PPCA, highlighting the computational challenges introduced by the couplings due to the pairwise comparisons for the M-Step. With this motivation, we reformulate the generative model, which leads to a decoupling of the likelihood and a shift of the parameters into a newly defined pseudo-prior. This formulation greatly simplifies the M-Step, that can now be performed more efficiently by noticing similarities to the analytic ML soluton of PPCA in the fully observed case.

We begin by introducing additional notation.

3.2.2.1 Preliminaries

Let $P_u = |\mathbb{O}_u|$ denote the number of pairs observed for each user. We write differences between entries of $\boldsymbol{f}_u$ corresponding to multiple pairs in $\mathbb{O}_u$ using matrix notation as follows. Let $\omega_u : \mathbb{O}_u \mapsto [P_u]$ be a bijection. Then we write

$$\bar{\boldsymbol{f}}_u = \boldsymbol{B}_u \boldsymbol{f}_u \tag{3.40}$$

where $\boldsymbol{B}_u$ is such for any $(i, j) \in \mathbb{O}_u$, the k-th entry of $\bar{\boldsymbol{f}}_u$ is

$$\bar{f}_{\omega_u(i,j)} = f_{iu} - f_{ju} = (\boldsymbol{\delta}_i - \boldsymbol{\delta}_j)^T \boldsymbol{f}_u \tag{3.41}$$

---

[5] According to the authors. Probably, this can be reduced to $O(DI^3)$) due to the structure of the problem.

where $k = \omega_u(i, j)$. Hence, we can rewrite the probit likelihood

$$P(i \succ_u j \mid \boldsymbol{f}_u) = \Phi(f_{iu} - f_{ju}) = \Phi(\bar{f}_{ku}). \tag{3.42}$$

From this perspective, and as mentioned e.g. by Houlsby et al. [2012], we recover a standard linear probit-classification model, with pseudo-features $\boldsymbol{B}_u$ and weight vector $\boldsymbol{f}_u$.

One way to represent the pseudo-features $\boldsymbol{B}_u$ is in form of a sparse matrix with $2P_u$ non-zero entries. See Algorithm 5 for pseudo-code.

Algorithm 5 Construct Pseudo-feautres $\boldsymbol{B}_u$

```
Require: $\mathbb{O}_u$
Ensure: Returns $\boldsymbol{B}_u$
rowIdx ← []
colIdx ← []
values ← []
$r = 0$
for $(i, j) \in \mathbb{O}_u$ do
    append(rowIdx, r)
    append(colIdx, i)
    append(values, 1)
    append(rowIdx, r)
    append(colIdx, j)
    append(values, -1)
    $r \leftarrow r + 1$
end for
return sparse(rowIdx, colIdx, values)
```

3.2.2.2 PPCA for Coupled Likelihoods

To facilitate the discussion, we consider a simplified PPCA [Tipping and Bishop, 1999] model.

In PPCA, data is assumed to be generated as

$$\boldsymbol{u}_u \sim \mathcal{N}(0, \boldsymbol{I}_D) \tag{3.43}$$

$$\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \boldsymbol{I}_I) \tag{3.44}$$

$$\boldsymbol{y}_u = \boldsymbol{V}\boldsymbol{u}_u + \sigma\boldsymbol{\varepsilon} \quad \Rightarrow \quad P(\boldsymbol{y}_u \mid \boldsymbol{u}_u, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{y}_u \mid \boldsymbol{V}\boldsymbol{u}_u, \sigma^2\boldsymbol{I}_I) \tag{3.45}$$

with parameters $\boldsymbol{\theta} = \left\{\boldsymbol{V}, \sigma^2\right\}$. The ML objective is

$$\begin{aligned}\sum_u \log P(\boldsymbol{y}_u \mid \boldsymbol{\theta}) &= \sum_u \log \mathcal{N}(\boldsymbol{y}_u \mid 0, \boldsymbol{C}) && (3.46)\\ &\propto -\log|\boldsymbol{C}| - \operatorname{tr}\left(\boldsymbol{C}^{-1}\hat{\boldsymbol{\Sigma}}\right) && (3.47)\end{aligned}$$

where $\boldsymbol{C} = \boldsymbol{V}\boldsymbol{V}^T + \sigma^2 \boldsymbol{I}_I$ and $\hat{\boldsymbol{\Sigma}}$ is the empirical covariance. Tipping and Bishop [1999] show that the maximum likelihood solution is given by

$$\begin{aligned}\boldsymbol{V}_{ML} &= \boldsymbol{Q}_D(\operatorname{diag}(\boldsymbol{\lambda}_D) - \sigma^2 \boldsymbol{I}_D)^{\frac{1}{2}}\boldsymbol{R} && (3.48)\\ \sigma^2_{ML} &= \frac{1}{I-D}\sum_{i=D+1}^{I}\lambda_i && (3.49)\end{aligned}$$

where $\hat{\boldsymbol{\Sigma}} = \boldsymbol{Q}\operatorname{diag}(\boldsymbol{\lambda})\boldsymbol{Q}^T$, subscript D denotes the leading D eigenvectors and values and $\boldsymbol{R}$ is an arbitrary orthonormal matrix.

For partially missing data, an iterative EM procedure is necessary. In a slight abuse of notation, we override the index set of observed entries $\mathbb{O}_u \subseteq [I]$. We focus on the M-Step update of $\boldsymbol{V}$, assuming for the moment σ^2 to be known. Denote by $\boldsymbol{S}_u = \mathrm{E}\left[\boldsymbol{u}_u\boldsymbol{u}_u^T\right]$, $\boldsymbol{L}_{\boldsymbol{u}}$ its Cholesky factor, $\boldsymbol{l}_{ud} = \boldsymbol{L}_{\boldsymbol{u}}\boldsymbol{\delta}_d$ and $\boldsymbol{s}_u = \mathrm{E}\left[\boldsymbol{u}_u\right]$ the previously gathered E-Step statistics. In the M-Step, we seek to minimize

$$\begin{aligned}\sum_u \mathrm{E}\left[\|\boldsymbol{y}_{\mathbb{O}_u} - \boldsymbol{I}_{\mathbb{O}_u}\boldsymbol{V}\boldsymbol{u}_u\|^2\right] &\doteq \sum_u \mathrm{E}\left[\boldsymbol{u}_u^T\boldsymbol{V}^T\boldsymbol{I}_{\mathbb{O}_u}^T\boldsymbol{I}_{\mathbb{O}_u}\boldsymbol{V}\boldsymbol{u}_u\right] - 2\boldsymbol{s}_u^T\boldsymbol{V}^T\boldsymbol{I}_{\mathbb{O}_u}^T\boldsymbol{y}_{\mathbb{O}_u} && (3.50)\\ &= \sum_u \mathrm{E}\left[\operatorname{tr}\left(\boldsymbol{V}^T\boldsymbol{I}_{\mathbb{O}_u}^T\boldsymbol{I}_{\mathbb{O}_u}\boldsymbol{V}\boldsymbol{u}_u\boldsymbol{u}_u^T\right)\right] - 2\boldsymbol{s}_u^T\boldsymbol{V}^T\boldsymbol{I}_{\mathbb{O}_u}^T\boldsymbol{y}_{\mathbb{O}_u} && (3.51)\\ &= \sum_u \operatorname{tr}\left(\boldsymbol{V}^T\boldsymbol{I}_{\mathbb{O}_u}^T\boldsymbol{I}_{\mathbb{O}_u}\boldsymbol{V}\boldsymbol{S}_u\right) - 2\boldsymbol{s}_u^T\boldsymbol{V}^T\boldsymbol{I}_{\mathbb{O}_u}^T\boldsymbol{y}_{\mathbb{O}_u} && (3.52)\end{aligned}$$

This is a quadratic function in $\boldsymbol{V}$, which becomes obvious by rewriting it in terms of the row-major vectorization of $\boldsymbol{V}$, $\boldsymbol{v} = \operatorname{vec}_r(\boldsymbol{V})$, and then applying the equality $\boldsymbol{V}\boldsymbol{w} = \left(\boldsymbol{I}_I \otimes \boldsymbol{w}^T\right)\boldsymbol{v}$. Then the linear term is

$$\sum_u \boldsymbol{s}_u^T\boldsymbol{V}^T\boldsymbol{I}_{\mathbb{O}_u}^T\boldsymbol{y}_{\mathbb{O}_u} = \boldsymbol{v}^T\sum_u (\boldsymbol{I}_I \otimes \boldsymbol{s}_u)\boldsymbol{I}_{\mathbb{O}_u}^T\boldsymbol{y}_{\mathbb{O}_u} \qquad (3.53)$$

The quadratic term can be rewritten as

$$\sum_u \operatorname{tr}\left(\boldsymbol{V}^T \boldsymbol{I}_{\mathbb{O}_u}^T \boldsymbol{I}_{\mathbb{O}_u} \boldsymbol{V} \boldsymbol{L_u} \boldsymbol{L_u}^T\right) = \sum_u \operatorname{tr}\left(\boldsymbol{L_u}^T \boldsymbol{V}^T \boldsymbol{I}_{\mathbb{O}_u}^T \boldsymbol{I}_{\mathbb{O}_u} \boldsymbol{V} \boldsymbol{L_u}\right) \tag{3.54}$$

$$= \sum_u \sum_{d=1}^{D} \left(\boldsymbol{\delta}_d^T \boldsymbol{L_u}^T \boldsymbol{V}^T \boldsymbol{I}_{\mathbb{O}_u}^T \boldsymbol{I}_{\mathbb{O}_u} \boldsymbol{V} \boldsymbol{L_u} \boldsymbol{\delta}_d\right) \tag{3.55}$$

$$= \sum_u \sum_{d=1}^{D} \left(\boldsymbol{l}_{ud}^T \boldsymbol{V}^T \boldsymbol{I}_{\mathbb{O}_u}^T \boldsymbol{I}_{\mathbb{O}_u} \boldsymbol{V} \boldsymbol{l}_{ud}\right) \tag{3.56}$$

$$= \sum_u \sum_{d=1}^{D} \left(\boldsymbol{v}^T \left(\boldsymbol{I}_I \otimes \boldsymbol{l}_{ud}\right) \boldsymbol{I}_{\mathbb{O}_u}^T \boldsymbol{I}_{\mathbb{O}_u} \left(\boldsymbol{I}_I \otimes \boldsymbol{l}_{ud}^T\right) \boldsymbol{v}\right) \tag{3.57}$$

$$= \boldsymbol{v}^T \left(\sum_u \sum_{d=1}^{D} \left(\left(\boldsymbol{I}_I \otimes \boldsymbol{l}_{ud}\right) \boldsymbol{I}_{\mathbb{O}_u}^T \boldsymbol{I}_{\mathbb{O}_u} \left(\boldsymbol{I}_I \otimes \boldsymbol{l}_{ud}^T\right)\right)\right) \boldsymbol{v} \tag{3.58}$$

$$= \boldsymbol{v}^T \left(\sum_u \sum_{d=1}^{D} \left(\boldsymbol{I}_{\mathbb{O}_u}^T \otimes \boldsymbol{l}_{ud}\right) \left(\boldsymbol{I}_{\mathbb{O}_u} \otimes \boldsymbol{l}_{ud}^T\right)\right) \boldsymbol{v} \tag{3.59}$$

$$= \boldsymbol{v}^T \left(\sum_u \sum_{d=1}^{D} \left(\boldsymbol{I}_{\mathbb{O}_u}^T \boldsymbol{I}_{\mathbb{O}_u} \otimes \boldsymbol{l}_{ud} \boldsymbol{l}_{ud}^T\right)\right) \boldsymbol{v} \tag{3.60}$$

$$= \boldsymbol{v}^T \left(\sum_u \left(\boldsymbol{I}_{\mathbb{O}_u}^T \boldsymbol{I}_{\mathbb{O}_u} \otimes \boldsymbol{L_u} \left(\sum_{d=1}^{D} \boldsymbol{\delta}_d \boldsymbol{\delta}_d^T\right) \boldsymbol{L_u}^T\right)\right) \boldsymbol{v} \tag{3.61}$$

$$= \boldsymbol{v}^T \left(\sum_u \boldsymbol{I}_{\mathbb{O}_u}^T \boldsymbol{I}_{\mathbb{O}_u} \otimes \boldsymbol{S}_u\right) \boldsymbol{v} \tag{3.62}$$

The matrix in the middle has a block diagonal structure, that enables us to solve independently for each row of $\boldsymbol{V}$ a $D \times D$ linear system.

We return to the pairwise setting, where we can perform the same analysis for a coupled likelihood, again for the sake of simplicity using Gaussian likelihoods. We essentially obtain the derivation by substituting $\boldsymbol{I}_{\mathbb{O}_u}$ with $\boldsymbol{B}_u$. The system matrix therefore becomes

$$\left(\sum_u \boldsymbol{B}_u^T \boldsymbol{B}_u \otimes \boldsymbol{S}_u\right) \tag{3.63}$$

As we have seen before, structure in this matrix, i.e. its block-sparsity pattern, leads to an efficient M-Step. Based on the following proposition, we argue that due to the couplings induced by the pairwise comparisons, it is unlikely to have a sparse linear system to work with.

Proposition 4. *The matrix* $\boldsymbol{B}_u^T \boldsymbol{B}_u$ *has a off-diagonal non-zero entry at* (i, j) *for all*

$(i, j) \in \mathbb{O}_u$.

Proof. From (3.41) we know that rows of $\boldsymbol{B}_u$ are differences of indicator vectors, i.e. for each $(i, j) \in \mathbb{O}_u$ there is a row in $\boldsymbol{B}_u$ that is equal to $(\boldsymbol{\delta}_i - \boldsymbol{\delta}_j)^T$ Therefore, $\boldsymbol{w}_{uk_1} := \boldsymbol{B}_u \boldsymbol{\delta}_{k_1}$ will be a vector of length P_u, with a non-zero at every $k = \omega_u(i, j)$ for all $(i, j) \in \mathbb{O}_u$ where $i = k_1$ or $j = k_1$. Thus, for any k_2 such that $(k_1, k_2) \in \mathbb{O}_u$ or $(k_2, k_1) \in \mathbb{O}_u$, we have $\boldsymbol{w}_{k_1}^T \boldsymbol{w}_{k_2} \neq 0$. Moreover, these off-diagonal entries will be negative, since we have differences of indicators, so that there will be no cancellation. □

The non-zero pattern of (3.63) is thus data-dependent. Moreover, summing over all users makes it unlikely for an useful sparsity pattern to emerge. Solving the resulting system directly is difficult, since the system matrix is of size $ID \times ID$. The same holds for gradient-based methods as evaluating the gradient involves matrix-vector multiplications with this matrix. Even coordinate descent methods, that apply due to strong convexity of the objective, might be expensive involving repeated passes over all users to recompute columns of the matrix. Similar insights in the context of non-probabilistic pairwise matrix factorization were presented by Takács and Tikk [2012].

3.2.2.3 Our Approach: EM Algorithm under Reparameterization

We show the graphical model corresponding to our preference model in Figure 3.3. It illustrates the coupling of the parameters through the likelihood. Next, we discuss the implications of slightly reformulating the model. Recall the utility model in (3.36)

$$\boldsymbol{u}_u \sim \mathcal{N}(0, \boldsymbol{I}) \tag{3.64}$$
$$\boldsymbol{g}_u \sim \mathrm{GP}(0, k(\cdot, \cdot)) \tag{3.65}$$
$$f_{iu} = \boldsymbol{v}_i^T \boldsymbol{u}_u + b_i + \sigma_s g_u(\boldsymbol{x}_i) \tag{3.66}$$

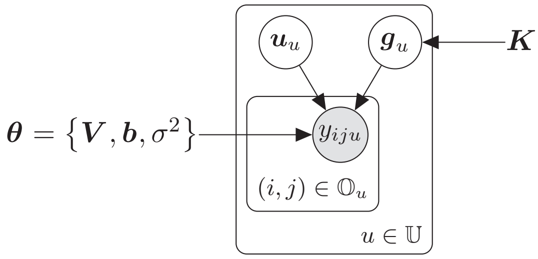

$\boldsymbol{u}_u$
$\boldsymbol{g}_u$
$\boldsymbol{K}$
$\boldsymbol{\theta} = \{\boldsymbol{V}, \boldsymbol{b}, \sigma^2\}$
y_{iju}
$(i, j) \in \mathbb{O}_u$
$u \in \mathbb{U}$

Figure 3.3 – Original Preference Model

Instead of working explicitly with $\boldsymbol{u}_u$ and $\boldsymbol{g}_u$, we could introduce a pseudo-prior and work directly with $\boldsymbol{f}_u$, which is Gaussian as well, distributed as

$$P(\boldsymbol{f}_u \mid \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{f}_u \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{3.67}$$

with parameters

$$\boldsymbol{\mu} = \boldsymbol{b} \qquad \boldsymbol{\Sigma} = \boldsymbol{V}\boldsymbol{V}^T + \sigma^2\boldsymbol{K} \tag{3.68}$$

The corresponding graphical model is depicted in Figure 3.4 and corresponds to marginalizing out latent variables in PPCA setting [Tipping and Bishop, 1999]. In the remainder of this part we develop our modified EM algorithm, an overview of which can be found in Algorithm 6. The algorithm can be summarized as follows:

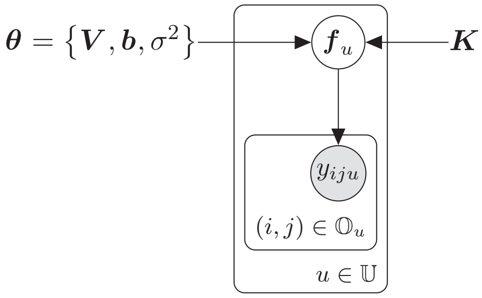


Figure 3.4 – Reparameterized Preference Model

E-Step Inference in the E-Step decomposes over the users and can be seen as independent linear probit regression problems with pseudo-features $\boldsymbol{B}_u$. Since $\boldsymbol{f}_u$ is Gaussian, this can be cast into standard GP classification [Rasmussen and Williams, 2006]. As can be seen in Line 14 of Algorithm 6, the inference problems are of size P_u. We run parallel-updating Expectation Propagation, such that the dominating cost is $O(P_u^3)$ for a total of $O(\sum_{u \in \mathbb{U}} P_u^3)$. We gather statistics $\boldsymbol{S}, \boldsymbol{s}$ as described in detail below.

M-Step We show that in this formulation, the M-Step can be reduced to a PPCA M-Step [Tipping and Bishop, 1999]. It requires the accumulation of E-Step statistics of size $O(I^2)$. Instead of a direct eigendecomposition at cubic cost, we can use iterative methods, that leads to a cost of $O(DI^2)$.

We proceed by first recovering PPCA in the M-Step in 3.2.2.4 and then describing which statistics are necessary for the update in 3.2.2.5.

Algorithm 6 EM for Collaborative Preference Learning

```
Require: $\boldsymbol{K}, \mathbb{O} = \{\mathbb{O}_u\}$
Ensure: Updated parameters $\boldsymbol{\theta}$
// Initialization
$\boldsymbol{L}_K \leftarrow$ chol($\boldsymbol{K}$)
initialize $\bar{\boldsymbol{V}}, \boldsymbol{b}, \sigma^2$
$\boldsymbol{V} \leftarrow \boldsymbol{L}_K \bar{\boldsymbol{V}}$
$\boldsymbol{S} \leftarrow$ zeros($I, I$)
$\boldsymbol{s} \leftarrow$ zeros($I$)
// EM-Algorithm
repeat
    // E-Step (Section 3.2.2.5)
    for $u \in \mathbb{U}$ do
        $\boldsymbol{B}_u \leftarrow$ pseudoFeatures($\mathbb{O}_u$) (Algorithm 5)
        $\bar{\boldsymbol{\mu}} \leftarrow \boldsymbol{B}_u \boldsymbol{b}$
        $\bar{\boldsymbol{\Sigma}} \leftarrow \boldsymbol{B}_u \boldsymbol{V} \boldsymbol{V}^T \boldsymbol{B}_u^T + \sigma^2 \boldsymbol{B}_u \boldsymbol{L}_K \boldsymbol{L}_K^T \boldsymbol{B}_u^T$
        $[\boldsymbol{\pi}_u, \boldsymbol{\beta}_u] \leftarrow$ inferGP($\bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\Sigma}}$, likelihood="probit") // Equation (3.88)
        $[\boldsymbol{E}_u, \boldsymbol{e}_u] \leftarrow$ stats($\boldsymbol{\pi}_u, \boldsymbol{\beta}_u, \bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\Sigma}}$) // Equations (3.94) and (3.101)
        $\boldsymbol{S} \leftarrow \boldsymbol{S} + \frac{1}{U} \boldsymbol{B}_u^T (\boldsymbol{E}_u - \boldsymbol{e}_u \boldsymbol{e}_u^T) \boldsymbol{B}_u$ // Equation (3.113)
        $\boldsymbol{s} \leftarrow \boldsymbol{s} + \frac{1}{U} \boldsymbol{B}_u^T \boldsymbol{e}_u$ // Equation (3.104)
    end for
    // M-Step (Section 3.2.2.4)
    $\left[\bar{\boldsymbol{V}}, \sigma^2\right] \leftarrow$ PPCAeigs($\boldsymbol{S}, \bar{\boldsymbol{V}}, \sigma^2, \boldsymbol{L}_K$)
    $\boldsymbol{b} \leftarrow \boldsymbol{b} + \boldsymbol{V} \left(\boldsymbol{V}^T \boldsymbol{s}\right) + \sigma^2 \boldsymbol{L}_K \left(\boldsymbol{L}_K^T \boldsymbol{s}\right)$
    $\boldsymbol{V} \leftarrow \boldsymbol{L}_K \bar{\boldsymbol{V}}$
until Convergence
```

3.2.2.4 M-Step

At the end of the E-Step, we have computed an approximate Gaussian posterior

$$Q(\boldsymbol{F}) = \prod_{u \in \mathbb{U}} \mathcal{N}(\boldsymbol{f}_u \mid \boldsymbol{\xi}_u, \boldsymbol{\Xi}_u) \tag{3.69}$$

to which we will come back later.

Using (3.27), the M-Step objective can then be written as

$$\mathcal{L}(\boldsymbol{\theta}) = \mathrm{E}_Q\left[\log \frac{P(\mathbb{O} \mid \boldsymbol{F})P(\boldsymbol{F} \mid \boldsymbol{\theta})}{Q(\boldsymbol{F})}\right] \tag{3.70}$$

$$\doteq -\mathrm{E}_Q\left[\log \frac{Q(\boldsymbol{F})}{P(\boldsymbol{F} \mid \boldsymbol{\theta})}\right] \tag{3.71}$$

$$= -\sum_{u\in\mathbb{U}} D_{\mathrm{KL}}\left[Q(\boldsymbol{f}_u)\|P(\boldsymbol{f}_u \mid \boldsymbol{\theta})\right] \tag{3.72}$$

$$= -\frac{1}{2}\sum_{u\in\mathbb{U}}\left(\mathrm{tr}\left(\boldsymbol{\Sigma}^{-1}\boldsymbol{\Xi}_u\right) + \left(\boldsymbol{\mu}-\boldsymbol{\xi}_u\right)^T\boldsymbol{\Sigma}^{-1}\left(\boldsymbol{\mu}-\boldsymbol{\xi}_u\right) + \log\frac{|\boldsymbol{\Sigma}|}{|\boldsymbol{\Xi}_u|} - I\right) \tag{3.73}$$

$$\doteq -\frac{1}{2}\sum_{u\in\mathbb{U}}\left(\mathrm{tr}\left(\boldsymbol{\Sigma}^{-1}\boldsymbol{\Xi}_u\right) + \left(\boldsymbol{\mu}-\boldsymbol{\xi}_u\right)^T\boldsymbol{\Sigma}^{-1}\left(\boldsymbol{\mu}-\boldsymbol{\xi}_u\right) + \log|\boldsymbol{\Sigma}|\right) \tag{3.74}$$

$$\propto -\,\mathrm{tr}\left(\boldsymbol{\Sigma}^{-1}\frac{1}{U}\sum_{u\in\mathbb{U}}\left(\boldsymbol{\Xi}_u + \left(\boldsymbol{\mu}-\boldsymbol{\xi}_u\right)\left(\boldsymbol{\mu}-\boldsymbol{\xi}_u\right)^T\right)\right) - \log|\boldsymbol{\Sigma}| \tag{3.75}$$

$$= -\,\mathrm{tr}\left(\boldsymbol{\Sigma}^{-1}\boldsymbol{C}\right) - \log|\boldsymbol{\Sigma}| \tag{3.76}$$

where we defined $\boldsymbol{C} := \frac{1}{U}\sum_{u\in\mathbb{U}}\left(\boldsymbol{\Xi}_u + \left(\boldsymbol{\mu}-\boldsymbol{\xi}_u\right)\left(\boldsymbol{\mu}-\boldsymbol{\xi}_u\right)^T\right)$.

We first state the algorithm to give an overview of the method and derive each step.

There are two challenges associated with this formulation:

1. Recall the definitions in (3.67), that show the functional dependence of $\boldsymbol{\Sigma}$ on $\boldsymbol{\theta}$. While continuously differentiable, optimizing it naively may be cumbersome. We address this issue by recognizing in (3.76) the PPCA M-Step objective (3.46). Given the similarity of the generative model of PPCA to our utility model, we show how to make use of the analytic solution (3.48) and (3.49) to solve our M-Step.

2. A major obstacle for doing so is the accumulation of sufficient statistics to perform the M-Step: The straightforward way to obtain

   $$\boldsymbol{C} = \frac{1}{U}\sum_{u\in\mathbb{U}}\left(\boldsymbol{\Xi}_u + \left(\boldsymbol{\mu}-\boldsymbol{\xi}_u\right)\left(\boldsymbol{\mu}-\boldsymbol{\xi}_u\right)^T\right)$$

   from $Q(\boldsymbol{F})$ costs $O(I^3U + I^2U)$. Using ideas from [Yu et al., 2009], we show how to use the structure in the problem to potentially speed this computation up. The main idea is to express both the dimensionality of the E-Step problem as well as the required statistics in terms of the number of pairs. While the number of pairs can be large in the worst case, this formulation introduces a single tuning parameter to reduce the cost of the algorithm, by running a stochastic version of EM by only presenting a random subset of the pairs to the algorithm.

Reduction to PPCA. Recall that the pseudo-prior on $\boldsymbol{f}_u$ is $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ parameterized in terms of $\boldsymbol{\theta} = \left\{\boldsymbol{V}, \boldsymbol{b}, \sigma^2\right\}$ as

$$\boldsymbol{\mu} = \boldsymbol{b} \qquad \boldsymbol{\Sigma} = \boldsymbol{V}\boldsymbol{V}^T + \sigma^2\boldsymbol{K} \tag{3.77}$$

Instead of an isotropic noise term $\boldsymbol{\varepsilon}$ in (3.43) our utility model incorporates a Gaussian process term $\boldsymbol{g}_u$ with kernel matrix $\boldsymbol{K}$ (3.64)[6]. Let $\boldsymbol{K} = \boldsymbol{L}_K\boldsymbol{L}_K^T$ be $\boldsymbol{K}$'s Cholesky factorization. Define

$$\begin{aligned} \bar{\boldsymbol{\Sigma}} &:= \boldsymbol{L}_K^{-1}\boldsymbol{\Sigma}\boldsymbol{L}_K^{-T} && (3.78)\\ &= \boldsymbol{L}_K^{-1}\boldsymbol{V}\boldsymbol{V}^T\boldsymbol{L}_K^{-T} + \sigma^2\boldsymbol{L}_K^{-1}\boldsymbol{K}\boldsymbol{L}_K^{-T} && (3.79)\\ &= \bar{\boldsymbol{V}}\bar{\boldsymbol{V}}^T + \sigma^2\boldsymbol{I} && (3.80) \end{aligned}$$

which is in the desired form. Substituting $\boldsymbol{\Sigma}$ in (3.76) by

$$\boldsymbol{\Sigma} = \boldsymbol{L}_K\bar{\boldsymbol{\Sigma}}\boldsymbol{L}_K^T \tag{3.81}$$

we get the following objective

$$-\operatorname{tr}\left(\bar{\boldsymbol{\Sigma}}^{-1}\boldsymbol{L}_K^{-1}\boldsymbol{C}\boldsymbol{L}_K^{-T}\right) - \log\left|\boldsymbol{L}_K\bar{\boldsymbol{\Sigma}}\boldsymbol{L}_K^T\right| \doteq -\operatorname{tr}\left(\bar{\boldsymbol{\Sigma}}^{-1}\bar{\boldsymbol{C}}\right) - \log\left|\bar{\boldsymbol{\Sigma}}\right| \tag{3.82}$$

Hence, we can compute $\bar{\boldsymbol{V}}$ and σ^2 using (3.48) and (3.49) but with an eigendecomposition of $\bar{\boldsymbol{C}} := \boldsymbol{L}_K^{-1}\boldsymbol{C}\boldsymbol{L}_K^{-T}$. We recover $\boldsymbol{V}$ using

$$\boldsymbol{V} = \boldsymbol{L}_K\bar{\boldsymbol{V}} \tag{3.83}$$

Since we are only interested in the D leading eigenvectors and -values, we can use iterative Krylov methods that require D matrix-vector multiplications with $\bar{\boldsymbol{C}}$ per round. These methods are standard routines shipped with all widely available numerical libraries. We describe a more efficient implementation of multiplication with $\bar{\boldsymbol{C}}$ in (3.2.2.6).

The M-Step update of $\boldsymbol{b}$ can be derived in the same way as for normal PPCA using pseudo-observations $\boldsymbol{\xi}_u$. For the update, consider (3.74), which is a sum of quadratic terms in $\boldsymbol{\mu} = \boldsymbol{b}$. We maximize it by finding the minimizer of:

$$\sum_{u\in\mathbb{U}} \left(\boldsymbol{\mu} - \boldsymbol{\xi}_u\right)^T \boldsymbol{\Sigma}^{-1} \left(\boldsymbol{\mu} - \boldsymbol{\xi}_u\right) \tag{3.84}$$

---

[6] $\boldsymbol{K}$ is assumed to be invertible, which we ensure as in [Neal, 1997] by adding a $\alpha\boldsymbol{I}$

The gradient is

$$\nabla_{\boldsymbol{b}} \sum_{u \in \mathbb{U}} (\boldsymbol{\mu} - \boldsymbol{\xi}_u)^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu} - \boldsymbol{\xi}_u) = \sum_{u \in \mathbb{U}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu} - \boldsymbol{\xi}_u) \tag{3.85}$$

$$= \boldsymbol{\Sigma}^{-1} \left(U \boldsymbol{\mu} - \sum_{u \in \mathbb{U}} \boldsymbol{\xi}_u \right) \overset{!}{=} 0 \tag{3.86}$$

leading to the following update:

$$\boldsymbol{b} \leftarrow \frac{1}{U} \sum_{u \in \mathbb{U}} \boldsymbol{\xi}_u \tag{3.87}$$

3.2.2.5 E-Step

In the E-Step, we compute a posterior approximation $Q(\boldsymbol{f}_u)$ for each $u \in \mathbb{U}$ using Expectation Propagation. The resulting posterior has the following form[7]

$$\begin{aligned}
Q(\boldsymbol{f}_u) &= \mathcal{N}(\boldsymbol{f}_u \mid \boldsymbol{\xi}_u, \boldsymbol{\Xi}_u) && (3.88)\\
&\propto \prod_{(i,j) \in \mathbb{O}_u} \mathcal{N}\left(\bar{f}_{\omega_u(i,j)} \mid \tilde{\mu}_{\omega_u(i,j)}, \tilde{\sigma}^2_{\omega_u(i,j)} \right) \mathcal{N}(\boldsymbol{f}_u \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) && (3.89)\\
&\propto e^{\boldsymbol{f}_u^T \boldsymbol{B}_u^T \boldsymbol{\beta}_u - \frac{1}{2} \boldsymbol{f}_u^T \boldsymbol{B}_u^T \operatorname{diag}(\boldsymbol{\pi}_u) \boldsymbol{B}_u \boldsymbol{f}_u} \mathcal{N}(\boldsymbol{f}_u \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) && (3.90)\\
\Rightarrow \quad \boldsymbol{\xi}_u &= \boldsymbol{\Xi}_u \left(\boldsymbol{B}_u^T \boldsymbol{\beta}_u + \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} \right) \qquad \boldsymbol{\Xi}_u = \left(\boldsymbol{\Sigma}^{-1} + \boldsymbol{B}_u^T \operatorname{diag}(\boldsymbol{\pi}_u) \boldsymbol{B}_u \right)^{-1} && (3.91)
\end{aligned}$$

where $\boldsymbol{\pi}_u = \left[\tilde{\sigma}^{-2}_{\omega_u(i,j)} \right]_{(i,j) \in \mathbb{O}_u}$ and $\boldsymbol{\beta}_u = \left[\tilde{\mu}_{\omega_u(i,j)} \tilde{\sigma}^{-2}_{\omega_u(i,j)} \right]_{(i,j) \in \mathbb{O}_u}$ are the natural parameters of the approximate Gaussians, that are computed using EP. We rewrite the posterior parameters, where the user specific computations are expressed in terms of the number of pairs per user.

Using the Woodbury formula for matrix inversion, we can rewrite $\boldsymbol{\Xi}_u$ as

$$\begin{aligned}
\boldsymbol{\Xi}_u &= \left(\boldsymbol{\Sigma}^{-1} + \boldsymbol{B}_u^T \operatorname{diag}(\boldsymbol{\pi}_u) \boldsymbol{B}_u \right)^{-1} && (3.92)\\
&= \boldsymbol{\Sigma} - \boldsymbol{\Sigma} \boldsymbol{B}_u^T \left(\operatorname{diag}(\boldsymbol{\pi}_u)^{-1} + \boldsymbol{B}_u \boldsymbol{\Sigma} \boldsymbol{B}_u^T \right)^{-1} \boldsymbol{B}_u \boldsymbol{\Sigma} && (3.93)\\
&= \boldsymbol{\Sigma} - \boldsymbol{\Sigma} \boldsymbol{B}_u^T \boldsymbol{E}_u \boldsymbol{B}_u \boldsymbol{\Sigma} && (3.94)
\end{aligned}$$

i.e. we define $\boldsymbol{E}_u^{-1} := \left(\operatorname{diag}(\boldsymbol{\pi}_u)^{-1} + \boldsymbol{B}_u \boldsymbol{\Sigma} \boldsymbol{B}_u^T \right) \in \mathbb{R}^{P_u \times P_u}$.

---

[7] The same holds for a Variational Gaussian approximation [Opper and Archambeau, 2009] and local bounding methods.

Next, we can re-write $\boldsymbol{\xi}_u$ as follows.

$$
\begin{aligned}
\boldsymbol{\xi}_u &= \boldsymbol{\Xi}_u \left(\boldsymbol{B}_u^T \boldsymbol{\beta}_u + \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right) && (3.95)\\
&= \left(\boldsymbol{\Sigma} - \boldsymbol{\Sigma}\boldsymbol{B}_u^T \boldsymbol{E}_u \boldsymbol{B}_u \boldsymbol{\Sigma}\right)\left(\boldsymbol{B}_u^T \boldsymbol{\beta}_u + \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right) && (3.96)\\
&= \boldsymbol{\Sigma}\boldsymbol{B}_u^T\boldsymbol{\beta}_u - \boldsymbol{\Sigma}\boldsymbol{B}_u^T\boldsymbol{E}_u\boldsymbol{B}_u\boldsymbol{\Sigma}\boldsymbol{B}_u^T\boldsymbol{\beta}_u + \boldsymbol{\mu} - \boldsymbol{\Sigma}\boldsymbol{B}_u^T\boldsymbol{E}_u\boldsymbol{B}_u\boldsymbol{\mu} && (3.97)\\
&= \boldsymbol{\mu} + \boldsymbol{\Sigma}\boldsymbol{B}_u^T\boldsymbol{E}_u\left(\boldsymbol{E}_u^{-1}\boldsymbol{\beta}_u - \boldsymbol{B}_u\boldsymbol{\Sigma}\boldsymbol{B}_u^T\boldsymbol{\beta}_u - \boldsymbol{B}_u\boldsymbol{\mu}\right) && (3.98)\\
&= \boldsymbol{\mu} + \boldsymbol{\Sigma}\boldsymbol{B}_u^T\boldsymbol{E}_u\Big(\operatorname{diag}\left(\boldsymbol{\pi}_u\right)^{-1}\boldsymbol{\beta}_u + \boldsymbol{B}_u\boldsymbol{\Sigma}\boldsymbol{B}_u^T\boldsymbol{\beta}_u - \boldsymbol{B}_u\boldsymbol{\Sigma}\boldsymbol{B}_u^T\boldsymbol{\beta}_u - \boldsymbol{B}_u\boldsymbol{\mu}\Big) && (3.99)\\
&= \boldsymbol{\mu} + \boldsymbol{\Sigma}\boldsymbol{B}_u^T\boldsymbol{E}_u\Big(\operatorname{diag}\left(\boldsymbol{\pi}_u\right)^{-1}\boldsymbol{\beta}_u - \boldsymbol{B}_u\boldsymbol{\mu}\Big) && (3.100)\\
&= \boldsymbol{\mu} + \boldsymbol{\Sigma}\boldsymbol{B}_u^T\boldsymbol{e}_u && (3.101)
\end{aligned}
$$

where $\boldsymbol{e}_u := \boldsymbol{E}_u\Big(\operatorname{diag}\left(\boldsymbol{\pi}_u\right)^{-1}\boldsymbol{\beta}_u - \boldsymbol{B}_u\boldsymbol{\mu}\Big) \in \mathbb{R}^{P_u}$. Thus,we can rewrite the update of $\boldsymbol{b}$ as

$$
\begin{aligned}
\boldsymbol{b} &\leftarrow \frac{1}{U}\sum_{u\in\mathbb{U}}\boldsymbol{\xi}_u && (3.102)\\
&= \boldsymbol{\mu} + \boldsymbol{\Sigma}\frac{1}{U}\sum_{u\in\mathbb{U}}\boldsymbol{B}_u^T\boldsymbol{e}_u && (3.103)\\
&= \boldsymbol{\mu} + \boldsymbol{\Sigma}\boldsymbol{s} && (3.104)
\end{aligned}
$$

with $\boldsymbol{s} = \frac{1}{U}\sum_{u\in\mathbb{U}}\boldsymbol{B}_u^T\boldsymbol{e}_u$.

We now revisit the computation of $\boldsymbol{C} = \frac{1}{U}\sum_{u\in\mathbb{U}}\Big(\boldsymbol{\Xi}_u + \left(\boldsymbol{\mu} - \boldsymbol{\xi}_u\right)\left(\boldsymbol{\mu} - \boldsymbol{\xi}_u\right)^T\Big)$. For convenience, we denote the average over users as $\langle f(u)\rangle := \frac{1}{U}\sum_{u\in\mathbb{U}} f(u)$, i.e. $\boldsymbol{C} = \Big\langle \boldsymbol{\Xi}_u + \left(\boldsymbol{\mu} - \boldsymbol{\xi}_u\right)\left(\boldsymbol{\mu} - \boldsymbol{\xi}_u\right)^T\Big\rangle$.

In order to make clearer the distinction between new value after the update and the previous value, we use $\boldsymbol{b}$ to denote the updated value and $\boldsymbol{\mu}$ to denote the previous value of the bias term. Recall, that after the update we have

$$
\boldsymbol{b} = \boldsymbol{\mu} + \boldsymbol{\Sigma}\boldsymbol{s} \tag{3.105}
$$

Plugging (3.94) for $\boldsymbol{\Xi}_u$ and $\boldsymbol{b}$ for $\boldsymbol{\mu}$ in $\boldsymbol{C}$ and noting that $\boldsymbol{b} - \boldsymbol{\xi}_u = \boldsymbol{b} - \boldsymbol{\mu} + \boldsymbol{\mu} - \boldsymbol{\xi}_u =$

$\boldsymbol{\Sigma s} - \boldsymbol{\Sigma B}_u^T \boldsymbol{e}_u = \boldsymbol{\Sigma}\left(\boldsymbol{s} - \boldsymbol{B}_u^T \boldsymbol{e}_u\right)$, we obtain:

$$\begin{aligned}
\boldsymbol{C} &= \left\langle \boldsymbol{\Xi}_u - (\boldsymbol{\mu} - \boldsymbol{\xi}_u)(\boldsymbol{\mu} - \boldsymbol{\xi}_u)^T \right\rangle && (3.106)\\
&= \boldsymbol{\Sigma} - \left\langle \boldsymbol{\Sigma B}_u^T \boldsymbol{E}_u \boldsymbol{B}_u \boldsymbol{\Sigma} - (\boldsymbol{\mu} - \boldsymbol{\xi}_u)(\boldsymbol{\mu} - \boldsymbol{\xi}_u)^T \right\rangle && (3.107)\\
&= \boldsymbol{\Sigma} - \left\langle \boldsymbol{\Sigma B}_u^T \boldsymbol{E}_u \boldsymbol{B}_u \boldsymbol{\Sigma} - \boldsymbol{\Sigma}\left(\boldsymbol{s} - \boldsymbol{B}_u^T \boldsymbol{e}_u\right)\left(\boldsymbol{s} - \boldsymbol{B}_u^T \boldsymbol{e}_u\right)^T \boldsymbol{\Sigma} \right\rangle && (3.108)\\
&= \boldsymbol{\Sigma} - \boldsymbol{\Sigma}\left\langle \boldsymbol{B}_u^T \boldsymbol{E}_u \boldsymbol{B}_u - \left(\boldsymbol{s} - \boldsymbol{B}_u^T \boldsymbol{e}_u\right)\left(\boldsymbol{s} - \boldsymbol{B}_u^T \boldsymbol{e}_u\right)^T \right\rangle \boldsymbol{\Sigma} && (3.109)\\
&= \boldsymbol{\Sigma} - \boldsymbol{\Sigma}\left\langle \boldsymbol{B}_u^T \boldsymbol{E}_u \boldsymbol{B}_u - \boldsymbol{s}\boldsymbol{s}^T + 2\boldsymbol{s}\boldsymbol{B}_u^T \boldsymbol{e}_u - \boldsymbol{B}_u^T \boldsymbol{e}_u \boldsymbol{e}_u^T \boldsymbol{B}_u \right\rangle \boldsymbol{\Sigma} && (3.110)\\
&= \boldsymbol{\Sigma} - \boldsymbol{\Sigma}\left(\left\langle \boldsymbol{B}_u^T \left(\boldsymbol{E}_u - \boldsymbol{e}_u \boldsymbol{e}_u^T\right) \boldsymbol{B}_u \right\rangle - \boldsymbol{s}\boldsymbol{s}^T + 2\boldsymbol{s}\left\langle \boldsymbol{B}_u^T \boldsymbol{e}_u \right\rangle\right) \boldsymbol{\Sigma} && (3.111)\\
&= \boldsymbol{\Sigma} - \boldsymbol{\Sigma}\left(\left\langle \boldsymbol{B}_u^T \left(\boldsymbol{E}_u - \boldsymbol{e}_u \boldsymbol{e}_u^T\right) \boldsymbol{B}_u \right\rangle + \boldsymbol{s}\boldsymbol{s}^T\right) \boldsymbol{\Sigma} && (3.112)\\
&= \boldsymbol{\Sigma} - \boldsymbol{\Sigma S \Sigma} && (3.113)
\end{aligned}$$

Running EP as it is costs $O(I^3)$ as we are computing marginal variances over $\boldsymbol{f}_u$. As initially noted, instead of running inference over $\boldsymbol{f}_u$, we can equivalently run EP to compute $Q(\bar{\boldsymbol{f}}_u)$, using a prior $P(\bar{\boldsymbol{f}}_u) = \mathcal{N}\left(\bar{\boldsymbol{f}}_u \mid \boldsymbol{B}_u \boldsymbol{\mu}, \boldsymbol{B}_u \boldsymbol{\Sigma} \boldsymbol{B}_u^T\right)$ to obtain $\boldsymbol{\pi}_u, \boldsymbol{\beta}_u$ for $\boldsymbol{E}_u, \boldsymbol{e}_u$. This is convenient, also because this formulation is a canonical GP classification with a conditionally independent likelihood for which standard EP code can be used.

3.2.2.6 Implementation Details

In order to run a Krylov method to compute parts of the spectrum of $\bar{\boldsymbol{C}} = \boldsymbol{L}_K^{-1} \boldsymbol{C} \boldsymbol{L}_K^{-T}$ iteratively, we notice that it is not necessary to form $\bar{\boldsymbol{C}}$ explicitly. Instead, we provide a linear operator that computes $\boldsymbol{x} \mapsto \bar{\boldsymbol{C}} \boldsymbol{c}$ by using the structure of $\bar{\boldsymbol{C}}$.

We rewrite $\bar{\boldsymbol{C}}$ by plugging in the definition of $\boldsymbol{C}$:

$$\begin{aligned}
\bar{\boldsymbol{C}} &= \boldsymbol{L}_K^{-1} \boldsymbol{C} \boldsymbol{L}_K^{-T} && (3.114)\\
&= \boldsymbol{L}_K^{-1} \left(\boldsymbol{\Sigma} - \boldsymbol{\Sigma S \Sigma}\right) \boldsymbol{L}_K^{-T} && (3.115)\\
&= \boldsymbol{L}_K^{-1} \boldsymbol{\Sigma} \boldsymbol{L}_K^{-T} - \boldsymbol{L}_K^{-1} \boldsymbol{\Sigma S \Sigma} \boldsymbol{L}_K^{-T} && (3.116)\\
&= \bar{\boldsymbol{\Sigma}} - \bar{\boldsymbol{\Sigma}} \boldsymbol{L}_K \boldsymbol{S} \boldsymbol{L}_K^T \bar{\boldsymbol{\Sigma}} && (3.117)
\end{aligned}$$

where we can multiply with $\bar{\boldsymbol{\Sigma}}$, given in (3.78), in $O(DM)$ by performing $\boldsymbol{x} \mapsto \bar{\boldsymbol{V}}(\bar{\boldsymbol{V}}^T)\boldsymbol{x} + \sigma^2 \boldsymbol{x}$, since we maintain $\bar{\boldsymbol{V}}$. Furthermore, we need two triangular and one symmetric matrix-vector multiplications, which costs $O(I^2)$. Thus, we save the $O(I^3)$ cost of forming $\bar{\boldsymbol{C}}$ and get away with an only marginally more expensive cost than multiplying with $\bar{\boldsymbol{C}}$ directly. To compute the leading D eigenvectors and -values, we need to perform $O(D)$ multiplications, for a total of $O(DI^2)$ for the M-Step.

For the E-Step, we note that the statistics $\boldsymbol{E}_u$ and $\boldsymbol{e}_u$ can be independently computed per user. The accumulation of $\boldsymbol{S}$ and $\boldsymbol{s}$ is thus in the form of a map-reduce operation.

We implemented a parallelized map-reduce version of our code that runs E-step in parallel, using multi-threading in C++ with Boost. For simplicity and to avoid synchronization, in our implementation, each thread manages a local copy of the sufficient statistic matrix which are then added in the end in parallel, requiring only logarithmic number of operations in the number of threads. We implemented the parallel reduction using Intel's Threading Building Blocks. We opted for a shared memory environment for ease of implementation, as the scale of the experiments did not justify the communication overhead of a distributed environment.

3.2.3 Experiments

In this section we report experimental results. In particular, we are interested in the following aspects. In Section 3.2.3.2, we compare against various GP-based models based on existing work. There, we investigate the impact of the different components of the model, as well as the accuracy of the inference method. While in Section 3.2.3.2, we are limited to small datasets due to the complexity of the GP-based methods, we compare against, in Section 3.2.3.3, we investigate the performance and running time characteristics of our method on several larger datasets.

We begin by describing the error metric used as well as the datasets.

3.2.3.1 Preliminaries

Error Metric. All methods we compare are probabilistic in nature. Therefore, we chose negative log predictive probabilities as error metric. The predictive distribution over an unseen pair $y^* = (i^*, j^*)$ is defined as

$$P(y^* \mid \mathbb{O}) \approx \int P(i^* \succ_u j^* \mid \boldsymbol{f}_u) Q(\boldsymbol{f}_u) \,\mathrm{d}\boldsymbol{f}_u \tag{3.118}$$

$$= \int P(i^* \succ_u j^* \mid \bar{f}_{\omega_u(i^*,j^*)}) Q(\bar{f}_{\omega_u(i^*,j^*)}) \,\mathrm{d}\bar{f}_{\omega_u(i^*,j^*)} \tag{3.119}$$

Formally, this corresponds to the computation necessary for an EP update, which in case of probit likelihood amounts to the evaluation of a normal CDF [Rasmussen and Williams, 2006].

Datasets. In our experiments we used two types of datasets, summarized in Table 3.10. The Movielens dataset is an explicit rating dataset, while the Sushi dataset consists of rankings, from both of which we derive preference pairs.

The perhaps most used preference benchmark dataset is the Sushi (SSH) dataset [Kamishima, 2003], that records the preferences of 5000 restaurant clients for different types of sushi. The data comes in two versions: For the smaller one (SSH-A), each user was asked to give a ranking of the same 10 types of sushi. The ranking can be decomposed into individual comparisons for a total of 45 comparisons per user. For the larger dataset (SSH-B), the item set consists of 100 types of sushi. Clients were asked to rank random subsets of 10 types of sushi each.

This dataset provides both, user and item features. There are 18 item features (2 binary, 4 numerical and a 12-state categorical feature, that we represent using a one-hot encoding) and 9 user features.

We randomly drew training and test sets as follows. For SSH-A, we subsampled $U = 200$ users, then extracted 3 training and 1 test pair per user. This very small dataset size allowed us to compare against methods which scale cubically in the total number of training pairs. For SSH-B, we used all $U = 5000$ users, and extracted 10 training and 1 test pair per user.

The Movielens (ML) dataset [Miller et al., 2003] consists of explicit movie ratings and is available in 3 different sizes. Similar to Mnih and Teh [2012], we derive comparisons from the ratings for each user by pairing all of the users items with highest rating with all of the users items with lowest rating. For each user we randomly select 4 of these pairs for testing. Users with insufficient number of pairs are removed.

	SSH-A	SSH-B	ML-100K	ML-1M	ML-10M
Items	10	100	1.5k	3.5k	9.5k
Users	5k	5k	715	4.6k	45.6k
Available Pairs	225K	225K	187k	3.5M	18.5M
TestPairs	5k	5k	1.3k	9.9k	77k

Table 3.4 – Versions of Sushi (SSH) and Movielens (ML) datasets for preference learning. We refer to the text for details on preprocessing and selection of test pairs.

3.2.3.2 Comparison to Alternative Utility Models

We compare our method to other probabilistic preference learning models. Due to the limited scalability of some of them, the comparison is conducted on the relatively small Sushi dataset. We compare the following utility models:

GPfull Given item features $\boldsymbol{x}^{(i)} \in \mathbb{R}^{D^{(i)}}$ and user features $\boldsymbol{x}^{(u)} \in \mathbb{R}^{D^{(u)}}$, the Bonilla et al. [2010] model the utility matrix $\boldsymbol{F} \in \mathbb{R}^{I \times U}$ using a latent function $f(\boldsymbol{x}^{(u)}, \boldsymbol{x}^{(i)})$ drawn from a coupled GP with kernel $k(\boldsymbol{x}_u^{(u)}, \boldsymbol{x}_v^{(u)}, \boldsymbol{x}_i^{(i)}, \boldsymbol{x}_j^{(i)}) = k(\boldsymbol{x}_u^{(u)}, \boldsymbol{x}_v^{(u)}) k(\boldsymbol{x}_i^{(i)}, \boldsymbol{x}_j^{(i)})$. The kernel matrix is $\boldsymbol{K}^{(i)} \otimes \boldsymbol{K}^{(u)} \in \mathbb{R}^{IU \times IU}$. Inference is performed using the Laplace approximation. We use the code provided by the authors[8].

---

[8] http://ebonilla.github.io/gppe/

GPitem The model of Birlutiu et al. [2010] models columns of $\boldsymbol{F}$ by latent functions $f_u(\boldsymbol{x}^{(i)})$ with kernel $k(\boldsymbol{x}_i^{(i)}, \boldsymbol{x}_j^{(i)})$. Kernel parameters are shared across users. We use EP and Laplace approximations.

VU This model consists only of the bi-linear term $\boldsymbol{F} = \boldsymbol{V}\boldsymbol{U}^T + \boldsymbol{b}\boldsymbol{1}_U^T$ and does not make use of item or user features. Salimans et al. [2012] described this model and used a fully factorized posterior over $\boldsymbol{V}$, $\boldsymbol{U}$ and $\boldsymbol{b}$. In contrast, here, we estimate $\boldsymbol{V}$.

GPVU This is our model combining GPitem with VU.

Apart from the utility models, we were interested in the impact of our choice of inference method. Thus, for the models GPitem and GPVU, we compare the performance of the EP approximation against other, simpler ones. For GPVU, these are a local bounding method proposed by Jaakkola [1997], designated JK, that approximates the logistic sigmoid instead of the probit. For GPitem we compare against the Laplace approximation (LP) that was used by Bonilla et al. [2010], Birlutiu et al. [2010].

For all GPs, we use the squared-exponential kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sigma^2 \exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/\psi)$. For GPfull, there are two sets of hyperparameters (σ_i^2, ψ_i) and (σ_u^2, ψ_u) for item and user kernel, respectively. While it is possible to learn these parameters, we perform a grid search to minimize the validation error, except for our method, that learns σ^2. Similarly, for VU and GPVU, we select the best value for D. We improve conditioning of all kernel matrices by adding $\alpha\boldsymbol{I}$ with $\alpha = 0.1$.

Results on SSH-A. The SSH-A dataset is small enough to allow us to run all methods, including GPfull. The corresponding errors are shown in Table 3.5. The rankings are sparsely observed, a setting in which bi-linear models appear to excel. Another reason could be the quality of the features. Especially the user features seem to be not very useful, a result consistent with previous findings [Houlsby et al., 2012]. A content based component in the model can nevertheless be advantageous, e.g. for cold start scenarios.

Model	GPfull	GPitem	VU	GPVU
Error	0.897	0.875	0.856	**0.798**

Table 3.5 – SSH-A: Impact of model on performance. GPFull: Bonilla et al. [2010], GPitem: Birlutiu et al. [2010], VU: PPCA version of Salimans et al. [2012], GPVU: ours.

Next, we were interested in the impact of the inference method on the performance. Methods like the Laplace approximation and local bounds have been shown to be less representative of skewed distributions of posterior mass than the EP approximation [Kuss and Rasmussen, 2005, Nickisch and Rasmussen, 2008]. Table 3.6 shows the results of the comparison. As parts of the performance gain of our method is due to the choice of inference approximation, it is noteworthy, that our formulation enables us to reliably run parallel EP for probit classification without modifications, such as damping, achieving convergence over all users typically within around 10 iterations.

Method	GPitem-LP	GPitem-EP	GPVU-JK	GPVU-EP
Error	0.910	0.875	0.854	**0.798**

Table 3.6 – SSH-A: Impact of inference method on performance. LP: Laplace approximation, JK: Jaakkola approximation, EP: Expectation Propagation.

Results on SSH-B. The second Sushi dataset contains the full set of 5000 users. While we admit more pairs, the item size increases 10-fold compared to SSH-B. In this setting, we were interested if item features have a positive effect. We compared VU and GPVU and obtain errors as shown in Table 3.7. While the accuracy is slightly better for GPVU, it required careful tuning of the kernel parameters.

Model	VU	GPVU
Error	0.730	**0.690**

Table 3.7 – SSH-B: Performance improvement due to item features.

3.2.3.3 Movielens Datasets

Sub-sampling Observations. Inference for an user u scales cubically with the number of pairs P_u observed for this user. Real-world collaborative filtering datasets can exhibit heavy-tailed statistics for the number of observations, where for a fraction of users most of the observations were made. Inference for these few users would dominate the computation of the E-Step statistics. Therefore, we implement a simple heuristic to impose a limit on the computational budget available per user: if $P_u > P_{\max}$, we sample $P_{\max}$ observations for a user uniformly at random. In Figure 3.5 we show different runs of our method for different values of $P_{\max}$ and corresponding P_E on the ML-100K dataset. There appears to be a stable regime, where variations in $P_{\max}$ does not have a strong influence on the error achieved. This mechanism thus constitutes a simple way to adjust the cost of the algorithm to the available computational budget.

Running Time. In Table 3.8, we report running times of a single iteration of GPVU-EP on the different MovieLens datasets for different values of P_E. The running times were achieved on a Intel 2.8GHz i7-2600S processor with 4 physical cores. We see that on commodity hardware with four cores, we can run up to 3M pairs within an hour. The inherent parallelism in the problem allows us to scale further. On a 32-cores system, we were able to further reduce the processing time for 3M pairs to 10 minutes.

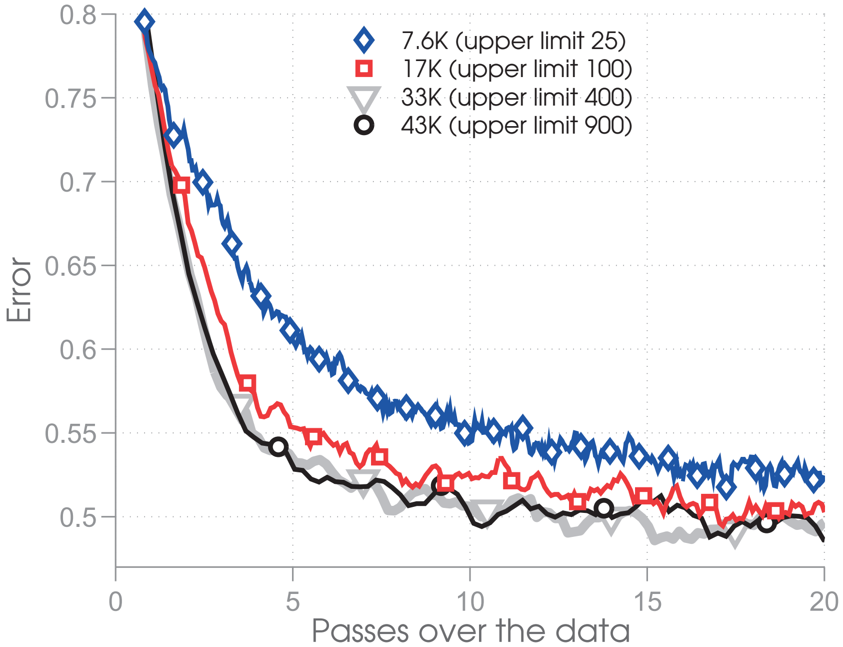


Figure 3.5 – ML-100K: GPVU-EP, $D = 30, \psi = 1$. We show runs with different values of upper limits of pairs $P_{\max}$. Different values $P_{\max}$ imply a different number of pairs processed during an E-Step. We show test errors as a function of passes over the full set of pairs. Markers are set after 20 EM iterations.

P_E	ML-100K	P_E	ML-1M	P_E	ML-10M
17k	1s, 1s	147k	9s, 3s	1M	1m, 20s
33k	7s, 1s	349k	1m, 3s	2M	7m, 20s
43k	27s, 1s	534k	15m, 3s	3M	58m, 20s

Table 3.8 – Running times of an iteration of GPVU-EP on MovieLens datasets for different total numbers of training pairs processed in the E-Step (P_E) due to different values of $P_{\max}$. We report running times of a single EM iteration (in s(econds) or m(inutes)).

Non-parametric PCA. In this setting, we compare our method to the non-parametric PCA (NPCA) method by Yu et al. [2009]. Their EM algorithm served as inspiration of ours. The motivation for NPCA was the observation that for the Netflix prediction problem, increasing the dimensionality of latent factors seemed to improve the test RMSE. Instead of a low-rank decomposition, they are after a bi-linear model with full rank, i.e. $D = I$. There is no content-based aspect to their utility model. They derive a PPCA-like model with this assumption, that can be trained using an EM algorithm close to ours, but with an M-Step that imposes no rank constraint on the prior covariance $\mathbf{\Sigma}$. The criterion is the same as ours in (3.76). Since they optimize for an unstructured covariance, their M-Step update is just $\mathbf{\Sigma} \leftarrow \mathbf{\Sigma} - \mathbf{\Sigma} \boldsymbol{C} \mathbf{\Sigma}$, which costs two matrix-matrix multiplications at $O(I^3)$. The E-Step is identical to ours. We compare the resulting method to GPVU on the Movielens datasets and report test errors in Table 3.9. We see

	ML-100K	ML-1M	ML-10M
NPCA-EP	0.525	0.375	0.576
GPVU-EP	**0.483**	**0.368**	**0.390**

Table 3.9 – Average test log-loss for NPCA-EP and GPVU-EP on MovieLens (100K, 1M, 10M). Scores are averaged over 5 repetitions each (different seeds), as well as over the last few EM iterations of each run. GPVU outperforms NPCA in all cases, the differences being statistically significant for 100K and 10M.

that for the sparser datasets ML-100K and ML-10M NPCA appears to be overfitting, while our low-rank assumption seems more robust. On ML-1M the errors are much closer, which, we conjecture, is due to a higher density of observations.

3.2.3.4 Visualization of Results

Genre clustering. Figure 3.6 illustrates the parameters $\boldsymbol{V}, \boldsymbol{b}$ learned on ML-10M. We further decompose $\boldsymbol{V}$ into its SVD and display the five leading singular vectors and $\boldsymbol{b}$. Since each row corresponds to a movie, we sort and annotate the rows by the corresponding genres.We recognize distinct patterns in the latent factors for the different genre-clusters. This representation of items as real-valued vector could be useful for subsequent tasks. Note, however, that movie features are part of the item features.

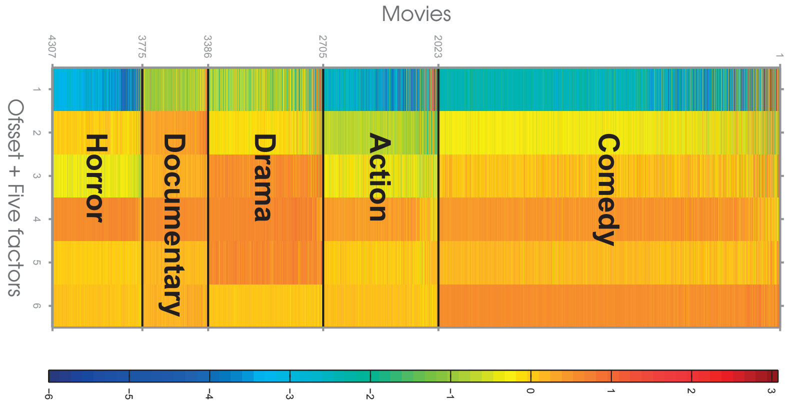


Figure 3.6 – ML-10M: Visualization of $\boldsymbol{b}$ and the leading components of $\boldsymbol{V}$. We recover an approximate genre clustering of movies.

Quantifying Uncertainty. Preference learning is often considered in an active learning context, where the goal is to elicit a user's preferences using a minimal amount of queries [Chu and Ghahramani, 2005, Brochu et al., 2008, Bonilla et al., 2010, Houlsby et al., 2012]. Such decision making problems can be solved in a principled way using probabilistic models by quantifying the remaining uncertainty given the current set of observations. Specifically, by observing certain behavior repeatedly, revealed by corresponding choices, the uncertainty about making such choices should be reduced. Retrieving more comparisons of this sort would therefore not be as informative as others and should be avoided. We illustrate the capability of our model to reflect this in the following experiment. We create an artificial preference dataset by creating comparisons in favor of a certain genre G_P. We hold out 100 of these pairs for testing and successively train our model on the remaining pairs. As more of these pairs are presented to the model, the uncertainty over the test set reduces as illustrated in Figure 3.7 for $G_P =$ Comedy and $G_P =$ Action. The red curve corresponds to the average probability to correctly predict in favor of G_P on the test set. The blue curve corresponds to unrelated comparisons for which we have not received sufficient information.

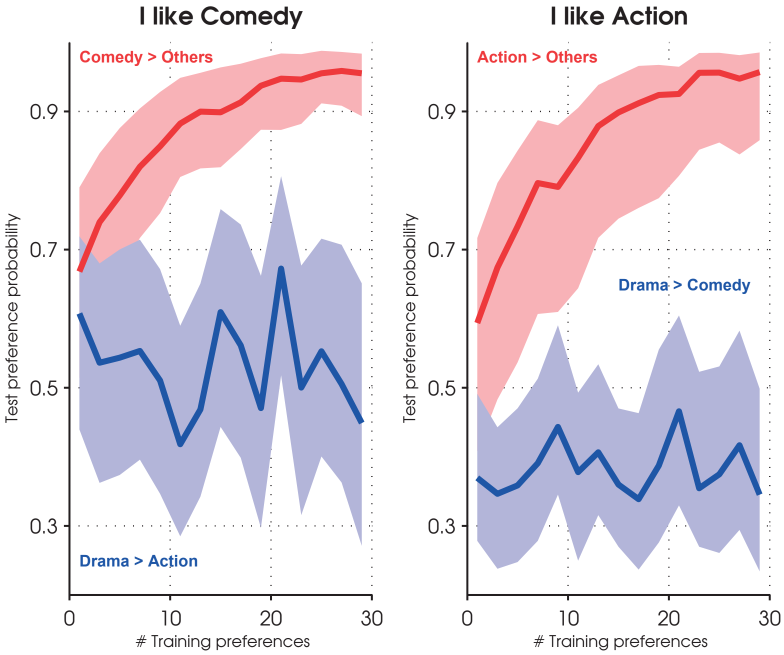


Figure 3.7 – ML-10M: Simulation of uncertainty reduction for genre preference. We create artificial preference pairs in favor of comedy movies (left) or action movies (right). As the number of comparisons in favor of the preferred genre trained on increases, the uncertainty about predicting this preference decreases (red curves), while unrelated comparisons remain uncertain (blue curves).

3.2.4 Discussion

In this section we introduced a new utility model for collaborative preference learning from pairwise comparisons, combining bilinear latent factor models with non-parametric regression. Bilinear models have been successfully used for collaborative filtering and are complemented here by a content-based aspect that can help to overcome sparsity. We discussed issues of scalability for bilinear models and found that the difficulty of learning such models do not merely stem from intractable inference in the E-Step. While still non-conjugate, we have powerful approximate inference techniques that are effective in this setting. Instead, learning this model is complicated due to the latent couplings induced by the observations, leading to an expensive M-Step. We address this structural problem by a reformulation from which we recover a standard PPCA M-Step. Although our model is not fully Bayesian, we feature a full posterior covariance model over items, useful for decision making such as in active learning.

While our algorithm scales better than other methods based on GPs, there are limitations due to the dense $I \times I$ matrices, we need to maintain, effectively limiting us in practice to around 10^5 items. Also, in order to reduce the complexity of the M-Step, the E-Step now scales cubically in the number of pairs per user. Assuming no repetitions, there can be $O(I^2)$ pairs in the worst case. We introduced a simple heuristic to reduce this number, for which, however, no theoretical guarantees exist. This heuristic could be replaced or complemented by other approximation techniques, such as sparse GP approximations. Another useful alternative could be online EM algorithms that update the parameters more frequently before making a full pass over the users.

3.3 Collaborative Recurrent Neural Networks

3.3.1 Overview and Related Work

As ever larger parts of the population routinely consume online an increasing amount of digital goods and services, and with the proliferation of affordable storage and computing resources, to gain valuable insight, content-, product- and service-providing organizations face the challenge and the opportunity of tracking at a large scale the activity of their users. Modeling the underlying mechanisms that govern a users' choice for a particular item at a particular time is useful for, e.g., boosting user engagement or sales by assisting users in navigating overwhelmingly large product catalogs through recommendations, or by using profits from accurately targeted advertisement to keep services free of charge. Developing appropriate methodologies that use the available data effectively is therefore of great practical interest.

In recent years, the recommendation problem has often been cast into an explicit-rating-prediction problem, possibly facilitated by the availability of appropriate datasets [e.g., Bennett and Lanning, 2007, Miller et al., 2003]. Incorporating a temporal aspect into the explicit-rating recommendation scenario is an active area of research [e.g., Rendle, 2010, Koren, 2010, Koenigstein et al., 2011, Chi and Kolda, 2012]. However, concerns are increasingly being raised about the suitability of explicit rating prediction as an effective paradigm for user modeling. Featuring prominently among the criticism are concerns about the availability and reliability of explicit ratings, as well as the static nature of this paradigm [Yi et al., 2014, Du et al., 2015], in which the tastes and interests of users are assumed to be captured by a one-time rating, thus neglecting the immediate context of the rating at the instant it is issued by the user.

In contrast, the implicit-feedback scenario is advantageous due to abundantly available data that can be gathered unintrusively in the background [Rendle et al., 2009]. Although methods for processing implicit feedback often only consider a static aggregate [e.g., Ko and Khan, 2014], the raw data typically consist of user activity logs, i.e., sequences of user interactions with a service, and is thus a far more suitable basis for dynamic user models. Data of this form is generated in many domains where user interactions can range from navigating a website by following links, to consuming multi-media content on a streaming site or to announcing the current physical locations on social media, thus making appropriate methods widely applicable. Moreover, a user's behavior is intimately linked to the context in which it is observed: factors such as mood, current activity, social setting, location, time-of-day etc. can temporarily cause a shift in a user's item preferences. While the relevant contextual factors themselves are hard to observe or even to define, their effect might be visible in the form of particular temporal patterns in the activity logs, the exploitation of which would then lead to more fine-grained and accurate user models. The purpose of this work is therefore to develop and evaluate methods for analyzing activity sequences generated by a heterogeneous set of users.

The desired traits of a model for this kind of data are (1) flexibility, (2) the ability to model complex, potentially long-range dependencies in the sequences, and (3) collaboration between users.

Flexibility. Although we often have at our disposal datasets richer than the sequences described above, these sequences are a common denominator for many applications. Thus, we devise practical methods suitable for processing sequences, but with the flexibility for handling different tasks with minor modifications, ideally in an end-to-end fashion, and for tapping into existing sources of metadata to augment the sequences.

Long-range Dependencies. The model needs to be powerful enough to represent complex dependencies within long sequences that might be of different length, in contrast to methods that, in order to achieve scalability, have to make compromises regarding the number of past events taken into account [Rendle et al., 2010, Wang et al., 2015].

Collaboration. Conceivably, there are two extreme cases: each individual user's behavior is modeled in isolation, or all users are described by a single prototypical behavioral profile. There are several drawbacks to both extremes: the first approach clearly relies on the sufficient availability of observations about each user, whereas realistically, the distribution of ratings over users is often heavily skewed in favor of a few very active users. Furthermore, users typically only access a subset of the items, thus making it difficult to recommend anything outside of this subset. The second approach, i.e. ignoring to the concept of individual users, implies that a single model is trained on all sequences pooled together. Although pooling, to an extent, mitigates the sparsity problem, it lacks personalization and thus neglects potentially important differences in user behavior. Instead of these extremes, our model needs to be personalized yet collaborative to make efficient use of the available data by identifying parameters that can be learned across users.

In this work, we propose a model that fulfills these requirements based on recurrent neural networks [RNN, Rumelhart et al., 1988, Werbos, 1990], that combines ideas from language modeling with collaborative-filtering to take into account the multi-user structure of the problem. We study different architectural variants of our model and find that including the collaborative aspect can lead to an efficient, practical method. To test the versatility of our approach, we present experimental results on two very different real-world tasks, music recommendation and mobility prediction, where we find that the new model significantly outperforms various baseline methods.

Related Work. As previously mentioned, there has been much work dedicated to the rating-prediction problem. Some of these methods also take into account the effect of time [Rendle, 2010, Koren, 2010, Koenigstein et al., 2011, Chi and Kolda, 2012]. The difference is that recurrent temporal patterns are not apparent in the underlying datasets, because ratings are one-time events. Du et al. [2015] propose a model for recurrent user activities, but use a fundamentally different approach: they use continuous-time stochastic processes to explicitly model time and tie the recommendation to a specific point in time and focus more on modeling *when* the next event might occur by using the self-excitation property of the Hawkes process. A problem closely related to our setup is the next-basket prediction for e-commerce applications [Rendle et al., 2010, Wang et al., 2015], which models sequences of sets of items that represent the contents of shopping carts bought by users. Wang et al. [2015] learn a hierarchical latent-factor representation reminiscent of a multi-layer perceptron arguing that the non-linearity in their model is a crucial component. Rendle et al. [2010] use a Markov chain per user and assume low-rank structure for the transition tensor and optimize a BPR-inspired objective [Rendle et al., 2009]. Our work differs in mainly two ways: first, we model events at a finer granularity, i.e., per event, which is more appropriate for the applications we considered. Second, the use of RNNs enables us to model long-range dependencies, whereas their models use much stricter factorization assumptions for computational reasons. To the best of our knowledge, this is the first use of RNNs in this context.

The remainder of this section is organized as follows. We begin in Section 3.3.2 by introducing notation, and formalizing the task by defining the error metric we will use for comparing different methods. In Section 3.3.3 we introduce various baseline models and discuss their properties. In Section 3.3.4, we describe our collaborative RNN model, the learning algorithm and present variants of the neural network. We present the experimental evaluation in Section 3.3.5.

3.3.2 Preliminaries

Let $\mathbb{U}$ denote a set of users of cardinality $|\mathbb{U}| = U$ and let $\mathbb{I}$ denote a set of items of cardinality $|\mathbb{I}| = I$. As usual, the sets of users and items are assumed to be fixed. We use indices u and i to refer to users and items, respectively. For every user $u \in \mathbb{U}$ we observe a sequence $\boldsymbol{y}^{(u)} = [y_1^{(u)}, \dots, y_{T_u}^{(u)}]$ with elements $y_t^{(u)} \in \mathbb{I}$, i.e., each event in a user sequence is an exclusive choice over the set of items. We will refer to a user making such a choice as *consuming* the item. Similar to [Rendle et al., 2010], we focus on sequences without taking into account the absolute time at which the events occurred, because here, we are interested in exploiting temporal patterns expressed by the ordering of events. We denote by $\boldsymbol{y}_{<t}^{(u)} = [y_1^{(u)}, \dots, y_{t-1}^{(u)}]$ and $\boldsymbol{y}_{\geq t}^{(u)} = [y_t^{(u)}, \dots, y_{T_u}^{(u)}]$ subsequences up to and from index t, respectively.

The basic query a sequence model should support is the quantification of the likelihood of a user u to consume items at $t = k, k+1, k+2, \dots$ given access to the history $\boldsymbol{y}_{<k}^{(u)}$ of events.

Our approach for such prediction tasks is to model the observed sequences probabilistically. Given the ordering imposed by time, it is reasonable to choose a factorization of the joint distribution over a sequence that respects that order:

$$P_{\boldsymbol{\theta}}(\boldsymbol{y}^{(u)}) = \prod_{t=1}^{T_u} P_{\boldsymbol{\theta}}(y_t^{(u)} \mid \boldsymbol{y}_{<t}^{(u)}), \tag{3.120}$$

where $\boldsymbol{\theta}$ denotes a set of model parameters.

As we model a discrete set of outcomes, the conditionals are multinomial distributions over $\mathbb{I}$, represented by probability vectors $\boldsymbol{p}_t^{(u)}$ of dimension I, whose concrete parameterization and form is determined by the assumptions that underlie a particular model:

$$P_{\boldsymbol{\theta}}(y_t^{(u)} \mid \boldsymbol{y}_{<t}^{(u)}) = \boldsymbol{p}_t^{(u)}(\boldsymbol{\theta}, \boldsymbol{y}_{<t}^{(u)}). \tag{3.121}$$

The evaluation metric we report is the average negative log likelihood computed for a held-out part of the sequence: For every user, we split $\boldsymbol{y}^{(u)}$ into $\boldsymbol{y}_{<r_u}^{(u)}$ and $\boldsymbol{y}_{\geq r_u}^{(u)}$ at $1 < r_u \leq T_u$ and define the error as

$$E(\boldsymbol{y}_{\geq r_1}^{(1)}, \ldots, \boldsymbol{y}_{\geq r_U}^{(U)} \mid \boldsymbol{\theta}) = -\frac{1}{U} \sum_{u \in \mathbb{U}} \frac{1}{T_u - r_u + 1} \log P_{\boldsymbol{\theta}}(\boldsymbol{y}_{\geq r_u}^{(u)}). \tag{3.122}$$

3.3.3 Baseline Models

Many approaches to specify a model within the framework of Eq. 3.120 come to mind. Here, we present several straightforward models, that differ in whether or not they have a mechanism to represent dynamics and collaboration, and against which we will benchmark our method.

Static Uniform. A trivial, yet illustrative, baseline is the uniform distribution over $\mathbb{I}$:

$$P(y_t^{(u)} \mid \boldsymbol{y}_{<t}^{(u)}) = P(y_t^{(u)}) = \frac{1}{I}. \tag{3.123}$$

n-grams. n-gram models, popular in NLP [Brown et al., 1992], are distributions over co-occurrences of n consecutive items and thus have to maintain $O(I^n)$ parameters. Due to data sparsity, we consider only $n = 1$ (unigram) and $n = 2$ (bigram). We define the following quantities:

$$c_i = \sum_{u \in \mathbb{U}} \sum_{t=1}^{T_u} \mathbb{1}(y_t^{(u)} = i), \qquad b_{ij} = \sum_{u \in \mathbb{U}} \sum_{t=2}^{T_u} \mathbb{1}(y_t^{(u)} = i, y_{t-1}^{(u)} = j). \tag{3.124}$$

Then, we define the unigram model as

$$P(y_t^{(u)} = i \mid \boldsymbol{y}_{<t}^{(u)}) = P(y_t^{(u)} = i) = \frac{c_i + \epsilon}{\sum_{k \in \mathbb{I}} (c_k + \epsilon)}. \tag{3.125}$$

Similarly, the bigram model is defined as

$$P(y_t^{(u)} = i \mid \boldsymbol{y}_{<t}^{(u)}) = P(y_t^{(u)} = i \mid y_{t-1}^{(u)} = j) = \frac{b_{ij} + \epsilon}{\sum_{k \in \mathbb{I}} (b_{kj} + \epsilon)}, \tag{3.126}$$

where we added Laplace-smoothing parameterized by ϵ.

Matrix Factorization. Methods based on matrix factorization have become a widely used tool for collaborative filtering due to their early success for recommender systems [Koren et al., 2009]. Although the low-rank representation of a rating matrix is helpful for sparse static rating data by transferring knowledge between similar users and items, they are less suitable for sequential data. Approaches, such as tensor factorization [Xiong et al., 2010, Chi and Kolda, 2012] generalize matrix factorization to more than two dimensions, one of which can be used to represent time under the assumptions that observations per user are temporally aligned. Related to this issue is the a-priori unknown optimal size of time slices for grouping together related observations. These challenges could greatly increase the complexity of the model or of the processing pipeline. Nevertheless, a static method based on matrix factorization is an interesting baseline to compare against in order to study the effect of a collaborative component in isolation. From the sequential data, we construct a matrix of log-counts[9], i.e., we construct a (sparse) matrix $\boldsymbol{M}$ with entries $m_{iu} = \log\left(c_i^{(u)} + 1\right)$. Then, we compute the low-rank decomposition of $\boldsymbol{M} \approx \boldsymbol{V}^T\boldsymbol{U}$ with $\boldsymbol{V} \in \mathbb{R}^{D \times I}$ and $\boldsymbol{U} \in \mathbb{R}^{D \times U}$ by solving the weighted-λ-regularized formulation by Zhou et al. [2008]:

$$\min_{\boldsymbol{U},\boldsymbol{V}} \sum_{(i,u):m_{iu}>0} \left(m_{iu} - \boldsymbol{v}_i^T \boldsymbol{u}_u\right)^2 + \lambda \left(\sum_{u \in \mathbb{U}} d_u \|\boldsymbol{u}_u\|^2 + \sum_{i \in \mathbb{I}} d_i \|\boldsymbol{v}_i\|^2 \right), \tag{3.127}$$

where with a slight abuse of notation, we denote by $d_u = \sum_{i \in \mathbb{I}} \mathbb{1}(m_{iu} > 0)$ and $d_i = \sum_{u \in \mathbb{U}} \mathbb{1}(m_{iu} > 0)$. Next-step distributions can then be defined as

$$P(y_t^{(u)} \mid \boldsymbol{y}_{<t}^{(u)}) = P(y_t^{(u)}) = \sigma_m(\boldsymbol{z}_u, y_t^{(u)}), \qquad \boldsymbol{z}_u = \boldsymbol{V}^T \boldsymbol{u}_u, \tag{3.128}$$

$$\sigma_m(\boldsymbol{z}, j) = \frac{\exp(z_j)}{\sum_{k \in [|\boldsymbol{z}|]} \exp(z_k)}. \tag{3.129}$$

This model is collaborative and makes the whole set $\mathbb{I}$ accessible for each user, but it neglects the sequential nature of the data.

Hidden Markov Model (HMM). Another natural baseline to compare against is the

---

[9] We use log counts to combat over-dispersion due to the heavy-tailed nature of the data. Note that as a side effect the output of the softmax (Eq. 3.128) can then be interpreted as a ratio of pseudo-counts.

HMM. We consider a standard formulation and refer to Rabiner and Juang [1986].

3.3.4 The Collaborative Recurrent Neural Network Model

We motivate our Collaborative RNN (C-RNN) model as follows. We begin by postulating a latent feature embedding that characterizes the items and that can be considered static for the time-scale of interest, in the same spirit as latent factor models [Koren et al., 2009] and word embeddings [Mikolov et al., 2013]. Next, we view user behavior as governed by a dynamical state intrinsic to each user. This internal state is hidden and needs to be inferred from recent activity and could be due to a mechanism too complicated to explicitly describe. Recurrent neural networks are sequence models that operate in precisely this way: They maintain a hidden state that is updated when the next element in the input sequence is presented to the network, by a complex, non-linear parametric function of the previous events, whose parameters are learned from the data itself. Moreover, the parameters in the input layer can be interpreted as item embeddings, which therefore can be learned, too, from variable length sequences.

In recent years, practical advances in the area of recurrent neural networks and their variants [Hochreiter and Schmidhuber, 1997, Cho et al., 2014b] have led to enormously successful methods to tackle various interesting, yet extremely challenging, tasks that are inherently sequential in nature, most prominently in, but not restricted to, the area of Natural Language Processing (NLP). The success of RNN-based models in these applications demonstrates their versatility, flexibility and ability to capture complex patterns that require the model to retain information gathered at various previous time steps, possibly in the distant past. Therefore, in spite of shortcomings such as the lack of theoretical guarantees on the performance, we deemed RNNs to be a solid foundation on top of which we developed a model that fulfills the previously posed requirements.

We found RNN language models as described in Section 1.2.2, to be well-suited for incorporating the dynamics into our collaborative model. In particular, the RNN Language Model [RNNLM, Mikolov et al., 2010] was the main source of inspiration for this work, as we found their problem formulation to closely match ours. For language models, $\mathbb{I}$ represents the vocabulary, and user sequences correspond to sentences. The crucial difference to our setup is that the goal in language modeling is to learn a *single* underlying concept that ties together all sequences: a generative model for the language. Thus, it makes sense to not seek to model the dynamics per sequence, but to do so for the whole corpus. The necessary model-complexity for capturing the nuances of a language is achieved by adding more layers [Graves et al., 2013, Sutskever et al., 2014]. This perspective is to an extent supported by the type of data itself: For language models, a corpus consists of a plethora of relatively short sequences from the same language, whereas in our case, sequences are very long and are expressions of relatively few distinctly individual preferences. In Section 3.3.4.1, we will give details on how the RNNLM processes a single sequence. This procedure will serve as the building block to our simple collaborative extension in Section 3.3.4.2.

3.3.4.1 Processing a Single Sequence

Recall from Section 1.2.2, that RNNs compute a mapping from the input sequence to a corresponding sequence of real-valued hidden state vectors of dimension D:

$$\mathrm{RNN}\left([y_1, \ldots, y_T]\right) \;=\; [\boldsymbol{h}_1, \ldots, \boldsymbol{h}_T]\;, \quad \boldsymbol{h}_t \in \mathbb{R}^D. \tag{3.130}$$

The hidden state $\boldsymbol{h}_t$ is a flexible representation, meant to summarize the sequence seen up to time t, that enables us to use the network for different tasks by defining an appropriate output layer. RNNs can be seen as non-linear dynamic systems that define the computation of the hidden state recursively. In graphical terms, they are a generalization of the directed, acyclic graph structure of feed-forward networks by allowing cycles to represent dependencies on the past state of the network. The RNNLM uses a simple network architecture that goes back to Elman [1990] and expresses the dependency on the past through the following recursive definition of the hidden state vectors:

$$\begin{aligned} \boldsymbol{a}_t &= \boldsymbol{W}_h \boldsymbol{h}_{t-1} + \boldsymbol{W}_{in} \boldsymbol{\delta}_{y_t}, \\ \boldsymbol{h}_t &= \sigma(\boldsymbol{a}_t). \end{aligned} \tag{3.131}$$

The recurrence is initialized by a constant vector $\boldsymbol{h}_0 = \epsilon \mathbf{1}$ with small $\epsilon \geq 0$. The matrices $\boldsymbol{W}_h \in \mathbb{R}^{D \times D}$ and $\boldsymbol{W}_{in} \in \mathbb{R}^{D \times I}$ are parameters of the RNN and $\sigma(\cdot)$ is a non-linear function that is applied element-wise to its input, such as the logistic sigmoid, the hyperbolic tangent or, more recently, the rectified linear function. The input is presented to the network as one-hot encoded vectors denoted by $\boldsymbol{\delta}_{y_t}$, in which case the corresponding matrix-vector product $\boldsymbol{W}_{in}\boldsymbol{\delta}_{y_t}$ reduces to projecting out the y_t-th column of $\boldsymbol{W}_{in}$. Thus, the columns of $\boldsymbol{W}_{in}$ can be thought of as latent features describing he corresponding item. Note, that the network can be trivially extended to accept arbitrary side information that characterizes the input at time t. In practice, more sophisticated architectures implementing (3.130) are in use [Hochreiter and Schmidhuber, 1997, Cho et al., 2014b].

To obtain a distribution over the next item, we can linearly map the hidden state to $\mathbb{R}^I$ using a matrix $\boldsymbol{W}_{out} \in \mathbb{R}^{I \times D}$ and pass the output vector $\boldsymbol{y}_t \in \mathbb{R}^I$ through the softmax function from (3.129):

$$\begin{aligned} \boldsymbol{z}_t &= \boldsymbol{W}_{out} \boldsymbol{h}_t, \\ P(y_{t+1} \mid \boldsymbol{y}_{<t+1}) &= \sigma_m\left(\boldsymbol{z}_t, y_{t+1}\right). \end{aligned} \tag{3.132}$$

This likelihood together with the recursion in (3.131) enables us to sample sequences, by drawing from the multinomial (3.132) and presenting the sample as input to the network for the next timestep.

The network is parameterized by $\boldsymbol{\theta} = \{\boldsymbol{W}_{in}, \boldsymbol{W}_h, \boldsymbol{W}_{out}\}$ and can be trained by using maximum likelihood. The resulting continuous optimization problem is solved using stochastic gradient descent, where gradients are approximated using backpropagation through time [BPTT, Williams and Zipser, 1995], which we briefly describe below and amounts to unrolling the recurrence described by the forward equations (3.131) for a fixed number of steps.

Parameter Learning. We describe how the RNNLM processes a single sequence as we will use this procedure as a building block for learning the model in Section 3.3.4.2. As loss function Mikolov et al. [2010] use the log-probability of sequences (see Eq. 3.120), which is differentiable with respect to all parameter matrices. The network is trained using backpropagation through time [BPTT, Williams and Zipser, 1995]. BPTT computes the gradient by unrolling the RNN in time and by treating it as a multi-layer feed-forward neural network with parameters tied across every layer and error signals at every layer. For computational reasons, the sequence unrolling is truncated to a fixed size B. This is a popular approximation for processing longer sequences computationally more efficiently. This method is summarized in Algorithm 7. To train on a full corpus, Algorithm 7 is used on individual sentences in a stochastic fashion. The learning rule in the original RNNLM is a gradient descent step with a scalar step size[10]. The training process is regularized by using early stopping and small amounts of Tikhonov regularization on the network weights.

Algorithm 7 `processSequence()`

```
Require: Sequence $\boldsymbol{y}$, $\boldsymbol{\theta} = \{\boldsymbol{W}_{in}, \boldsymbol{W}_h, \boldsymbol{W}_{out}\}$, batch size $B$
Ensure: Updated parameters $\boldsymbol{\theta}$
$\mathcal{B} \leftarrow$ Split $\boldsymbol{y}$ into sub-sequences of length $B$
$\boldsymbol{h}_0 \leftarrow \epsilon\mathbf{1}$
for $\boldsymbol{b} \in \mathcal{B}$ do
    $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_B \leftarrow \mathrm{RNN}\,(\boldsymbol{b}, \boldsymbol{h}_0; \boldsymbol{\theta})$          (Forward pass using Eq. 3.131)
    $\nabla_{\boldsymbol{\theta}} \log E \leftarrow \mathrm{BPTT}\,(\boldsymbol{b}, \boldsymbol{h}_1, \ldots, \boldsymbol{h}_B; \boldsymbol{\theta})$    (Backward pass, see below)
    $\boldsymbol{\theta} \leftarrow \mathrm{LearningRule}\,(\nabla_{\boldsymbol{\theta}}, \boldsymbol{\theta})$
    $\boldsymbol{h}_0 = \boldsymbol{h}_B$
end for
```

Backpropagation Through Time. We derive the BPTT equations for the RNNLM for the gradient computations in Algorithm 7. This is done in the same manner as back-propagation is derived for feed-forward networks. The unrolled RNN differs from a feed-forward network by having input and error terms at each layer. Furthermore, gradients are approximate due to truncation. Note, that modern neural network libraries

---

[10] There are more nuances to their algorithm, but for the purpose of our development this basic exposition is sufficient.

do not require writing gradient code, which is especially useful for more complex RNN architectures. Nevertheless, we found it instructive to understand the basic computations involved in training neural networks.

Based on the forward equations (3.131) and (3.132), we define the sequence likelihood using the following quantities:

$$\boldsymbol{p}_t := [\sigma_m(\boldsymbol{z}_t, i)]_{i\in\mathbb{I}} \tag{3.133}$$

$$\ell_t := \log \sigma_m(\boldsymbol{z}_t, y_{t+1}) = \log \boldsymbol{p}_{t,y_{t+1}} \tag{3.134}$$

$$L := \textstyle\sum_{t=0}^{B-1} \ell_t \tag{3.135}$$

To compute $\nabla_{\boldsymbol{\theta}} L$, it is convenient to first compute the gradients $\nabla_{\boldsymbol{a}_t} L, \forall t$. The forward equations reveal the dependencies on a particular $\boldsymbol{a}_t$ as being two-fold: both ℓ_t and $\boldsymbol{a}_{t+1}$ contribute to $\nabla_{\boldsymbol{a}_t} L$. Therefore, the gradient $\nabla_{\boldsymbol{a}_t} L$ is given by (assuming row vectors)

$$\nabla_{\boldsymbol{a}_t} L = \nabla_{\boldsymbol{a}_t} \ell_t + \sum_k \frac{\partial L}{\partial a_{t+1,k}} \frac{\partial a_{t+1,k}}{\partial \boldsymbol{a}_t} = \nabla_{\boldsymbol{a}_t} \ell_t + (\nabla_{\boldsymbol{a}_{t+1}} L)(\Delta_{\boldsymbol{a}_t} \boldsymbol{a}_{t+1}) \tag{3.136}$$

The recursion is initialized by $\nabla_{\boldsymbol{a}_{B-1}} L = \nabla_{\boldsymbol{a}_{B-1}} \ell_{B-1}$ since only the output ℓ_{B-1} depends on the last activation. The gradient and Jacobian used above are given by:

$$\nabla_{\boldsymbol{a}_t} \ell_t = \sum_i \frac{\partial \ell_t}{\partial y_{t,i}} \frac{\partial y_{t,i}}{\partial \boldsymbol{a}_t} = (\boldsymbol{\delta}_{y_{t+1}} - \boldsymbol{p}_t)^T \boldsymbol{W}_{out} \operatorname{diag}\big(\sigma'(\boldsymbol{a}_t)\big) \tag{3.137}$$

$$\Delta_{\boldsymbol{a}_t} \boldsymbol{a}_{t+1} = \boldsymbol{W}_h \operatorname{diag}\big(\sigma'(\boldsymbol{a}_t)\big) \tag{3.138}$$

Now, the parameter gradients can be easily obtained using $\nabla_{\boldsymbol{W}_\cdot} L = \sum_{t,k} \frac{\partial L}{\partial a_{t,k}} \frac{\partial a_{t,k}}{\partial \boldsymbol{W}_\cdot}$ and

$$\frac{\partial a_{t,k}}{\partial \boldsymbol{W}_h} = \frac{\partial}{\partial \boldsymbol{W}_h} \boldsymbol{\delta}_k^T (\boldsymbol{W}_h \boldsymbol{h}_{t-1} + \boldsymbol{W}_{in} \boldsymbol{\delta}_{y_t}) = \boldsymbol{\delta}_k \boldsymbol{h}_{t-1}^T \tag{3.139}$$

$$\frac{\partial a_{t,k}}{\partial \boldsymbol{W}_{in}} = \frac{\partial}{\partial \boldsymbol{W}_{in}} \boldsymbol{\delta}_k^T (\boldsymbol{W}_h \boldsymbol{h}_{t-1} + \boldsymbol{W}_{in} \boldsymbol{\delta}_{y_t}) = \boldsymbol{\delta}_k \boldsymbol{\delta}_{y_t}^T \tag{3.140}$$

Plugging this in, we get

$$\nabla_{\boldsymbol{W}_h} L = \sum_t (\nabla_{\boldsymbol{a}_t} L)\, \boldsymbol{h}_{t-1}^T \tag{3.141}$$

$$\nabla_{\boldsymbol{W}_{in}} L = \sum_t (\nabla_{\boldsymbol{a}_t} L)\, \boldsymbol{\delta}_{y_t}^T \tag{3.142}$$

The output weights are not affected by the recurrent structure. Thus, we only need to accumulate terms

$$\nabla_{\boldsymbol{W}_{out}} \ell_t = (\nabla_{\boldsymbol{z}_t} \ell_t)\, \boldsymbol{h}_t^T \tag{3.143}$$

3.3.4.2 The Collaborative Recurrent Neural Network (C-RNN)

We extend the RNN to the collaborative setting in a straightforward way. We explained previously that our model needs to represent hidden contextual state that can vary per user as a function of a user u's activity. We therefore define our collaborative RNN as a function that maps a user's sequence to a personalized sequence of hidden states:

$$\text{C-RNN}\left([y_1^{(u)}, \ldots, y_{T_u}^{(u)}]\right) = \left[\boldsymbol{h}_1^{(u)}, \ldots, \boldsymbol{h}_{T_u}^{(u)}\right], \quad \boldsymbol{h}_t^{(u)} \in \mathbb{R}^D, \forall u \in \mathbb{U}. \tag{3.144}$$

Using a RNNLM-type network per user would result in several problems: the limitations to individual item libraries and excessive, potentially redundant, parameterization, with potentially insufficient data to learn the parameters effectively. Concretely, we would have to handle $O(2DIU + UD^2)$ parameters, where the problem stems from the IU term as in general $D \ll I$.

We propose the following compromise: the input and output parameter matrices $\boldsymbol{W}_{in}$ and $\boldsymbol{W}_{out}$ can be thought of as real-valued embeddings, akin to latent factors in matrix factorization models [Koren et al., 2009] or word embeddings [Mikolov et al., 2013]. With this interpretation, we assume that such latent factors embody certain global traits of items, valid across users and static on the time-scale of interest, and we attribute the dynamics to users by maintaining per-user the part of the model responsible for capturing the dynamics, i.e., the relatively small matrix $\boldsymbol{W}_h$. The parameterization is thus reduced to a more tolerable[11] $O(2DI + UD^2)$, i.e., $\boldsymbol{\theta} = \left\{\boldsymbol{W}_{in}, \boldsymbol{W}_{out}, \boldsymbol{W}_h^{(1)}, \ldots, \boldsymbol{W}_h^{(U)}\right\}$. Correspondingly, the forward equations become personalized:

$$\begin{aligned}
\boldsymbol{a}_t^{(u)} &= \boldsymbol{W}_h^{(u)} \boldsymbol{h}_{t-1}^{(u)} + \boldsymbol{W}_{in} \boldsymbol{\delta}_{y_t}^{(u)} && (3.145)\\
\boldsymbol{h}_t^{(u)} &= \sigma(\boldsymbol{a}_t^{(u)}) && (3.146)\\
\boldsymbol{z}_t^{(u)} &= \boldsymbol{W}_{out} \boldsymbol{h}_t^{(u)} && (3.147)\\
P(y_t^{(u)} \mid \boldsymbol{y}_{<t}^{(u)}) &= \sigma_m(\boldsymbol{z}_t^{(u)}, y_t^{(u)}). && (3.148)
\end{aligned}$$

Regularization and Learning Algorithm. We train the network by using Algorithm 8 that uses the BPTT procedure from Algorithm 7 as a building block. For the learning rule, we found it more effective to use RMSprop [Tieleman and Hinton, 2012]. For the regularization, we use early stopping. Additionally, we implemented dropout [Srivastava et al., 2014], following the insights about its application to RNNs presented by Zaremba et al. [2014], but found it to not to be effective. Similarly to Jozefowicz et al. [2015], we initialize the weights by randomly drawing each component i.i.d. from $N(0, D^{-1})$.

---

[11] For comparison, a latent factor model would typically have $O(D(I + U))$ parameters.

Algorithm 8 Collaborative RNN Training

```
Require: Sequences $\boldsymbol{y}^{(1)}, \dots, \boldsymbol{y}^{(U)}$, Batch size $B$
Init $\boldsymbol{\theta} = \left\{\boldsymbol{W}_{in}, \boldsymbol{W}_{out}, \boldsymbol{W}_h^{(1)}, \dots, \boldsymbol{W}_h^{(U)}\right\}$
repeat
    $\mathcal{R} = randperm(U)$                    (process users in random order)
    for $u \in \mathcal{R}$ do
        $\boldsymbol{\theta}_u \leftarrow \left\{\boldsymbol{W}_{in}, \boldsymbol{W}_{out}, \boldsymbol{W}_h^{(u)}\right\}$
        $\boldsymbol{W}_{in}, \boldsymbol{W}_{out}, \boldsymbol{W}_h^{(u)} \leftarrow$ processSequence $\left(\boldsymbol{y}^{(u)}, \boldsymbol{\theta}_u, B\right)$          (Algorithm 7)
    end for
until Early Stopping Criterion holds
```

3.3.4.3 Variants

Recently, since the RNNLM was first introduced, great strides have been made in the area of RNNs. Special attention has been paid to the fact that RNNs are known to be difficult to train due to vanishing and exploding gradients [Hochreiter et al., 2001].

Among the approaches to mitigating this issue [Pascanu et al., 2012, Mikolov et al., 2014], architectural modifications to the standard RNN stand out as being easily adoptable and effective in practice. The Long Short-Term Memory (LSTM) cell introduced by Hochreiter and Schmidhuber [1997] is the first and perhaps best-known variant, specifically designed to counteract the vanishing-gradient problem by including a gating mechanism to enable the network to retain information for much longer sequences. This method is especially appealing as it requires virtually no modifications to standard stochastic gradient descent-based learning techniques, but this comes at the cost of an increase in the number of parameters by a constant factor.

In this work, we choose to use the Gated Recurrent Unit [GRU, Cho et al., 2014a], a simplified architecture that is similar in spirit to the LSTM cell and that works well in practice [Jozefowicz et al., 2015]. The forward equations for our Collaborative GRU model are

$$\boldsymbol{w}_t^{(u)} = \sigma_l \left(\boldsymbol{W}_{zh}^{(u)} \boldsymbol{h}_{t-1}^{(u)} + \boldsymbol{W}_{zi} \boldsymbol{\delta}_{y_t}\right), \tag{3.149}$$

$$\boldsymbol{r}_t^{(u)} = \sigma_l \left(\boldsymbol{W}_{rh}^{(u)} \boldsymbol{h}_{t-1}^{(u)} + \boldsymbol{W}_{ri} \boldsymbol{\delta}_{y_t}\right), \tag{3.150}$$

$$\tilde{\boldsymbol{h}}_t^{(u)} = \sigma(\boldsymbol{W}_h(\boldsymbol{r}_t^{(u)} \circ \boldsymbol{h}_{t-1}^{(u)}) + \boldsymbol{W}_{in} \boldsymbol{\delta}_{y_t}), \tag{3.151}$$

$$\boldsymbol{h}_t^{(u)} = (1 - \boldsymbol{w}_t^{(u)}) \circ \boldsymbol{h}_{t-1}^{(u)} + \boldsymbol{w}_t^{(u)} \tilde{\boldsymbol{h}}_t^{(u)}. \tag{3.152}$$

Here, $\sigma_l(\cdot)$ denotes the logistic sigmoid, that maps inputs to $[0, 1]$, such that the vectors $\boldsymbol{w}$ and $\boldsymbol{r}$ can be thought of as implementing a soft gating mechanism. The resulting additive update of the hidden state is more robust than the non-linear update of the basic RNN. The number of parameters is increased by a small factor to $O(4DI + 3UD^2)$.

3.3.4.4 Implementation Details

We implement the learning algorithm by using Theano [Theano Development Team, 2016] and execute the code on NVIDIA Titan X (Maxwell) GPUs on Intel Xeon E5-2680 servers with 256 GB of RAM. As we back-propagate on the fixed-sized sub-segments of the original sequence[12], we can unroll the Theano scan operation[13], trading a minor increase in compilation time for significant speedups in execution.

3.3.5 Empirical Evaluation

In this part, we present our experimental findings. We begin by introducing the datasets that we use in Section 3.3.5.1 and then address the following points. In Section 3.3.5.2, we investigate the influence of the RNN architecture, as well as that of the collaborative aspect of the model in contrast to the user-agnostic version. In Section 3.3.5.3, we show the comparison against various baselines. Lastly, in Section 3.3.5.4 we shed some light on the difficulty of the problem by characterizing the dependence of the performance on properties of the user profile.

3.3.5.1 Datasets

We used two publicly available datasets for our evaluation: Brightkite[14] (BK) and LastFM[15] (LFM).

Brightkite, discontinued in 2011, used to be a location-based social network where users could actively announce (“check in”) their location and find their nearby friends. For our purposes, we focus on the check-in logs consisting of triplets of the form (user id, location id, check-in time).

The LastFM[16] dataset consists of sequences of songs played by a user’s music player collected by using a tracking plug-in. In our evaluation, we consider the sequences of artists.

Errors are measured using the average negative log-likelihood (3.122).

We apply the following preprocessing steps: As Rendle et al. [2010], we start with a 10-core

---

[12] In our experiments we use a window of length 128.

[13] This had a great impact on performance, as we document in http://yjk21.github.io/unrolling.html

[14] https://snap.stanford.edu/data/loc-brightkite.html

[15] http://www.dtic.upf.edu/~ocelma/MusicRecommendationDataset/lastfm-1K.html

[16] http://last.fm

subset[17] and remove pathologically homogeneous user profiles (e.g., overall reporting only a single place but hundreds or thousands of times). We prepare test sets that consist of the last 2.5% of the data clipping their length to the interval $[3, 500]$. Additionally, in the case of LFM, we restrict the maximum sequence length to 5120, because we did not find qualitative differences beyond that length. The set of items $\mathbb{I}$ is taken to be the union of the individual training sequences. The resulting datasets, along with the error of a uniform baseline predictor (3.123), are summarized in Table 3.10. We see that the total number of events of the two datasets differ by an order of magnitude, suggesting that results on the LFM dataset might be more meaningful.

	Users	Items	Events	$E_{\text{Unif.}}$
BK	1679	23 243	599 618	10.05
LFM	954	48 188	4 320 170	10.78

Table 3.10 – Summary of the datasets. $E_{\text{Unif.}}$ is the negative log probability assigned to a sequences by the uniform baseline given in Eq. 3.123.

3.3.5.2 Comparison of RNN Variants

The aforementioned difficulties in training RNNs has sparked numerous architectural variants of the basic tanh RNN. Understanding their effect on performance is an interesting aspect in its own right with major relevance for practitioners. Therefore, e.g., Jozefowicz et al. [2015] conducted large empirical studies comparing many thousands of variants, focusing on modifications of the original LSTM cell [Hochreiter and Schmidhuber, 1997] in comparison to the GRU [Cho et al., 2014b]. Although they did not identify a single best model, they corroborated the observation that such advanced RNN architectures as the GRU and LSTM outperform the tanh RNN. We choose GRUs over LSTMs for their simpler formulation and parameterization and state-of-the-art performance for many tasks, and we compared it to the collaborative tanh RNN (C-tanh). We found that the advantage of the gated version over the basic RNN carries over to our setup.

Furthermore, we compared our collaborative model to the user-agnostic GRU model that we will refer to as pooled RNN (P-RNN). It is conceivable that a RNN with a large-enough number of hidden units can eventually capture even distinct, user-specific patterns. However, we found that for our problem the collaborative RNN, tailored to the structure of the problem, is far more efficient in terms of computation. This is due to the fact that, whilst the collaborative RNN maintains recurrent connection matrices per-user thus more parameters in absolute numbers, each of these matrices can be of smaller dimension. As the learning algorithms are otherwise virtually equivalent, the pooled model incurs a major performance hit. At this point, we do not rule out that the P-RNN can match the collaborative version. But with the large running times required,

---

[17] We keep only users with at least 10 observations and items that were consumed at least 10 times.

due to the large number of hidden units required to come close to the performance of the collaborative model, thorough model selection for even higher-dimensional models becomes extremely cumbersome. We summarize these findings in Figure 3.8.

In a side note, another advantage of the collaborative model, apart from the computational aspect, is that it meets the potential requirement of discovering user specific embeddings.

	P-GRU	C-tanh	C-GRU
BK	6.45	6.08	6.02
LFM	4.58	4.90	4.51

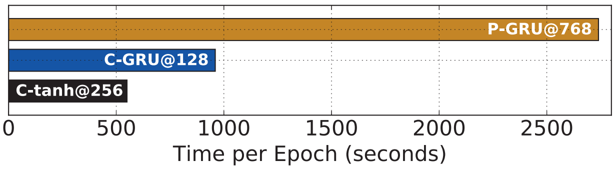


Figure 3.8 – **Left:** Errors of different RNN models on the two datasets. **Right:** Running time comparison of a single Epoch of different RNN models on the larger LFM dataset, demonstrating the difference in the number of hidden units for each model to perform well. We found that the P-RNN achieves a similar performance requiring an excessive amount of hidden units ($D = 768$). The negative impact on running time prevented us from exploring even higher-dimensional models.

3.3.5.3 Baseline Comparison

In this part, we present the comparison of our model against the various baselines that we introduced earlier. The static methods, i.e., those that do not take into account time (unigram, matrix factorization), do not perform very well, whereas the simple bigram model does (Table 3.11).

For HMM we used the GHMM[18] library and for MF we used the parallel coordinate descent algorithm by Yu et al. [2012] implemented in the libpmf[19] library.

	1-gram	MF	HMM	2-gram	C-GRU
BK	9.53	9.40	8.81	6.73	6.02
LFM	8.60	8.86	7.66	5.87	4.51

Table 3.11 – Comparison with baseline methods on different datasets.

3.3.5.4 Characterization of Error

To conclude this section, we examine the error incurred on individual users and relate it to a notion of difficulty. An intuitive way to quantify the difficulty of predicting the behavior of a user is to use the Shannon entropy of the empirical distribution of items. A lower value corresponds to a lower difficulty with a lower bound of 0, which corresponds to a user having consumed the same item at all times. We divided users into three equally sized

---

[18] http://ghmm.org
[19] http://www.cs.utexas.edu/~rofuyu/libpmf/

bins, according to the entropies of their sequences; in Figure 3.9 we show the distribution of errors in each bin. Unsurprisingly, there is a distinct correlation between our proxy for difficulty and the median error. We see that the C-RNN not only outperforms the closest competing baseline on average, but also in terms of variability of the errors over all bins.

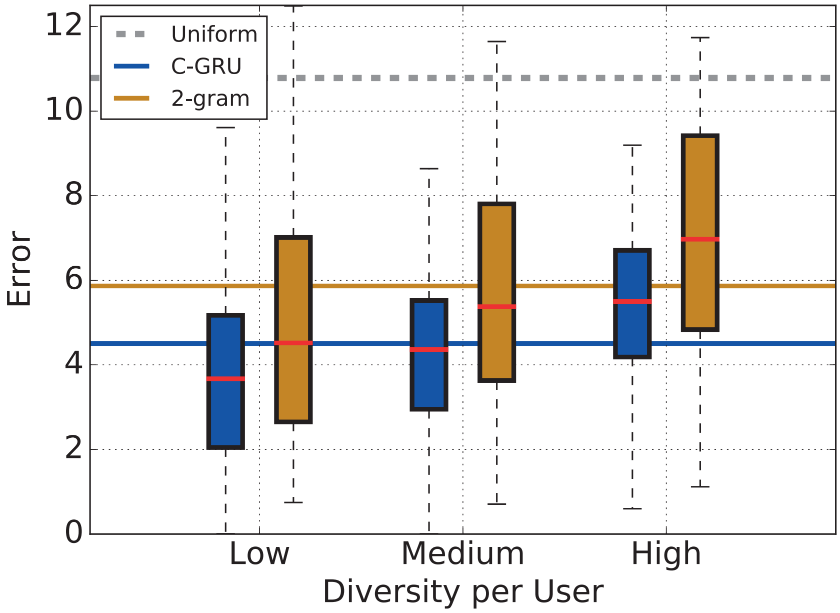


Figure 3.9 – LFM: Distribution of errors per user for three different difficulty regimes. These regimes were found by dividing the users into three equally-sized groups according to the entropy of their sequences. Solid lines indicate the mean errors across all users. The dashed line is the error associated with the uniform baseline. Errors are correlated with difficulty. The RNN-based model appears to not only incur a lower error on average, but also to perform more consistently.

3.3.6 Discussion

We presented a novel collaborative RNN-based model to analyze sequences of user activity useful for modern recommender systems based on specific assumptions and modeling goals. We empirically showed that these models consistently outperform straightforward baseline methods on two real-world datasets. These results are to be taken with multiple grains of salt. It is not entirely clear, what the model is learning, especially for the artist prediction task, where it may be very well the case, that at least parts of the sequences were generated by player shuffling songs randomly. We tried to shed light on this issue by measuring sequence entropy. Therefore, additional experiments with datasets where user intent is more clearly displayed might be necessary. Furthermore, we also tried to visualize the high-dimensional item features but did not found any particular structure, e.g. indicating genre.

Scalability to large datasets can be improved by using techniques such as those of Chen et al. [2015] for the sub-linear evaluation of the output layer in case of large item sets, a major factor in the overall running time, and those of Recht et al. [2011] to parallelize training, which can work due to sparse updates of the parameters.

We can apply tensor factorization of the $O(UD^2)$ transition tensor to better handle sparsity, similar to the factorization of the transition tensor in the personalized Markov model, described by Rendle et al. [2010].

A drawback of our approach, is the lack of representing the time passed between events One way to make temporal information available, is to augment the input by the time passed since the last event. Another way is to introduce a special symbol (similar to an end-of-sequence or unknown-word symbol in language models) to indicate a gap.

4 Concluding Remarks

In this dissertation we discussed applications of approximate inference and learning for probabilistic models relevant for two particular application.

In Chapter 2, we discussed inference problems mainly motivated by the problem of reconstructing images from an underdetermined system of linear measurements that are corrupted by noise, that are formally also closely related to supervised learning problems. We studied in Section 2.1 rectified-linear Poisson regression, a particular likelihood model for counts, useful for modeling noise in photon-limited acquisition scenarios. We described an efficient dynamic program for computing Gaussian expectations of this likelihood, enabling us to run Expectation Propagation faster and more robustly than with a generic quadrature implementation. We experimentally found that properly modeling the distinctly non-Gaussian noise characteristics in low intensity regimes is important for reconstruction performance. Next, in Section 2.2, we modeled the hierarchical correlations between the coefficients of an orthonormal multi-scale Wavelet transform using a tree-structured super-Gaussian mixture model. To use this model as an image prior for image denoising and inpainting tasks, we derived an approximate inference algorithm, able to infer full posterior covariances for realistic image sizes. We confirmed that using additional structure helps to regularize difficult reconstruction problems if a proper probabilistic inference method is used to deal with the increased latent space.

In Chapter 3, we discussed the problem of inferring user preferences for items from three different types of implicit feedback: aggregate counts of users interacting with items, binary outcomes of pairwise preference statements and user activity logs at the granularity of single events. In Section 3.1 we modeled count data using a latent Gaussian bi-linear model with Poisson likelihood. For this model, we developed an approximate inference algorithm based on the variational Gaussian lower-bounding technique and derive a closed-form constrained variational objective, that can be optimized by a simple alternating scheme. Our experiments confirmed the advantages of probabilistic methods previously described in related studies and show that these advantages are available at a cost only marginally larger than for point estimation. In Section 3.2, we addressed the problem of inferring user preferences from the binary outcomes of pairwise preference statements. We proposed a utility model that combines a low-rank bi-linear model with non-parametric item-feature regression. We illustrated the difficulty of learning such a model due to latent couplings and presented an approximate variational Expectation Maximization algorithm that mitigates these computational challenges. We extended the notion to the pairwise case, that latent bilinear models are an excellent representation compared to purely content-based models. In Section 3.3, we aimed at modeling sequences of user activity at the granularity of single interaction events in contrast to the aggregate counts in Section 3.1. Routinely gathered in the background at a large scale in many applications, such sequences can reveal temporal and contextual aspects of user behavior through recurrent patterns, which we tried to uncover by using a collaborative sequence model based on recurrent neural networks. As a concession to scale and complexity of the model, we used a generative likelihood model for sequences, but without representing uncertainty in the weights or hidden states and trained the network using simple maximum likelihood estimation. We found that this model performed favorably in two different sequence-prediction tasks compared to various simple baselines.

Looking back, we began by introducing probabilistic models as having long been recognized as a versatile and expressive framework for data analysis. Making inferences in probabilistic models in a principled way, however, often entails high-dimensional integration for which no analytic expressions exist. This leaves us in a trade-off situation, where we can either compromise by neglecting uncertainty in latent quantities or make only approximate inferences. For most of this dissertation we took the second approach. We advocated deterministic approximations, in particular Gaussian approximations, but soon found ourselves confronted with several more trade-offs on different levels about the type of approximation, factorization assumptions or more subtle points, such as the number of iterations for numerical primitives. From an optimistic perspective, this leaves us with the freedom and the fine-grained control necessary to adapt these methods to our computational budget, often enabling us to begin with methods, that are not much more costly than point estimation.

Arguably the biggest benefit of a probabilistic perspective is, however, the mindset that it may help cultivating. It forces us to make explicit our assumptions and encourages us to reason in terms of uncertainties. Moreover, by conditioning on the data we have observed, we acknowledge that our inferences are very much tied to the particular sample with all its imperfections. This seems good practice, especially when trying to predict phenomena related to something as complex, and even erratic at times, as human behavior.

Bibliography

H. Attias. A Variational Bayesian Framework for Graphical Models. In *NIPS 12*, pages 209–215, 2000.

D. Barber. *Bayesian Reasoning and Machine Learning.* Cambridge University Press, 2012.

M. Beal. *Variational Algorithms for Approximate Bayesian Inference*. PhD thesis, Gatsby Unit, 2003.

J. Bennett and S. Lanning. The Netflix Prize. In *Proceedings of KDD Cup and Workshop*, 2007.

M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image Inpainting. In *ACM SIGGRAPH 27*, pages 417–424, 2000.

D. Bertsekas. *Nonlinear Programming.* Athena Scientific, 2nd edition, 1999.

A. Birlutiu, P. Groot, and T. Heskes. Multi-task Preference Learning with an Application to Hearing Aid Personalization. *Neurocomputing*, 73(79):1177–1185, 2010.

C. M. Bishop. Bayesian PCA. *NIPS 13*, pages 382–388, 1999.

C. M. Bishop. *Pattern Recognition and Machine Learning.* Springer, 1st edition, 2006.

C. M. Bishop, D. Spiegelhalter, and J. Winn. VIBES: A Variational Inference Engine for Bayesian Networks. In *NIPS 15*, pages 777–784, 2002.

D. Blei, A. Ng, and M. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

E. Bonilla, S. Guo, and S. Sanner. Gaussian Process Preference Elicitation. In *NIPS 23*, pages 262–270, 2010.

S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.

E. Brochu, N. De Freitas, and A. Ghosh. Active Preference Learning with Discrete Choice Data. In *NIPS 20*, pages 409–416. 2008.

T. Broderick, N. Boyd, A. Wibisono, A. Wilson, and M. Jordan. Streaming Variational Bayes. In *NIPS 26*, pages 1727–1735, 2013.

P. F. Brown, P. V. Desouza, R. L. Mercer, V. J. Della Pietra, and J. C. Lai. Class-based n-gram Models of Natural Language. *Computational Linguistics*, 18(4), 1992.

L. Buesing, M. Sahani, and J. H. Macke. Spectral Learning of Linear Dynamics from Generalised-linear Observations with Application to Neural Population Data. In *NIPS 26*, pages 1682–1690, 2012.

W. Buntine. Variational Extensions to EM and Multinomial PCA. In *ECML 13*, pages 23–34, 2002.

C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to Rank using Gradient Descent. In *ICML 22*, pages 89–96. ACM, 2005.

E. Candès, J. Romberg, and T. Tao. Robust Uncertainty Principles: Exact Signal Reconstruction from Highly Incomplete Frequency Information. *IEEE Transactions on Information Theory*, 52(2):489–509, 2006.

M. Carlavan and L. Blanc-Féraud. Sparse Poisson Noisy Image Deblurring. *IEEE Transactions on Image Processing*, 21(4):1834–1846, 2012.

E. Challis and D. Barber. Concave Gaussian Variational Approximations for Inference in Large-Scale Bayesian Linear Models. In *AISTATS 14*, pages 199–207, 2011.

C. Chen and J. Huang. Compressive Sensing MRI with Wavelet Tree Sparsity. In *NIPS 26*, pages 1115–1123, 2012.

X. Chen, X. Liu, M. Gales, and P. C. Woodland. Recurrent Neural Network Language Model Training with Noise Contrastive Estimation for Speech Recognition. In *IEEE ICASSP*, pages 5411–5415, 2015.

E. C. Chi and T. G. Kolda. On Tensors, Sparsity, and Nonnegative Factorizations. *SIAM Journal on Matrix Analysis and Applications*, 33(4):1272–1299, 2012.

K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. In *Proceedings of the Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, 2014a.

K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *EMNLP*, pages 1724–1734, 2014b.

W. Chu and Z. Ghahramani. Gaussian Processes for Ordinal Regression. *Journal of Machine Learning Research*, 6(7):1019–1041, 2005.

M. Collins, S. Dasgupta, and R. Schapire. A Generalization of Principal Components Analysis to the Exponential Family. In *NIPS 14*, pages 617–624, 2002.

M. Crouse, R. Nowak, and R. Baraniuk. Wavelet-based Statistical Signal Processing using Hidden Markov Models. *IEEE Transactions on Signal Processing*, 46:886–902, 1998.

A. Dempster, N. Laird, and D. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of Royal Statistical Society: Series B*, 39:1–38, 1977.

P. Diggle, P. Moraga-Serrano, B. Rowlingson, and B. Taylor. Spatial and Spatio-temporal Log-Gaussian Cox processes: Extending the Geostatistical Paradigm. *Statistical Science*, 28(4):542–563, 2013.

D. Donoho. Compressed Sensing. *IEEE Transactions on Information Theory*, 52(4): 1289–1306, 2006.

D. Donoho, A. Maleki, and A. Montanari. Message-passing Algorithms for Compressed Sensing. *Proceedings of the National Academy of Sciences*, 106(45):18914–18919, 2009.

N. Du, Y. Wang, N. He, J. Sun, and L. Song. Time-Sensitive Recommendation From Recurrent User Activities. In *NIPS 28*, pages 3492–3500. 2015.

M. Duarte, M. Davenport, D. Takhar, J. Laska, T. Sun, K. Kelly, and R. Baraniuk. Single Pixel Imaging via Compressive Sampling. *IEEE Signal Processing Magazine*, 25(2): 83–91, 2008.

F. X. Dupé, M. J. Fadili, and J. L. Starck. Deconvolution of Confocal Microscopy Images using Proximal Iteration and Sparse Representations. In *5th IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pages 736–739. IEEE, 2008.

J. L. Elman. Finding Structure in Time. *Cognitive science*, 14(2):179–211, 1990.

Y. Gal and Z. Ghahramani. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. In *NIPS 29*, pages 1019–1027, 2016.

A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis.* Texts in Statistical Sciences. Chapman & Hall, 1st edition, 1995.

M. van Gerven, B. Cseke, F. de Lange, and T. Heskes. Efficient Bayesian Multivariate fMRI Analysis using a Sparsifying Spatio-Temporal Prior. *Neuroimage*, 50:150–161, 2010.

S. Gerwinn, J. Macke, M. Seeger, and M. Bethge. Bayesian Inference for Spiking Neuron Models with a Sparsity Prior. In *NIPS 20*, pages 529–536, 2008.

S. Gerwinn, J. Macke, and M. Bethge. Bayesian Inference for Generalized Linear Models for Spiking Neurons. *Frontiers in Computational Neuroscience*, 4, 2010.

Z. Ghahramani. Probabilistic Machine Learning and Artificial Intelligence. *Nature*, 521 (7553):452–459, 2015.

M. Girolami. A Variational Method for Learning Sparse and Overcomplete Representations. *Neural Computation*, 13(11):2517–2532, 2001.

X. Glorot, A. Bordes, and Y. Bengio. Deep Sparse Rectifier Neural Networks. In *AISTATS 15*, number 106, 2011.

A. Graves. Practical Variational Inference for Neural Networks. In *NIPS 24*, pages 2348–2356, 2011.

A. Graves, A. Mohamed, and G. Hinton. Speech Recognition with Deep Recurrent Neural Networks. In *IEEE ICASSP*, pages 6645–6649, 2013.

Grouplens. HetRec2011, 2011. URL http://grouplens.org/datasets/hetrec-2011.

L. He and L. Carin. Exploiting Structure in Wavelet-Based Bayesian Compressive Sensing. *IEEE Transactions on Signal Processing*, 57(9):3488–3497, 2009.

L. He, H. Chen, and L. Carin. Tree-Structured Compressive Sensing with Variational Bayesian Analysis. *IEEE Sig. Proc. Letters*, 17(3):233–236, 2010.

R. Herbrich, T. Minka, and T. Graepel. TrueSkill: A Bayesian Skill Rating System. In *NIPS 19*, pages 569–576, 2007.

S. Hochreiter and J. Schmidhuber. Long Short-term Memory. *Neural Computation*, 9(8): 1735–1780, 1997.

S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient Flow in Recurrent Nets: The Difficulty of Learning Long-term Dependencies, 2001.

M. Hoffman, D. Blei, C. Wang, and J. Paisley. Stochastic Variational Inference. *Journal of Machine Learning Research*, 14:1303–1347, 2013.

N. Houlsby, J. Hernandez-Lobato, F. Huszar, and Z. Ghahramani. Collaborative Gaussian Processes for Preference Learning. In *NIPS 25*, pages 2105–2113, 2012.

A. Ilin and T. Raiko. Practical Approaches to Principal Component Analysis in the Presence of Missing Values. *Journal of Machine Learning Research*, 11(7):1957–2000, 2010.

T. Jaakkola. *Variational Methods for Inference and Estimation in Graphical Models.* PhD thesis, MIT, 1997.

T. Jaakkola and M. Jordan. Improving the Mean Field Approximation via the Use of Mixture Distributions. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 163–173. 1998.

R. G. Jarrett. A Note on the Intervals between Coal-mining Disasters. *Biometrika*, 66(1): 191–193, 1979.

M. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul. An Introduction to Variational Methods in Graphical Models. In M. I. Jordan, editor, *Learning in Graphical Models*. 1997.

R. Jozefowicz, W. Zaremba, and I. Sutskever. An Empirical Exploration of Recurrent Network Architectures. In *ICML 32*, 2015.

Kaggle. Million Song Data Set Challenge, 2012. URL https://www.kaggle.com/c/msdchallenge/data.

T. Kamishima. Nantonac Collaborative Filtering: Recommendation based on Order Responses. In *SIGKDD 9*, pages 583–588, 2003.

M. E. Khan, Y.-J. Ko, and M. Seeger. Scalable Collaborative Bayesian Preference Learning. In *AISTATS 17*, 2014.

A. Klami. Polya-gamma Augmentations for Factor Models. In *ACML 6*, 2014.

Y.-J. Ko and M. E. Khan. Variational Gaussian Inference for Bilinear Models of Count Data. In *ACML 6*, 2014.

Y.-J. Ko and M. Seeger. Large-scale Variational Bayesian Inference for Structured Scale-mixture Models. In *ICML 29*, 2012.

Y.-J. Ko and M. Seeger. Expectation Propagation for Rectified Linear Poisson Regression. In *ACML 7*, 2015.

Y.-J. Ko, L. Maystre, and M. Grossglauser. Collaborative Recurrent Neural Networks for Dynamic Recommender Systems. In *ACML 8*, 2016.

N. Koenigstein, G. Dror, and Y. Koren. Yahoo! Music Recommendations: Modeling Music Ratings with Temporal Dynamics and Item Taxonomy. In *ACM RecSys 5*, pages 165–172, 2011.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 1st edition, 2009.

Y. Koren. Collaborative Filtering with Temporal Dynamics. *Communications of the ACM*, 53(4):89–97, 2010.

Y. Koren, R. Bell, C. Volinsky, et al. Matrix Factorization Techniques for Recommender Systems. *Computer*, 42(8):30–37, 2009.

A. Kucukelbir, R. Ranganath, A. Gelman, and D. Blei. Automatic Variational Inference in Stan. In *NIPS 28*, pages 568–576, 2015.

M. Kuss and C. Rasmussen. Assessing Approximate Inference for Binary Gaussian Process Classification. *Journal of Machine Learning Research*, 6:1679–1704, 2005.

M. Kuusela and V. M. Panaretos. Empirical Bayes Unfolding of Elementary Particle Spectra at the Large Hadron Collider. *The Annals of Applied Statistics*, 9(3):1671–1705, 2015.

P. Lamere. The LastFM ArtistTags2007 Dataset, 2008. URL http://musicmachinery.com/2010/11/10/lastfm-artisttags2007.

D. Lee and H. Seung. Learning the Parts of Objects by Non-negative Matrix Factorization. *Nature*, 401:788–791, 1999.

A. Levin, Y. Weiss, F. Durand, and W. Freeman. Understanding and Evaluating Blind Deconvolution Algorithms. In *CVPR*, pages 1964–1971, 2009.

A. Levin, Y. Weiss, F. Durand, and W. T. Freeman. Understanding Blind Deconvolution Algorithms. *IEEE PAMI*, 33(12):2354–2367, 2011.

Y. Lim and Y.-W. Teh. Variational Bayesian Approach to Movie Rating Prediction. In *Proceedings of KDD Cup and Workshop*, 2007.

Tie-Yan Liu. Learning to Rank for Information Retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

L. B. Lucy. An Iterative Technique for the Rectification of Observed Distributions. *The Astronomical Journal*, 79:745, 1974.

A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier Nonlinearities Improve Neural Network Acoustic Models. In *ICML 30*, 2013.

D. MacKay. Bayesian Interpolation. *Neural Computation*, 4(3):415–447, 1992.

S. Mallat. *A Wavelet Tour of Signal Processing.* Academic press, 1999.

P. McCullach and J.A. Nelder. *Generalized Linear Models.* Number 37 in Monographs on Statistics and Applied Probability. Chapman & Hall, 1st edition, 1983.

X. Meng, S. Wu, L. Kuang, and J. Lu. An Expectation Propagation Perspective on Approximate Message Passing. *IEEE Signal Processing Letters*, 22(8):1194–1197, 2015.

T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur. Recurrent Neural Network based Language Model. In *Interspeech*, volume 2, 2010.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space. *ICLR Workshop*, 2013.

T. Mikolov, A. Joulin, S. Chopra, M. Mathieu, and M. A. Ranzato. Learning Longer Memory in Recurrent Neural Networks. *arXiv preprint arXiv:1412.7753*, 2014.

B. N. Miller, I. Albert, S. K. Lam, J. A. Konstan, and J. Riedl. MovieLens Unplugged: Experiences with an Occasionally Connected Recommender System. In *IUI 8*, 2003.

T. Minka. *A Family of Algorithms for Approximate Bayesian Inference*. PhD thesis, MIT, January 2001a.

T. Minka. Expectation Propagation for Approximate Bayesian Inference. In *Uncertainty in AI 17*, 2001b.

T. Minka. Divergence Measures and Message Passing. Technical Report MSR-TR-2005-173, Microsoft Research, Cambridge, 2005.

A. Mnih and Y. W. Teh. Learning Label Trees for Probabilistic Modelling of Implicit Feedback. In *NIPS 25*, pages 2825–2833, 2012.

S. Mohamed, K. Heller, and Z. Ghahramani. Bayesian Exponential Family PCA. In *NIPS 21*, pages 1089–1096, 2008.

K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

S. Nakajima and M. Sugiyama. Analysis of Empirical MAP and Empirical Partially Bayes: Can They be Alternatives to Variational Bayes? In *AISTATS 17*, pages 20–28, 2014.

S. Nakajima, M. Sugiyama, S. D. Babacan, and R. Tomioka. Global Analytic Solution of Fully-observed Variational Bayesian Matrix Factorization. *Journal of Machine Learning Research*, 14(Jan):1–37, 2013.

R. M. Neal. Monte Carlo Implementation of Gaussian Process Models for Bayesian Classification and Regression. Technical Report 9702, Department of Statistics, University of Toronto, January 1997.

H. Nickisch. *Bayesian Inference and Experimental Design for Large Generalised Linear Models*. PhD thesis, 2010.

H. Nickisch and C. Rasmussen. Approximations for Binary Gaussian Process Classification. *Journal of Machine Learning Research*, 9:2035–2078, 2008.

B. Olshausen and D. Field. Emergence of Simple-Cell Receptive Field Properties by Learning a Sparse Code for Natural Images. *Nature*, 381:607–609, 1996.

B. Olshausen and D. Field. Sparse Coding with an Overcomplete Basis Set: A Strategy employed by V1? *Vision Research*, 37:3311–3325, 1997.

M. Opper and C. Archambeau. The Variational Gaussian Approximation Revisited. *Neural Computation*, 21(3):786–792, 2009.

M. Opper and O. Winther. Expectation Consistent Approximate Inference. *Journal of Machine Learning Research*, 6:2177–2204, 2005.

M. Opper, U. Paquet, and O. Winther. Perturbative Corrections for Approximate Inference in Gaussian Latent Variable Models. *Journal of Machine Learning Research*, 14(1):2857–2898, 2013.

J. Palmer, D. Wipf, K. Kreutz-Delgado, and B. Rao. Variational EM Algorithms for Non-Gaussian Latent Variable Models. In *NIPS 18*, pages 1059–1066, 2006.

L. Paninski. Maximum Likelihood Estimation of Cascade Point-Process Neural Encoding Models. *Network: Computation in Neural Systems*, 15:243–262, 2004.

G. Papandreou and A. Yuille. Gaussian Sampling by Local Perturbations. In *NIPS 23*, pages 1858–1866, 2010.

G. Papandreou and A. Yuille. Efficient Variational Inference in Large-scale Bayesian Compressed Sensing. In *ICCV Workshop*, pages 1332–1339. IEEE, 2011.

G. Papandreou, P. Maragos, and A. Kokaram. Image Inpainting with a Wavelet Domain Hidden Markov Tree Model. In *IEEE ICASSP*, pages 773–776, 2008.

M. Park and J. W. Pillow. Bayesian Inference for Low-rank Spatiotemporal Neural Receptive Fields. In *NIPS 26*, pages 2688–2696, 2013.

M. Park, J. P. Weller, G. D. Horwitz, and J. W. Pillow. Bayesian Active Learning of Neural Firing Rate Maps with Transformed Gaussian Process Priors. *Neural Computation*, 26 (8):1519–41, 2014.

R. Pascanu, T. Mikolov, and Y. Bengio. On the Difficulty of Training Recurrent Neural Networks. *arXiv preprint arXiv:1211.5063*, 2012.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann, 1988.

T. Pham, V. Bluche, C. Kermorvant, and J. Louradour. Dropout Improves Recurrent Neural Networks for Handwriting Recognition. In *ICFHR 14*, pages 285–290, 2014.

J. W. Pillow. Likelihood-based Approaches to Modeling the Neural Code. *Bayesian Brain: Probabilistic Approaches to Neural Coding*, pages 53–70, 2007.

J. Portilla, V. Strela, M. Wainwright, and E. Simoncelli. Image Denoising using Gaussian Scale Mixtures in the Wavelet Domain. *IEEE Transactions on Image Processing*, 12: 1338–1351, 2003.

L. R. Rabiner and B. H. Juang. An Introduction to Hidden Markov Models. *IEEE ASSP Magazine*, pages 4–15, January 1986.

R. Ranganath, S. Gerrish, and D. Blei. Black Box Variational Inference. In *AISTATS 17*, 2014.

C. E. Rasmussen and H. Nickisch. Gaussian Processes for Machine Learning (GPML) Toolbox. *Journal of Machine Learning Research*, 11(Nov):3011–3015, 2010.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning.* MIT Press, 2006.

B. Recht, C. Re, S. Wright, and F. Niu. Hogwild: A Lock-free Approach to Parallelizing Stochastic Gradient Descent. In *NIPS 24*, pages 693–701, 2011.

S. Rendle. Time-variant Factorization Models. In *Context-Aware Ranking with Factorization Models.* 2010.

S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Uncertainty in AI 25*, pages 452–461, 2009.

S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Factorizing Personalized Markov Chains for Next-basket Recommendation. In *WWW 19*, 2010.

W. H. Richardson. Bayesian-based iterative method of image restoration. *Journal of the Optical Society of America*, 62(1):55–59, 1972.

R. Rockafellar. *Convex Analysis.* Princeton University Press, 1970.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning Representations by Back-propagating Errors. *Cognitive Modeling*, 5(3), 1988.

R. Salakhutdinov and A. Mnih. Probabilistic Matrix Factorization. In *NIPS 20*, pages 1257–1264, 2008a.

R. Salakhutdinov and A. Mnih. Bayesian Probabilistic Matrix Factorization using Markov Chain Monte Carlo. In *ICML 25*, pages 880–887, 2008b.

T. Salimans, U. Paquet, and T. Graepel. Collaborative Learning of Preference Relations. In *ACM RecSys 6*, 2012.

C. Sanderson. Armadillo: An Open Source C++ Linear Algebra Library for Fast Prototyping and Computationally Intensive Experiments. Technical report, 2010.

L. Saul, T. Jaakkola, and M. I. Jordan. Mean Field Theory for Sigmoid Belief Networks. *Journal of AI Research*, 4:61–76, 1996.

B. Schölkopf and A. Smola. *Learning with Kernels.* MIT Press, 1st edition, 2002.

M. Seeger. Bayesian Model Selection for Support Vector Machines, Gaussian Processes and Other Kernel Classifiers. In *NIPS 12*, pages 603–609, 2000.

M. Seeger. Bayesian Inference and Optimal Design for the Sparse Linear Model. *Journal of Machine Learning Research*, 9:759–813, 2008.

M. Seeger and G. Bouchard. Fast Variational Bayesian Inference for Non-Conjugate Matrix Factorization Models. In *AISTATS 15*, 2012.

M. Seeger and H. Nickisch. Compressed Sensing and Bayesian Experimental Design. In *ICML 25*, pages 912–919, 2008.

M. Seeger and H. Nickisch. Large Scale Bayesian Inference and Experimental Design for Sparse Linear Models. *SIAM Journal of Imaging Sciences*, 4(1):166–199, 2011a.

M. Seeger and H. Nickisch. Fast Convergent Algorithms for Expectation Propagation Approximate Bayesian Inference. In *AISTATS 14*, 2011b.

E. Simoncelli. Modeling the joint statistics of images in the Wavelet domain. In *Proceedings 44th SPIE*, pages 188–195, 1999.

S. Som and P. Schniter. Compressive Imaging using Approximate Message Passing and a Markov-tree Prior. *IEEE Transactions on Signal Processing*, 60(7):3439–3448, 2012.

N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(1), 2014.

J.-L. Starck and F. Murtagh. *Astronomical Image and Data Analysis*. Springer, 2002.

I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to Sequence Learning with Neural Networks. In *NIPS 27*, pages 3104–3112, 2014.

G. Takács and D. Tikk. Alternating Least Squares for Personalized Ranking. In *ACM RecSys 6*, pages 83–90, 2012.

Theano Development Team. Theano: A Python Framework for Fast Computation of Mathematical Expressions. *arXiv e-prints*, abs/1605.02688, 2016.

R. Tibshirani. Regression Shrinkage and Selection via the Lasso. *Journal of Royal Statistical Society: Series B*, 58:267–288, 1996.

T. Tieleman and G. E. Hinton. Lecture 6.5-rmsprop: Divide the Gradient by a Running Average of its Recent Magnitude. *COURSERA: Neural Networks for Machine Learning*, 2012.

M. Tipping and C. Bishop. Mixtures of Probabilistic Principal Component Analyzers. *Neural Computation*, 11(2):443–482, 1998.

M. Tipping and C. Bishop. Probabilistic Principal Component Analysis. *Journal of Royal Statistical Society: Series B*, 61(3):611–622, 1999.

M. Titsias and M. Lázaro-Gredilla. Spike and Slab Variational Inference for Multi-Task and Multiple Kernel Learning. In *NIPS 24*, pages 2339–2347, 2011.

J. Vanhatalo, V. Pietiläinen, and A. Vehtari. Approximate Inference for Disease Mapping with sparse Gaussian Processes. *Statistics in Medicine*, 29(15):1580–1607, 2010.

J. Vanhatalo, J. Riihimäki, J. Hartikainen, P. Jylänki, V. Tolvanen, and A. Vehtari. GPstuff: Bayesian modeling with Gaussian Processes. *Journal of Machine Learning Research*, 14(4):1175–1179, 2013.

M. J. Wainwright and M. I. Jordan. Graphical Models, Exponential Families, and Variational Inference. *FTML*, 1(1–2):1–305, 2008.

P. Wang, J. Guo, Y. Lan, J. Xu, S. Wan, and X. Cheng. Learning Hierarchical Representation Model for NextBasket Recommendation. In *ACM SIGIR 38*, 2015.

P. J. Werbos. Backpropagation Through Time: What it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

R. J. Williams and D. Zipser. Gradient-based Learning Algorithms for Recurrent Networks and their Computational Complexity. *Back-propagation: Theory, Architectures and Applications*, 1995.

L. Xiong, X. Chen, T.-K. Huang, J. G. Schneider, and J. G. Carbonell. Temporal Collaborative Filtering with Bayesian Probabilistic Tensor Factorization. *SDM*, 2010.

X. Yi, L. Hong, E. Zhong, N. N. Liu, and S. Rajan. Beyond Clicks: Dwell Time for Personalization. In *ACM RecSys 8*, 2014.

H.-F. Yu, C.-J. Hsieh, I. Dhillon, et al. Scalable Coordinate Descent Approaches to Parallel Matrix Factorization for Recommender Systems. In *IEEE ICDM 12*, 2012.

K. Yu, S. Zhu, J. Lafferty, and Y. Gong. Fast Nonparametric Matrix Factorization for Large-Scale Collaborative Filtering. In *ACM SIGIR 32*, pages 211–218, 2009.

A. Yuille and A. Rangarajan. The Concave-Convex Procedure. *Neural Computation*, 15 (4):915–936, 2003.

W. Zaremba, I. Sutskever, and O. Vinyals. Recurrent Neural Network Regularization. *arXiv preprint arXiv:1409.2329*, 2014.

M. D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, et al. On Rectified-Linear Units for Speech Processing. In *IEEE ICASSP*, pages 3517–3521. IEEE, 2013.

Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan. Large-scale Parallel Collaborative Filtering for the Netflix Prize. In *Algorithmic Aspects in Information and Management.* Springer, 2008.

Young-Jun Ko

EPFL IC ISC LCA4, INR 140 (Batiment INR), Station 14
CH-1015 Lausanne, Switzerland

✉ youngjun.ko@gmail.com | 📱 +41 78 63 33 55 8 | 🌐 yjk21.github.io

Education

02.2011 - 02.2017	**PhD at EPFL, Switzerland** Supervisors: Prof. Matthias Grossglauser, Dr. Matthias W. Seeger Research: Probabilistic Modeling and Variational Approximate Inference applied to Low Level Computer Vision and Collaborative Filtering, Recurrent Neural Networks
10.2007 - 06.2010	**MSc in Comp. Sci. at Saarland University, Germany (GPA[1]: 1.3)** Supervisor: Prof. Matthias Hein Topic: Stability of Feature Selection Methods
10.2002 - 09.2005	**Dipl.-Inf.(BA)/BSc(OU) in Applied Computer Science at the University of Cooperative Education, Germany (GPA: 1.4)** Topic: Rule-based Invoicing of IBM Infrastructure Services for Deutsche Bank
1993 - 2002	**Abitur at Lessing Gymnasium, Germany (GPA: 1.3)**

Publications

2016	Ko, L. Maystre, M. Grossglauser, Collaborative Recurrent Neural Networks for Dynamic Recommender Systems, *ACML 2016*
2015	Ko, M. Seeger, Expectation Propagation for Rectified Linear Poisson Regression, *ACML 2015*
2014	Ko, M. E. Khan, Variational Gaussian Inference for Bilinear Models of Count Data, *ACML 2014* M. E. Khan, Ko, M. Seeger, Scalable Collaborative Bayesian Preference Learning, *AISTATS 2014*
2013	M. E. Khan, Ko, M. Seeger, Scalable Bayesian Preference Learning, *NIPS 2013* Workshop on Personalization 2013
2012	Ko, M. Seeger, Large Scale Variational Bayesian Inference for Structured Scale Mixture Models, *ICML 2012*

Scholarships

04.2015	Google internship offer, Mountain View, California
07.2010 - 09.2010	PhD Scholarship of the International Max Planck Research School for Computer Science (Cancelled due to the move to EPFL)
04.2008 - 09.2009	Scholarship of the Saarbrücken Graduate School in Computer Science
Summer 2008	Tuition Fee Waiver (Grades in the top 5%, afterwards fees were abolished)

---

[1] All GPAs on a German scale: 1.0 (Best) - 6.0

Work Experience

02.2011 - 02.2017	**Research- and Teaching Assistant at EPFL**
06.2010 - 02.2011	**Research Assistant at Saarland University**
10.2006 - 10.2007	**Software Developer at SAP AG - Analytical Banking**
	SAP Bank Analyzer 6 Process Infrastructure Development in ABAP Objects on the Netweaver Plattform
10.2005 - 09.2006	**Software Developer at IBM Systems & Technology Group**
	IBM Total Storage Productivity Center Development (Storage Area Network Management based on SNIA SMI-S Interface)
01.2005 - 03.2005	**Internship at IBM ITS Consulting & Architecture**
	Development of a network connection analysis tool in J2EE
06.2004 - 09.2004	**Internship at IBM AMS Solution Implementation**
	Design and implementation of a framework for backend system integration
01.2004 - 03.2004	**Internship at IBM BCS Sector Finance**
	Design and implementation of a web framework for frontoffice work places

Professional Activities

Program Committee	Senior member/meta reviewer, ACML2017

Languages and Technologies

Languages:	English(fluent), German(fluent), Korean(intermediate), French(basic), Latinum
Programming:	Python, Julia, Matlab, C/C++, Java, SQL
Technologies:	Theano and Tensorflow, Hadoop Map/Reduce, Linux, git, web (flask, javascript)

Lausanne, March 22, 2017