Understanding Music Semantics and User Behavior with Probabilistic Latent Variable Models

Dawen Liang

Submitted in partial fulfillment of the
requirements for the degree
of Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2016

Abstract

Understanding Music Semantics and User Behavior with Probabilistic Latent Variable Models

Dawen Liang

Bayesian probabilistic modeling provides a powerful framework for building flexible models to incorporate latent structures through likelihood model and prior. When we specify a model, we make certain assumptions about the underlying data-generating process with respect to these latent structures. For example, the latent Dirichlet allocation (LDA) model assumes that when generating a document, we first select a latent *topic* and then select a word that often appears in the selected *topic*. We can uncover the latent structures conditioned on the observed data via posterior inference. In this dissertation, we apply the tools of probabilistic latent variable models and try to understand complex real-world data about music semantics and user behavior.

We first look into the problem of automatic music tagging – inferring the semantic tags (e.g., “jazz”, “piano”, “happy”, etc.) from the audio features. We treat music tagging as a matrix completion problem and apply the Poisson matrix factorization model jointly on the vector-quantized audio features and a “bag-of-tags” representation. This approach exploits the shared latent structure between semantic tags and acoustic codewords. We present experimental results on the Million Song Dataset for both annotation and retrieval tasks, illustrating the steady improvement in performance as more data is used.

We then move to the intersection between music semantics and user behavior: music recommendation. The leading performance in music recommendation is achieved by collaborative filtering methods which exploit the similarity patterns in user's listening history. We address the fundamental *cold-start* problem of collaborative filtering: it cannot recommend new songs that no one has listened to. We train a neural network on semantic tagging information as a content model and use it as a prior in a collaborative filtering model. The proposed system is evaluated on the Million Song Dataset and shows comparably better result than the collaborative filtering approaches, in addition to the favorable performance in the *cold-start* case.

Finally, we focus on general recommender systems. We examine two different types of data: implicit and explicit feedback, and introduce the notion of *user exposure* (whether or not a user is exposed to an item) as part of the data-generating process, which is latent for implicit data and observed for explicit data. For implicit data, we propose a probabilistic matrix factorization model and infer the user exposure from data. In the language of causal analysis (Imbens and Rubin, 2015), *user exposure* has close connection to the assignment mechanism. We leverage this connection more directly for explicit data and develop a causal inference approach to recommender systems. We demonstrate that causal inference for recommender systems leads to improved generalization to new data.

Exact posterior inference is generally intractable for latent variables models. Throughout this thesis, we will design specific inference procedure to tractably analyze the large-scale data encountered under each scenario.

Contents

List of Figures

List of Tables

List of Abbreviations

- AROC - Area under the Receiver-Operator Curve
- ATE - Average Treatment Effect
- CDF - Cumulative Distribution Function
- CTPF - Collaborative Topic Poisson Factorization
- CTR - Collaborative Topic Regression
- ELBO - Evidence Lower BOund
- EM - Expectation-Maximization
- LDA - Latent Dirichlet Allocation
- MAP - Mean Average Precision
- MAR - Mean Average Rank
- MCMC - Markov Chain Monte Carlo
- MIR - Music Information Retrieval

- MSD - Million Song Dataset
- NDCG - Normalized Discounted Cumulative Gain
- NMF - Nonnegative Matrix Factorization
- PLP - Predictive Log Tail Probability
- PMF - Poisson Matrix Factorization
- WMF - Weighted Matrix Factorization

Acknowledgments

It has been a wonderful journey over the last four years. I am very lucky to have an overall extremely positive PhD experience, thanks to many people who have helped me along the way.

First of all, I want to express my deepest gratitude to my advisor Dan Ellis and co-advisor David Blei. Thanks Dan for allowing me a great deal of freedom to explore anything that I found interesting. Thanks Dave for taking me into his group when Dan was happily enjoying his sabbatical at Google – I feel extremely fortunate, as I know how many people would like to have the opportunity to work with Dave. I learned a great deal from both of them about how to become a better researcher, and more importantly, a better person. I couldn't have possibly asked for better mentors along this journey.

I would also like to thank Shih-Fu Chang, John Paisley, and Matt Hoffman for serving on my thesis committee and providing valuable and insightful feedbacks. Special thanks to Matt, who also served as my mentor when I interned at Adobe Research. He is not only a good friend, but also has practically become a "co-advisor" to me during my early years in the PhD program. My research trajectory would have been completely different (probably in a bad way) if not with his guidance and encouragement.

I am glad that I had LabROSA and Blei Lab as my intellectual home at Columbia. I will remember LabROSA as this room without a window but with a lot of fun, thanks to those who I have met and worked with over the years: Zhuo Chen, Colin Raffel, Brian McFee, Thierry Bertin-Mahieux, Courtenay Cotton, Rachel Kurtz, Minshu Zhan, Matt McVicar, Jiaming Liu, Cyril Gaudefroy, Hélène Papadopoulos, Diego Furtado Silva and others that I must have missed. I also thank all the folks in Blei Lab: Laurent Charlin, Jaan Altosaar, James McInerney, Maja Rudolph, Dustin Tran, Rajesh Ranganath, Alp Kucukelbir, Allison Chaney, Adji Dieng, Kriste Krstovski, Stephan Mandt, Francisco Ruiz, Kui Tang, and Yixin Wang. It's been a pleasure. I will remember all the insightful discussions and the eventual silences in the causal inference thematic meetings, as well as the late night beer (for me it was non-alcoholic) we had together, and I look forward to more in the future in the bars of NIPS/ICML/etc. Our NYU neighbor in the Music and Audio Research Lab also gave me a good reason to visit NYC downtown: Rachel Bittner, Eric Humphrey, Oriol Nieto and Justin Salamon, it has been great to know you all.

I would like to thank Roger Dannenberg and Guangyu Xia in the Compute Music Group at Carnegie Mellon University for getting me started my research on music information retrieval which led me to apply for a PhD program in the first place. I've already thanked them once in my master's thesis, so I will keep it short here.

Every summer, I did an internship and got to work with great mentors: Gautham Mysore, Erik Schmidt, and Keki Burjorjee, as well as many smart fellow interns, from whom I learned a lot. My experience at both Adobe Research and Pandora has been extremely rewarding thanks to all of them.

Being in a PhD program for a few years would be much more difficult without friends. Lucky for me, I got to be schoolmate again with one of my best friends from high school on the other side of the globe. I would like to thank Yanan Pei for listening to my rambling (or me listening to hers) about research and life, all the great (or not so great) food we have explored over the years, and many other things that would be so boring if I were to do them alone. Sorry that you will have to find someone else to have lunch/dinner with, I am sure you cannot find anyone better. Best of luck finishing up your PhD and hope our paths will cross again in the future.

Finally, I am grateful to my parents, Jianguo Liang and Jianmin Li, for everything. Not being able to be with you more often over the last few years hasn't been easy. Thank you for the unconditional support and love whenever I need them the most. Last but not least, I want to thank my wife Qimin Xu. Thank you for taking good care of me and being patient with me when I get cranky. It would have been significantly more difficult to survive graduate school were it not for your support every step of the way – I look forward to our next adventure together. I don't say I love you very often but you know I do.

Chapter 1

Introduction

1.1 Motivation

Understanding complex real-world data is crucial as we are overwhelmed with the massive amount of information that is constantly generated. For example, there are hundreds or even thousands of new songs being released every day. Ideally we would like to get personalized spot-on recommendation without browsing through every single one of them. This requires understanding both the music semantics ("Is this song a standard upbeat pop tune or a 20-minute multi-sectional epic progressive rock masterpiece?") from acoustic waveform, as well as our music preferences, which can be so subtle that sometimes even we cannot describe precisely ourselves.

As another example, we consider arXiv.org[1] where scholars upload the latest scientist discovery everyday. For a field that is rapidly developing, like machine learning, there can be tens or even hundreds of new papers being uploaded each day. Even though the papers on

[1] `http://arxiv.org` is a pre-print repository for scientific papers.

arXiv.org are categorized by subject, it is still impractical to even skim through every new paper in the subject of our interests. On the other hand, we also do not want to miss the latest scientific progress that is happening in our field. Ideally we would like to get personalized recommendations from the massive amount of new papers. Similar to music recommendation, this requires understanding both the content of the paper ("Is this paper about unsupervised learning or reinforcement learning?") from the text, as well as our preferences/fields of interest.

In this dissertation, we aim to understand the complex real-world data (e.g., music, articles, and user feedback) by applying the tools of Bayesian probabilistic models, or more precisely, latent variable models. We give a high-level introduction to probabilistic latent variable models below, and put everything into the context of music and paper recommendation examples outlined above.

1.2 Probabilistic latent variable models

The basic idea behind probabilistic latent variable models is that we assume there exists some process that stochastically generates the data we observe. We further assume that the data-generating process is governed by some latent structures. As a concrete example, let's consider latent Dirichlet allocation model (Blei et al., 2003), a widely used probabilistic latent variable model for documents. The model assumption is that when generating a document, we first select a latent *topic* (e.g., business, sports, or politics), then select a word that often appears in the selected *topic* (e.g., the word "election" will commonly appear in a topic about politics), and repeat this process for every word. Here we do not observe the latent structures (topics). However, when we make such assumption and fit the model with text data, we are able to discover the latent topics which help us organize, browse, and retrieve large text corpus more easily and efficiently.

As mentioned above, when we design a model, we make certain assumptions about the latent structures and data-generating process. The data-generating process does not have to be absolutely correct. (In fact, they are never correct, as George E. P. Box once put it: “All models are wrong, but some are useful.”) Related to both music and paper recommendation examples, a commonly used probabilistic model for recommendation is matrix factorization (details in Section 2.2.2.1). The general model assumption of matrix factorization for recommendation is that a user’s feedback towards an item is generated by the combination of two latent variables: user preference and item attribute. The item attribute can itself be generated from a prior, or generated from a probabilistic model of the actual item content (acoustic waveform or document text). This seems to be an overly-simplified data-generating process. However, it proves effective in many scenarios and is widely used in commercial recommender systems.

When we fit the model via posterior inference (we give details about general inference procedure in Section 2.1), we uncover these latent structures. These latent structures reveal interesting aspects of the data, e.g., we know for some users, they enjoy classical music and papers about Bayesian statistics. Concretely, in Chapter 3, we fit a latent variable model to a music collection of 370k tracks to predict music semantic tags (e.g., genre, instrumentation, mood, etc.) from acoustic features. By exploring the model in Table 3.3, we can get an idea of what portion of the acoustic space is being captured by the latent variables, and whether it is musically coherent. In Chapter 5, we introduce the notation of *user exposure* (whether a user is exposed to an item or not) in recommender systems as a latent variable. After fitting the model, we are able to reason about if the model believes a user has been exposed to certain items in Figure 5.2 and Figure 5.3. These exploratory studies can help us better understand the complex data at hand and provide insight into what the model is capturing.

1.3 Contributions

Below we outline the contributions of this dissertation. Since we address a number of different problems in subsequent chapters, we provide more thorough literature reviews of specific prior work in each chapter.

1.3.1 Music understanding and recommendation

Scalable music tagging with Poisson factorization. We develop scalable solution to automatic music tagging – inferring the semantic tags from the audio features. We treat music tagging as a matrix completion problem and apply the Poisson factorization model to a large collection of music data. We explore the fitted model and identify what portion of the acoustic codeword space is being captured by the latent variables.

Content-aware collaborative music recommendation. We address the fundamental cold-start problem of collaborative filtering (it cannot recommend new songs that no one has listened to) by pre-training a multi-layered neural network on semantic tagging information as a content model and using it as a prior in a collaborative filtering model. The proposed system shows comparably better result than the state-of-the-art collaborative filtering approaches, in addition to the favorable performance in the cold-start case.

1.3.2 Probabilistic models for recommender systems

Modeling user exposure in recommendation. We develop a probabilistic matrix factorization model to capture the latent user exposure (whether or not a user is exposed to an item). In doing so, we recover one of the most successful state-of-the-art approaches as a special case of our model (Hu et al., 2008), and provide a plug-in method for conditioning exposure on various forms of exposure covariates (e.g., topics in text, venue locations). In four datasets from various domains, we show that our model outperforms existing benchmarks both with and without exposure covariates.

Causal inference for recommendation. We develop a causal inference approach to recommender systems. We use inverse propensity weighting to correct for the bias which exists in observational recommendation data. Through extensive empirical study, we demonstrate that this causal approach to recommender systems leads to improved generalization to new data.

1.4 Related publications

The work presented in this dissertation is largely based on published articles in various conference proceedings: Chapter 3 and Chapter 4 are based on papers presented in ISMIR 2014 (Liang et al., 2014) and ISMIR 2015 (Liang et al., 2015), respectively. Chapter 5 is based on our paper presented in WWW 2016 (Liang et al., 2016b). Chapter 6 is based on our paper which is currently in submission (Liang et al., 2016a).

Chapter 2

Background

In this chapter, we discuss some background knowledge and previous work helpful to understanding the rest of the dissertation. Broadly speaking, we will make use of inference techniques for probabilistic modeling, collaborative filtering method for recommender systems, and causal inference. We give a high-level overview of the three fields and introduce some necessary definitions that will be used in the subsequent chapters.

2.1 Probabilistic modeling and inference techniques

We begin by defining the general problem setup for probabilistic latent variable models. We observe data $\boldsymbol{x} = \{x_1, \dots, x_N\}$. We assume the data is generated stochastically by a model $p(\boldsymbol{x} \mid \boldsymbol{z}, \theta)$ that is governed by some latent variables $\boldsymbol{z} = \{z_1, \dots, z_N\}$ as well as model parameters $\boldsymbol{\theta}$[1]. We can also incorporate priors $p(\boldsymbol{z}, \boldsymbol{\theta})$. We leave the dependency structure of prior generic. This class of models covers a wide range of commonly used models, to

[1]The distinction between latent variables $\boldsymbol{z}$ and model parameters $\boldsymbol{\theta}$ can be somewhat arbitrary. Here we follow the convention that the dimensionality of the latent variables grows with the number of observations

name a few: mixture models, hidden Markov models, probabilistic matrix factorization (Salakhutdinov and Mnih, 2008), and mixed-membership models (e.g., latent Dirichlet allocation (Blei et al., 2003), and stochastic blockmodels (Airoldi et al., 2008)).

Figure 2.1 demonstrates the graphical model representation for the general latent variable models described above. Shaded nodes represent observed variables. Unshaded nodes represent hidden (unobserved) variables. A directed edge from node a to node b denotes that the variable b depends on the value of variable a. Plates denote replication by the value in the lower corner of the plate. We use doted line to indicate that the dependency between model parameters $\boldsymbol{\theta}$ and latent variables $\boldsymbol{z}$ are optional – it depends on how the prior $p(\boldsymbol{z}, \boldsymbol{\theta})$ is specified.

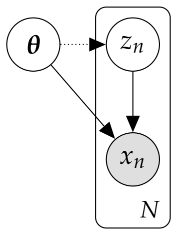


Figure 2.1: Graphical model representation for the general latent variable models. Doted line is used to indicate that the dependency between model parameters $\boldsymbol{\theta}$ and latent variables $\boldsymbol{z}$ are optional – it depends on how the prior $p(\boldsymbol{z}, \boldsymbol{\theta})$ is specified.

In Bayesian inference, the goal is to reason about the posterior distribution over the model parameters and latent variables conditioned on the data, which is given by Bayes' rule:

$$p(\boldsymbol{\theta}, \boldsymbol{z} \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid \boldsymbol{z}, \boldsymbol{\theta}) p(\boldsymbol{z}, \boldsymbol{\theta})}{p(\boldsymbol{x})} = \frac{p(\boldsymbol{x} \mid \boldsymbol{z}, \boldsymbol{\theta}) p(\boldsymbol{z}, \boldsymbol{\theta})}{\int_{\boldsymbol{\theta}} \int_{\boldsymbol{z}} p(\boldsymbol{x} \mid \boldsymbol{z}, \boldsymbol{\theta}) p(\boldsymbol{z}, \boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{z} \, \mathrm{d}\boldsymbol{\theta}} \tag{2.1}$$

Through posterior inference, we are able to uncover the latent structure induced by the model. Except in very simple models, posterior $p(\boldsymbol{\theta}, \boldsymbol{z} \mid \boldsymbol{x})$ is generally intractable to compute due to

(hence both $\boldsymbol{x}$ and $\boldsymbol{z}$ are indexed by n), while that of model parameters does not. Latent variables and model parameters loosely correspond to *local variables* and *global variables* in Hoffman et al. (2013)

the normalizing constant $p(\boldsymbol{x})$ which requires computing the integral in the denominator of Eq. 2.1. In practice, people normally resort to Markov chain Monte Carlo (MCMC) methods (Neal, 1993; Robert and Casella, 2013) to obtain samples from the posterior distribution to form a Monte Carlo estimator about the predictive quantities. Despite the asymptotic guarantees, MCMC methods are generally unable to analyze large-scale data. Scaling Bayesian inference to large-scale data is an active research area (see Angelino et al. (2016) for an extensive survey). In Section 2.1.2, we will introduce variational inference, a scalable deterministic alternative to MCMC.

Alternatively, it is also possible (and computationally simpler) to only obtain a point estimate of the parameters of interest instead of reasoning about the entire posterior via maximum likelihood estimation (MLE) or maximum *a posteriori* (MAP):

$$\boldsymbol{\theta}^{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{x} \mid \boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \log \int_{\boldsymbol{z}} p(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta})\, \mathrm{d}\boldsymbol{z} \tag{2.2}$$

$$\boldsymbol{\theta}^{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}, \boldsymbol{z} \mid \boldsymbol{x}) = \arg\max_{\boldsymbol{\theta}} \log \int_{\boldsymbol{z}} p(\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{x})\, \mathrm{d}\boldsymbol{z} \tag{2.3}$$

For models with latent variables, expectation-maximization (EM) (Dempster et al., 1977) algorithm is usually required to obtain these point estimates, which we will turn to next.

2.1.1 Parameter estimation via expectation-maximization

In this section, we introduce the EM algorithm for parameter estimation. There exist different derivations of the algorithm in the literature. Here we choose to introduce it in the variational framework to highlight its close connection to variational inference, which we will introduce in the following section. We derive the EM algorithm for maximum likelihood estimation (Eq. 2.2) – it only requires minor modification for maximum *a posteriori* (Eq. 2.3).

To obtain the maximum likelihood estimates, typically we write down the so-called *observable data log-likelihood* $\log p(\boldsymbol{x} \mid \boldsymbol{\theta})$ and optimize it with respect to the model parameters $\boldsymbol{\theta}$. One of the problems with directly optimizing the observable data log-likelihood for models with latent variables $\boldsymbol{z}$ is that it requires to integrate over all the latent variables $\log \int_{\boldsymbol{z}} p(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta})\, \mathrm{d}\boldsymbol{z}$, which is generally intractable. As a workaround, we instead optimize the *complete data log-likelihood* $\log p(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta})$ by introducing a *variational distribution* $q(\boldsymbol{z})$ and applying Jensen's inequality:

$$\begin{aligned}
\log p(\boldsymbol{x} \mid \boldsymbol{\theta}) &= \log \int_{\boldsymbol{z}} p(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta})\, \mathrm{d}\boldsymbol{z} \\
&= \log \int_{\boldsymbol{z}} q(\boldsymbol{z}) \frac{p(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta})}{q(\boldsymbol{z})}\, \mathrm{d}\boldsymbol{z} \\
&\geq \int_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{p(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta})}{q(\boldsymbol{z})}\, \mathrm{d}\boldsymbol{z} \\
&= \mathbb{E}_q \left[\log p(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta})\right] - \mathbb{E}_q \left[\log q(\boldsymbol{z})\right].
\end{aligned} \tag{2.4}$$

We obtain a lower bound of the log-likelihood $\log p(\boldsymbol{x} \mid \boldsymbol{\theta})$ that we are interested in optimizing. The tightness of this bound depends on the variational distribution $q(\boldsymbol{z})$. We could of course find out the optimal $q(\boldsymbol{z})$ by exploring when the equality holds for Jensen's inequality. However, we will solve it from a different angle here. We first explore the slack by applying

Jensen's inequality:

$$\begin{aligned}\Delta &= \log p(\boldsymbol{x} \mid \boldsymbol{\theta}) - \int_z q(z) \log \frac{p(\boldsymbol{x}, z \mid \boldsymbol{\theta})}{q(z)} \, \mathrm{d}z \\ &= \int_z q(z) \log p(\boldsymbol{x} \mid \boldsymbol{\theta}) - \int_z q(z) \log \frac{p(\boldsymbol{x}, z \mid \boldsymbol{\theta})}{q(z)} \, \mathrm{d}z \\ &= \int_z q(z) \log \frac{p(\boldsymbol{x} \mid \boldsymbol{\theta}) q(z)}{p(\boldsymbol{x}, z \mid \boldsymbol{\theta})} \, \mathrm{d}z \\ &= \int_z q(z) \log \frac{q(z)}{p(z \mid \boldsymbol{x}, \boldsymbol{\theta})} \, \mathrm{d}z \equiv \mathrm{KL}(q_z \| p_{z \mid \boldsymbol{x}, \boldsymbol{\theta}})\end{aligned} \tag{2.5}$$

Thus, the difference is the Kullback-Leibler (KL) divergence between the variational distribution $q(z)$ and the posterior distribution $p(z \mid \boldsymbol{x}, \boldsymbol{\theta})$. We can re-write the log-likelihood as:

$$\log p(\boldsymbol{x} \mid \boldsymbol{\theta}) = \underbrace{\mathbb{E}_q \left[\log p(\boldsymbol{x}, z \mid \boldsymbol{\theta})\right] - \mathbb{E}_q \left[\log q(z)\right]}_{\mathcal{L}(q, \boldsymbol{\theta})} + \mathrm{KL}(q_z \| p_{z \mid \boldsymbol{x}, \boldsymbol{\theta}}).$$

Since the KL-divergence $\mathrm{KL}(q_z \| p_{z \mid \boldsymbol{x}, \boldsymbol{\theta}})$ is non-negative and equals 0 only if $q(z) = p(z \mid \boldsymbol{x}, \boldsymbol{\theta})$, *a.s.*, $\mathcal{L}(q, \boldsymbol{\theta})$ acts as a tight lower-bound that equals $\log p(\boldsymbol{x} \mid \boldsymbol{\theta})$ when we set $q(z)$ to $p(z \mid \boldsymbol{x}, \boldsymbol{\theta})$. EM algorithm optimizes $\mathcal{L}(q, \boldsymbol{\theta})$ by iteratively applying the following E(xpectation) and M(aximization) steps:

E-step: By setting $q(z) = p(z \mid \boldsymbol{x}, \boldsymbol{\theta})$, we obtain the optimal variational distribution, closing the gap between log-likelihood $\log p(\boldsymbol{x} \mid \boldsymbol{\theta})$ and the lower bound $\mathcal{L}(q, \boldsymbol{\theta})$.

M-step: We have $q(z)$ fixed from E-step and optimize $\mathcal{L}(q, \boldsymbol{\theta})$ with respect to the model parameters $\boldsymbol{\theta}$, which is equivalent to the following:

$$\boldsymbol{\theta}^{\text{new}} = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_q \left[\log p(\boldsymbol{x}, z \mid \boldsymbol{\theta})\right],$$

since $\mathbb{E}_q\left[\log q(\boldsymbol{z})\right]$ does not depend on $\boldsymbol{\theta}$.[2] Unless we have reached a stationary point, the lower bound $\mathcal{L}(q, \boldsymbol{\theta})$ will increase with the new parameters $\boldsymbol{\theta}^{\text{new}}$. The new parameters $\boldsymbol{\theta}^{\text{new}}$ will also make the KL-divergence $\text{KL}(q_{\boldsymbol{z}} \| p_{\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{\theta}})$ greater than 0, which creates gap between the log-likelihood and $\mathcal{L}(q, \boldsymbol{\theta})$. This gap will be closed in the next E-step. Chapter 9 of Bishop (2006) provides a clear illustrative demonstration of the EM algorithm.

EM algorithm is a typical example of *coordinate ascent* (Bertsekas, 1999), where in each E- and M-step, we fix one of the variables of interest—$\boldsymbol{\theta}$ in E-step and $q(\boldsymbol{z})$ in M-step—and optimize with respect to the other one.

2.1.2 Variational inference

Variational inference is a deterministic alternative to MCMC methods (Jordan et al., 1999; Wainwright and Jordan, 2008; Blei et al., 2016). In Bayesian inference, we aim to reason about the posterior $p(\boldsymbol{z}, \boldsymbol{\theta} \mid \boldsymbol{x})$ which is almost always intractable to compute. The basic idea behind variational inference is to choose a tractable family of variational distributions $q(\boldsymbol{z}, \boldsymbol{\theta})$ to approximate the intractable posterior $p(\boldsymbol{z}, \theta | \boldsymbol{x})$, so that the KL-divergence between the variational distribution and the true posterior $\text{KL}(q_{\boldsymbol{z}, \boldsymbol{\theta}} \| p_{\boldsymbol{z}, \boldsymbol{\theta} \mid \boldsymbol{x}})$ is minimized.

Variational inference turns the problem of Bayesian inference into a one of optimization, which enables us to leverage the advances from optimization community, e.g., by making use of stochastic optimization, we can scale variational inference to massive dataset (Hoffman et al., 2013).

[2] In principle, $\boldsymbol{\theta}^{\text{new}}$ does not have to fully optimize the objective $\mathbb{E}_q\left[\log p(\boldsymbol{x}, \boldsymbol{z}, \mid \boldsymbol{\theta})\right]$, as long as the new values increase it (e.g., by taking a few gradient steps). This is referred as generalized EM (Neal and Hinton, 1998).

To utilize variational inference for approximate Bayesian inference, we introduce the variational distribution $q(\boldsymbol{z}, \boldsymbol{\theta})$ and lower bound the marginal likelihood, similar to Eq. 2.4:

$$\begin{aligned}\log p(\boldsymbol{x}) &= \log \int_{\boldsymbol{z},\boldsymbol{\theta}} p(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\theta})\, \mathrm{d}\boldsymbol{z}\, \mathrm{d}\boldsymbol{\theta} \\ &\geq \mathbb{E}_q \left[\log p(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\theta})\right] - \mathbb{E}_q \left[\log q(\boldsymbol{z}, \boldsymbol{\theta})\right] \triangleq \mathcal{L}.\end{aligned} \tag{2.6}$$

Here the model parameters $\boldsymbol{\theta}$ are treated the same as latent variables $\boldsymbol{z}$ with some pre-specified prior $p(\boldsymbol{z}, \boldsymbol{\theta})$.[3] $\mathcal{L}$ is usually referred as evidence lower bound (ELBO), since it is a lower bound of the model evidence $\log p(\boldsymbol{x})$. Based on derivation similar to Eq. 2.5 (omitted for brevity), we can show that optimizing ELBO is equivalent to minimizing the KL divergence between the variational distribution $q(\boldsymbol{z}, \boldsymbol{\theta})$ and the posterior of interest $p(\boldsymbol{z}, \boldsymbol{\theta} \mid \boldsymbol{x})$. Once we obtain the approximating posterior that minimizes the KL-divergence, we can use it as a proxy of the true posterior to form prediction.

So far we haven't specified how to select the variational distribution $q(\boldsymbol{z}, \boldsymbol{\theta})$. One popular choice is the mean-field family which is completely factorized: $q(\boldsymbol{z}, \boldsymbol{\theta}) = \left(\prod_d q_d(z_d)\right)\left(\prod_i q_i(\theta_i)\right)$. With mean-field family, we can obtain the general form for the optimal variational distributions:

$$\begin{aligned} q_d^*(z_d) &\propto \exp\{\mathbb{E}_{q_{-d}} \left[\log p(z_d, \boldsymbol{x}, z_{-d}, \boldsymbol{\theta})\right]\} \\ q_i^*(\theta_i) &\propto \exp\{\mathbb{E}_{q_{-i}} \left[\log p(\theta_i, \boldsymbol{x}, \boldsymbol{z}, \theta_{-i})\right]\},\end{aligned} \tag{2.7}$$

where z_{-d} is used to index all of $\boldsymbol{z}$ except the dth dimension, and $\mathbb{E}_{q_{-d}}[\cdot]$ denotes taking expectation with respect to everything except $q_d(z_d)$. (θ_{-i} and $\mathbb{E}_{q_{-i}}[\cdot]$ are similarly defined.)

[3]Variational inference can also be applied to the MLE/MAP case in Section 2.1.1 where we only marginalize out latent variables $\boldsymbol{z}$ to obtain point estimates of model parameters $\boldsymbol{\theta}$. This happens in the E-step, when the posterior $p(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{\theta})$ is intractable to compute exactly, which leads to the variational EM algorithm (Beal, 2003).

For conditional conjugate model where the complete conditionals $p(z_d \mid \boldsymbol{x}, z_{-d}, \boldsymbol{\theta})$ and $p(\theta_i \mid \boldsymbol{x}, \boldsymbol{z}, \theta_{-i})$ are in exponential family, the distributional form of Eq. 2.7 can be computed exactly. This leads to the standard coordinate ascent variational inference algorithm: we iteratively set $q_d(z_d)$ and $q_i(\theta_i)$ to its optimal form while keeping everything else fixed across the dimensions and repeat this procedure until convergence.

2.2 Recommender systems

Making good recommendations is an important problem on the web. In the recommendation problem, we observe how a set of users interacts with a set of items, and our goal is to show each user a set of previously unseen items that she will like. Broadly speaking, recommender systems use historical data to infer users' preferences, and then use the inferred preferences to suggest items. Good recommender systems are essential as the web grows; users are overwhelmed with choice.

2.2.1 Explicit and implicit feedback

Traditionally there are two modes of the recommendation problem: recommendation from explicit data and recommendation from implicit data. With explicit data, users rate some items (positively, negatively, or along a spectrum) and we can predict their missing ratings (the task of *rating prediction*, popularized by the Netflix Prize[4]). This is called explicit data because users' preferences are expressed in an explicit fashion: positively rated items indicate types of items that they like; negatively rated items indicate items that they do not like. For explicit data, it is enough to only use the rated items to infer a user's preferences as we have

[4]`http://www.netflixprize.com/`

both positive and negative examples. Explicit data is of great value, but it is often difficult to obtain.

In implicit data, each user expresses a binary decision about items[5]—for example this can be clicking, purchasing, viewing—and we aim to predict unclicked items that she would want to click on. (We use the verb “click” throughout this dissertation for concreteness; this can be any type of interaction, including “download,” “purchase,” “listen,” or “watch.”) Unlike ratings data, implicit data is easily accessible. While ratings data requires action on the part of the users, implicit data is often a natural byproduct of their behavior, e.g., browsing histories, click logs, and past purchases. Despite the ease of access, implicit data is inherently noisy, as users’ preferences are expressed through implicit actions and we only observe positive signals: we know users click on items they like, but we do not know why an item is unclicked. We will explore recommender systems for both implicit and explicit data in this dissertation in Chapter 5 and Chapter 6, respectively.

2.2.2 Collaborative filtering for recommender systems

Collaborative filtering is the workhorse of recommender systems. It is widely used in many commercial websites, e.g., Amazon uses various collaborative filtering algorithms to suggest products (“Customers Who Bought This Item Also Bought”), and Netflix uses collaborative filtering algorithms extensively in their homepage to suggest new movies and TV shows to watch.

Collaborative filtering analyzes user preferences for items by exploiting the similarity patterns across users. There are two major classes of collaborative filtering algorithms: neighborhood-

[5]In principle, implicit data can go beyond binary: For example, the number of times a user listened to certain songs can also be considered as implicit feedback. However, in practice we find that the binary indicator of *interaction* tends to carry the most signal.

based model (Sarwar et al., 2001) and the matrix factorization model (Koren et al., 2009). In this dissertation, we will mainly focus on the matrix factorization for collaborative filtering.

2.2.2.1 Matrix factorization for collaborative filtering

User-item preference data, whether explicit or implicit, can be encoded in a user by item matrix. Throughout this dissertation, a user is indexed by $u \in \{1, \ldots, U\}$, an item is indexed by $i \in \{1, \ldots, I\}$, and we will refer to this user by item matrix as the *click matrix* or the *interaction matrix*. Given the observed entries in this matrix $\{y_{ui} : (u, i) \in \mathcal{O}\}$, the recommendation task is often framed as filling in the unobserved entries. Matrix factorization models, which infer (latent) user preferences and item attributes by factorizing the click matrix, are standard in recommender systems (Koren et al., 2009).

Figure 2.2 demonstrates the basic idea behind matrix factorization for collaborative filtering.[6] In this illustrative example, we have three users and three items, where each user consumes only one item. Matrix factorization aims to find a latent space to embed all of the users and items. If a user consumes an item, this user and item pair will be embedded closer in this latent space. The locations (coordinates) in this latent space correspond to the user and item latent factors obtained by factorizing the click matrix. To make recommendations for each user, we select the unconsumed items which have high dot products with the user's latent factor.

How would this work? Consider a metalhead who has listened to a lot of *Metallica* but not *Iron Maiden*. It is reasonable to assume that there are many other users with similar tastes listened to songs from both bands, which makes the latent factors for songs by both *Metallica*

[6]This figure is only an illustrative example: typical matrix factorization models use inner products, not Euclidean distance, to measure similarity.

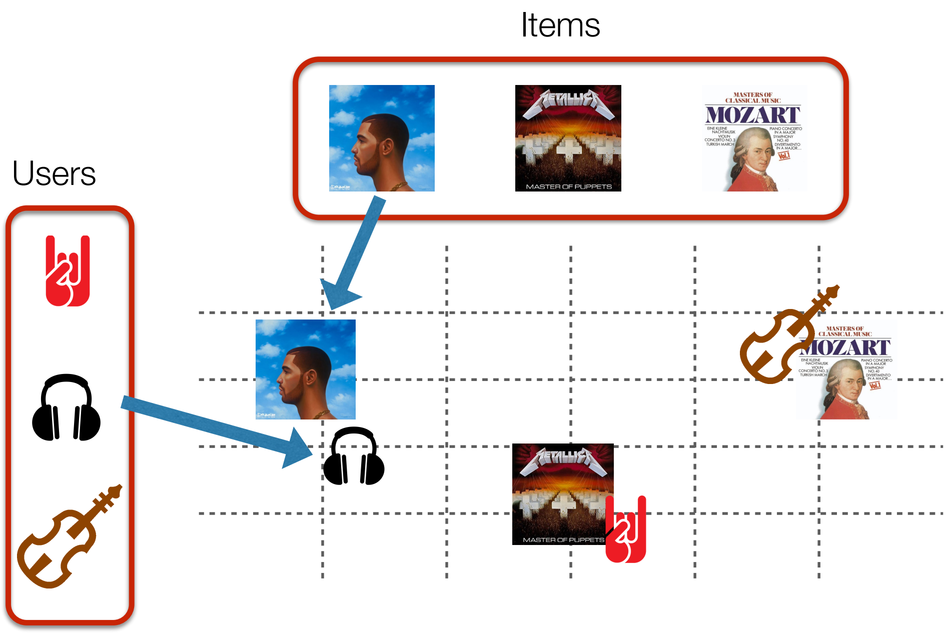


Figure 2.2: An illustrative example of matrix factorization for collaborative filtering. Matrix factorization aims to find a latent preference space to embed both users and items such that if a user consumes an item, they will be embedded closer in this latent space.

and *Iron Maiden* very close in the latent space. Therefore, when making recommendations for this user, the songs from *Iron Maiden* will likely have higher dot products, which will be recommended by the learned matrix factorization model.

From a generative modeling perspective, the model can be understood as first drawing user and item latent factors corresponding, respectively, to user preferences and item attributes. Then drawing observations from a specific distribution (e.g., a Poisson or a Gaussian) with its mean parametrized by the dot product between the user and the item factors. Formally, Gaussian matrix factorization is (Salakhutdinov and Mnih, 2008):

$$\begin{aligned}\boldsymbol{\theta}_u &\sim \mathcal{N}(\mathbf{0}, \lambda_\theta^{-1}\mathrm{I}_K) \quad \text{for } u = 1, \ldots, U,\\ \boldsymbol{\beta}_i &\sim \mathcal{N}(\mathbf{0}, \lambda_\beta^{-1}\mathrm{I}_K) \quad \text{for } i = 1, \ldots, I, \\ y_{ui} &\sim \mathcal{N}(\boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i, \lambda_y^{-1}) \quad \text{for } (u,i) \in \mathcal{O},\end{aligned} \tag{2.8}$$

where $\boldsymbol{\theta}_u$ and $\boldsymbol{\beta}_i$ represent user u's latent preferences and item i's attributes respectively. We use the mean and (co)variance to parametrize the Gaussian distribution. λ_θ, λ_β, and λ_y can be treated as hyperparameters, or be given priors for a full Bayesian treatment. I_K stands for the identity matrix of dimension K. A graphical model representation of the Gaussian matrix factorization model is shown in Figure 2.3.

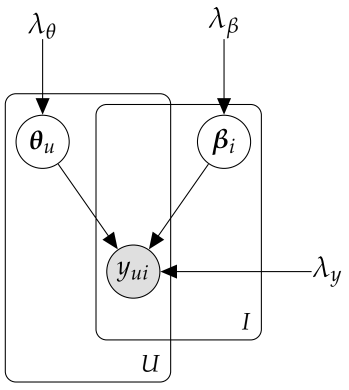


Figure 2.3: Graphical model representation for the Gaussian matrix factorization.

We derive coordinate updates to obtain the maximum *a posteriori* estimates of the Gaussian matrix factorization model, as they are closely related to the model inference we develop in the later chapters. Since we are only obtaining point estimates of the model parameters, we can always scale λ_θ and λ_β by λ_y to obtain the same solution. Without loss of generality, we set $\lambda_y = 1$. The complete log-likelihood of the model is:

$$\mathcal{L} = -\sum_{(u,i)\in\mathcal{O}} \frac{1}{2}(y_{ui} - \boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i)^2 - \frac{\lambda_\theta}{2}\sum_u \|\boldsymbol{\theta}_u\|_2^2 - \frac{\lambda_\beta}{2}\sum_i \|\boldsymbol{\beta}_i\|_2^2. \tag{2.9}$$

As we can see, the maximum *a posteriori* estimates of the Gaussian matrix factorization model is equivalent to the solution of minimizing the squared loss between the estimated and actual preferences $\sum_{(u,i)\in\mathcal{O}}(y_{ui} - \boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i)^2$ with ℓ_2 regularization on the latent factors.

The basic idea of the coordinate updates for the Gaussian matrix factorization model is to only update one of the user or item factor ($\boldsymbol{\theta}_u$ or $\boldsymbol{\beta}_i$) at a time when keeping everything else fixed. Taking the gradient of the complete log-likelihood (Eq. 2.9) with respect to one of the latent factors and setting it to 0, we obtain the following updates:

$$\boldsymbol{\theta}_u^{\text{new}} \leftarrow \Big(\sum_{i:(u,i)\in\mathcal{O}} \boldsymbol{\beta}_i\boldsymbol{\beta}_i^\top + \lambda_\theta \mathrm{I}_K\Big)^{-1}\Big(\sum_{i:(u,i)\in\mathcal{O}} y_{ui}\boldsymbol{\beta}_i\Big) \tag{2.10}$$

$$\boldsymbol{\beta}_i^{\text{new}} \leftarrow \Big(\sum_{u:(u,i)\in\mathcal{O}} \boldsymbol{\theta}_u\boldsymbol{\theta}_u^\top + \lambda_\beta \mathrm{I}_K\Big)^{-1}\Big(\sum_{u:(u,i)\in\mathcal{O}} y_{ui}\boldsymbol{\theta}_u\Big) \tag{2.11}$$

Every single update resembles that of ridge regression (Hastie et al., 2009) where the responses are y_{ui} and the covariates are the latent factors. Therefore, the coordinate updates for the Gaussian matrix factorization is often called *alternating least squares* (ALS). The full algorithm is summarized in Algorithm 1. Note that the updates are embarrassingly parallelizable across users and items.

2.2.2.2 Collaborative filtering for implicit data

The model described in Eq. 2.8 can be equally applied to both explicit and implicit data. The real difference is how to define the observed set $\mathcal{O}$: For explicit data, it can simply be the

Algorithm 1: ALS Alternating least squares for the Gaussian matrix factorization

```
Input: A set of observed entires in the click matrix $\{y_{ui} : (u, i) \in \mathcal{O}\}$, regularization
       parameters $\lambda_\theta$ and $\lambda_\beta$
Output: A set of user latent factors $\boldsymbol{\theta}_{1:U}$ and item latent factors $\boldsymbol{\beta}_{1:I}$
Randomly initialize $\boldsymbol{\theta}_{1:U}$, $\boldsymbol{\beta}_{1:I}$
while not converged do
    for $u \leftarrow 1$ to $U$ do
        Update user factor $\boldsymbol{\theta}_u$ (Eq. 2.10)
    end
    for $i \leftarrow 1$ to $I$ do
        Update item factor $\boldsymbol{\beta}_i$ (Eq. 2.11)
    end
end
return $\boldsymbol{\theta}_{1:U}$, $\boldsymbol{\beta}_{1:I}$
```

user-item pairs where user u has clicked on (rated) item i. However, we can not copy the same definition for implicit data. The reason is that the data is binary and thus, when inferring a user's preferences, we must use unclicked items (otherwise, it would be like training a classifier with only positive labels[7]), i.e., $\mathcal{O}$ contains all the entires in the click matrix.

Mirroring methods for explicit data, many methods treat unclicked items as those a user does not like. But this assumption is mistaken, and overestimates the effect of the unclicked items. Some of these items—many of them, in large-scale settings—are unclicked because the user didn't see them, rather than because she chose not to click them. This is the crux of the problem of analyzing implicit data: we know users click on items they like, but we do not know why an item is unclicked.

Weighted matrix factorization (WMF), the standard factorization model for implicit data, selectively downweights evidence from the click matrix (Hu et al., 2008). WMF uses a simple heuristic where all unobserved user-item interactions are equally downweighted vis-a-vis the

[7] Recommendation from implicit data is also known as one-class collaborative filtering (Pan et al., 2008).

observed interactions. Under WMF an observation is generated from:

$$y_{ui} \sim \mathcal{N}(\boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i, c_{y_{ui}}^{-1}),$$

where the "confidence" c is set such that $c_1 > c_0$. This dependency between a click and itself is unorthodox; because of it WMF is not a generative model. As we will describe in Chapter 5 we obtain a proper generative model by adding an exposure latent variable.

The maximum *a posteriori* estimates of WMF can also be obtained via ALS with minor modification. For notational convenience, we define $c_{ui} \triangleq c_{y_{ui}}$. The complete log-likelihood for WMF is (recall that observed set $\mathcal{O}$ contains all the entries in the click matrix):

$$\mathcal{L} = -\sum_{u,i} \frac{c_{ui}}{2}(y_{ui} - \boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i)^2 - \frac{\lambda_\theta}{2}\sum_u \|\boldsymbol{\theta}_u\|_2^2 - \frac{\lambda_\beta}{2}\sum_i \|\boldsymbol{\beta}_i\|_2^2$$

Again, we take the gradient with respect to one of the factors and set it to 0, which leads to the following ALS updates:

$$\boldsymbol{\theta}_u^{\text{new}} \leftarrow (\sum_i c_{ui}\boldsymbol{\beta}_i\boldsymbol{\beta}_i^\top + \lambda_\theta \mathrm{I}_K)^{-1}(\sum_i c_{ui} y_{ui} \boldsymbol{\beta}_i) \tag{2.12}$$

$$\boldsymbol{\beta}_i^{\text{new}} \leftarrow (\sum_u c_{ui}\boldsymbol{\theta}_u\boldsymbol{\theta}_u^\top + \lambda_\beta \mathrm{I}_K)^{-1}(\sum_u c_{ui} y_{ui} \boldsymbol{\theta}_u) \tag{2.13}$$

However, unlike Eq. 2.10 and Eq. 2.11, the summation inside of the matrix inversion is over all the users or items, which can be computationally challenging, especially considering that we will have to do this computation for every single factor update (there are in total $U + I$ factors to be updated in one iteration). Hu et al. (2008) propose a clever trick to speed up the computation substantially by breaking up the summation into two parts as follows (here we only demonstrate the case for updating the user factor $\boldsymbol{\theta}_u$, the same can be applied to item factor $\boldsymbol{\beta}_i$):

$$
\begin{aligned}
&(\sum_i c_{ui}\boldsymbol{\beta}_i\boldsymbol{\beta}_i^\top + \lambda_\theta \mathrm{I}_K)^{-1}(\sum_i c_{ui}y_{ui}\boldsymbol{\beta}_i) \\
=\ &(\sum_i (c_{ui} - c_0)\boldsymbol{\beta}_i\boldsymbol{\beta}_i^\top + \underbrace{\sum_i c_0\boldsymbol{\beta}_i\boldsymbol{\beta}_i^\top + \lambda_\theta \mathrm{I}_K}_{\text{precompute once per iteration}})^{-1}(\sum_i c_{ui}y_{ui}\boldsymbol{\beta}_i).
\end{aligned}
$$

The second part $\sum_i c_0\boldsymbol{\beta}_i\boldsymbol{\beta}_i^\top + \lambda_\theta \mathrm{I}_K$ is shared across all the user factor updates, thus can be precomputed once per iteration. The first part $\sum_i (c_{ui} - c_0)\boldsymbol{\beta}_i\boldsymbol{\beta}_i^\top$ can be efficiently computed because $c_{ui} - c_0$ is non-zero only when $y_{ui} = 1$ and normally the click matrix is highly sparse. Furthermore, just like ALS for the Gaussian matrix factorization, all the updates are embarrassingly parallelizable across users and items. The full algorithm of ALS for WMF is summarized in Algorithm 2. With the speed-up trick and embarrassing parallelization, ALS for WMF can be easily applied to datasets with millions of users and items.

Algorithm 2: W-ALS Alternating least squares for WMF

```
Input: Click matrix $y_{ui}$, the confidence for clicked $c_1$ and unclicked $c_0$, regularization
       parameters $\lambda_\theta$ and $\lambda_\beta$
Output: A set of user latent factors $\boldsymbol{\theta}_{1:U}$ and item latent factors $\boldsymbol{\beta}_{1:I}$
Randomly initialize $\boldsymbol{\theta}_{1:U}$, $\boldsymbol{\beta}_{1:I}$
while not converged do
    Precompute $\sum_i c_0\boldsymbol{\beta}_i\boldsymbol{\beta}_i^\top + \lambda_\theta \mathrm{I}_K$
    for $u \leftarrow 1$ to $U$ do
        Update user factor $\boldsymbol{\theta}_u$ (Eq. 2.12)
    end
    Precompute $\sum_u c_0\boldsymbol{\theta}_u\boldsymbol{\theta}_u^\top + \lambda_\beta \mathrm{I}_K$
    for $i \leftarrow 1$ to $I$ do
        Update item factor $\boldsymbol{\beta}_i$ (Eq. 2.13)
    end
end
return $\boldsymbol{\theta}_{1:U}$, $\boldsymbol{\beta}_{1:I}$
```

WMF treats the collaborative filtering problem with implicit data as a regression problem.

Concretely, consumed user-item pairs are assigned a value of one and unobserved user-item pairs are assigned a value of zero. Bayesian personalized ranking (BPR) (Rendle et al., 2009; Rendle and Freudenthaler, 2014) instead treats the problem as a one of ranking consumed user-item pairs above unobserved pairs. In a similar vein, the weighted approximate-ranking pairwise (WARP) loss proposed in Weston et al. (2011) approximately optimizes Precision@k. To deal with the non-differentiable nature of the ranking loss, these methods typically design specific (stochastic optimization) methods for parameter estimation.

2.3 Causal inference

Causal inference is aiming to answer the cause-and-effect question: does X cause Y? If so, how much is the effect of X on Y? Causal inference helps us learn about how things work and predict what happens when certain things change (Morgan and Winship, 2014; Imbens and Rubin, 2015). In this section, we review some basic concepts of causal inference that will be used in the later chapters of this dissertation when we make a connection between causal inference and recommendation.

2.3.1 Potential outcome framework

The potential outcome framework of causal inference (Rubin, 1974) is the most widely used causality formulation. We use random variable $A = a$ as an indicator of treatment assignment and assume the treatment is binary, i.e., a is either 1 (assigned the treatment) or 0 (not assigned the treatment). In this framework, each individual has two *potential outcomes* $Y(a)$ depending on the value of a. For example, in a medical trial, for each patient, we assume there is a potential outcome $Y(1)$ if she receives the treatment and $Y(0)$ if she receives the placebo.

One measurement of the causal effect is the average difference (over individuals) between those potential outcomes. It is formally formulated as the average treatment effect (ATE): $\mathbb{E}[\delta] = \mathbb{E}[Y(1)] - \mathbb{E}[Y(0)]$, where the expectation is taken over the whole population of interest. In the language of graphical models (Pearl, 2009), this is framed as evaluating the impact of an intervention on random variables in a probabilistic graph. The difficulty of causal inference is that we can only observe one realization of all the potential outcomes $Y(a)$, for $a \in \{0, 1\}$.

2.3.2 Randomized experiments and observational studies

There are two types of data commonly encountered in causal analysis: data collected from randomized experiments and data collected from observational studies.

Randomized experiments are the experiments that each unit receives treatment randomly (e.g., a medical trial where a random proportion of patients receives treatment). They allow the great reliability and validity of statistical estimates of causal effects. A naive ATE estimator of the difference between the treated and untreated with data from randomized experiments is unbiased. Such data is of great quality, but sometimes it is impossible to obtain.

Observational data, on the other hand, is collected from an observational study where we have no control over the assignment mechanism. This can happen when it is impractical to perform a randomized experiment (e.g., for ethical reasons) or when we cannot control the data collecting process. Special care is required when making causal statement with observational data, since the naive ATE estimator is generally biased. Despite such difficulty, observational data is easily accessible comparing to data collected from randomized experiments.

If we treat recommending an item to a user as assigning a treatment, the data collected from a typical recommender system is an example of observational data. We leverage this connection in Chapter 6 to develop a causal inference approach to recommendation.

Chapter 3

Scalable Music Tagging with Poisson Factorization

Automatic music tagging is an important but challenging problem within Music Information Retrieval (MIR). In this chapter, we treat music tagging as a matrix completion problem. We apply the Poisson matrix factorization model jointly on the vector-quantized audio features and a "bag-of-tags" representation. This approach exploits the shared latent structure between semantic tags and acoustic codewords. Leveraging the stochastic variational inference, the model can tractably analyze massive music collections. We present experimental results on the CAL500 dataset and the Million Song Dataset for both annotation and retrieval tasks, illustrating the steady improvement in performance as more data is used.

3.1 Introduction

Automatic music tagging is the task of analyzing the audio content (waveform) of a music recording and assigning to it human-relevant semantic tags (Turnbull et al., 2008) – which may relate to style, genre, instrumentation, or more subtle aspects of the music, such as those contributed by users on social media sites. Such "autotagging" (Eck et al., 2007) relies on labeled training examples for each tag, and performance typically improves with the number of training examples consumed, although training schemes also take longer to complete. In the era of "Big Data", it is necessary to develop models which can rapidly handle massive amount of data; a starting point for music data is the Million Song Dataset (Bertin-Mahieux et al., 2011), which includes user tags from Last.fm.

In this chapter, we treat the automatic music tagging as a matrix completion problem, and use the techniques of stochastic variational inference to be able to learn from large amounts of data presented in an online fashion (Hoffman et al., 2013). The "matrix completion" problem treats each track as a row in a matrix, where the elements describe both the acoustic properties (represented, for instance, as a histogram of occurrences of vector-quantized acoustic features) and the relevance of a large vocabulary of tags (we describe the details about data representation in Section 3.2): We can regard the tag information as incomplete or missing for some of the rows, and seek to "complete" these rows based on information inferred from the complete, present rows.

3.1.1 Related work

There have been a large number of papers on automatic tagging of music audio in recent years. In addition to the papers mentioned above, work particularly relevant to this paper includes the Codeword Bernoulli Average (CBA) approach of Hoffman et al. (2009), which uses a similar vector-quantization (VQ) histogram representation of the audio to build a simple but effective probabilistic model for each tag in a discriminative fashion. Xie et al. (2011) directly fits a regularized logistic regression model to the normalized acoustic codeword histograms to predict each tag and achieves state-of-the-art results, and Ellis et al. (2013) further improves tagging accuracy by employing multiple generative models that capture different characteristics of a music piece, which are combined in an optimized “bag-of-systems”.

Much of the previous work has been performed on the CAL500 dataset (Turnbull et al., 2008) of 502 Western popular music tracks that were carefully labeled by at least three human annotators with their relevance to 149 distinct labels spanning instrumentation, genre, emotions, vocal characteristics, and use cases. This small dataset tends to reward approaches that can maximize the information extracted from the sparse data regardless of the computational cost. A relatively larger dataset in this domain is CAL10k (Tingle et al., 2010) with over 10,000 tracks described by over 500 tags, mined from Pandora’s website[1]. However, neither of these datasets can be considered industrial scale, which implies handling millions of tracks and tens of thousands of tags.

Matrix factorization techniques, in particular, nonnegative matrix factorization (NMF), have been widely used to analyze music signals (Hoffman et al., 2010; Liang et al., 2013) in the context of source separation. Paisley et al. (2015) derived scalable Bayesian NMF for topic modeling, which we develop here. To our knowledge, this is the first application of matrix factorization to VQ acoustic features for automatic music tagging.

[1] `http://www.pandora.com/`

3.2 Data representation

We first describe the data that is used in the matrix completion problem. For our automatic tagging system, the data comes from two sources: vector-quantized audio features and a "bag-of-tags" representation.

- **Vector-quantized audio features.** Instead of directly working with audio features, we vector quantize all the features following the standard procedure: We run the k-means algorithm on a subset of randomly selected training data to learn J cluster centroids (codewords). Then for each song, we assign each frame to the cluster with the smallest Euclidean distance to the centroid. We form the VQ feature $y_{\text{VQ}} \in \mathbb{N}^J$ by counting the number of assignments to each cluster across the entire song.

- **Bag-of-tags.** Similar to the bag-of-words representation, which is commonly used to represent documents, we represent the tagging information (whether or not the tag applies to a song) with a binary bag-of-tags vector $y_{\text{BoT}} \in \{0, 1\}^{|V|}$, where V is the set of all tags.

For each song, we will simply concatenate the VQ feature y_{VQ} and the bag-of-tags vector y_{BoT}, thus the dimension of the data is $D = J + |V|$. Figure 3.1 demonstrates the workflow of the proposed automatic tagging system. The data (left) consists of both acoustic features and bag-of-tags vectors. When we apply the matrix factorization model to this data, the latent factors we learn (rightmost) will exploit the shared latent structure between semantic tags and acoustic codewords. Therefore, we can utilize the shared latent structure to predict tags when only given the audio features.

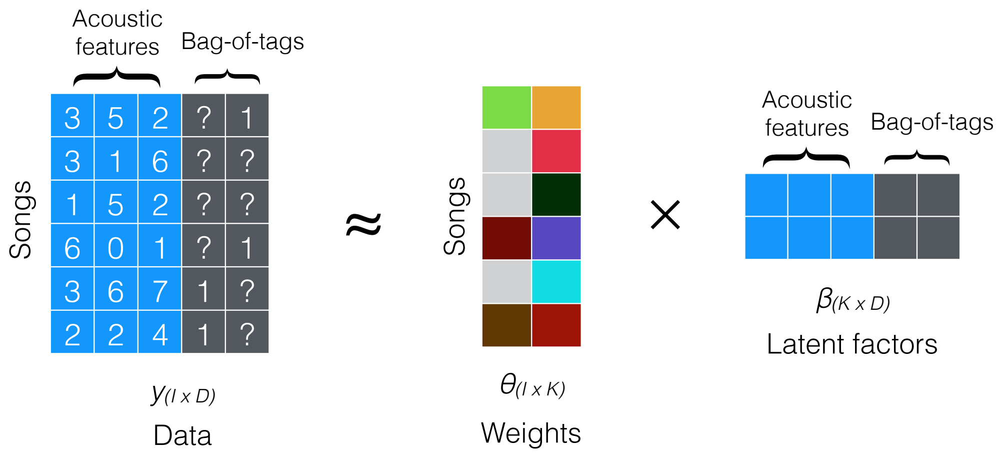


Figure 3.1: The workflow of the proposed automatic tagging system. The data (left) consists of both acoustic features and bag-of-tags vectors. When we apply the matrix factorization model to this data, the latent factors we learn (rightmost) will exploit the shared latent structure between semantic tags and acoustic codewords.

3.3 Poisson matrix factorization

We adopt the notational convention that bold letters (e.g. $\boldsymbol{y}, \boldsymbol{\theta}, \boldsymbol{\beta}$) denote matrices. $i \in \{1, \cdots, I\}$ is used to index songs. $d \in \{1, \cdots, D\}$ is used to index feature dimensions. $k \in \{1, \cdots, K\}$ is used to index latent factors from the matrix factorization model. Given the data $\boldsymbol{y} \in \mathbb{N}^{I \times D}$ as described in Section 3.2, the Poisson matrix factorization model is formulated as follows:

$$\begin{aligned} \theta_{ik} &\sim \text{Gam}(a, ac), \\ \beta_{kd} &\sim \text{Gam}(b, b), \\ y_{id} &\sim \text{Pois}\Big(\sum_{k=1}^{K} \theta_{ik}\beta_{kd}\Big), \end{aligned} \tag{3.1}$$

where $\beta_k \triangleq [\beta_{k1}, \ldots, \beta_{kD}]^\top \in \mathbb{R}_+^D$ denote the kth latent factor and $\theta_i \triangleq [\theta_{i1}, \ldots, \theta_{iK}]^\top \in \mathbb{R}_+^K$ denote the weights for song i. a and b are model hyperparameters. c is a scalar on the weights that we tune to maximize the likelihood. A graphical model representation for the Poisson matrix factorization is shown in Figure 3.2.

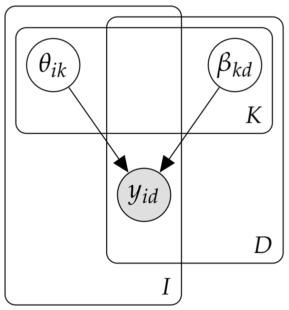


Figure 3.2: Graphical model representation for the Poisson matrix factorization.

There are a couple of reasons to choose a Poisson model over a more traditional Gaussian model (Salakhutdinov and Mnih, 2008). First, the Poisson distribution is a more natural choice to model count data. Secondly, real-world tagging data is extremely noisy and sparse. If a tag is not associated with a song in the data, it could be either because that tag does not apply to the song, or simply because no one has labeled the song with the tag yet. The Poisson matrix factorization model has the desirable property that it does not penalize values of 0 as strongly as the Gaussian distribution (Paisley et al., 2015; Gopalan et al., 2015). Therefore, even weakly labeled data can be used to learn the Poisson model.

3.4 Variational inference

To learn the latent factors $\boldsymbol{\beta}$ and the corresponding decomposition weights $\boldsymbol{\theta}$ from the training data $\boldsymbol{y}$, we need to compute the posterior distribution $p(\boldsymbol{\theta}, \boldsymbol{\beta}|\boldsymbol{y})$. However, no closed-form expression exists for this hierarchical model. We therefore employ mean-field variational inference to approximate this posterior as described in Section 2.1.2.

We choose a fully-factorized family of variational distributions,

$$q(\boldsymbol{\theta}, \boldsymbol{\beta}) = \prod_{k=1}^{K} \Big(\prod_{i=1}^{I} q_{ik}(\theta_{ik}) \Big) \Big(\prod_{d=1}^{D} q_{kd}(\beta_{kd}) \Big),$$

to approximate the posterior $p(\boldsymbol{\theta}, \boldsymbol{\beta} | \boldsymbol{y})$, so that the KL-divergence between the variational distribution and the true posterior is minimized. Following a further approximation discussed in the next section, the factorized distribution allows for a closed-form expression of this variational objective, and thus tractable inference. Here we choose variational distributions from the same family as the prior (we use the shape and rate parametrization for gamma distribution):

$$q_{ik}(\theta_{ik}) = \text{Gam}(\theta_{ik}; \gamma_{ik}, \chi_{ik}),$$
$$q_{kd}(\beta_{kd}) = \text{Gam}(\beta_{kd}; \nu_{kd}, \lambda_{kd}).$$

Minimizing the KL-divergence is equivalent to maximizing the following variational objective (ELBO):

$$\mathcal{L} = \mathbb{E}_q \left[\ln p(\boldsymbol{y}, \boldsymbol{\theta}, \boldsymbol{\beta}) \right] + H(q),$$

where $H(q)$ is the entropy of the variational distribution q. We can optimize the variational objective using coordinate ascent via two approaches: batch inference, which requires processing of the entire dataset for every iteration; or stochastic inference, which only needs a small batch of data for each iteration and can be potentially scale to much larger datasets where batch inference is no longer computationally feasible.

3.4.1 Batch inference

Although the model in Eq. 3.1 is not conditionally conjugate by itself, as demonstrated in Cemgil (2009), we can introduce latent auxiliary random variables $z_{idk} \sim \text{Poisson}(\theta_{ik}\beta_{kd})$ ($y_{id} = \sum_k z_{idk}$) with the variational distribution being $q(z_{idk}) = \text{Mult}(z_{id}; \phi_{id})$, where $z_{id} \in \mathbb{N}^K$, $\phi_{idk} \geq 0$ and $\sum_k \phi_{idk} = 1$. This makes the model conditionally conjugate, which means that closed-form coordinate ascent updates are available.

Following the standard results of variational inference for conditionally conjugate model (e.g. Hoffman et al. (2013)), we can obtain the updates for θ_{ik}:

$$\begin{aligned} \gamma_{ik} &= a + \sum_{d=1}^{D} y_{id}\phi_{idk}, \\ \chi_{ik} &= ac + \sum_{d=1}^{D} \mathbb{E}_q\left[\beta_{kd}\right]. \end{aligned} \tag{3.2}$$

The scale c is updated as:

$$c^{-1} = \frac{1}{IK}\sum_{i,k}\mathbb{E}_q[\theta_{ik}].$$

Similarly, we can obtain the updates for β_{kd}:

$$\begin{aligned} \nu_{kd} &= b + \sum_{i=1}^{I} y_{id}\phi_{idk}, \\ \lambda_{kd} &= b + \sum_{i=1}^{I} \mathbb{E}_q[\theta_{ik}]. \end{aligned} \tag{3.3}$$

Finally, for the auxiliary variables z_{idk}, the following update is applied:

$$\phi_{idk} \propto \exp\{\mathbb{E}_q[\ln \theta_{ik}\beta_{kd}]\}.$$

Note that this update should be applied after either updating θ_{ik} or β_{kd}. The necessary expectations for θ_{ik} are:

$$\mathbb{E}_q[\theta_{ik}] = \gamma_{ik}/\chi_{ik},$$
$$\mathbb{E}_q[\ln \theta_{ik}] = \psi(\gamma_{ik}) - \ln \chi_{ik},$$

where $\psi(\cdot)$ is the digamma function. The expectations for β_{kd} have the same form, but use ν_{kd} and λ_{kd}. The full algorithm of batch inference for the Poisson matrix factorization is summarized in Algorithm 3.

3.4.2 Stochastic inference

Batch inference will alternate between updating $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ using the entire data at each iteration until convergence to a local optimum, which could be computationally intensive for large datasets. We can instead adopt stochastic optimization by selecting a subset (mini-batch) of the data at iteration t, indexed by $B_t \subset \{1, \cdots, I\}$, and optimizing over a noisy version of the variational objective $\mathcal{L}$:

$$\mathcal{L}_t = \frac{I}{|B_t|} \sum_{i \in B_t} \mathbb{E}_q\left[\ln p(y_i, \theta_i | \boldsymbol{\beta})\right] + \mathbb{E}_q\left[\ln p(\boldsymbol{\beta})\right] + H(q). \tag{3.4}$$

By optimizing $\mathcal{L}_t$, we are optimizing $\mathcal{L}$ in expectation.

The updates for weights θ_{ik} and auxiliary variables z_{idk} are essentially the same as those of batch inference, except that now we are only inferring weights for the mini-batch of data for $i \in B_t$. The optimal scale c is updated accordingly:

$$c^{-1} = \frac{1}{|B_t|K} \sum_{i \in B_t, k} \mathbb{E}_q[\theta_{ik}].$$

Algorithm 3: BatchVI Batch variational inference for the Poisson matrix factorization

```
Input: Acoustic features and bag-of-tags vectors $\boldsymbol{y}$, hyperparameters $a$ and $b$
Output: Variatioanl parameters $\boldsymbol{\gamma}$, $\boldsymbol{\chi}$, $\boldsymbol{\nu}$, $\boldsymbol{\lambda}$
Randomly initialize variational parameters $\boldsymbol{\gamma}$, $\boldsymbol{\chi}$, $\boldsymbol{\nu}$, $\boldsymbol{\lambda}$
while not converged do
    for $(i, d) : y_{id} > 0$ do
        for $k \leftarrow 1$ to $K$ do
            Update auxiliary variables $\phi_{idk} \propto \exp\{\mathbb{E}_q[\ln \theta_{ik}\beta_{kd}]\}$.
        end
    end
    for $i \leftarrow 1$ to $I$ do
        for $k \leftarrow 1$ to $K$ do
            Update variational parameters $\gamma_{ik}$ and $\chi_{ik}$ for weights $\theta_{ik}$ (Eq. 3.2)
        end
    end
    Update scale $c^{-1} = \frac{1}{IK}\sum_{i,k}\mathbb{E}_q[\theta_{ik}]$
    for $(i, d) : y_{id} > 0$ do
        for $k \leftarrow 1$ to $K$ do
            Update auxiliary variables $\phi_{idk} \propto \exp\{\mathbb{E}_q[\ln \theta_{ik}\beta_{kd}]\}$.
        end
    end
    for $d \leftarrow 1$ to $D$ do
        for $k \leftarrow 1$ to $K$ do
            Update variational parameters $\nu_{kd}$ and $\lambda_{kd}$ for latent factors $\beta_{kd}$ (Eq. 3.3)
        end
    end
end
return $\boldsymbol{\gamma}$, $\boldsymbol{\chi}$, $\boldsymbol{\nu}$, $\boldsymbol{\lambda}$
```

After alternating between updating weights θ_{ik} and latent variables z_{idk} until convergence, we can take a gradient step, preconditioned by the inverse Fisher information matrix of variational distribution $q_{kd}(\beta_{kd})$, to optimize β_{kd} (see Hoffman et al. (2013) for more technical details),

$$\nu_{kd}^{(t)} = (1-\rho_t)\nu_{kd}^{(t-1)} + \rho_t\left(b + \frac{I}{|B_t|}\sum_{i\in B_t} y_{id}\phi_{idk}\right),$$
$$\lambda_{kd}^{(t)} = (1-\rho_t)\lambda_{kd}^{(t-1)} + \rho_t\left(b + \frac{I}{|B_t|}\sum_{i\in B_t} \mathbb{E}_q[\theta_{ik}]\right),$$

where $\rho_t > 0$ is a step size at iteration t. To ensure convergence (Bottou, 1998), the following conditions must be satisfied:

$$\textstyle\sum_{t=1}^{\infty} \rho_t = \infty, \quad \sum_{t=1}^{\infty} \rho_t^2 < \infty.$$

One possible choice of ρ_t is $\rho_t = (t_0 + t)^{-\kappa}$ for $t_0 > 0$ and $\kappa \in (0.5, 1]$. It has been shown (Hoffman et al., 2013) that this update corresponds to stochastic optimization with a natural gradient step, which better fits the geometry of the parameter space for probability distributions. The full algorithm for stochastic variational inference is summarized in Algorithm 4. Note that unlike Algorithm 3 where the variational parameters for weights γ and χ are also returned, here we only return the learned variational parameters for latent factors $\boldsymbol{\nu}$ and $\boldsymbol{\lambda}$ to demonstrate the "online" natural of the stochastic variational inference algorithm: the data is processed in mini-batches and there is no need to keep track of any old data that has been already analyzed.

Algorithm 4: SVI Stochastic variational inference for the Poisson matrix factorization

```
Input: Acoustic features and bag-of-tags vectors $\boldsymbol{y}$, hyperparameters $a$, $b$, $t_0$, $\kappa$, and
       mini-batch size
Output: Variatioanl parameters $\boldsymbol{\nu}$, $\boldsymbol{\lambda}$
Randomly initialize variational parameters $\boldsymbol{\nu}$, $\boldsymbol{\lambda}$
for $t \leftarrow 1, \dots$ do
    Subsample a mini-batch of data $B_t$
    while not converged do
        for $(i, d) : i \in B_t$ and $y_{id} > 0$ do
            for $k \leftarrow 1$ to $K$ do
                Update auxiliary variables $\phi_{idk} \propto \exp\{\mathbb{E}_q[\ln \theta_{ik}\beta_{kd}]\}$.
            end
        end
        for $i \in B_t$ do
            for $k \leftarrow 1$ to $K$ do
                Update variational parameters $\gamma_{ik}$ and $\chi_{ik}$ for weights $\theta_{ik}$ (Eq. 3.2)
            end
        end
        Update scale $c^{-1} = \frac{1}{|B_t|K} \sum_{i \in B_t, k} \mathbb{E}_q[\theta_{ik}]$
    end
    Set step size $\rho_t = (t_0 + t)^{-\kappa}$
    for $d \leftarrow 1$ to $D$ do
        for $k \leftarrow 1$ to $K$ do
            Take natural gradient steps for latent factors:
                $\nu_{kd}^{(t)} = (1 - \rho_t)\nu_{kd}^{(t-1)} + \rho_t \left( b + \frac{I}{|B_t|} \sum_{i \in B_t} y_{id}\phi_{idk} \right)$
                $\lambda_{kd}^{(t)} = (1 - \rho_t)\lambda_{kd}^{(t-1)} + \rho_t \left( b + \frac{I}{|B_t|} \sum_{i \in B_t} \mathbb{E}_q[\theta_{ik}] \right)$
        end
    end
end
return $\boldsymbol{\nu}$, $\boldsymbol{\lambda}$
```

3.4.3 Generalizing to new songs

Once the latent factor $\boldsymbol{\beta} \in \mathbb{R}_{+}^{K \times D}$ is inferred, we can naturally divide it into two blocks: the VQ part $\boldsymbol{\beta}_{\mathrm{VQ}} \in \mathbb{R}_{+}^{K \times J}$, and the bag-of-tags part $\boldsymbol{\beta}_{\mathrm{BoT}} \in \mathbb{R}_{+}^{K \times |V|}$. See the rightmost of Figure 3.1 for an illustration.

Given a new song, we can first obtain the VQ feature y_{VQ} and fit it with $\boldsymbol{\beta}_{\mathrm{VQ}}$ to compute posterior of the weights $p(\theta | y_{\mathrm{VQ}}, \boldsymbol{\beta}_{\mathrm{VQ}})$. We can approximate this posterior with the variational inference algorithm in Section 3.4.1 with $\boldsymbol{\beta}$ fixed. Then to predict tags, we can compute the expectation of the dot product between the weights θ and $\boldsymbol{\beta}_{\mathrm{BoT}}$ under the variational distribution:

$$\hat{y}_{\mathrm{BoT}} = \mathbb{E}_q \left[\theta^T \boldsymbol{\beta}_{\mathrm{BoT}} \right] . \tag{3.5}$$

Since for different songs the weights θ may be scaled differently, before computing the dot product we normalize $\mathbb{E}_q [\theta]$ so that it lives on the probability simplex. To do automatic tagging, we could annotate the song with top M tags according to $\hat{y}_{\mathrm{BoT}}$. To compensate for a lack of diversity in the annotations, we adopt the same heuristic used in Hoffman et al. (2009) by introducing a "diversity factor" d: For each predicted score, we subtract d times the mean score for that tag. In our system, we set $d = 3$.

3.5 Evaluation

We evaluate the model's performance on an annotation task and a retrieval task using CAL500 (Turnbull et al., 2008) and Million Song Dataset (MSD) (Bertin-Mahieux et al., 2011). Unlike the CAL500 dataset where tracks are carefully-annotated, the Last.fm dataset associated with MSD comes from real-world user tagging, and thus contains only weakly labeled data with a tagging vocabulary that is much larger and more diverse.

We compare our results on these tasks with two other sets of codebook-based methods: Codeword Bernoulli Average (CBA) (Hoffman et al., 2009) and ℓ_2-regularized logistic regression (Xie et al., 2011). Like the Poisson matrix factorization model, both methods are easy to train and can scale to relatively large dataset on a single machine. However, since both methods perform optimization in a batch fashion, we will later refer to them as "batch algorithms", along with the Poisson model with batch inference described in Section 3.4.1.

For the hyperparameters of the Poisson matrix factorization model, we set $a = b = 0.1$, and the number of latent factors $K = 100$. To learn the latent factors $\boldsymbol{\beta}$, we followed the procedure in Algorithm 3 for batch inference until the relative increase on the ELBO is less than 0.05%. For stochastic inference, we followed the procedure in Algorithm 4 and used a mini-batch size $|B_t| = 1,000$ unless otherwise specified and took a full pass of the randomly permuted data. As for the learning rate, we set $t_0 = 1$ and $\kappa = 0.6$. All the source code in Python is available online[2].

3.5.1 Annotation task

The purpose of annotation task is to automatically tag unlabeled songs. To evaluate the model's ability for annotation, we computed the average per-tag precision, recall, and F-score on a test set. Per-tag precision is defined as the average fraction of songs that the model annotates with tag v that are actually labeled v. Per-tag recall is defined as the average fraction of songs that are actually labeled v that the model also annotates with tag v. F-score is the harmonic mean of precision and recall, and is one overall metric for annotation performance.

[2]`http://github.com/dawenl/stochastic_PMF`

3.5.2 Retrieval task

The purpose of the retrieval task is, when given a query tag v, to provide a list of songs which are related to tag v. To evaluate the models' retrieval performance, for each tag in the vocabulary we ranked each song in the test set by the predicted score from Eq. 3.5. We evaluated the area under the receiver-operator curve (AROC) and mean average precision (MAP) for each ranking. AROC is defined as the area under the curve, which plots the true positive rate against the false positive rate, and MAP is defined as the mean of the average precision (AP) for each tag, which is the average of the precisions at each possible level of recall.

3.5.3 Results on CAL500

Following the procedure similar to that described in Hoffman et al. (2009); Xie et al. (2011), we performed a 5-fold cross-validation to evaluate the annotation and retrieval performance on CAL500. We selected the top 78 tags, which are annotated more than 50 times in the dataset, and learned a codebook of size $J = 2000$. For the annotation task, we labeled each song with the top 10 tags based on the predicted score. Since CAL500 is a relatively small dataset, we only performed batch inference for the Poisson matrix factorization model.

The results are reported in Table 3.1, which shows that the Poisson model has comparable performance on the annotation task, and does slightly worse on the retrieval task. As mentioned in Section 3.3, the Poisson matrix factorization model is particularly suitable for noisy and sparse data where 0's are not necessarily interpreted as explicit observations. However, this may not be the case for CAL500, as the vocabulary was well-chosen and the data was collected from a survey where the tagging quality is understandably higher than the actual tagging data in the real world, like the one from Last.fm. Therefore, this task

Model	Prec	Recall	F-score	AROC	MAP
CBA	0.41	0.24	0.29	0.69	0.47
ℓ_2 LogRegr	0.48	0.26	0.34	0.72	0.50
PMF-Batch	0.42	0.23	0.30	0.67	0.45

Table 3.1: Results for the top 78 popular tags on CAL500, for Codeword Bernoulli Average (CBA), ℓ_2 regularized logistic regression (ℓ_2 LogRegr), and Poisson matrix factorization with batch inference (PMF-Batch). The results for CBA and ℓ_2 LogRegr are directly copied from Xie et al. (2011).

cannot fully exploit the advantage brought by the Poisson model. Meanwhile, the amount of data in CAL500 is fairly small – the data $\boldsymbol{y}$ fit to the model is simply a 502-by-2078 matrix. This prevents us from adopting stochastic inference, which will be shown being much more effective than batch inference even on a 10,000-song dataset in Section 3.5.4.

3.5.4 Results on MSD

To demonstrate the scalability of the Poisson matrix factorization model, we conducted experiments using MSD and the associated Last.fm dataset. To our knowledge, there has not been any previous work where music tagging results are reported on the MSD.

Since the Last.fm dataset contains 522,366 unique tags, it is not realistic to build the model with all of them. We first selected the tags with more than 1,000 appearances and removed those which do not carry discriminative information (e.g. “my favorite”, “awesome”, “seen live”, etc.). Then we ran the stemming algorithm implemented in *NLTK*[3] to further reduce the potential duplications and correct for alternate spellings (e.g. “pop-rock” v.s. “pop rock”, “love song” v.s. “love songs”), which gave us a vocabulary of 561 tags. Using the default train/test artist split from MSD, we filtered out the songs which have been labeled with tags from the selected vocabulary. This gave us 371,209 songs for training. For test set, we further

[3]http://www.nltk.org/

selected those which have at least 20 tags (otherwise, it is likely that this song is very weakly labeled). This gave us a test set of 2,757 songs. The feature we used is the Echo Nest's timbre feature, which is very similar to MFCC.

We randomly selected 10,000 songs as the data which can fit into the memory nicely for all the batch algorithms, and trained the following models with different codebook sizes $J \in \{256, 512, 1024, 2048\}$: Codeword Bernoulli Average (CBA), ℓ_2-regularized logistic regression (ℓ_2 LogRegr), Poisson matrix factorization with batch inference (PMF-Batch) and stochastic inference by a single pass of the data (PMF-Stoc-10K). Here we used batch size $|B_t| = 500$ for PMF-Stoc-10K, as otherwise there will only be 10 mini-batches from the subset. However, given enough data, in general larger batch size will lead to relatively superior performance, since the variance of the noisy variational objective in Eq. 3.4 is smaller. To demonstrate the effectiveness of the Poisson model on massive amount of data (exploiting the stochastic algorithm's ability to run without loading the entire dataset into memory), we also trained the model with the full training set with stochastic inference (PMF-Stoc-full). For the annotation task, we labeled each song with the top 20 tags based on the predicted score.

The results are reported in Table 3.2. In general, the performance of Poisson matrix factorization is comparably better for smaller codebook size J. Specifically, for stochastic inference, even if the amount of training data is relatively small, it is not only significantly faster than batch inference, but can also help improve the performance by quite a large margin. Finally, not surprisingly, PMF-Stoc-full dominates all the metrics, regardless of the size of the codebook, because it is able to learn from more data.

Figure 3.3 illustrates how the metrics improve as more data becomes available for the Poisson matrix factorization model, showing how the F-score, AROC, and MAP improve with the number of (1000-element) mini-batches consumed up to the entire 371k training set. We see

Codebook size	Model	Precision	Recall	F-score	AROC	MAP
$J = 256$	CBA	0.112 (0.007)	0.121 (0.008)	0.116	0.695 (0.005)	0.112 (0.006)
	ℓ_2 LogRegr	0.091 (0.008)	0.093 (0.006)	0.092	0.692 (0.005)	0.110 (0.006)
	PMF-Batch	0.113 (0.007)	0.105 (0.006)	0.109	0.647 (0.005)	0.094 (0.005)
	PMF-Stoc-10K	0.116 (0.007)	0.127 (0.007)	0.121	0.682 (0.005)	0.105 (0.006)
	PMF-Stoc-full	**0.127 (0.008)**	**0.143 (0.008)**	**0.134**	**0.704 (0.005)**	**0.115 (0.006)**
$J = 512$	CBA	0.120 (0.007)	0.127 (0.008)	0.124	0.689 (0.005)	0.117 (0.006)
	ℓ_2 LogRegr	0.096 (0.008)	0.108 (0.007)	0.101	0.693 (0.005)	0.113 (0.006)
	PMF-Batch	0.111 (0.007)	0.108 (0.006)	0.109	0.645 (0.005)	0.098 (0.005)
	PMF-Stoc-10K	0.112 (0.007)	0.128 (0.007)	0.120	0.687 (0.005)	0.110 (0.006)
	PMF-Stoc-full	**0.130 (0.008)**	**0.154 (0.008)**	**0.141**	**0.715 (0.005)**	**0.122 (0.006)**
$J = 1024$	CBA	0.118 (0.007)	0.126 (0.007)	0.122	0.692 (0.005)	0.117 (0.006)
	ℓ_2 LogRegr	0.113 (0.008)	0.129 (0.008)	0.120	0.698 (0.005)	0.115 (0.006
	PMF-Batch	0.112 (0.007)	0.109 (0.006)	0.111	0.635 (0.005)	0.098 (0.006)
	PMF-Stoc-10K	0.111 (0.007)	0.127 (0.007)	0.118	0.687 (0.005)	0.111 (0.006)
	PMF-Stoc-full	**0.127 (0.008)**	**0.146 (0.008)**	**0.136**	**0.712 (0.005)**	**0.120 (0.006)**
$J = 2048$	CBA	**0.124 (0.007)**	0.129 (0.007)	0.127	0.689 (0.005)	0.117 (0.006)
	ℓ_2 LogRegr	0.115 (0.008)	0.137 (0.008)	0.125	0.698 (0.005)	**0.118 (0.006)**
	PMF-Batch	0.109 (0.007)	0.110 (0.006)	0.110	0.637 (0.005)	0.098 (0.006)
	PMF-Stoc-10K	0.107 (0.007)	0.124 (0.007)	0.115	0.682 (0.005)	0.106 (0.006)
	PMF-Stoc-full	0.120 (0.007)	**0.147 (0.008)**	**0.132**	**0.712 (0.005)**	**0.118 (0.006)**

Table 3.2: Annotation (evaluated using precision, recall, and F-score) and retrieval (evaluated using area under the receiver-operator curve (AROC) and mean average precision (MAP)) performance on the Million Song Dataset with various codebook sizes, from Codeword Bernoulli Average (CBA), ℓ_2 regularized logistic regression (ℓ_2 LogRegr), Poisson matrix factorization with batch inference (PMF-Batch) and stochastic inference by a single pass of the subset (PMF-Stoc-10K) and full data (PMF-Stoc-full). One standard error is reported in the parenthesis.

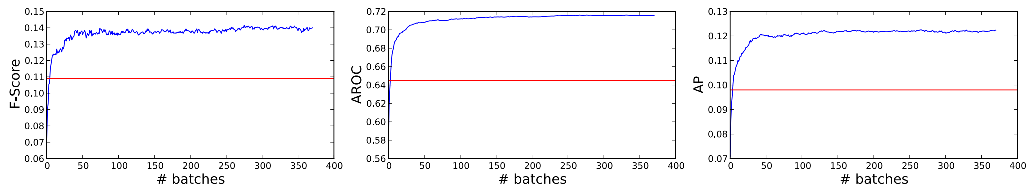


Figure 3.3: Improvement in performance with the number of mini-batches consumed for the PMF-Stoc-full system with $J = 512$. Red lines indicate the performance of PMF-Batch which is trained on 10k examples; that system's performance is exceeded after fewer than 5 mini-batches.

that initial growth is rapid, thanks to the natural gradient, with much of the benefit obtained after just 50 batches. However, we see continued improvement beyond this; it is reasonable to believe that if more data becomes available, the performance can be further improved. On the other hand, we also observe that the performance is limited by the modeling capacity of a (bi-)linear factorization model. In Chapter 4, we show that superior performance can be achieved with a deep neural net.

Table 3.3 contains information on the qualitative performance of our model. The tagging model works by capturing correlations between semantic tags and acoustic codewords in each latent factor β_k. As discussed, when a new song arrives with missing tag information, only the portion of β_k corresponding to acoustic codewords is used, and the semantic tag portion of β_k is used to make predictions of the missing tags. Similar to related topic models (Hoffman et al., 2013), we can therefore look at the highly probable tags for each β_k to understand what portion of the acoustic codeword space is being captured by that factor, and whether it is musically coherent. We show an example of this in Table 3.3, where we list the top 7 tags from 9 latent factors β_k learned by our model with $J = 512$. We sort the tags according to expected relevance under the variational distribution $\mathbb{E}_q[\beta_{kd}]$. This shows which tags are considered to have high probability for a song that has the given factor expressed. As is evident, each factor corresponds to a particular aspect of a music genre. We note that other factors contained similarly coherent tag information.

3.6 Summary

We present a codebook-based scalable music tagging model with Poisson matrix factorization. The system learns the joint behavior of acoustic features and semantic tags, which can be used to infer the most appropriate tags given the audio alone. The Poisson model is naturally less sensitive to zero values than some alternatives, making it a good match to “noisy” training examples derived from real users’ taggings, where the fact that no user has applied a tag does not necessarily imply that the term is irrelevant. By learning this model using stochastic variational inference, we are able to efficiently exploit much larger training sets than are tractable using batch approaches, making it feasible to learn from an entire set of over 370k tagged examples. Although much of the improvement comes in the earlier iterations, we see continued improvement implying this approach can benefit from much larger, effectively unlimited sources of tagged examples, as might be available on a commercial music service with millions of users.

There are a few areas where our model can be easily developed. For example, stochastic variational inference requires we set the learning rate parameters t_0 and κ, which is application-dependent. By using adaptive learning rates for stochastic variational inference (Ranganath et al., 2013), model inference can converge faster and to a better local optimal solution. From a modeling perspective, currently the hyperparameters for weights $\boldsymbol{\theta}$ are fixed, indicating that the sparsity level of the weight for each song is assumed to be the same *a priori*. Alternatively we could put *song-dependent* hyper-priors on the hyperparameters of $\boldsymbol{\theta}$ to encode the intuition that some of the songs might have denser weights because more tagging information is available. This would offer more flexibility to the current model.

"Pop"	*"Indie"*	*"Jazz"*	*"Classical"*	*"Metal"*	*"Reggae"*	*"Electronic"*	*"Experimental"*	*"Country"*
pop	indie	chillout	piano	metal	reggae	house	instrumental	country
female vocal	rock	lounge	instrumental	death metal	funk	electro	ambient	classic country
dance	alternative	chill	ambient	thrash metal	funky	electronic	experimental	male vocal
electronic	indie rock	downtempo	classic	brutal death metal	dance	dance	electronic	blues
sexy	post punk	smooth jazz	beautiful	grindcore	hip-hop	electric house	psychedelic	folk
love	psychedelic	relax	chillout	heavy metal	party	techno	progressive	love songs
synth pop	new wave	ambient	relax	black metal	sexy	minimal	rock	americana

Table 3.3: Top 7 tags from 9 latent factors for PMF-Stoc-full with $J = 512$. For each factor, we assign the closest music genre on top. As is evident, each factor corresponds to a particular aspect of a music genre.

Chapter 4

Content-Aware Collaborative Music Recommendation

Although content is fundamental to our music listening preferences (or at least we believe so), the leading performance in music recommendation is achieved by collaborative filtering methods which exploit the similarity patterns in user's listening history rather than the audio content of songs. Meanwhile, collaborative filtering has the well-known "cold-start" problem, i.e., it is unable to work with new songs that no one has listened to. Efforts on incorporating content information into collaborative filtering methods have shown success in many non-musical applications, such as scientific article recommendation. Inspired by the related work, we train a neural network on semantic tagging information as a content model and use it as a prior in a collaborative filtering model. Such a system still allows the user listening data to "speak for itself". The proposed system is evaluated on the Million Song Dataset and shows comparably better result than the collaborative filtering approaches, in addition to the favorable performance in the cold-start case.

4.1 Introduction

Music recommendation is an important yet difficult task in MIR. A recommendation system that accurately predicts users' listening preferences bears enormous commercial value. However, the high complexity and dimensionality of music data and the scarcity of user feedback makes it difficulty to create a successful music recommendation system.

Two primary approaches exist in recommendation[1]: collaborative filtering and content-based methods. For music, the state-of-the-art recommendation results have been achieved by collaborative filtering methods (e.g., all the top-ranked submissions to the Kaggle Million Song Dataset Challenge[2] are based on collaborative filtering even though the dataset also comes with content features and meta-data), which requires only information on users' listening history rather than the musical content for recommendation. The central assumption of this model is that a user is likely to accept a song that is liked by users who have similar taste. A major category of collaborative filtering approaches is based on latent factor (matrix factorization) model. It assumes that a low-dimensional representation exists for both users and songs such that the compatibility between a user and a song, modeled as their inner product in this latent space, predicts the user's fondness of the song. As discussed in Section 2.2.1, in the case that user feedback is *implicit* (e.g., whether or not the user has listened to a particular song), WMF (Hu et al., 2008) works particularly well. WMF is described in Section 2.2.2.2.

On the other hand, modeling musical content for the purpose of taste prediction is difficult due to the structural complexity present in music data which is hard to capture by simple models. Deep learning has shown its power in various pattern recognition tasks with its capability of

[1]Collective intelligence also plays a huge role in commercial recommender systems. However, it is beyond the scope of this dissertation.

[2]`https://www.kaggle.com/c/msdchallenge`

extracting hierarchical representations from raw data. In music recommendation, van den Oord et al. (2013) have experimented with neural networks on predicting the song latent representation from musical content.

It is natural to combine collaborative filtering and content models in recommendation to utilize different sources of information. A successful attempt from Wang and Blei (2011), which joins a content model on article with collaborative filtering, achieves good performance on scientific article recommendation.

Inspired by these mentioned above, we create a content-aware collaborative music recommendation system. As the name suggests, the system has two components: the content model and the collaborative filtering model. To obtain a powerful content model, we pre-train a multi-layer neural network to predict semantic tags from vector-quantized acoustic feature. The output of the last hidden layer is treated as a high-level representation of the musical content, which is used as a prior for the song latent representation in collaborative filtering. We evaluate our system on the Million Song Dataset and show competitive performance to the state-of-the-art system.

4.2 Related work

In this section we review two closely relevant models: collaborative topic model for article recommendation and deep content-based music recommendation. We also review other related work which hybridizes content and collaborative filtering models.

4.2.1 Collaborative topic model

As reviewed in Section 2.2.2, a widely used approach to recommender systems is collaborative filtering, where items are recommended to a user based on other users with similar patterns of item consumption. Matrix factorization models (Hu et al., 2008; Koren et al., 2009) are among the most successful collaborative filtering methods. In the case of the Gaussian matrix factorization (Salakhutdinov and Mnih, 2008), efficient *alternating least squares* update (Algorithms 1 and 2) exists for scalable inference, makes it an attractive option for commercial recommender systems.

Due to its content-free nature, collaborative filtering approaches can be applied in a wide range of domains. They perform well on what is called *in-matrix* predictions, i.e., recommending items that have been consumed by some users. However, this approach suffers from the well-known problem that it is unable to recommend new items that no user has consumed, or making *out-of-matrix* predictions, where content-based models are better suited. Figure 4.1 provides illustrative examples for both *in-matrix* and *out-of-matrix* predictions. Many efforts have been made to incorporate content into collaborative filtering. Wang and Blei (2011) propose the CTR model for scientific article recommendation, which is particularly relevant to our proposed method.

There are two components in CTR: a matrix factorization collaborative filtering model (WMF as described in Section 2.2.2.2) and a latent Dirichlet allocation (LDA) article content model. LDA (Blei et al., 2003) is a mixed-membership model on documents. Assuming there are K topics $\Phi = \boldsymbol{\phi}_{1:K}$, each of which is a distribution over a fixed set of vocabulary, LDA treats each document as a mixture of these topics where the topic proportion $\boldsymbol{\pi}_i$ is inferred from the data. One can understand LDA as representing documents in a low-dimensional "topic" space with the topic proportion being their coordinates. With this interpretation, the generative

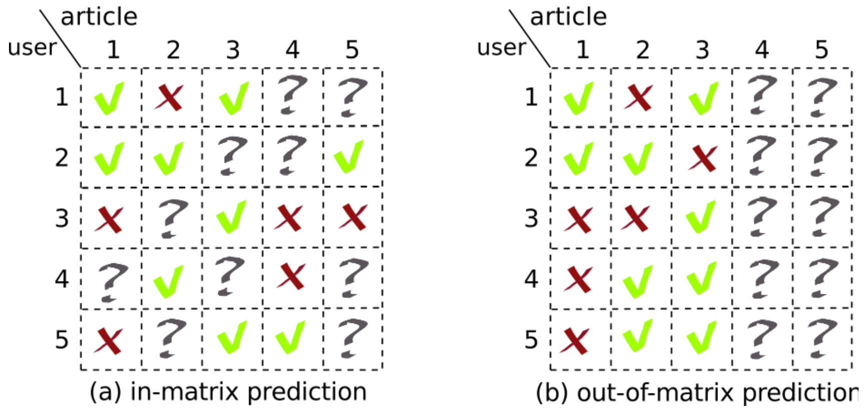


Figure 4.1: An illustration of the *in-matrix* (left) and *out-of-matrix* (right) predictions, where ✓ indicates "like", x indicates "dislike", and ? indicates "unknown". The goal of both predictions is to predict the values of ?'s. (Copied from Wang and Blei (2011))

process of CTR is as follows:

- For user $u = 1, \dots, U$, draw user latent factor: $\boldsymbol{\theta}_u \sim \mathcal{N}(\mathbf{0}, \lambda_\theta^{-1} \mathrm{I}_K)$,
- For document $i = 1, \dots, I$,
  - Draw topic proportion $\boldsymbol{\pi}_i \sim \mathrm{Dir}(\alpha)$,
  - For word n in document i,
    - Draw topic assignment $z_{in} \sim \mathrm{Discrete}(\boldsymbol{\pi}_i)$,
    - Draw word $w_{in} \sim \mathrm{Discrete}(\boldsymbol{\phi}_{z_{in}})$,
  - Draw latent factor $\boldsymbol{\beta}_i \sim \mathcal{N}(\boldsymbol{\pi}_i, \lambda_\beta^{-1} \mathrm{I}_K)$,
- For user-document pair (u, i), draw feedback: $r_{ui} \sim \mathcal{N}(\boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i, c_{ui}^{-1})$.

Here the confidence c_{ui} is set the same as in WMF. We can see CTR differs from WMF in that CTR assumes that the item latent factor $\boldsymbol{\beta}_i$ is close to the topic proportion $\boldsymbol{\pi}_i$ but could deviate from it if necessary. This allows the user-item interaction data to "speak for itself". An attractive characteristic of CTR is its capability of making *out-of-matrix* predictions. This is done by using the topic proportion $\boldsymbol{\pi}_i$ alone as the item latent factor: $\hat{r}_{ui} = \boldsymbol{\theta}_u^\top \boldsymbol{\pi}_i$, which is not possible in the traditional collaborative filtering model.

Although CTR achieves better recommendation performance than WMF, it does not scale well with large data. Since the model is not conditionally conjugate: the prior on $\boldsymbol{\beta}_i$ comes from a Dirichlet-distributed random variable $\boldsymbol{\pi}_i$, topic proportion $\boldsymbol{\pi}_i$ cannot be updated analytically and slower numerical optimization method is required. To address this problem, Gopalan et al. (2014) propose the collaborative topic Poisson factorization (CTPF). This model replaces the Gaussian likelihood and Gaussian prior in CTR with Poisson likelihood and gamma prior, thus becoming conditionally conjugate with closed-form updates. Experiments on large-scale scientific article recommendation demonstrate that CTPF performs significantly better than CTR.

The main difference that sets our method apart from collaborative topic model is the content model. As a feature extractor, LDA can only produce linear factors due to its bilinear nature. On the other hand, multi-layer neural network used by in our system is capable of capturing the non-linearities in the feature space.

4.2.2 Deep content-based music recommendation

Previous attempts on content-based music recommendation have achieved promising results. van den Oord et al. (2013) utilize a neural network to map acoustic features to the song latent factors learned from WMF. As a result, given a new song that no one has ever listened to, a latent factor can still be predicted from the network and recommendation can be done in the same fashion as with a regular collaborative filtering model.

Our method is very similar to this approach, but we will point out two major differences:

- First, the neural network is used for different purposes. We use it as a content feature extractor, just like LDA in the collaborative topic model. The neural network in van den Oord et al. (2013) maps content directly to the latent factors learned from pure collaborative filtering, and the resulting model is expected to operate similarly to collaborative filtering even when usage data is absent.

- Since the neural network is trained to map content to the latent factors learned from WMF, the performance of van den Oord et al. (2013) is unlikely to surpass that of WMF. What we propose in this paper, on the other hand, uses content as an *addition* to WMF, in a similar manner as the collaborative topic model described in Section 4.2.1. As we show in the experiment, we are able to achieve better result than WMF when we only have limited amount of user feedback.

Other approaches that hybridize content and collaborative models include Yoshii et al. (2006), McFee et al. (2010), and Wang and Wang (2014). Yoshii et al. (2006) train a three-way probabilistic model that joins user, item, and content by a latent "topic" variable; the model focuses on explicit feedback (user ratings). McFee et al. (2010) take a similar approach to van den Oord et al. (2013) and learn a content-based similarity function from collaborative filtering via metric learning. Wang and Wang (2014) also use a neural network to incorporate music content into the collaborative filtering model. The major difference is that in Wang and Wang (2014) the output of the neural network is treated as item latent factor and the neural network is trained to minimize a loss function that is based on collaborative filtering. Therefore the content model itself does not have explicit musicological meaning, as opposed to neural network in our system which is trained to predict semantic music tags.

4.3 Content-aware music recommendation

Adopting the same structure as that of CTR, our system consists of two components: a content model which is based on a pre-trained neural network and a collaborative filtering model based on matrix factorization.

4.3.1 Supervised pre-training

Inspired by the success of transfer learning in computer vision which exploits deep convolutional neural networks (Krizhevsky et al., 2012), in our system we pre-train a multi-layer neural network in a supervised semantic tagging prediction task and use it as the content model.

Our training data is the same from Section 3.5 which consists of 370k tracks from the Million Song Dataset and the pre-processed Last.fm data with a vocabulary of 561 tags, including genre, mood, instrumentation, etc. We use the Echonest's timbre feature, which is very similar to MFCC. To get the song-level features, we vector-quantize all the timbre features following the standard procedure: We run the k-means algorithm on a subset of randomly selected training data to learn $V = 1024$ cluster centroids (codewords). Then for each song, we assign each segment (frame) to the cluster with the smallest Euclidean distance to the centroid. We aggregate the VQ feature of song i ($\boldsymbol{x}_i \in \mathbb{R}_+^V$) by counting the number of assignments to each cluster across the entire song and then normalize it to have unit ℓ_1 norm to account for the various lengths.

We treat music tagging as a binary classification problem: For each tag, we make independent predictions on whether the song is tagged with it or not. We fit the output of the network $f(\boldsymbol{x}_i) \in \mathbb{R}^{561}$ into logistic regression classifiers with independent cross entropy loss. Therefore, given tag labels $y_{it} \in \{-1, 1\}$ for song i and tag t, the network is trained to minimize the following objective:

$$\mathcal{L}_{\text{tag}} = \sum_{i,t} \log(1 + \exp(-y_{it} f_t(\boldsymbol{x}_i)))$$

Here we use a network with three fully-connected hidden layers and rectified linear units (ReLU) activations with dropout ($p = 0.5$) (Srivastava et al., 2014). Each layer has 1,200 neurons. Stochastic gradient descent with mini-batch of size 100 is used with AdaGrad (Duchi et al., 2011) for adjusting the learning rate[3]. We notice that both dropout and AdaGrad are crucial for getting the good performance. The tagging performance is reported in Section 4.4.1.

4.3.2 Content-aware collaborative filtering

We can interpret the output of the last hidden layer $\boldsymbol{h}_i \in \mathbb{R}^{F_h}$ (here $F_h = 1200$) as a latent content representation of song i. Because of the way the network is trained, this latent representation is supposed to be highly correlated to the semantic tags ("topics" of music). Therefore, we can take a similar approach to the collaborative topic model and use this representation in a collaborative filtering model.

The generative process for the proposed model is as follows:

- For user $u = 1, \ldots, U$, draw user latent factor: $\boldsymbol{\theta}_u \sim \mathcal{N}(\mathbf{0}, \lambda_\theta^{-1} \mathrm{I}_K)$.

[3]The source code for training the neural network is available at: `https://github.com/dawenl/deep_tagging`

- For each song $i = 1, \dots, I$, draw song latent factor: $\boldsymbol{\beta}_i \sim \mathcal{N}(\boldsymbol{W}^\top \boldsymbol{h}_i, \lambda_\beta^{-1} I_K)$.
- For each user-song pair (u, i), draw implicit feedback (whether user u listened to song i): $r_{ui} \sim \mathcal{N}(\boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i, c_{ui}^{-1})$.

Since the dimensionality of $\boldsymbol{h}_i$ is generally much higher than that of the song latent factor $\boldsymbol{\beta}_i$, we use a weight matrix $\boldsymbol{W} \in \mathbb{R}^{F_h \times K}$ that transforms the learned content representation from the neural networks into the collaborative filtering latent space via $\boldsymbol{W}^\top \boldsymbol{h}_i$. The precision parameter λ_β balances how the song latent vector $\boldsymbol{\beta}_i$ deviates from the content feature: larger λ_β will force the song latent factors to stay close to the content feature. We set the confidence c_{ui} following the same way as in Hu et al. (2008):

$$c_{ui} = 1 + \alpha \log(1 + r_{ui}/\epsilon)$$

where α and ϵ are tunable hyperparameters. A graphical model representation of the content-aware collaborative filtering model is shown in Figure 4.2. We use $f(\cdot)$ to indicate that the content feature $\boldsymbol{h}_i$ comes from the pre-trained neural network.

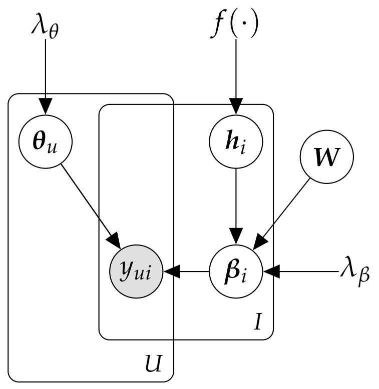


Figure 4.2: Graphical model representation for the content-aware collaborative filtering model.

We want to emphasize that our proposed model is *content-aware* instead of *content-based*.

Just like collaborative topic model, our proposed model is still fundamentally based on collaborative filtering (WMF, to be more precise). The content model is only used as a prior and can be deviated if the model thinks it is necessary to explain the data. As a matter of fact, with sufficient amount of feedback data, the model will almost always choose to derivate from the content feature, because the collaborative filtering part of the model will find it better at explaining the feedback data.

Inference. We estimate the model parameters $\{\boldsymbol{\theta}_{1:U}, \boldsymbol{\beta}_{1:I}, \boldsymbol{W}\}$ via maximum *a posteriori* since it enables us to tractably fit the model to large-scale datasets. The complete log-likelihood of the model is written as:

$$\mathcal{L} = -\sum_{u,i} \frac{c_{ui}}{2}(r_{ui} - \boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i)^2 - \frac{\lambda_\theta}{2}\sum_u \|\boldsymbol{\theta}_u\|_2^2 - \frac{\lambda_\beta}{2}\sum_i \|\boldsymbol{\beta}_i - \boldsymbol{W}^\top \boldsymbol{h}_i\|_2^2.$$

From this objective, we can also interpret the model as regularizing the song latent factors $\boldsymbol{\beta}_i$ towards something musicologically meaningful ($\boldsymbol{h}_i$), instead of $\mathbf{0}$. Taking the gradient of the complete log-likelihood with respect to the model parameters $\boldsymbol{\theta}_u$, $\boldsymbol{\beta}_i$, and $\boldsymbol{W}$, and setting it to 0, respectively, we can obtain the following closed-form coordinate updates:

$$\boldsymbol{\theta}_u^{\text{new}} \leftarrow (\sum_i c_{ui}\boldsymbol{\beta}_i\boldsymbol{\beta}_i^\top + \lambda_\theta \mathrm{I}_K)^{-1}(\sum_i c_{ui} r_{ui}\boldsymbol{\beta}_i) \tag{4.1}$$

$$\boldsymbol{\beta}_i^{\text{new}} \leftarrow (\sum_u c_{ui}\boldsymbol{\theta}_u\boldsymbol{\theta}_u^\top + \lambda_\beta \mathrm{I}_K)^{-1}(\sum_u c_{ui} r_{ui}\boldsymbol{\theta}_u + \lambda_\beta \boldsymbol{W}^\top \boldsymbol{h}_i) \tag{4.2}$$

$$\boldsymbol{W}^{\text{new}} \leftarrow (\sum_i \boldsymbol{h}_i\boldsymbol{h}_i^\top + \lambda_W \mathrm{I}_{F_h})^{-1}(\sum_i \boldsymbol{h}_i\boldsymbol{\beta}_i^\top) \tag{4.3}$$

When updating $\boldsymbol{W}$, we add a small ridge term λ_W to the diagonal of the matrix to regularize and avoid numerical problems when inverting. These updates are very similar to *alternating least squares* (ALS) of WMF. The main difference is in how we update $\boldsymbol{\beta}_i$ in the proposed model. We can view each update of ALS as a weighted ridge regression. Therefore, the update for β_i is collectively performing ridge regression with two sources of information: the click data r_{ui} and the (transformed) content feature $\boldsymbol{W}^\top \boldsymbol{h}_i$. Alternating between updating $\boldsymbol{\theta}_{1:U}$, $\boldsymbol{\beta}_{1:I}$, and $\boldsymbol{W}$, we are guaranteed to reach a stationary point of the complete log-likelihood.

The same technique used in Hu et al. (2008) to speed up computation (described in Section 2.2.2.2) can be applied here. This enables us to apply our model to large-scale music corpus and user-item interaction, which is not possible for CTR. The full algorithm is summarized in Algorithm 5.

Algorithm 5: CA-ALS Content-aware collaborative filtering inference

```
Input: Click matrix $r_{ui}$, the confidence for clicked $c_1$ and unclicked $c_0$, regularization
       parameters $\lambda_\theta$, $\lambda_\beta$ and $\lambda_W$
Output: User latent factors $\boldsymbol{\theta}_{1:U}$, item latent factors $\boldsymbol{\beta}_{1:I}$, and weight matrix $\boldsymbol{W}$
Randomly initialize $\boldsymbol{\theta}_{1:U}$, $\boldsymbol{\beta}_{1:I}$, and $\boldsymbol{W}$
while not converged do
    Precompute $\sum_i c_0 \boldsymbol{\beta}_i \boldsymbol{\beta}_i^\top + \lambda_\theta \mathrm{I}_K$
    for $u \leftarrow 1$ to $U$ do
        Update user factor $\boldsymbol{\theta}_u$ (Eq. 4.1)
    end
    Precompute $\sum_u c_0 \boldsymbol{\theta}_u \boldsymbol{\theta}_u^\top + \lambda_\beta \mathrm{I}_K$
    for $i \leftarrow 1$ to $I$ do
        Update item factor $\boldsymbol{\beta}_i$ (Eq. 4.2)
    end
    Update weight matrix $\boldsymbol{W}$ (Eq. 4.3)
end
return $\boldsymbol{\theta}_{1:U}$, $\boldsymbol{\beta}_{1:I}$, $\boldsymbol{W}$
```

Prediction. After the model is trained, we can make *in-matrix* prediction by $\hat{r}_{ui} = \boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i$. Similar to the collaborative topic model, we can also make *out-of-matrix* prediction for songs that no one has listened to by only using the content $\hat{r}_{ui} = \boldsymbol{\theta}_u^\top (\boldsymbol{W}\boldsymbol{h}_i)$.

4.4 Evaluation

We first evaluate our system on the pre-training tag prediction task to ensure the quality of the extracted features, and then measure its recommendation performance in comparison with related models[4].

4.4.1 Tag prediction

Evaluation tasks and metrics. We evaluate the pre-trained neural network on semantic tags with an annotation task and a retrieval task. We use the same dataset in Section 3.5 from the Million Song Dataset (Bertin-Mahieux et al., 2011) and compare with the result in Section 3.5 which, to our knowledge, is the state-of-the-art performance on large-scale tag prediction. Note that we only use tag prediction as a proxy to measure the quality of the content model and do not argue for our approach as an optimal one to automatic music tagging.

For the annotation task we seek to automatically tag unlabeled songs. To evaluate the model's ability to annotate songs, we compute the average per-tag precision, recall, and F-score on the held-out test set. For the retrieval task, given a query tag we seek to provide a list of songs which are related to that tag. To evaluate retrieval performance, for each tag in the vocabulary we ranked each song in the test set by the predicted probability. We then calculate the area under the receiver-operator curve (AROC) and mean average precision (MAP) for each ranking. The detailed description of the metrics can be found in Section 3.5.

Tagging performance and discussion. The results are reported in Table 4.1, which shows that the pre-trained neural network performs significantly better than the approach based on

[4] `https://github.com/dawenl/content_wmf` contains the source code for training the proposed model and reproducing the experimental results for recommendation in Section 4.4.2.

Model	Prec	Recall	F-score	AROC	MAP
SPMF	0.127	0.146	0.136	0.712	0.120
NNet	0.184	0.207	0.195	0.781	0.178

Table 4.1: Annotation and retrieval performance on the Million Song Dataset from Poisson matrix factorization with stochastic inference (SPMF) (described in Chapter 3) and the pre-trained neural network (NNet) described in Section 4.3.1. The standard error is on the order of 0.01, thus not included here.

Poisson matrix factorization in Chapter 3. This is not surprising for two reasons: 1) Here we treat tag prediction as a supervised task and train a multi-layer neural network, while in Chapter 3 the problem is formulated as an unsupervised learning task to account for the uncertainty in the user-generated tags (which incidentally can be considered as a typical example of implicit feedback). 2) Similar to LDA, Poisson matrix factorization can only capture linear factor, whose expressive power is much weaker than that of a multi-layer neural network.

Nevertheless, the results confirm that our pre-trained neural network can be considered as an effective content feature extractor and we will use the output of the last hidden layer as the content feature.

Note that our neural network has relatively simple structure and does not directly use raw acoustic features (e.g., log-mel spectrograms) as input. It is reasonable to believe that with a more complex network structure and low-level acoustic feature, we should be able to achieve better tagging performance and obtain a more powerful content feature extractor, which could further boost the performance of our proposed recommendation method.

4.4.2 Recommendation

Data preparation. We use the Taste Profile Subset which is part of the Million Song Dataset to evaluate the recommendation performance. It contains listening history in the form of play counts from one million users with more than 40 million (user, song, play count) triplets. We first binarize all the play counts[5] and create two complementary subsets, a dense one (DEN) and a sparse one (SPR):

For the dense subset (DEN), we intend to create a subset that is reasonably dense so that the traditional collaborative filtering model will have good performance. We remove the users who have less than 20 songs in their listening history and songs that are listened to by less than 50 users, obtaining a subset with 613,682 users and 97,414 songs with more than 38 million user-song pairs (sparsity level 0.064%). For the sparse subset (SPR), on the contrary, we only keep the users who have less than 20 songs in their listening history and songs that are listened to by less than 50 users, yielding a highly sparse (0.002%) subset with 564,437 users and 260,345 songs.

We select 5% of the songs from DEN (4,871) for *out-of-matrix* prediction. For both subsets we split 20% and 10% as test and validation sets, respectively. Validation set is used to select hyperparameters, as well as monitor convergence by computing predictive likelihood.

Competing methods. We compare our proposed method (denoted as *CF* + *deep*) with WMF (Hu et al., 2008), as well as the following three methods:

- *CF* + *shallow*: A simple baseline where we directly use the normalized VQ feature $\boldsymbol{x}_i$ in place of the feature extracted from the neural network $\boldsymbol{h}_i$. This baseline is mainly

[5]In practice, we find that the performances using actual play counts and binarized indicators are very close for our model.

used to demonstrate the necessity of an effective feature extractor for *out-of-matrix* prediction.

- Poisson matrix factorization (PMF) (Gopalan et al., 2015): Just like WMF, PMF is a matrix factorization model for collaborative filtering. Instead of Gaussian likelihood and priors on the latent factors, it utilizes Poisson likelihood model and gamma priors to learn nonnegative embeddings for both users and items. Concretely, it follows the following generative process:

  - For user $u = 1, \ldots, U$, draw user latent factor $\theta_{uk} \sim \mathrm{Gam}(a, b)$,
  - For item $i = 1, \ldots, I$, draw item latent factor $\beta_{ki} \sim \mathrm{Gam}(c, d)$,
  - For each user-item pair (u, i), draw feedback: $r_{ui} \sim \mathrm{Pois}(\boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i)$.

  The biggest advantage of PMF is computational. As shown in Gopalan et al. (2015), the inference algorithm has complexity that scales linearly with the number of non-zero entries in the user-item matrix, which is the same as that of WMF.

- CTPF (Gopalan et al., 2014): As mentioned in Section 4.2.1, CTR cannot scale to large datasets due to the non-conjugacy of the model. CTPF is proposed as a workaround: it incorporates the content information into PMF in the same way as CTR incorporates the content into WMF. The generative process is (recall $V = 1024$ is the size of the codebook for vector-quantization and we use $v \in \{1, \ldots, V\}$ to index codeword):

  - For topic $k = 1, \ldots, K$, draw topic $\gamma_{vk} \sim \mathrm{Gam}(c, d)$,
  - For user $u = 1, \ldots, U$, draw user latent factor $\theta_{uk} \sim \mathrm{Gam}(a, b)$,
  - For item $i = 1, \ldots, I$,

  - $*$ Draw topic intensities: $\beta_{ik} \sim \text{Gam}(e, f)$,
  - $*$ Draw additive offset $\epsilon_{ik} \sim \text{Gam}(g, h)$,
  - $*$ Draw codeword count $c_{iv} \sim \text{Pois}(\boldsymbol{\beta}_i^\top \gamma_v)$,
- For each user-item pair (u, i), draw feedback: $r_{ui} \sim \text{Pois}(\boldsymbol{\theta}_u^\top (\boldsymbol{\beta}_i + \boldsymbol{\epsilon}_i))$.

Additionally, it is conditionally conjugate with closed-form variational inference updates and enjoys the same computational efficiency as PMF. Therefore, it can be applied to large-scale dataset without delicate engineering.

Based on our argument in Section 4.2.2, we do not directly compare with van den Oord et al. (2013) because it is sufficient to compare with WMF. For *out-of-matrix* recommendation evaluation, we compare with CTPF and CF + shallow. In all the experiments, the dimensionality of the latent space $K = 50$. We select $\alpha = 2$ and $\epsilon = 10^{-6}$ to compute the confidence c_{ui}. For WMF, CF + shallow, and CF + deep, the model parameters $\boldsymbol{\theta}_{1:U}$, $\boldsymbol{\beta}_{1:I}$ and $\boldsymbol{W}$ (if any) are randomly initialized to the same values.

Evaluation metrics. To evaluate different algorithms, we produce a ranked list of all the songs (excluding those in the training and validation sets) for each user based on the predicted preference $\hat{r}_{ui}$ for $i \in \{1, \ldots, I\}$.

Precision and recall are commonly used evaluation metrics. However, for implicit feedback data, the zeros can mean either the user is not interested in the song or more likely, the user is not aware of the song. This makes the precision less interpretable. However, since the non-zero r_{ui}'s are known to be true positive, we instead report *Recall@M*, which only considers songs within the top M in the ranked list. For each user, the definition of *Recall@M* is

$$Recall@M = \frac{\text{\# of songs that the user listened to in top } M}{\text{total \# of songs the user has listened to}}.$$

Model	$R@40$	$R@80$	$R@120$	$R@160$	$R@200$	NDCG
PMF (Gopalan et al., 2015)	0.1021	0.1533	0.1908	0.2206	0.2456	0.2419
CTPF (Gopalan et al., 2014)	0.1031	0.1511	0.1861	0.2138	0.2370	0.2395
WMF (Hu et al., 2008)	0.1722	0.2367	0.2803	0.3133	0.3397	0.2881
CF + shallow	0.1724	0.2368	0.2803	0.3131	0.3396	0.2883
CF + deep	0.1722	0.2365	0.2800	0.3129	0.3394	0.2882

Table 4.2: *In-matrix* performance on the DEN subset with proposed and competing methods. We can see that with sufficient amount of user feedback, there is almost no difference in terms of recommendation performance among WMF, CF + shallow, and CF + deep.

In addition to *Recall@M*, we also report (untruncated) normalized discounted cumulative gain (NDCG) (Järvelin and Kekäläinen, 2002). Unlike *Recall@M* which only focuses on top M songs in the predicted list, NDCG measures the global quality of recommendation. In the meantime, it also prefers algorithms that place held-out test items higher in the list by applying a discounted weight. Given a ranked list of songs from the recommendation algorithm, for each user NDCG can be computed as follows:

$$\text{DCG} = \sum_{i=1}^{I} \frac{2^{rel_i} - 1}{\log_2(i+1)}; \quad \text{NDCG} = \frac{\text{DCG}}{\text{IDCG}}.$$

Given our binarized data, the reverence rel_i is also binary: 1 if song i is in the held-out user listening history and 0 otherwise. IDCG is the optimal DCG score where all the held-out test songs are ranked top in the list. Therefore, larger NDCG values indicate better performance.

Results on the DEN subset. The model hyperparameters $\lambda_\theta = \lambda_W = 10$ and $\lambda_\beta = 100$ are selected from the validation set based on NDCG. The *in-matrix* and *out-of-matrix* performances are reported in Tables 4.2 and 4.3, respectively. All the metrics are averaged across 612,232 users in the held-out test user-item pairs.

Model	$R@40$	$R@80$	$R@120$	$R@160$	$R@200$	NDCG
CTPF (Gopalan et al., 2014)	0.0256	0.0700	0.1440	0.1869	0.2086	0.1271
CF + shallow	0.0503	0.0894	0.1218	0.1514	0.1778	0.1429
CF + deep	**0.0910**	**0.1461**	**0.1881**	**0.2241**	**0.2550**	**0.1605**

Table 4.3: *Out-of-matrix* performance on the DEN subset with proposed and competing methods. We can see a larger margin between CF + deep and CF + shallow, as compared to their close performance on *in-matrix* predictions in Table 4.2. This suggests the importance of a powerful feature extractor in the absence of usage data.

We can see that with sufficient amount of user feedback, there is almost no difference in performance among WMF, CF + shallow, and CF + deep[6] – there is not a single model which is consistently better. This is understandable, since both CF + shallow and CF + deep are fundamentally collaborative filtering models. With enough user feedback, the model is able to produce meaningful recommendation without resorting to the content features. Moreover, CF + shallow, which has access to more content information, does slightly better than CF + deep.

One observation from Table 4.2 is that adding content features does not necessarily improve the recommendation performance. Unlike CF + deep, CTPF falls behind its content-free counterpart PMF on both *Recall@M* and NDCG. This is possibly due to the insufficient feature extraction capability of the topic model (LDA) on the rich musical data.

The superiority of CF + deep is more obvious on the *out-of-matrix* predictions performance shown in Table 4.3. We can see a larger margin between CF + deep and CF + shallow, as compared to their close performance on *in-matrix* predictions. This suggests the importance of a powerful feature extractor in the absence of usage data. Even a simple linear LDA model in CTPF can be more effective than CF + shallow at predicting songs that the users listened to

[6]There is little point in arguing for the statistical significance of the difference, since given the number of users to average over, the standard error is vanishingly small.

Model	R@40	R@80	R@120	R@160	R@200	NDCG
WMF (Hu et al., 2008)	0.1137	0.1286	0.1378	0.1449	0.1505	0.1415
CF + shallow	0.1138	0.1286	0.1377	0.1449	0.1504	0.1416
CF + deep	**0.1140**	**0.1289**	**0.1378**	**0.1451**	**0.1507**	**0.1417**

Table 4.4: *In-matrix* performance on the SPR subset with proposed and competing methods. With very limited user feedback, both CF + shallow and CF + deep outperform the content-free WMF.

in the held-out test set.

Results on the SPR subset. We repeat the *in-matrix* evaluation on the highly sparse SPR subset. The model hyperparameters $\lambda_\theta = \lambda_W = 10^{-2}$ and $\lambda_\beta = 1$ are selected from the validation set. The performance is reported in Table 4.4. All the metrics are averaged across 564,437 users in the held-out test user-item pairs.

Again, the overall differences among all three methods are relatively minor. However, with very limited user feedback, both CF + shallow and CF + deep outperform the content-free WMF. More importantly, CF + deep consistently improves over CF + shallow, which indicates the importance of an effective feature extractor.

4.5 Summary

In this chapter we present a content-aware collaborative music recommendation system that joins a multi-layer neural network content model with a collaborative filtering model. The system achieves the state-of-the-art performance in music recommendation given content and implicit feedback data.

A possible future direction is to incorporate ranking-based loss function, e.g., the weighted approximate-rank pairwise (WARP) loss in Weston et al. (2011) into the collaborative filtering model. We normally evaluate recommendation algorithms using ranking-based metrics (e.g. *Recall@M* and NDCG), but the model is trained using squared loss function. This discrepancy can be problematic sometimes, as it can mislead the learning algorithm to focus on optimizing unimportant portion of the model. It would be more natural to directly optimize a ranking-based loss function.

Chapter 5

Modeling User Exposure in Recommendation

In this chapter and the next chapter, we will focus on general models for recommender systems. Collaborative filtering analyzes user preferences for items (e.g., books, movies, restaurants, academic papers) by exploiting the similarity patterns across users. In implicit feedback settings, all the items, including the ones that a user did not consume, are taken into consideration. But this assumption does not accord with the common sense understanding that users have a limited scope and awareness of items. For example, a user might not have heard of a certain paper, or might live too far away from a restaurant to experience it. In the language of causal analysis (Imbens and Rubin, 2015), the assignment mechanism (i.e., the items that a user is exposed to) is a latent variable that may change for various user/item combinations. In this paper, we propose a new probabilistic approach that directly incorporates *user exposure* to items into collaborative filtering. The exposure is modeled as a latent variable and the model infers its value from data. In doing so, we recover one of the most successful state-of-the-art approaches WMF as a special case of our model (Hu et al., 2008), and provide a plug-in method for conditioning exposure on various forms of *exposure covariates* (e.g., topics in text, venue locations). We show that our scalable inference algorithm outperforms existing benchmarks in four different domains both with and without exposure covariates.

5.1 Introduction

As motivated in Section 2.2, it is crucial to make good recommendation on the web, as users are overwhelmed with choice. In this chapter, we focus on recommendation with implicit data (see Section 2.2.1 for definition).

Existing approaches account for this by downweighting the unclicked items. In WMF (Hu et al., 2008) the data about unclicked items are given a lower "confidence", expressed through the variance of a Gaussian random variable. In Bayesian personalized ranking (Rendle et al., 2009), the unclicked items are artificially subsampled at a lower rate in order to reduce their influence on the estimation. These methods are effective, but they involve heuristic alterations to the data.

We take a direct approach to solving this problem. We develop a probabilistic model for recommendation called Exposure MF (abbreviated as ExpoMF) that separately captures whether a user has been exposed to an item from whether a user has ultimately decided to click on it. This leads to an algorithm that iterates between estimating the user preferences and estimating the exposure, i.e., why the unclicked items were unclicked. When estimating preferences, it naturally downweights the unclicked items that it expected the user will like, because it imagines that she was not exposed to them.

Concretely, imagine a music listener with a strong preference for alternative rock bands such as Radiohead. Imagine that, in a dataset, there are some Radiohead tracks that this user has not listened to. There are different reasons which may explain unlistened tracks (e.g., the user has a limited listening budget, a particular song is too recent or is unavailable from a particular online service). According to that user's listening history these unlistened tracks would likely make for good recommendations. In this situation our model would assume that the user does not know about these tracks—she has not been exposed to them—and downweight their (negative) contribution when inferring that user's preferences.

Further, by separating the two sides of the problem, our approach enables new innovations in implicit recommendation models. Specifically, we can build models of users' exposure that are guided by additional information such as item content, if exposure to the items typically happens via search, or user/item location, if the users and items are geographically organized.

As an example imagine a recommender system for diners in New York City and diners in Las Vegas. New Yorkers are only exposed to restaurants in New York City. From our model's perspective, unvisited restaurants in New York are therefore more informative in deriving a New Yorker's preferences compared to unvisited restaurants in Las Vegas. Accordingly for New York users our model will upweight unvisited restaurants in New York while downweighting unvisited Las Vegas restaurants.

We studied our method with user listening history from a music intelligence company, clicks from a scientific e-print server, user bookmarks from an online reference manager, and user checkins at venues from a location-based social network. In all cases, ExpoMF matches or surpasses the state-of-the-art method of WMF (Hu et al., 2008). Furthermore, when available, we use extra information to inform our user exposure model. In those cases using the extra information outperforms the simple ExpoMF model. Further, when using document content information our model also outperforms a method specially developed for recommending documents using content and user click information (Wang and Blei, 2011). We illustrate the alternative-rock-listener and the New-York-dinner examples using real data fit with our models in Figure 5.2 and Figure 5.3. Finally, we demonstrate the versatility of ExpoMF by showcasing a couple of examples which incorporate exposure from different sources (e.g., the authors of a paper, or the friends in a social network).

5.2 Exposure matrix factorization

We present exposure matrix factorization (ExpoMF). In Section 5.2.1, we describe the main model. In Section 5.2.2 we discuss several ways of incorporating external information into ExpoMF (i.e., topics from text, locations). We derive inference procedures for our model (and variants) in Section 5.2.3. Finally we discuss how to make predictions given our model in Section 5.2.4.

5.2.1 Model description

For every combination of users $u = 1, \ldots, U$ and items $i = 1, \ldots, I$, consider two sets of variables. The first matrix $\boldsymbol{A} = \{a_{ui}\}$ indicates whether user u has been exposed to item i (exposure matrix). The second matrix $\boldsymbol{Y} = \{y_{ui}\}$ indicates whether or not user u clicked on item i (click matrix).

Whether a user is exposed to an item comes from a Bernoulli. Conditional on being exposed, user's preference comes from a Gaussian matrix factorization model, which factorizes this conditional distribution to K user preferences $\theta_{i,1:K}$ and K item attributes $\beta_{u,1:K}$,

$$
\begin{aligned}
\boldsymbol{\theta}_u &\sim \mathcal{N}(\mathbf{0}, \lambda_\theta^{-1} \mathrm{I}_K) \\
\boldsymbol{\beta}_i &\sim \mathcal{N}(\mathbf{0}, \lambda_\beta^{-1} \mathrm{I}_K) \\
a_{ui} &\sim \mathrm{Bernoulli}(\mu_{ui}) \\
y_{ui} \,|\, a_{ui} = 1 &\sim \mathcal{N}(\boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i, \lambda_y^{-1}) \\
y_{ui} \,|\, a_{ui} = 0 &\sim \delta_0,
\end{aligned}
\tag{5.1}
$$

where δ_0 denotes that $p(y_{ui} = 0 \,|\, a_{ui} = 0) = 1$, and we introduced a set of hyperparameters denoting the inverse variance (λ_θ, λ_β, λ_y). μ_{ui} is the prior probability of exposure, we discuss various ways of setting or learning it in subsequent sections. A graphical representation of the model in Eq. 5.1 is given in Figure 5.1a.

We observe the complete click matrix $\boldsymbol{Y}$. These have a special structure. When $y_{ui} > 0$, we know that $a_{ui} = 1$. When $y_{ui} = 0$, then a_{ui} is latent. The user might have been exposed to item i and decided not to click (i.e., $a_{ui} = 1$, $y_{ui} = 0$); or she may have never seen the item (i.e., $a_{ui} = 0$, $y_{ui} = 0$). We note that since $\boldsymbol{Y}$ is usually sparse in practice, most a_{ui} will be latent.

The model described in Eq. 5.1 leads to the following log joint probability[1] of exposures and clicks for user u and item i,

$$
\begin{aligned}
&\log p(a_{ui}, y_{ui} \,|\, \mu_{ui}, \boldsymbol{\theta}_u, \boldsymbol{\beta}_i, \lambda_y^{-1}) \\
&\quad = \log \mathrm{Bernoulli}(a_{ui} \,|\, \mu_{ui}) + a_{ui} \log \mathcal{N}(y_{ui} \,|\, \boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i, \lambda_y^{-1}) \\
&\qquad + (1 - a_{ui}) \log \mathbb{I}[y_{ui} = 0],
\end{aligned}
\tag{5.2}
$$

where $\mathbb{I}[b]$ is the indicator function that evaluates to 1 when b is true, and 0 otherwise.

[1]N.B., we follow the convention that $0 \log 0 = 0$ to allow the log joint to be defined when $y_{ui} > 0$.

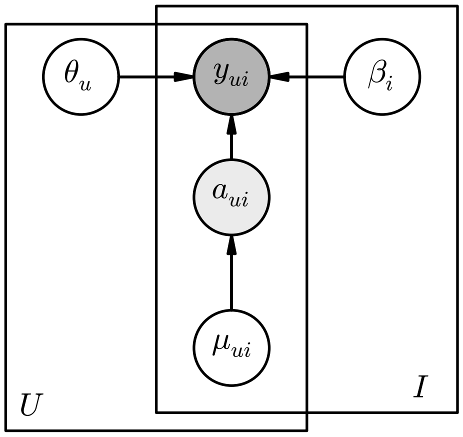
What does the distribution in Eq. 5.2 say about the model's exposure beliefs when no clicks are observed? When the predicted preference is high (i.e., when $\boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i$ is high) then the log-likelihood of no clicks $\log \mathcal{N}(0 \mid \boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i, \lambda_y^{-1})$ is low and likely non-positive. This feature penalizes the model for placing probability mass on $a_{ui} = 1$, forcing us to believe that user u is *not* exposed to item i. (The converse argument also holds for low values of $\boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i$). Interestingly, a low value of a_{ui} downweights the evidence for $\boldsymbol{\theta}_u$ and $\boldsymbol{\beta}_i$ (this is clear by considering extreme values: when $a_{ui} = 0$, the user and item factors do not affect the log joint in Eq. 5.2 at all; when $a_{ui} = 1$, we recover standard matrix factorization). Like WMF (Hu et al., 2008), ExpoMF shares the same feature of selectively downweighting evidence from the click matrix.

In ExpoMF, fixing the entries of the exposure matrix to a single value (e.g., $a_{ui} = 1, \forall u, i$) recovers Gaussian probabilistic matrix factorization (Salakhutdinov and Mnih (2008), see Section 2.2.2.1). WMF is also a special case of our model which can be obtained by fixing ExpoMF's exposure matrix using the confidence c_0 and c_1 (see Section 2.2.2.2).

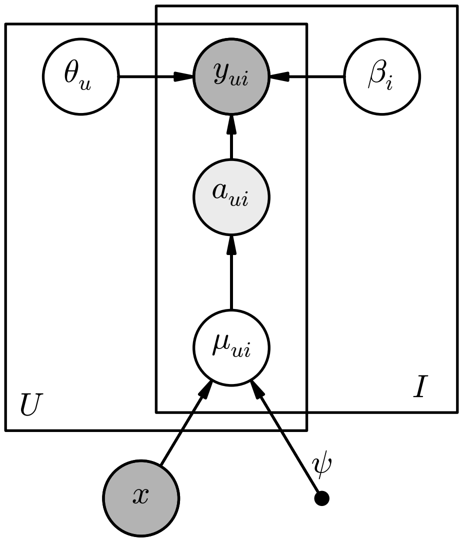
The intuitions we developed for user exposure from the joint probability do not yet involve μ_{ui}, the prior belief on exposure. As we noted earlier, there are a rich set of choices available in the modeling of μ_{ui}. We discuss several of these next.

5.2.2 Hierarchical modeling of exposure

We now discuss methods for choosing and learning μ_{ui}. One could fix μ_{ui} at some global value for all users and items, meaning that the user factors, item factors, and clicks would wholly determine exposure (conditioned on variance hyperparameters). One could also fix μ_{ui} for specific values of u and i. This can be done when there is specific extra information that informs us about exposure (denoted as *exposure covariates*), e.g. the location of a



(a) Exposure MF.



(b) Exposure MF with exposure covariates.

Figure 5.1: Graphical representation of the exposure MF model (both with and without exposure covariates). The lightly shaded node a_{ui} indicates that it is partially observed (i.e., it is observed when $y_{ui} = 1$ and unobserved otherwise).

restaurant, the content of a paper. However, we found that empirical performance is highly sensitive to this choice, motivating the need to place models on the prior for μ_{ui} with flexible parameters.

We introduce observed exposure covariates $\boldsymbol{x}_i$ and exposure model parameters $\boldsymbol{\psi}_u$ and condition $\mu_{ui} \mid \boldsymbol{\psi}_u, \boldsymbol{x}_i$ according to some domain-specific structure. The extended graphical model with exposure covariates is shown in Figure 5.1b. Whatever this exposure model looks like, conditional independence between the priors for exposure and the more standard collaborative filtering parameters (given exposure) ensures that the updates for the model we introduced in Section 5.2.1 will be the same for many popular inference procedures (e.g., expectation-maximization, variational inference, Gibbs sampling), making the extension to exposure covariates a plug-in procedure. We discuss two possible choices of exposure model next.

Per-item μ_i. A direct way to encode exposure is via item popularity: if a song is popular, it is more likely that you have been exposed to it. Therefore, we choose an item-dependent conjugate prior on $\mu_i \sim \text{Beta}(\alpha_1, \alpha_2)$. This model does not use any external information (beyond clicks).

Text topics or locations as exposure covariates. In the domain of recommending text documents, we consider the exposure covariates as the set of words for each document. In the domain of location-based recommendation, the exposure covariates are the locations of the venues being recommended. We treat both in a similar way.

Consider a L-dimensional (L does not necessarily equal the latent space dimension K in the matrix factorization model) representation $\boldsymbol{x}_i$ of the content of document i obtained through natural language processing (e.g., word embeddings (Mikolov et al., 2013), latent Dirichlet allocation (Blei et al., 2003)), or the position of venue i obtained by first clustering all the venues in the data set then finding the expected assignment to L clusters for each venue. In both cases, $\boldsymbol{x}_i$ is all positive and normalizes to 1. Denoting $\sigma(\cdot)$ as the sigmoid function, we set

$$\mu_{ui} = \sigma(\boldsymbol{\psi}_u^\top \boldsymbol{x}_i),$$

where we learn the coefficients $\boldsymbol{\psi}_u$ for each user u. Furthermore, we can include intercepts with various levels and interactions (Gelman and Hill, 2006).

How to interpret the coefficients $\boldsymbol{\psi}_u$? The first interpretation is that of logistic regression, where the independent variables are $\boldsymbol{x}_i$, the dependent binary variables are a_{ui}, and the coefficients to learn are $\boldsymbol{\psi}_u$.

The second interpretation is from a recommender systems perspective: $\boldsymbol{\psi}_u$ represents the topics (or geographical points of interest) that a user is usually exposed to, restricting the choice set to documents and venues that match $\boldsymbol{\psi}_u$. For example, if the l^{th} topic represents neural networks, and x_{il} is high, then the user must be an avid consumer of neural network papers (i.e., $\boldsymbol{\psi}_{ul}$ must be high) for the model to include an academic paper i in the exposure set of u. In the location domain if the l^{th} cluster represents Brooklyn, and x_{il} is high, then the user must live in or visit Brooklyn often for the model to include venues near there in the exposure set of u.

5.2.3 Inference

We use EM algorithm to find the maximum *a posteriori* estimates of the unknown parameters of the model (see Section 2.1.1).[2] The algorithm is summarized in Algorithm 6.

[2]There are various other inference methods we could have used, such as MCMC (Neal, 1993; Robert and Casella, 2013) or variational inference (Jordan et al., 1999; Wainwright and Jordan, 2008). We chose EM for reasons of efficiency and simplicity, and find that it performs well in practice.

The EM inference procedure for the basic model, ExpoMF, can be found by writing out the full log-likelihood of the model, then alternating between finding the expectations of missing data (exposure) in the E(xpectation)-step and finding maximum of the likelihood with respect to the parameters in the M(aximization)-step. This procedure is analytical for our model because it is conditionally conjugate, meaning that the posterior distribution of each random variable is in the same family as its prior in the model.

Furthermore, as we mentioned in Section 5.2.2, conditional independence between the priors for μ_{ui} and the rest of the model (given μ_{ui}) means that the update for the latent exposure variables and user and item factors are not altered for any exposure model we use. We present these general updates first.

E-step. In the E-step, we compute expectation of the exposure latent variable $\mathbb{E}\left[a_{ui}\right]$ for all user and item combinations (u, i) for which there are no observed clicks (recall that the presence of clicks $y_{ui} > 0$ means that $a_{ui} = 1$ deterministically),

$$\mathbb{E}\left[a_{ui} \mid \boldsymbol{\theta}_u, \boldsymbol{\beta}_i, \mu_{ui}, y_{ui} = 0\right] = \frac{\mu_{ui} \cdot \mathcal{N}(0|\boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i, \lambda_y^{-1})}{\mu_{ui} \cdot \mathcal{N}(0|\boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i, \lambda_y^{-1}) + (1 - \mu_{ui})}. \tag{5.3}$$

where $\mathcal{N}(0 \mid \boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i, \lambda_y^{-1})$ stands for the probability density function of $\mathcal{N}(\boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i, \lambda_y^{-1})$ evaluated at 0.

M-step. For notational convenience, we define $p_{ui} = \mathbb{E}\left[a_{ui} \mid \boldsymbol{\theta}_u, \boldsymbol{\beta}_i, \mu_{ui}, y_{ui} = 0\right]$ computed from the E-step. Without loss of generality, we define $p_{ui} = 1$ if $y_{ui} = 1$. The update for the latent collaborative filtering factors is:

$$\boldsymbol{\theta}_u^{\text{new}} \leftarrow (\lambda_y \sum_i p_{ui} \boldsymbol{\beta}_i \boldsymbol{\beta}_i^\top + \lambda_\theta I_K)^{-1} (\sum_i \lambda_y p_{ui} y_{ui} \boldsymbol{\beta}_i) \tag{5.4}$$

$$\boldsymbol{\beta}_i^{\text{new}} \leftarrow (\lambda_y \sum_u p_{ui} \boldsymbol{\theta}_u \boldsymbol{\theta}_u^\top + \lambda_\beta I_K)^{-1} (\sum_u \lambda_y p_{ui} y_{ui} \boldsymbol{\theta}_u), \tag{5.5}$$

Again, this looks very similar to the *alternating least squares* update of WMF, which reveals the close connection between ExpoMF and WMF.

5.2.3.1 Inference for the exposure prior μ_{ui}

We now present inference for the hierarchical variants of ExpoMF. In particular we highlight the updates to μ_{ui} under the various models we presented in Section 5.2.2.

Update for per-item μ_i. Maximizing the log-likelihood with respect to μ_i is equivalent to finding the mode of the complete conditional $\text{Beta}(\alpha_1 + \sum_u p_{ui}, \alpha_2 + U - \sum_u p_{ui})$, which is:

$$\mu_i \leftarrow \frac{\alpha_1 + \sum_u p_{ui} - 1}{\alpha_1 + \alpha_2 + U - 2} \tag{5.6}$$

Update for exposure covariates (topics, location). Setting $\mu_{ui} = \sigma(\boldsymbol{\psi}_u^\top \boldsymbol{x}_i)$, where $\boldsymbol{x}_i$ is given by pre-processing (topic analysis or location clustering), presents us with the challenge of maximizing the log-likelihood with respect to exposure model parameters $\boldsymbol{\psi}_u$. Since there is no analytical solution for the mode, we resort to following the gradients of the log-likelihood with respect to $\boldsymbol{\psi}_u$,

$$\boldsymbol{\psi}_u^{\text{new}} \leftarrow \boldsymbol{\psi}_u + \eta \nabla_{\boldsymbol{\psi}_u} \mathcal{L}, \tag{5.7}$$

for some learning rate η, where

$$\nabla_{\boldsymbol{\psi}_u} \mathcal{L} = \frac{1}{I} \sum_i (p_{ui} - \sigma(\boldsymbol{\psi}_u^\top \boldsymbol{x}_i))\boldsymbol{x}_i.$$

This can be computationally challenging especially for large item-set sizes I. Therefore, we perform (mini-batch) stochastic gradient descent: at each iteration t, we randomly subsample a small batch of items B_t and take a noisy gradient steps:

$$\boldsymbol{\psi}_u^{\text{new}} \leftarrow \boldsymbol{\psi}_u + \eta_t \tilde{g}_t \tag{5.8}$$

for some learning rate η_t, where

$$\tilde{g}_t = \frac{1}{|B_t|} \sum_{i \in B_t} (p_{ui} - \sigma(\boldsymbol{\psi}_u^\top \boldsymbol{x}_i))\boldsymbol{x}_i.$$

For each EM iteration, we found it sufficient to do a single update to the exposure model parameter $\boldsymbol{\psi}_u$ (as opposed to updating until it reaches convergence). This partial M-step (Neal and Hinton, 1998) is much faster in practice.

Algorithm 6: Expo-ALS Inference for ExpoMF

```
Input: Click matrix $\boldsymbol{Y}$, exposure covariates $x_{1:I}$ (topics or locations, optional)
Output: User latent factors $\boldsymbol{\theta}_{1:U}$, item latent factors $\boldsymbol{\beta}_{1:I}$, exposure priors $\mu_{1:I}$ (for per-item
        $\mu_i$), OR exposure model parameters $\boldsymbol{\psi}_{1:U}$ (with exposure model)
Random initialization: $\boldsymbol{\theta}_{1:U}$, $\boldsymbol{\beta}_{1:I}$, $\mu_{1:I}$, OR $\boldsymbol{\psi}_{1:U}$
while performance on validation set increases do
    Compute expected exposure $\boldsymbol{A}$ (Eq. 5.3)
    Update user factors $\boldsymbol{\theta}_{1:U}$ (Eq. 5.4)
    Update item factors $\boldsymbol{\beta}_{1:I}$ (Eq. 5.5)
    if ExpoMF with per-item $\mu_i$ then
        Update priors $\mu_i$ (Eq. 5.6)
    end
    if ExpoMF with exposure model $\mu_{ui} = \sigma(\boldsymbol{\psi}_u^\top \boldsymbol{x}_i)$ then
        Update coefficients $\boldsymbol{\psi}_u$ (Eq. 5.7 or Eq. 5.8)
    end
end
return $\boldsymbol{\theta}_{1:U}$, $\boldsymbol{\beta}_{1:I}$, $\mu_{1:I}$, OR $\boldsymbol{\psi}_{1:U}$
```

5.2.3.2 Complexity and implementation details

A naive implementation of the WMF has the same complexity as ExpoMF in terms of updating the user and item factors. However, the trick that is used to speed up computations in WMF described in Section 2.2.2.2 cannot be applied to ExpoMF due to the non-uniformness of the exposure latent variable a_{ui}. On the other hand, the factor updates are still independent across users and items. These updates can therefore easily be parallelized.

In ExpoMF's implementation, explicitly storing the exposure matrix $\boldsymbol{A}$ is impractical for even medium-sized datasets. As an alternative, we perform the E-step on the fly: only the necessary part of the exposure matrix $\boldsymbol{A}$ is constructed for the updates of the user/item factors and exposure priors μ_{ui}. As shown in Section 5.4, with parallelization and the on-the-fly E-step, ExpoMF can be easily fit to medium-to-large datasets.[3]

5.2.4 Prediction

In matrix factorization collaborative filtering the prediction of y_{ui} is given by the dot product between the inferred user and item factors $\boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i$. This corresponds to the predictive density of ExpoMF $p(y_{ui} \,|\, \boldsymbol{Y})$ using point mass approximations to the posterior given by the EM algorithm[4]. However, ExpoMF can also make predictions by integrating out the uncertainty

[3]The source code to reproduce all the experimental results is available at: `https://github.com/dawenl/expo-mf`.

[4]This quantity is also the treatment effect $\mathbb{E}\left[y_{ui} \,|\, a_{ui} = 1, \boldsymbol{\theta}_u, \boldsymbol{\beta}_i\right] - \mathbb{E}\left[y_{ui} \,|\, a_{ui} = 0, \boldsymbol{\theta}_u, \boldsymbol{\beta}_i\right]$ in the potential outcomes framework (see Section 2.3.1), since $a_{ui} = 0$ deterministically ensures $y_{ui} = 0$.

from the exposure latent variable a_{ui}:

$$\begin{aligned}
\mathbb{E}_y[y_{ui} \mid \boldsymbol{\theta}_u, \boldsymbol{\beta}_i] &= \mathbb{E}_a\left[\mathbb{E}_y[y_{ui} \mid \boldsymbol{\theta}_u, \boldsymbol{\beta}_i, a_{ui}]\right] \\
&= \sum_{a_{ui} \in \{0,1\}} \mathbb{P}(a_{ui}) \mathbb{E}_y[y_{ui} \mid \boldsymbol{\theta}_u, \boldsymbol{\beta}_i, a_{ui}] \\
&= \mu_{ui} \cdot \boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i
\end{aligned}$$

We experimented with both predictions in our study and found that the simple dot product works better for ExpoMF with per-item μ_i while $\mathbb{E}\left[y_{ui} \mid \boldsymbol{\theta}_u, \boldsymbol{\beta}_i\right]$ works better for ExpoMF with exposure covariates. We provide further insights about this difference in Section 5.4.6.1.

5.3 Related work

In this section we highlight connections between ExpoMF and other similar research directions.

Causal inference. Our work borrows ideas from the field of causal inference (Pearl, 2009; Imbens and Rubin, 2015). Causal inference aims at understanding and explaining the effect of one variable on another.

One particular aim of causal inference is to answer counterfactual questions. For example, “would this new recommendation engine increase user click through rate?”. While online studies may answer such a question, they are typically expensive even for large electronic commerce companies. Obtaining answers to such questions using observational data alone (e.g., log data) is therefore of important practical interest (Bottou et al., 2013; Li et al., 2010; Swaminathan and Joachims, 2015).

We establish a connection with the potential outcome framework of Rubin (1974). In this framework one differentiates the assignment mechanism, whether a user is exposed to an item, from the potential outcome, whether a user consumes an item. In potential outcome terminology our work can thus be understood as a form a latent assignment model. In particular, while consumption implies exposure, we do not know which items users have seen but not consumed. Further the questions of interest to us, personalized recommendation, depart from traditional work in causal inference which aims at quantifying the effect of a particular treatment (e.g., the efficacy of a new drug).

Biased CF models. Authors have recognized that typical observational data describing user rating items is biased toward items of interest. Although this observation is somewhat orthogonal to our investigation, models that emerged from this line of work share commonalities with our approach. Specifically, Marlin et al. (2007); Ling et al. (2012) separate the *selection model* (the exposure matrix) from the *data model* (the matrix factorization). However, their interpretation, rooted in the theory of missing data (Little and Rubin, 1986), leads to a much different interpretation of the selection model. They hypothesize that the value of a rating influences whether or not a user will report the rating (this implicitly captures the effect that users mostly consume items they like *a priori*). This approach is also specific to explicit feedback data. In contrast, we model how (the value of) the exposure matrix affects user rating or consumption.

Modeling exposure with random graphs. The user-item interaction can also be encoded as a bipartite graph. Paquet and Koenigstein (2013) model exposure using a hidden *consider graph*. This graph plays a similar role as our exposure variable. One important difference is that during inference, instead of directly inferring the posterior as in ExpoMF (which is computationally more demanding), an approximation is developed whereby a random consider graph is stochastically sampled.

Exposure in other contexts. In zero-inflated Poisson regression, a latent binary random variable, similar to our exposure variable is introduced to "explain away" the structural zeros, such that the underlying Poisson model can better capture the count data (Lambert, 1992). This type of model is common in Economics where it is used to account for overly frequent zero-valued observations.

ExpoMF can also be considered as an instance of a spike-and-slab model (Ishwaran and Rao, 2005) where the "spike" comes from the exposure variables and the matrix factorization component forms the flat "slab" part.

Versatile CF models. As we show in Section 5.2.2, ExpoMF's exposure matrix can be used to model external information describing user and item interactions. This is in contrast to most CF models which are crafted to model a single type of data (e.g., document content when recommendation scientific papers (Wang and Blei, 2011)). An exception is factorization machines (FM) of Rendle (2010). FM models all types of (numeric) user, item or user-item features. FM considers the interaction between all features and learns specific parameters for each interaction.

5.4 Empirical study

In this section we study the recommendation performance of ExpoMF by fitting the model to several datasets. We provide further insights into ExpoMF's performance by exploring the resulting model fits. We highlight that:

- ExpoMF performs comparably better than the state-of-the-art WMF Hu et al. (2008) on four datasets representing user clicks, checkins, bookmarks and listening behavior.

	TPS	Mendeley	Gowalla	ArXiv
# of users	221,830	45,293	57,629	37,893
# of items	22,781	76,237	47,198	44,715
# interactions	14.0M	2.4M	2.3M	2.5M
% interactions	0.29%	0.07%	0.09%	0.15%

Table 5.1: Attributes of datasets after pre-processing. Interactions are non-zero entries (listening counts, clicks, and checkins). % interactions refers to the density of the user-item click matrix ($\boldsymbol{Y}$).

- When augmenting ExpoMF with exposure covariates its performance is further improved. ExpoMF with location covariates and ExpoMF with content covariates both outperform the simpler ExpoMF with per-item μ_i. Furthermore, ExpoMF with content covariates outperforms CTR (Wang and Blei, 2011), a state-of-the-art document recommendation model.

- Through posterior exploration we provide insights into ExpoMF's user-exposure modeling.

5.4.1 Datasets

Throughout this study we use four medium to large-scale user-item consumption datasets from various domains: 1) taste profile subset (TPS) of the million song dataset (Bertin-Mahieux et al., 2011); 2) scientific articles data from arXiv; 3) user bookmarks from Mendeley[5]; and 4) check-in data from the Gowalla dataset (Cho et al., 2011). In more details:

- *Taste Profile Subset (TPS):* contains user-song play counts collected by the music intelligence company Echo Nest.[6] We binarize the play counts and interpret them as implicit preference. We further pre-process the dataset by only keeping the users with

[5]http://mendeley.com
[6]http://the.echonest.com

at least 20 songs in their listening history and songs that are listened to by at least 50 users.

- *ArXiv:* contains user-paper clicks derived from log data collected in 2012 by the arXiv pre-print server. Multiple clicks by the same user on the same paper are considered to be a single click. We pre-process the data to ensure that all users and items have a minimum of 10 clicks.

- *Mendeley:* contains user-paper bookmarks as provided by the Mendeley service, a "reference manager". The behavior data is filtered such that each user has at least 10 papers in her library and the papers that are bookmarked by at least 20 users are kept. In addition this dataset contains the content of the papers which we pre-process using standard techniques to yield a 10K words vocabulary. In Section 5.4.6.1 we make use of paper content to inform ExpoMF's exposure model.

- *Gowalla:* contains user-venue checkins from a location-based social network. We pre-process the data such that all users and venues have a minimum of 20 checkins. Furthermore, this dataset also contains locations for the venues which we will use to guide location-based recommendation (Section 5.4.6.2).

The important attributes of these datasets are summarized in Table 5.1.

5.4.2 Experimental setup

For each dataset we randomly split the observed user-item interactions into training/test/validation sets with 70/20/10 proportions. In all the experiments, the dimension of the latent space for collaborative filtering model K is 100. The model is trained following the inference algorithm described in Algorithm 6. We monitor the convergence of the algorithm using the truncated normalized discounted cumulative gain (NDCG@100, see below for details) on the validation set. Hyper-parameters for ExpoMF-based models and baseline models are also selected according to the same criterion.

To make predictions, for each user u, we rank each item using its predicted preference $y^*_{ui} = \boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i$, $i = 1, \cdots, I$. We then exclude items from the training and validation sets and calculate all the metrics based on the resulting ordered list. Further when using ExpoMF with exposure covariates we found that performance was improved by predicting missing preferences according to $\mathbb{E}[y_{ui}|\boldsymbol{\theta}_u, \boldsymbol{\beta}_i]$ (see Section 5.2.4 for details).

5.4.3 Performance measures

To evaluate the recommendation performance, we report both Recall@k, a standard information retrieval measure, as well as two ranking-specific metrics: mean average precision (MAP@k) and NDCG@k.

We denote $\text{rank}(u, i)$ as the rank of item i in user u's predicted list and $\mathbf{y}_u^{\text{test}}$ as the set of items in the heldout test set for user u.

- Recall@k: For each user u, Recall@k is computed as follows:

$$\text{Recall@}k = \sum_{i \in \mathbf{y}_u^{\text{test}}} \frac{\mathbb{I}\{\text{rank}(u, i) \leq k\}}{\min(k, |\mathbf{y}_u^{\text{test}}|)}$$

  where $\mathbb{I}\{\cdot\}$ is the indicator function. In all our experiments we report both $k = 20$ and $k = 50$. This is slightly different from the metric we used in Section 4.4.2: the expression in the denominator evaluates to the minimum between k and the number of items consumed by user u. In this way, Recall@k is normalized to have a maximum of 1. This corresponds to successfully retrieving all the relevant items in top k of the list. We do not report Precision@k due to the noisy nature of the implicit feedback data: even if an item $i \notin \mathbf{y}_u^{\text{test}}$, it is possible that the user will consume it in the future. This makes Precision@k less interpretable since it is prone to fluctuations.

- MAP@k: Mean average precision calculates the mean of users' average precision. The (truncated) average precision for user u is:

$$\text{Average Precision@}k = \sum_{n=1}^{k} \frac{\text{Precision@}n}{\min(n, |\mathbf{y}_u^{\text{test}}|)}.$$

- NDCG@k: Similar to the untruncated NDCG that we used in Section 4.4.2, it emphasizes the importance of the top ranks by logarithmically discounting ranks. NDCG@k for each user is computed as follows:

$$\text{DCG@}k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{\log_2(i+1)}; \ \text{NDCG@}k = \frac{\text{DCG@}k}{\text{IDCG@}k}$$

  IDCG@k is a normalization factor that ensures NDCG lies between zero and one (perfect ranking). In the implicit feedback case the relevance is binary: $rel_i = 1$ if $i \in \mathbf{y}_u^{\text{test}}$, and 0 otherwise. In our study we always report the averaged NDCG across users.

For the ranking-based measure in all the experiments we set $k = 100$ which is a reasonable number of items to consider for a user. Results are consistent when using other values of k.

5.4.4 Baselines

We compare ExpoMF to WMF, the standard state-of-the-art method for collaborative filtering with implicit data (Hu et al., 2008).

We also experimented with Bayesian personalized ranking (BPR) (Rendle et al., 2009), a ranking model for implicit collaborative filtering which approximately optimizes the area under the ROC curve (AUC). However preliminary results were not competitive with other approaches. BPR is trained using stochastic optimization which can be sensible to hyper-parameter values (especially hyper-parameters related to the optimization procedure). A more exhaustive search over hyper-parameters could yield more competitive results.

We describe specific baselines relevant to modeling exposure covariates in their dedicated subsections.

5.4.5 Studying ExpoMF

Empirical evaluation. Results comparing ExpoMF to WMF on our four datasets are given in Table 5.2. Each metric is averaged across all the users. We notice that ExpoMF performs comparably better than WMF on most datasets (the standard errors are on the order of 10^{-4}) though the difference in performance is small. In addition, higher values of NDCG@100 and MAP@100 (even when Recall@50 is lower) indicate that the top-ranked items by ExpoMF tend to be more relevant to users' interests.

Exploratory analysis. We now explore the posterior distributions of the exposure latent variables of two specific users from the TSP dataset. This exploration provides insights into how ExpoMF infers user exposure.

	TPS		**Mendeley**		**Gowalla**		**ArXiv**	
	WMF	ExpoMF	WMF	ExpoMF	WMF	ExpoMF	WMF	ExpoMF
Recall@20	0.195	**0.201**	0.128	**0.139**	**0.122**	0.118	0.143	**0.147**
Recall@50	**0.293**	0.286	0.210	**0.221**	**0.192**	0.186	**0.237**	0.236
NDCG@100	0.255	**0.263**	0.149	**0.159**	**0.118**	0.116	0.154	**0.157**
MAP@100	0.092	**0.109**	0.048	**0.055**	**0.044**	0.043	0.051	**0.054**

Table 5.2: Comparison between WMF (Hu et al., 2008) and ExpoMF. While the differences in performance are generally small, ExpoMF performs comparably better than WMF across datasets.

The top figure of Figure 5.2 shows the inferred exposure latent variable $\mathbb{E}[a_{ui}]$ corresponding to $y_{ui} = 0$ for user A. $\mathbb{E}[a_{ui}]$ is plotted along with the empirical item popularity (measured by number of times a song was listened to in the training set). We also plot the interpolated per-item exposure prior μ_i learned using Eq. 5.6. There is a strong relationship between song popularity and consideration (this is true across users). User A's training data revealed that she has only listened to songs from either Radiohead or Interpol (both are alternative rock bands). Therefore, for most songs, the model infers that the probability of user A considering them is higher than the inferred prior, i.e., it is more likely that user A did not want to listen to them (they are *true zeros*). However, as pointed out by the rectangular box, there are a few "outliers" which mostly contain songs from Radiohead and Interpol that user A did not listen to (some of them are in fact held out in the test set). Effectively, a lower posterior $\mathbb{E}[a_{ui}]$ than the prior indicates that the model downweights these unlistened songs more. In contrast, WMF downweights all songs uniformly.

A second example is shown in the bottom figure of Figure 5.2. User B mostly listens to *indie* rock bands (e.g. Florence and the Machine, Arctic Monkeys, and The Kills). "Dog Days are Over" by Florence and the Machine is the second most popular song in this dataset, behind "Fireworks" by Katy Perry. These two songs correspond to the two rightmost dots on the figure. Given the user's listening history, the model clearly differentiates these two similarly

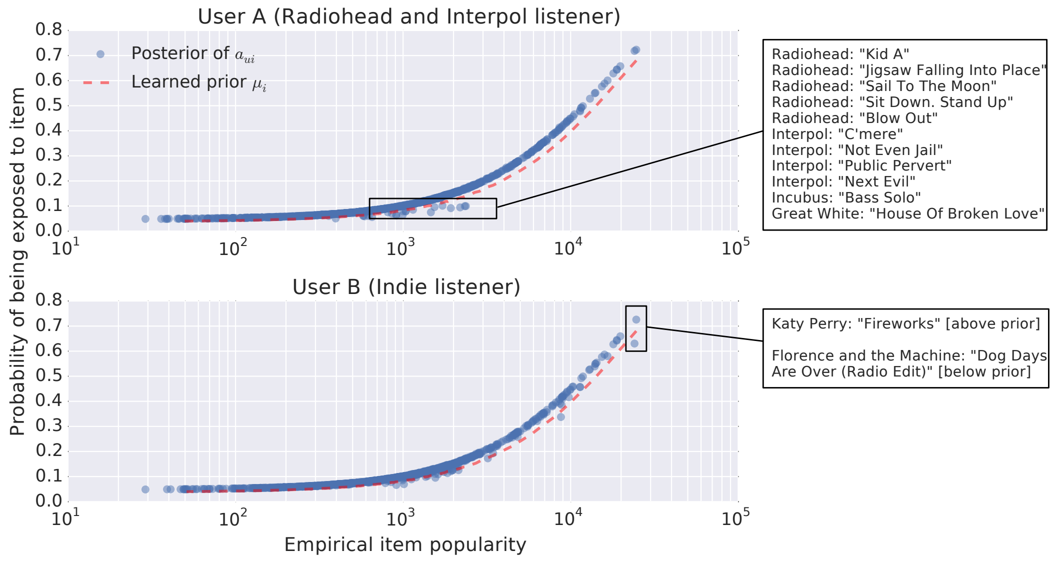


Figure 5.2: We compare the inferred posteriors of the exposure matrix for two users (denoted by blue dots) and compare against the prior probability for exposure (red dashed lined). On the top, user A is a fan of the bands Radiohead and Interpol. Accordingly, the model downweights unlistened songs from these two bands. User B has broader interests and notably enjoys listening to the very popular band Florence and the Machine. Similarly as for user A, unlistened tracks of Florence and the Machine get downweighted in the posterior.

popular songs. The fact that user B did not listen to “Dog Days are Over” (again in the test set) is more likely due to her not having been exposed to it. In contrast the model infers that the user probably did not like “Fireworks” even though it is popular.

5.4.6 Incorporating exposure covariates

Figure 5.2 demonstrates that ExpoMF strongly associates user exposure to item popularity. This is partly due to the fact that the model’s prior is parametrized with a per-item term μ_i. (This also explains why it is not a good idea to make a prediction with $\mathbb{E}[y_{ui} \mid \boldsymbol{\theta}_u, \boldsymbol{\beta}_i] = \mu_{ui} \cdot \boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i$ for ExpoMF without exposure covariates because it will be highly biased towards popular items.) Here we are interested in using exposure covariates to provide additional information about the (likely) exposure of users to items (see Figure 5.1b).

Recall that the role of these exposure covariates is to allow the matrix factorization component to focus on items that the user has been exposed to. In particular this can be done in the model by upweighting (increasing their probability of exposure) items that users were (likely) exposed to and downweighting items that were not. A motivating example with restaurant recommendations and New York City versus Las Vegas diners was discussed in Section 5.1.

In the coming subsections we compare content-aware and location-aware versions of ExpoMF which we refer to as Content ExpoMF and Location ExpoMF respectively. Studying each model in its respective domain we demonstrate that the exposure covariates improve the quality of the recommendations compared to ExpoMF with per-item μ_i.

5.4.6.1 Content covariates

Scientists—whether through a search engine, a personal recommendation or other means—have a higher likelihood of being exposed to papers specific to their own discipline. In this section we study the problem of using the content of papers as a way to guide inference of the exposure component of ExpoMF.

In this use case, we model the user exposure based on the topics of articles. We use LDA (Blei et al., 2003), a model of document collections, to model article content. LDA was briefly reviewed in Section 4.2.1.

	ExpoMF	Content ExpoMF	CTR (Wang and Blei, 2011)
Recall@20	0.139	**0.144**	0.127
Recall@50	0.221	**0.229**	0.210
NDCG@100	0.159	**0.165**	0.150
MAP@100	0.055	**0.056**	0.049

Table 5.3: Comparison between Content ExpoMF and ExpoMF on Mendeley. We also compare with CTR (Wang and Blei, 2011), a model makes use of the same additional information as Content ExpoMF. (CTR is reviewed in Section 4.2.1.)

We use the topic proportion $\boldsymbol{x}_i$ learned from the Mendeley dataset as exposure covariates. Following the notation of Section 5.2.2, our hierarchical ExpoMF is:

$$\mu_{ui} = \sigma(\boldsymbol{\psi}_u^\top \boldsymbol{x}_i + \gamma_u)$$

where we include a per-user bias term γ_u. Under this model, a molecular biology paper and a computer science paper that a computer scientist has not read will likely be treated differently: the model will consider the computer scientist has been exposed to the computer science paper, thus higher $\mathbb{E}[a_{ui}]$, yet not to the molecular biology paper (hence lower $\mathbb{E}[a_{ui}]$). The matrix factorization component of the model will focus on modeling computer science papers since that are more likely to be have been exposed.

Our model, Content ExpoMF, is trained following the algorithm in Algorithm 6. For updating exposure-related model parameters $\boldsymbol{\psi}_u$ and γ_u, we take mini-batch gradient steps with a batch-size of 10 users and a constant step size of 0.5 for 10 epochs.

Study. We evaluate the empirical performance of Content ExpoMF and report results in Table 5.3. We compare to CTR, a state-of-the-art method for recommending scientific papers

(Wang and Blei, 2011) combining both LDA and WMF.[7] We did not compare with the more recent and scalable CTPF (Gopalan et al., 2014) since the resulting performance differences may have been the result of CTPF Poisson likelihood (versus Gaussian likelihood for both ExpoMF and WMF). Both CTR and CTPF were reviewed in Chapter 4.

We note that CTR's performance falls in-between the performance of ExpoMF and WMF (from Table 5.1). CTR is particularly well suited to the *cold-start* case which is not the data regime we focus on in this study (i.e., recall that we have only kept papers that have been bookmarked by at least 20 users).

Figure 5.3 highlights the behavior of Content ExpoMF compared to that of regular ExpoMF. Two users are selected: User A (left column) is interested in statistical machine learning and Bayesian statistics. User B (right column) is interested in computer systems. Neither of them have read "Latent Dirichlet Allocation" (LDA), a seminal paper that falls within user A's interests. On the top row we show the posterior of the exposure latent variables $\mathbb{E}[a_{ui}]$ for two users (user A and user B) inferred from ExpoMF with per-item μ_i. LDA is shown using a white dot. Overall both users' estimated exposures are dominated by the empirical item popularity.

In contrast, on the bottom row we plot the results of Content ExpoMF. Allowing the model to use the documents' content to infer user exposure offers greater flexibility compared to the simple ExpoMF model. This extra flexibility may also explain why there is an advantage in using inferred exposure to predict missing observations (see Section 5.2.4). Namely when exposure covariates are available the model can better capture the underlying user exposures to items. In contrast using the inferred exposure to predict with the simple ExpoMF model performs worse.

[7]Note that to train CTR we first learned a document topic model, fixed it and then learned the user preference model. It was suggested by its authors that this learning procedure provided computational advantages while not hindering performance significantly (Wang and Blei, 2011).

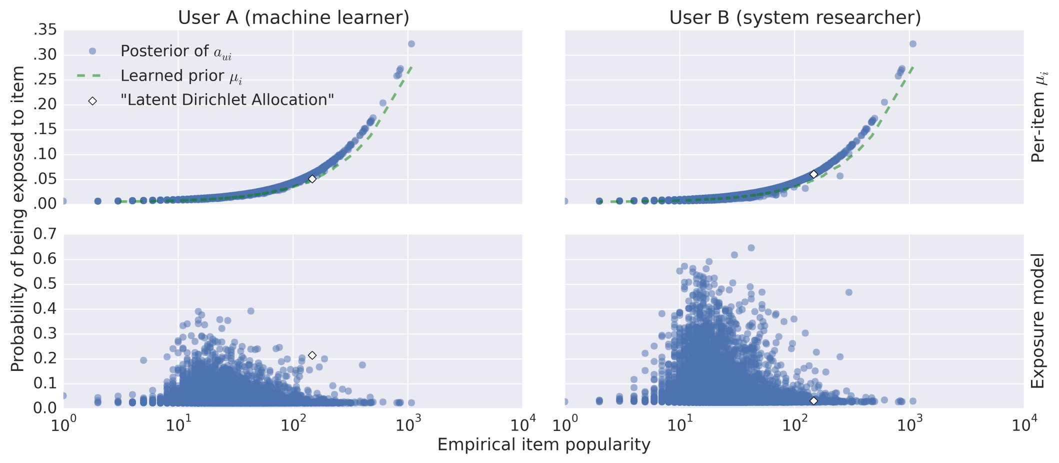


Figure 5.3: We compare the inferred exposure posterior of ExpoMF (top row) and Content ExpoMF (bottom row). On the left are the posteriors of user A who is interested in statistical machine learning while on the right user B is interested in computer system research. Neither users have read the "Latent Dirichlet Allocation" paper. ExpoMF infers that both users have about equal probability of having been exposed to it. As we discussed in Section 5.4.5 (and demonstrated in Figure 5.2) this is mostly based on the popularity of this paper. In contrast, Content ExpoMF infers that user A has more likely been exposed to this paper because of the closeness between that paper's content and user A's interest. Content ExpoMF therefore upweights the paper. Given user B's interests the paper is correctly downweighted by the model.

5.4.6.2 Location covariates

When studying the Gowalla dataset we can use venue location as exposure covariates.

Recall from Section 5.2.2 that location exposure covariates are created by first clustering all venues (using k-means) and then finding the representation of each venue in this clustering space. Similarly as in Content ExpoMF (Section 5.4.6.1), Location ExpoMF departs from ExpoMF:

$$\mu_{ui} = \sigma(\boldsymbol{\psi}_u^\top \boldsymbol{x}_i + \gamma_u)$$

	WMF	ExpoMF	Location ExpoMF
Recall@20	0.122	0.118	**0.129**
Recall@50	0.192	0.186	**0.199**
NDCG@100	0.118	0.116	**0.125**
MAP@100	0.044	0.043	**0.048**

Table 5.4: Comparison between Location ExpoMF and ExpoMF with per-item μ_i on Gowalla. Using location exposure covariates outperforms the simpler ExpoMF and WMF according to all metrics.

where x_{ik} is the venue i's expected assignment to cluster k and γ_u is a per-user bias term.[8]

Study. We train Location ExpoMF following the same procedure as Content ExpoMF. We report the empirical comparison between WMF, ExpoMF and Location ExpoMF in Table 5.4. We note that Location ExpoMF outperforms both WMF and the simpler version of ExpoMF.

For comparison purposes we also developed a simple baseline FilterWMF which makes use of the location covariates. FilterWMF filters out venues recommended by WMF that are inaccessible (too far) to the user. Since user location is not directly available in the dataset, we estimate it using the geometric median of all the venues the user has checked into. The median is preferable to the mean because it is better at handling outliers and is more likely to choose a typical visit location. However, the results of this simple FilterWMF baseline are worse than the results of the regular WMF. We attribute this performance to the fact that having a single focus of location is too strong an assumption to capture visit behavior of users well. In addition, since we randomly split the data, it is possible that a user's checkins at city A and city B are split between the training and test set. We leave the exploration of better location-aware baselines to future work.

[8]We named Content ExpoMF and Location ExpoMF differently to make it clear to the reader that they condition on content and location features respectively. Both models are in fact mathematically equivalent.

5.5 Extensions

Besides the Content and Location ExpoMF from Section 5.4, in this section we further demonstrate the versatility of the proposed ExpoMF model by incorporating exposure from various sources: 1) authors of a paper; 2) friends in a social network. We also demonstrate the "plug-in" nature of ExpoMF, showing how inference can be performed without much modification from the algorithm we developed in Section 5.2.3.

5.5.1 Author exposure

In Section 5.4.6.1, we assume that whether or not a scientist is exposed to a particular paper depends on the content of the paper. Here we make a different assumption: the exposure is dependent on the authors who wrote the paper – this is a reasonable assumption, as understandably more famous authors can get more attention because of their frames.

A straightforward model of author exposure would be to consider the author-paper matrix $\boldsymbol{C} \in \{0,1\}^{A \times I}$, where A is the total number of unique authors: $c_{ai} = 1$ if author a is included in the author list of paper i and $c_{ai} = 0$ otherwise. We treat each column $\boldsymbol{c}_i \in \{0,1\}^A$ as an exposure covariates vector. Then we can fit this model in the same fashion as Content and Location ExpoMF, where we learn $\boldsymbol{\psi}_{1:U}$ for user's exposure to authors.

However, there are two problems with this simple model: 1) Even though the inference is straightforward following Algorithm 6, it is almost impractical as it requires to store and constantly update a $U \times A$ dense exposure coefficient matrix $\boldsymbol{\psi}_{1:U}$ where A can be as large as tens of thousands. 2) More importantly, this model does not take into account the influence from co-authorship. For example, imagine Dave and John have written many papers together. If a user only read Dave's papers, then the above-mentioned exposure model will not consider a paper by John more likely to be exposed comparing to the papers from some other random authors.

To overcome the second problem, we can learn a latent representation for each author by factorizing the author-paper matrix $\boldsymbol{C}$. Back to our example above, with this setup John and Dave will end up being close in this learned latent space because of the similarity among co-authors. Consequently, the user who only read Dave's papers will also have higher likelihood of being exposed to John's papers.[9]

Let's assume we have learned author latent representation $\boldsymbol{x}_a$ ($a = 1, \cdots, A$) from author-paper matrix $\boldsymbol{C}$. The author-aware version of ExpoMF (we refer to as Author ExpoMF) is specified with the following hierarchical exposure prior (following the notation of Section 5.2.2):

$$\mu_{ui} = \sigma\Big(\Big(\frac{1}{|A_i|}\sum_{a\in A_i} \boldsymbol{x}_a\Big)^\top \boldsymbol{\psi}_u + \gamma_u\Big),$$

where A_i is the set of authors of paper i and γ_u is again a user-dependent bias term. Here we take the average of the author latent representations to account for various amount of authors per paper: for ArXiv dataset that we analyzed, the number of authors per paper ranges from one to more than 150.

Model inference for Author Exposure is similar to that of Content and Location ExpoMF, as we can treat $\frac{1}{|A_i|}\sum_{a\in A_i} \boldsymbol{x}_a$ as the exposure covariates. We use the same ArXiv dataset in Section 5.4 and pre-process the author-paper matrix $\boldsymbol{C}$ by only keeping the authors with at least 2 papers, which gives us $A = 38,627$ unique authors. We learn the author latent representation $\boldsymbol{x}_a$ with the Gaussian matrix factorization (Section 2.2.2.1) only on the observed 1's in the author-paper matrix $\boldsymbol{C}$.

[9]This could potentially help with the same author with different name spellings, which is very common in the ArXiv dataset.

	ExpoMF	Author ExpoMF
Recall@20	0.143	**0.151**
Recall@50	**0.236**	0.231
NDCG@100	0.157	**0.160**
MAP@100	0.054	**0.058**

Table 5.5: Comparison between Author ExpoMF and ExpoMF on ArXiv. We can see that incorporating author exposure improves the recommendation performance over the simple ExpoMF.

Quantitative results. We report the recommendation performance of Author ExpoMF in Table 5.5. We also compare to ExpoMF (the results are copied from Table 5.2). For Author ExpoMF, we predict the missing preferences according to $\mathbb{E}\left[y_{ui} \mid \boldsymbol{\theta}_u, \boldsymbol{\beta}_i\right]$. As we can see, incorporating author exposure improves the metrics over the simple ExpoMF, even though the difference in performance is generally small. We note that by filtering out inactive authors, some papers will be considered having "zero" authors, i.e., $\boldsymbol{x}_a = \mathbf{0}$ for $\forall a \in A_i$. This means that its exposure will be solely dependent on the user-dependent bias term γ_u, which could be restrictive.

Exploratory analysis. To help develop intuition on how Author ExpoMF helps with recommendation by making use of the author exposure, we look into a particular user who has read a couple of papers about network analysis and social networks by Mark Newman. From the dataset, we can see that Aaron Clauset has published quite a few papers with Mark Newman on the same topics, of which this user hasn't read any. The author latent representation $\boldsymbol{x}_a$ for Mark Newman and Aaron Clauset are very close with a cosine similarity of 0.85. Therefore, understandably Author ExpoMF recommends some of Aaron Clauset's papers on top of the recommended list. Furthermore, among these papers also include the ones that Aaron Clauset wrote with authors other than Mark Newman. For comparison, ExpoMF does not recommend any single paper from Aaron Clauset (except the one that he co-authored with Mark Newman) among the top 50.

5.5.2 Social network exposure

Users in a social network can be naturally influenced by their friends, e.g., we will be more likely to try out a song if a friend we trust recommends it. There have been many collaborative filtering models which exploits trust among friends in a social network (Ma et al., 2008, 2009, 2011; Guo et al., 2015; Chaney et al., 2015). In this section, we develop a social-network-aware version of ExpoMF (Social ExpoMF) where users' exposure to items is influenced by their social friends.

Social ExpoMF takes a similar approach to SocialPF (Chaney et al., 2015) with the following generative process:

$$
\begin{aligned}
\boldsymbol{\theta}_u &\sim \mathcal{N}(\mathbf{0}, \lambda_\theta^{-1} \mathrm{I}_K) \\
\boldsymbol{\beta}_i &\sim \mathcal{N}(\mathbf{0}, \lambda_\beta^{-1} \mathrm{I}_K) \\
\tilde{a}_{ui} &\sim \mathrm{Pois}(\lambda_{ui}) \\
a_{ui} &= \mathbb{I}\{\tilde{a}_{ui} > 0\} \\
y_{ui} \,|\, a_{ui} = 1 &\sim \mathcal{N}(\boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i, \lambda_y^{-1}) \\
y_{ui} \,|\, a_{ui} = 0 &\sim \delta_0,
\end{aligned}
$$

where $\lambda_{ui} = \sum_{v \in N(u)} \tau_{uv} y_{vi} + \gamma_u + \alpha_i$. $N(u)$ is the set of users who are social friends with user u. τ_{uv} can be interpreted as user v's influence on user u, which is learned as part of the inference procedure. We also include user- and item-dependent bias terms γ_u and α_i. We constrain exposure coefficients τ_{uv}, γ_u, and α_i to be nonnegative, so that the rate λ_{ui} to the Poisson-distributed random variable $\tilde{a}_{ui}$ is also nonnegative.

How to interpret Social ExpoMF model? If τ_{uv} is high for user u, that means user u is more likely to be exposed to the items that user v (a social friend of user u) clicked on. Equivalently, a small value of τ_{uv} indicates that the items user v clicked on will not likely expose to user u. If user u and v do not have any social overlap, then $\tau_{uv} = 0$. It is reasonable to assume the entire coefficient matrix $\boldsymbol{\tau} = \{\tau_{uv}\} \in \mathbb{R}_{+}^{U \times U}$ is very sparse in practice. Consequently, we add bias terms $\boldsymbol{\gamma} = \{\gamma_u\} \in \mathbb{R}_{+}^{U}$ and $\boldsymbol{\alpha} = \{\alpha_i\} \in \mathbb{R}_{+}^{I}$ to capture the user- and item-level exposure patterns beyond the influence from social friends.

Here we use a Poisson exposure model instead of the Bernoulli-logistic model that has been applied to other ExpoMF variations. This is mostly for computational concerns, as the social network data can be computationally demanding to work with. Gopalan et al. (2015) and Chaney et al. (2015) have demonstrated that Poisson model has the computational advantages especially on sparse data.

5.5.2.1 A variational EM algorithm

The inference for Social ExpoMF is more complicated due to the censored Poisson exposure model. We develop a variational EM algorithm for efficient model inference, where in the E-step we compute (approximate) the posterior of a_{ui} and in the M-step we update the exposure coefficients τ_{uv}, γ_u, and α_i via maximum likelihood estimation.

If we follow the general procedure of EM algorithm in Section 2.1.1 and write down the objective function, there will be a problematic term $\log \mathbb{P}(a_{ui} = 1) = \log(1 - \mathrm{e}^{-\lambda_{ui}})$ which prevents us from deriving closed-form coordinate updates. To work around it, we present a model augmentation strategy.[10]

We first observe that $1 - \mathrm{e}^{-\lambda}$ is the cumulative distribution function (CDF) of the $\text{Exponential}(\lambda)$ distribution evaluated at 1. Therefore, we can define an exponential distribution truncated at

[10]The augmentation strategy presented here comes from an unpublished note by Matthew D. Hoffman.

1:

$$\text{Exponential}_1(t;\lambda) = \frac{\lambda}{1-\mathrm{e}^{-\lambda}}\mathrm{e}^{-\lambda t}, t \in [0,1]$$

Furthermore, we define $\text{Exponential}_1(t;0) = \text{Unif}[0,1]$. We augment the model with $t_{ui} \mid a_{ui} \sim \text{Exponential}_1(t_{ui}; a_{ui}\lambda_{ui})$. Now the joint log-likelihood of a_{ui} and t_{ui} is:

$$\log p(a_{ui}, t_{ui}) = a_{ui}(\log \lambda_{ui} - t_{ui}\lambda_{ui}) - \lambda_{ui}(1 - a_{ui}),$$

where the problematic term $\log(1 - \mathrm{e}^{-\lambda_{ui}})$ gets canceled.

E-step. The complete data log-likelihood is

$$\mathcal{L} = \sum_{u,i} a_{ui} \log \mathcal{N}(y_{ui} \mid \boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i, \lambda_y^{-1}) + a_{ui}(\log \lambda_{ui} - t_{ui}\lambda_{ui}) - \lambda_{ui}(1 - a_{ui}) + \text{priors}.$$

In the E-step, we use a variational distribution $q(a_{ui}, t_{ui})$ to approximate the true posterior. Since we can compute the posterior expectation of a_{ui} exactly:

$$\mathbb{E}\left[a_{ui} \mid \boldsymbol{\theta}_u, \boldsymbol{\beta}_i, \lambda_{ui}, y_{ui} = 0\right] = \frac{(1 - \mathrm{e}^{-\lambda_{ui}}) \cdot \mathcal{N}(0 | \boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i, \lambda_y^{-1})}{(1 - \mathrm{e}^{-\lambda_{ui}}) \cdot \mathcal{N}(0 | \boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i, \lambda_y^{-1}) + \mathrm{e}^{-\lambda_{ui}}}, \tag{5.9}$$

we will only specify variational distribution for t_{ui} as $q(t_{ui}) = \text{Exponential}_1(t_{ui}; \rho_{ui})$. Define $p_{ui} = \mathbb{E}\left[a_{ui} \mid \boldsymbol{\theta}_u, \boldsymbol{\beta}_i, \lambda_{ui}, y_{ui} = 0\right]$ and without loss of generality, we set $p_{ui} = 1$ if $y_{ui} > 0$. We tune the variational parameters $\boldsymbol{\rho} = \{\rho_{ui}\} \in \mathbb{R}_+^{U \times I}$ to optimize the variational objective (ELBO) for t_{ui}:

$$\mathcal{L}_{ui}^{\text{VI}} = -p_{ui}\mathbb{E}_q[t_{ui}]\lambda_{ui} + \rho_{ui}\mathbb{E}_q[t_{ui}] - \log \rho_{ui} + \log(1 - \mathrm{e}^{-\rho_{ui}}) + \text{const.}$$

where the necessary expectation is $\mathbb{E}_q[t_{ui}] = \frac{1}{\rho_{ui}} - \frac{1}{e^{\rho_{ui}}-1}$. Take the derivative of ELBO with respect to ρ_{ui} and set it to 0, we get the following updates for the variational parameters:

$$\rho_{ui} \leftarrow p_{ui}\lambda_{ui} \tag{5.10}$$

M-step. The objective in the M-step is the expected complete data log-likelihood under the variational distribution:

$$\mathbb{E}_q[\mathcal{L}] = \sum_{u,i} p_{ui} \log \mathcal{N}(y_{ui} \mid \boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i, \lambda_y^{-1}) + p_{ui}(\log \lambda_{ui} - \mathbb{E}_q[t_{ui}]\lambda_{ui}) - \lambda_{ui}(1 - p_{ui}) + \text{const.}.$$

The updates for collaborative filtering latent factors $\boldsymbol{\theta}_u$ and $\boldsymbol{\beta}_i$ are identical to all the previous models (Eq. 5.4 and Eq. 5.5) because of the conditional independence between the exposure prior and the matrix factorization part of the model (given exposure).

The updates for exposure coefficients τ_{uv}, γ_u, and α_i are more difficult because of the problematic term $\log \lambda_{ui} = \log(\sum_{v \in N(u)} \tau_{uv} y_{vi} + \gamma_u + \alpha_i)$. We follow the standard strategy to lower bound it via Jensen's inequality:

$$\begin{aligned} \log(\sum_{v \in N(u)} \tau_{uv} y_{vi} + \gamma_u + \alpha_i) \geq & \sum_{v \in N(u)} \phi^{\tau}_{uiv}(\log \tau_{uv} y_{vi} - \log \phi^{\tau}_{uiv}) \\ & + \phi^{\gamma}_{ui}(\log \gamma_u - \log \phi^{\gamma}_{ui}) + \phi^{\alpha}_{ui}(\log \alpha_i - \log \phi^{\alpha}_{ui}) \end{aligned}$$

where $\phi^{\tau}_{uiv} \geq 0$, $\phi^{\gamma}_{ui} \geq 0$, $\phi^{\alpha}_{ui} \geq 0$, and $\sum_{v \in N(u)} \phi^{\tau}_{uiv} + \phi^{\gamma}_{ui} + \phi^{\alpha}_{ui} = 1$. To tighten the lower bound, we update $\phi^{\tau}_{uiv} = \frac{\tau_{uv} y_{vi}}{\lambda_{ui}}$, $\phi^{\gamma}_{ui} = \frac{\gamma_u}{\lambda_{ui}}$, and $\phi^{\alpha}_{ui} = \frac{\alpha_i}{\lambda_{ui}}$. After lower-bounding the objective, we take the gradients with respect to τ_{uv}, γ_u, and α_i and set them to 0, which leads to the following multiplicative updates:

$$
\begin{aligned}
\tau_{uv} &\leftarrow \frac{\sum_i \phi^{\tau}_{uiv} p_{ui}}{\sum_i y_{vi}(1 - p_{ui}(1 - \mathbb{E}_q[t_{ui}]))} \\
\gamma_u &\leftarrow \frac{\sum_i \phi^{\gamma}_{ui} p_{ui}}{\sum_i (1 - p_{ui}(1 - \mathbb{E}_q[t_{ui}]))} \\
\alpha_i &\leftarrow \frac{\sum_u \phi^{\alpha}_{ui} p_{ui}}{\sum_u (1 - p_{ui}(1 - \mathbb{E}_q[t_{ui}]))}
\end{aligned}
\tag{5.11}
$$

These updates closely resemble those of NMF with generalized KL-divergence loss function (Lee and Seung, 2001). The full algorithm for Social ExpoMF is summarized in Algorithm 7. In actual implementation, we perform on-the-fly E-step, similar to what was described in Section 5.2.3.2, for both a_{ui} and t_{ui}, as well as parallelization when updating both latent factors ($\boldsymbol{\theta}_{1:U}$, $\boldsymbol{\beta}_{1:I}$) and exposure coefficients ($\boldsymbol{\tau}$, $\boldsymbol{\gamma}$, $\boldsymbol{\alpha}$). This enables Social ExpoMF to tractably analyze large-scale user-item interaction data with social networks.

Algorithm 7: SOCIAL-EXPO-ALS Inference for Social ExpoMF

```
Input: Click matrix $\boldsymbol{Y}$, social network $N(u)$ for $u = 1, \ldots, U$
Output: User latent factors $\boldsymbol{\theta}_{1:U}$, item latent factors $\boldsymbol{\beta}_{1:I}$, exposure coefficients $\boldsymbol{\tau}$, $\boldsymbol{\gamma}$, and $\boldsymbol{\alpha}$
Random initialization: $\boldsymbol{\theta}_{1:U}$, $\boldsymbol{\beta}_{1:I}$, $\boldsymbol{\tau}$, $\boldsymbol{\gamma}$, $\boldsymbol{\alpha}$
while performance on validation set increases do
    Compute expected exposure $\boldsymbol{P} = \{p_{ui}\}$ (Eq. 5.9)
    Update variational parameter $\boldsymbol{\rho} = \{\rho_{ui}\}$ (Eq. 5.10)
    Update user factors $\boldsymbol{\theta}_{1:U}$ (Eq. 5.4)
    Update item factors $\boldsymbol{\beta}_{1:I}$ (Eq. 5.5)
    Update exposure coefficients $\boldsymbol{\tau}$, $\boldsymbol{\gamma}$, and $\boldsymbol{\alpha}$ (Eq. 5.11)
end
return $\boldsymbol{\theta}_{1:U}$, $\boldsymbol{\beta}_{1:I}$, $\boldsymbol{\tau}$, $\boldsymbol{\gamma}$, $\boldsymbol{\alpha}$
```

Data. We evaluate Social ExpoMF on Douban dataset (Ma et al., 2011). Douban (douban.com) is a Chinese social service where users record ratings for music, movies, and books. It contains 129K users and 57K items with 16M user-item interactions in the form of ratings on a 1-5 scale. The social network is undirected (i.e., if user v is user u's social friend, then the

	ExpoMF	Social ExpoMF
Recall@20	0.205	**0.208**
Recall@50	0.306	**0.320**
NDCG@100	0.225	**0.230**
MAP@100	**0.087**	0.086

Table 5.6: Comparison between Social ExpoMF and ExpoMF on Douban dataset. We can see that incorporating social network exposure improves the recommendation performance over the simple ExpoMF.

reverse is true) with 1.3M network connections. Following the experimental setup in Chaney et al. (2015), we remove network connections where the users have no items in common. Furthermore, we binarize the explicit ratings by only keeping ratings greater than or equal to 4 and treat them as implicit preferences.

Quantitative results. We evaluate the empirical performance of Social ExpoMF and report results in Table 5.6. We compare to the ExpoMF with per-item μ_i exposure prior. We can see that Social ExpoMF outperforms ExpoMF except on MAP@100. This indicates that incorporating exposure based on social network provides additional benefits on top of simple popularity-based exposure model. Unfortunately due to privacy reasons, most of the public user-item interaction datasets with social network do not have meta-data information about users and/or items, which prevents us from exploring the resulting model fit.

5.6 Summary

In this chapter, we presented a novel collaborative filtering mechanism that takes into account user exposure to items. In doing so, we theoretically justify existing approaches that downweight unclicked items for recommendation, and provide an extendable framework for specifying more elaborate models of exposure based on logistic regression. In empirical studies we found that the additional flexibility of our model helps it outperform existing approaches to matrix factorization on four datasets from various domains. We also demonstrate the versatility of our model by incorporating other sources of exposure, e.g., the authors of a paper, or the friends in a social network.

There are several promising avenues for future work. Consider a reader who keeps himself up to date with the “what’s new” pages of a website, or a tourist visiting a new city looking for a restaurant recommendation. The exposure processes are more dynamic in these scenarios and may be different during training and test time. We therefore seek new ways to capture exposure that include ever more realistic assumptions about how users interact with items.

Finally, we would like to evaluate our proposed model in a more realistic setting, e.g., in an online environment with user interactions. It would be instructive to evaluate the performance of ExpoMF in environments where it may be possible to observe items which users have been exposed to.

Chapter 6

Causal Inference for Recommendation

In this chapter, we move from implicit data to *explicit* data and develop a causal inference approach to recommender systems. Observational recommendation data contains two sources of information: which items each user decided to look at and which of those items each user liked. We assume these two types of information come from different models—the *exposure* data comes from a model by which users discover items to consider; the *click* data comes from a model by which users decide which items they like. Traditionally, recommender systems use the click data alone (or ratings data) to infer the user preferences. But this inference is biased by the exposure data, i.e., that users do not consider each item independently at random. We use causal inference to correct for this bias. On real-world data, we demonstrate that causal inference for recommender systems leads to improved generalization to new data.

6.1 Introduction

The goal of recommender systems is to infer users' preferences for items and then to predict items that users will like. We develop a causal inference approach to this problem.

Here is the idea. Observational recommendation data contains two sources of information (see the definition of observational data in Section 2.3.2): which items each user decided to look at and which of those items each user liked. For example, one of the data sets we analyze contains which movies each user watched and which of them each liked; another contains which scientific abstracts each user saw and which PDFs each decided to download.

We assume these two types of information come from different models—the *exposure* data comes from a model by which users discover items to consider; the *click* data comes from a model by which users decide which items they like. Traditionally, recommender systems use the click data alone (or ratings data) to infer the user preferences. But this inference is biased by the exposure data, i.e., that users do not consider each item independently at random.

We use causal inference to correct for this bias. First, we estimate the exposure model from the exposure data, a model of which items each user is likely to consider. Then we fit the preferences with weighted click data, where each click (or skip) is weighted by the inverse probability of exposure (from the exposure model). This is a propensity weighting approach to causal inference (Imbens and Rubin, 2015), i.e., we warp the observational click data as though it came from an "experiment" where users are randomly shown items. We study several variants of this strategy.

Why might this work? Consider the film enthusiast (from our data) who mostly watches popular drama but has also enjoyed a couple of documentaries ("Crumb" and "The Cruise").

A classical recommendation system will infer film preferences that center around drama. Our causal method detects a preference for drama too, but further up-weights the preference for documentaries. The reason is that the history of the user indicates that she is unlikely to have been exposed to many documentaries; the method values its signal from the two she did like. Consequently, when we recommend from among the unwatched films, our method promotes documentaries ("Fast, Cheap & Out of Control" and "Paris Is Burning") that the user (in held-out data) also liked. Across users, on real-world data, we demonstrate that causal inference for recommender systems leads to improved generalization to new data.

6.1.1 Related work

Marlin and Zemel (2009) first formalized statistical models for correcting rating-selection bias. They posit that a user's decision to rate an item depends on the user's opinion of the item. They propose a mechanism to correct for this self-selection bias, based first on generating a rating and then conditionally on whether the rating is observed. Others have proposed different rating models using this same mechanism (Ling et al., 2012; Hernández-Lobato et al., 2014). In contrast, our model (similar to ExpoMF (Liang et al., 2016b) from Chapter 5) first generates each user's exposure to an item and then her rating. Unlike ExpoMF, we work with explicit click data in this chapter. Thus we can use causal inference to de-bias the resulting inference of user preferences.

Solving recommendations using causality has been explored in the multi-arm bandit literature (e.g., Li et al. (2010); Vanchinathan et al. (2014); Zhao et al. (2013); Li et al. (2015)). They focus on unbiased evaluation of a recommendation policy, though using biased data (e.g., data collected in web log). This work typically uses importance sampling, weighting the probability of each observed click under the logging policy and under the (new) recommendation policy. We use the same tools for data re-weighting—propensity score weighting is equivalent to importance sampling—but we reason about preferences rather than recommendation policies. Further we work in a batch learning setting (as opposed to online learning).

The recent work of Schnabel et al. (2016) is closest to what we present in this chapter. The authors propose a causal inference approach to learning unbiased estimators from biased rating data. One important difference with our work is that their propensity weights depend on user preferences (either directly through ratings or indirectly through user and item covariates)—a process known as self-selection—rather than reflecting exposure, as in our work. Their formalization of the problem also differs: they appeal to empirical risk minimization while we take a Bayesian perspective.

6.2 A causal model for recommendation

In this section we develop our method. We describe explicit recommendation data, a joint model of exposure and clicks, how we do prediction, and how we do causal inference.

6.2.1 Data

Our data are *explicit data*: we know which items each user saw and which of those items each clicked (liked) or skipped (disliked). For example, in Section 6.4 we analyze a large collection of click data from arXiv.org. We know which arXiv abstracts a user has viewed and, among those, which PDFs she has downloaded. Our goal is to infer each user's latent preferences for items and then to use those preferences in a recommendation system.

We begin with notation for the data. There are two types of observations. The *exposure data* is a_{ui}, whether user u had the opportunity to click on item i. The *click data* is y_{ui}, an indicator of whether user u clicked on item i (liked) or decided to skip the item (disliked). This is the explicit counterpart of the setup in Chapter 5.

These data capture the users' clicks. There are some items which a user was exposed to ($a_{ui} = 1$) but did not click on ($y_{ui} = 0$); there are other items that a user was exposed to ($a_{ui} = 1$) and did click on ($y_{ui} = 1$); finally, there are items that a user was not exposed to ($a_{ui} = 0$) and, by definition, did not click on ($y_{ui} = 0$). A user cannot click on an item she is not exposed to.

6.2.2 Joint models of exposure and clicks

We build a joint model of the data described in Section 6.2.1: an exposure model of what the user sees and a click model of what the user clicks on, conditional on her seeing it. The key idea behind our approach is this. Given observational data, i.e., data collected by users exploring information and clicking on items, classical inference of the click model—of the user's preferences for clicking on items that she is exposed to—is incorrect because of the biases induced by the exposure model. We take a causal inference approach to this problem: we infer the user's preferences from an imagined experiment where each item is exposed with equal probability.

We first describe the *observation joint*, from which we observe our data set.

$$\begin{aligned} a_{ui} &\sim f(\cdot \,|\, \pi_{ui}) \\ y_{ui} \,|\, a_{ui} = 0 &\sim \delta_0(\cdot) \\ y_{ui} \,|\, a_{ui} = 1 &\sim g(\cdot \,|\, \mu_{ui}). \end{aligned}$$

Here the exposure and click models are generic. Each is governed by the exposure parameter π_{ui} and click parameter μ_{ui}, respectively.

For example, one exposure model we study is a Bernoulli with an item-specific parameter. We call this the popularity exposure because it allows some items to be more likely to be exposed (across users) than others,

$$a_{ui} \sim \text{Bernoulli}(\rho_i). \qquad \textit{(Popularity exposure)} \tag{6.1}$$

Alternatively, the exposure model can capture a user's preferences for seeking out items. We will also study Poisson factorization,

$$a_{ui} \sim \text{Poisson}(\boldsymbol{\pi}_u^\top \boldsymbol{\gamma}_i), \qquad \textit{(Poisson factorization exposure)} \tag{6.2}$$

which finds non-negative embeddings for users and items (Gopalan et al., 2015).[1]

For the click model, we use classical probabilistic matrix factorization (Salakhutdinov and Mnih, 2008) in Section 2.2.2.1. Conditional on being exposed, the click comes from a normal distribution, $y_{ui} \,|\, a_{ui} = 1 \sim \mathcal{N}(\boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i, \lambda_y^{-1})$. Here $\boldsymbol{\theta}_u$ is a latent K-vector of user preferences and $\boldsymbol{\beta}_i$ is a latent K-vector of item attributes. In all models, the conditional distribution of a click y_{ui} given that a user is not exposed to the item ($a_{ui} = 0$) is a point mass at zero.

6.2.3 Forming predictions

Our goal is to use this model to form future predictions about the users. We are given observed data $\mathcal{D} = \{(a_{ui}, y_{ui})\}$ of what each user was exposed to and what each user clicked on. We

[1] Though Poisson models capture count data, they are effective for binary data with many items (Gopalan et al., 2015).

want to predict what we should expose them to in the future, i.e., what they would like to see.

We will study two ways of predicting. One is to form conditional predictions as the probability that a user clicks on an item given that she is exposed to it,

$$\mathbb{E}\left[y_{ui} \mid a_{ui} = 1, \mathcal{D}\right]. \qquad \textit{(Conditional prediction)} \qquad (6.3)$$

Alternatively, we use marginal predictions, where we marginalize out the exposure variable

$$\mathbb{E}\left[y_{ui} \mid \mathcal{D}\right] = p(a_{ui} \mid \mu_{ui}, \mathcal{D})\mathbb{E}\left[y_{ui} \mid a_{ui} = 1, \mathcal{D}\right]. \qquad \textit{(Marginal prediction)} \qquad (6.4)$$

The marginal prediction uses that $y_{ui} = 0$ when $a_{ui} = 0$. It is apt when the exposure model also contains information about the user, i.e., information about what the user is likely to seek out.

Note that these methods require approximating the posterior predictive distribution of y_{ui} and a_{ui} given the data. We now turn to this inference problem.

6.2.4 Causal inference for recommendation

One way to solve the inference problem is with classical Bayesian inference, where we condition on the observed data and then use posterior prediction to recommend items. But there is an issue with using classical Bayesian inference to form recommendations: the data we observe $\mathcal{D}^{\text{obs}}$ is not the data from which we would like to infer the user's preferences and item attributes, i.e., the click model. The reason is that the exposure model—the distribution that governs what each user sees—biases our inference about the click model. Items that users are likely to be exposed exert too much of an influence; items that users are rarely exposed to have too little influence.

Ideally, we would infer preferences from an experiment, a model that randomly exposed each user to items and then recorded which items each one click on. We call this the *intervention joint*,

$$\begin{aligned} a_{ui} &\sim \text{Bernoulli}(\pi) \\ y_{ui} \,|\, a_{ui} = 0 &\sim \delta_0(\cdot) \\ y_{ui} \,|\, a_{ui} = 1 &\sim g(\cdot \,|\, \mu_{ui}). \end{aligned}$$

In this model, we have intervened on the mechanism from which users are exposed to items. (This is the "mutilated model" (Pearl, 2009).) Data from this model leads to better estimates of the click model (i.e., their preferences) and better generalization to the items that they will want to click on.

This is a causal approach to the problem. The observation joint is the model of how we collected the data; the intervention joint is a model of a randomized experiment that would (in theory) help us make the inferences that we need. The challenge is to use data from the observation joint to perform inference in the intervention joint.

How do we solve this problem? Assume for now that the exposure model is known and is the popularity model, i.e., $a_{ui} \sim \text{Bernoulli}(\rho_i)$. We will use *inverse propensity weighting* (Imbens and Rubin, 2015), which takes samples from the observation joint and weights them to look like samples from the intervention joint; this is essentially an importance sampling technique. Specifically, we weight each observation (a_{ui}, y_{ui}) by $1/\rho_i$ to estimate θ_u. (Because of the click model, this estimate only relies on those data where $a_{ui} = 1$.) When inferring a user's preferences, this down-weights the influence of popular items and up-weights the influence of unpopular items.

More formally, our goal is to obtain a data set $\mathcal{D} = \{(a_{ui}, y_{ui})\}$ from the intervention joint and then estimate $p(\boldsymbol{\theta}_u \mid \mathcal{D})$. We define the "do dataset" to be the observed data embellished with weights, $\mathcal{D}^{\text{do}} = \{(a_{ui}, y_{ui}, w_{ui})\}$. The posterior is

$$p(\boldsymbol{\theta}_u \mid \mathcal{D}^{\text{do}}) \propto p(\boldsymbol{\theta}_u) \prod_i p(y_{ui} \mid a_{ui})^{w_{ui}} \tag{6.5}$$

Intuitively, this assumes that we see each data point "w_{ui} times", and that the clicks are conditionally independent given the preferences.

How is this different from standard causality? One way is that, in typical causal settings, we have a single causal question (Imbens and Rubin, 2015). Here we have many causal questions (one per user-item pair). What is crucial is that the causal outcomes are related, each governed by the same set of parameters.

6.3 Inference

We first estimate the exposure model from the observed data. This can be the popularity model or Poisson factorization. Then, we use the fitted exposure model to weight the data (by the inverse probability) and fit the click model. Finally we use the posterior distribution of the exposure model and (causal) posterior distribution of the click model to form predictions. This procedure generalizes better than Bayesian inference, especially under intervention, i.e., when we change the distribution of which items a user is exposed to.

6.3.1 Fitting exposure model

Popularity model. For popularity exposure model $a_{ui} \sim \text{Bernoulli}(\rho_i)$, we obtain the maximum likelihood estimate $\hat{\rho}_i$ by counting the portion of the users who have been exposed to item i. The propensity score in this case is fixed across users:

$$\pi_{ui} = \hat{\rho}_i, \forall u \in \{1, \ldots, U\}. \tag{6.6}$$

Poisson factorization model. For Poisson factorization exposure model $a_{ui} \sim \text{Pois}(\boldsymbol{\pi}_u^\top \boldsymbol{\gamma}_i)$ with gamma prior on the latent embeddings $\boldsymbol{\pi}_u$ and $\boldsymbol{\gamma}_i$, we perform standard variational inference (Gopalan et al., 2015) on the exposure data a_{ui}. After obtaining the optimal variational distribution q on $\boldsymbol{\pi}_u$ and $\boldsymbol{\gamma}_i$ at convergence, we compute the propensity score,

$$\pi_{ui} = 1 - \mathbb{P}\{a_{ui} = 0\} = 1 - \exp\{-\mathbb{E}_q\left[\boldsymbol{\pi}_u^\top \boldsymbol{\gamma}_i\right]\}. \tag{6.7}$$

6.3.2 Fitting click model

The click model is a matrix factorization $y_{ui} \,|\, a_{ui} = 1 \sim \mathcal{N}(\boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i, \lambda_y^{-1})$. Following Section 2.2.2.1, we place a diagonal normal prior on both user preference $\boldsymbol{\theta}_u \sim \mathcal{N}(\mathbf{0}, \lambda_\theta^{-1} \mathrm{I}_K)$ and item attributes $\boldsymbol{\beta}_i \sim \mathcal{N}(\mathbf{0}, \lambda_\beta^{-1} \mathrm{I}_K)$. To fit the model, we compute the maximum *a posteriori* estimates of the parameters $\boldsymbol{\theta}_u$ and $\boldsymbol{\beta}_i$. Concretely, the objective for the inverse propensity weighted Gaussian matrix factorization model (Eq. 6.5) is:

$$\mathcal{L} = -\sum_{(u,i)\in\mathcal{O}} \frac{1}{2\pi_{ui}} (y_{ui} - \boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i)^2 - \frac{\lambda_\theta}{2} \sum_u \|\boldsymbol{\theta}_u\|_2^2 - \frac{\lambda_\beta}{2} \sum_i \|\boldsymbol{\beta}_i\|_2^2,$$

where the propensity score π_{ui} can be obtained by either Eq. 6.6 or Eq. 6.7. The observed set $\mathcal{O}$ contains all the entries with $a_{ui} = 1$. Similar to WMF, we can obtain the following coordinate updates by taking the gradients with respect to $\boldsymbol{\theta}_u$ and $\boldsymbol{\beta}_i$ and setting them to 0:

$$\boldsymbol{\theta}_u^{\text{new}} \leftarrow \Big(\sum_{i:(u,i)\in\mathcal{O}} \frac{1}{\pi_{ui}} \boldsymbol{\beta}_i \boldsymbol{\beta}_i^\top + \lambda_\theta \mathrm{I}_K \Big)^{-1} \Big(\sum_{i:(u,i)\in\mathcal{O}} \frac{1}{\pi_{ui}} y_{ui} \boldsymbol{\beta}_i \Big) \tag{6.8}$$

$$\boldsymbol{\beta}_i^{\text{new}} \leftarrow \Big(\sum_{u:(u,i)\in\mathcal{O}} \frac{1}{\pi_{ui}} \boldsymbol{\theta}_u \boldsymbol{\theta}_u^\top + \lambda_\beta \mathrm{I}_K \Big)^{-1} \Big(\sum_{u:(u,i)\in\mathcal{O}} \frac{1}{\pi_{ui}} y_{ui} \boldsymbol{\theta}_u \Big) \tag{6.9}$$

The full algorithm for the inverse propensity weighted Gaussian matrix factorization is summarized in Algorithm 8. Note that this algorithm only includes options for fitting the model causally (Eq. 6.5). In Section 6.4, we empirically explore different combinations of the exposure model, prediction method, and fitting procedures.

Algorithm 8: IPW-ALS Alternating least squares for the inverse propensity weighted Gaussian matrix factorization

```
Input: A set of observed entires in the click matrix $\{y_{ui} : (u,i) \in \mathcal{O}\}$, regularization
       parameters $\lambda_\theta$ and $\lambda_\beta$
Output: A set of user latent factors $\boldsymbol{\theta}_{1:U}$ and item latent factors $\boldsymbol{\beta}_{1:I}$
Fit exposure model to compute the propensity score (Eq. 6.6 or Eq. 6.7)
Randomly initialize $\boldsymbol{\theta}_{1:U}$, $\boldsymbol{\beta}_{1:I}$
while not converged do
    for $u \leftarrow 1$ to $U$ do
        Update user factor $\boldsymbol{\theta}_u$ (Eq. 6.8)
    end
    for $i \leftarrow 1$ to $I$ do
        Update item factor $\boldsymbol{\beta}_i$ (Eq. 6.9)
    end
end
return $\boldsymbol{\theta}_{1:U}$, $\boldsymbol{\beta}_{1:I}$
```

6.4 Empirical study

We studied causal recommender systems on several data sets. We compared models trained observationally with models trained causally; we compared predictions made marginally and those made conditional on exposure; we studied and evaluated different exposure models, both those based on popularity and based on personalized preferences; and we studied typical test sets and test sets that focus on rare items.

We highlight the following results:

- Poisson factorization (Eq. 6.2) is a better exposure model than the one based on item popularity (Eq. 6.1). We evaluate the exposure model both as a standalone model to predict held-out exposure and as a component in the whole recommender system.
- When the test set focuses on rare items, fitting causally (Eq. 6.5) gives better generalization than classical inference. Causal inference is important for generalizing to situations that we do not see in training.
- Accounting for exposure is important when making prediction—recommendation with marginal prediction (Eq. 6.4) significantly boosts the ranking-based recommendation performance.

We give details below. We describe the data, methods, metrics, and results.

6.4.1 Datasets

We study three types of data (and four data sets):

- *MovieLens (ML-1M and ML-10M).* User-movie ratings collected from a movie recommendation service.[2] The ratings are on a 1–5 scale.

- *Yahoo-R3.* Music ratings collected from Yahoo! Music services (Marlin and Zemel, 2009). The ratings are 1–5.

- *ArXiv.* User-paper clicks from the 2012 log-data of the arXiv pre-print server. The data are binarized: multiple clicks by the same user on the same paper are considered to be a single click. This data contains which papers a user downloaded and which she only read the abstract.

For ML-1M, ML-10M, and Yahoo-R3, we denote exposure $a_{ui} = 1$ as user u having rated item i. These three datasets are typically used for rating prediction. Because our end goal is recommendation, we binarize the ratings and encode preferences as being either positive or negative ($y_{ui} = 1$ if rating is greater than or equal to 3 and $y_{ui} = 0$ otherwise). This type of binarization gives better recommendation performance than directly using predicted ratings (Hu et al., 2008).[3]

In ArXiv we denote exposure $a_{ui} = 1$ as user u having viewed the abstract of paper i. Among papers that a user is exposed to, we set $y_{ui} = 1$ if she downloaded the paper and $y_{ui} = 0$ otherwise.

Table 6.1 summarizes the important attributes of our four datasets.

Data pre-processing. For each dataset, we create two training/validation/test splits: regular (REG) and skewed (SKEW). We create a regular split by randomly selecting the exposed items for each user into training/validation/test sets, following 70/10/20 proportions. In the

[2] http://grouplens.org/datasets/movielens/

[3] We note that the Yahoo! data set also contains a random test set, where a subset of the users are given 10 randomly selected songs to rate. But most of the ratings for this random test set are below 3. Rather, we created a skewed test set.

	ML-1M	ML-10M	Yahoo-R3	ArXiv
# of users	6,040	69,878	15,400	26,541
# of items	3,706	10,677	1,000	80,082
# of exposures	1.0M	10.0M	0.3M	1.9M
% of exposures	4.47%	1.34%	2.02%	0.09%

Table 6.1: Attributes of the data. # of exposures is the number of entries with $a_{ui} = 1$ (rated an item, viewed an abstract). % of exposure refers to the density of the user-item exposure matrix.

regular split, the test set has the same exposure distribution as the training and validation sets. This is how researchers typically evaluate recommendation models (with observational data).

The skewed split rebalances the splits to better approximate an intervention. We create it by first sampling a test set with roughly 20% of the total exposures, such that each item has uniform probability. Training and validation sets are then created from the remaining data (as in a regular split) with 70/10 proportions. For a skewed split, the test set will have a completely different exposure distribution from the training and validation sets. We use this split to demonstrate that causal inference for recommendation leads to improved generalization performance.

Figure 6.1 shows the scatter plots of the training exposure distribution (reflected by the empirical item popularity) against the test exposure distribution on regular and skewed splits of the ML-1M dataset. The empirical item popularity is computed by counting the number of users who have been exposed to each item. The skewed split has a roughly uniform exposure distribution across items, while in the regular split, both training and test sets follow similar exposure patterns.

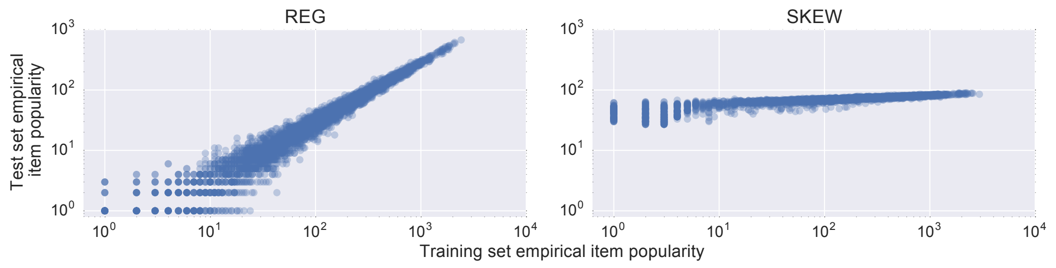


Figure 6.1: Scatter plots of the training exposure distribution (reflected by the empirical item popularity) against the test exposure distribution on REG (left) and SKEW (right) splits for ML-1M dataset. SKEW split has a roughly uniform exposure distribution across items, while in REG split both training and test sets follow similar exposure patterns.

6.4.2 Methods

There are several choices in the proposed method. We explored combinations of the exposure model, prediction method, and fitting procedure. The different choices are summarized below:

- *Exposure model.* Popularity (Pop, Eq. 6.1) or Poisson factorization (PF, Eq. 6.2).
- *Prediction.* Conditional prediction (Cond, Eq. 6.3) or marginal prediction (Mar, Eq. 6.4).
- *Model fitting.* Train the click model causally (CAU, Eq. 6.5), with inverse propensity weighting, or observationally (OBS), with classical inference.

Among these methods are two baselines. The models that are trained observationally (OBS) with conditional prediction (Cond) correspond to classical matrix factorization (Salakhutdinov and Mnih, 2008). The models that are trained causally (CAU) with conditional prediction (Cond) correspond to inverse propensity weighted matrix factorization proposed in Schnabel et al. (2016).[4] We note that this approach significantly outperformed the previous state-of-the-art model proposed in Hernández-Lobato et al. (2014) for the task of rating prediction (though the main focus of this chapter is on recommendation).

Hyperparameters. We perform grid search using $\lambda_\theta \in \{10^{-4}, \dots, 10^4\}$ and $\lambda_\beta \in \{10^{-4}, \dots, 10^4\}$ to select hyperparameters based on the normalized discounted cumulative gain (NDCG) (Järvelin and Kekäläinen, 2002) of the validation set.

We set the dimension of the latent space K to 30 and use the same random initialization of $\boldsymbol{\theta}_u$ and $\boldsymbol{\beta}_i$ in all settings. For the coordinate updates algorithm in Section 6.3, we declare convergence when the mean squared error on the validation set increases.

6.4.3 Metrics

We separately evaluate the exposure model, how well we predict which items a user will see, and the click model, which items a user will like. Note that causal inference of the click model uses the exposure model to compute the propensity score. Further, marginal prediction of clicks also uses the exposure model.

We evaluate the exposure model using model fitness to the data (predictive log-likelihood). We evaluate the click model with recommendation metrics, both a likelihood-based metric (a tail probability) and a ranking-based metric (mean average rank (Charlin et al., 2015)).[5] We describe the recommendation metrics in turn.

[4]Even though Schnabel et al. (2016) derive the model from empirical risk minimization framework, the model objective closely resembles the joint log-likelihood of the causally trained model (CAU) with conditional prediction (Cond).

[5]NDCG (Järvelin and Kekäläinen, 2002) is another commonly used ranking-based metric. It emphasizes the importance of the top ranks by logarithmically discounting ranks. MAR, on the other hand, makes no such discounting.

Predictive log tail probability (PLP). For y_{ui} in the heldout test set, we compute the predictive log-probability based on its value and whether we predict conditionally or marginally (see Eq. 6.4).

Conditional prediction uses $\mathbb{E}[y_{ui} \mid a_{ui} = 1, \mathcal{D}]$. If $y_{ui} = 1$, we compute right-tail conditional predictive log-probability for positively preferred items,

$$\log \mathbb{P}(y_{ui}^{\text{pred}} > 1 \mid a_{ui} = 1, \mathcal{D}).$$

Otherwise we compute left-tail conditional predictive log-probability

$$\log \mathbb{P}(y_{ui}^{\text{pred}} \leq 0 \mid a_{ui} = 1, \mathcal{D}).$$

Both correspond to Gaussian tail probability for matrix factorization.

Marginal prediction uses $\mathbb{E}[y_{ui} \mid \mathcal{D}]$. If $y_{ui} = 1$, we compute right-tail marginal predictive log-probability,

$$\log \mathbb{P}(y_{ui}^{\text{pred}} > 1 \mid \mathcal{D}) = \log \pi_{ui} + \log \mathbb{P}(y^{\text{pred}} > 1 | a_{ui} = 1, \mathcal{D})$$

(Recall that π_{ui} is the probability that user u is exposed to item i.) Otherwise we compute left-tail marginal predictive log-probability

$$\log \mathbb{P}(y_{ui}^{\text{pred}} \leq 0 \mid \mathcal{D}) = \log \left(\pi_{ui} \mathbb{P}(y_{ui}^{\text{pred}} \leq 0 \mid a_{ui} = 1, \mathcal{D}) + (1 - \pi_{ui})\right).$$

The intuition behind PLP is that we would like to have 0's and 1's in the heldout set well-separated. This is different from the commonly used metrics for rating prediction, e.g., mean squared error or mean absolute error, both of which penalize the model unless it predicts with a perfect 0 and 1. We report average PLP over all the heldout y_{ui} in the test set.

Mean Average Rank. We compute MAR as follows. For user u we calculate the ranking of all the items $i \in \{1, 2, \dots, I\}$ by sorting the predictions and excluding the items from the training and validation sets. Define $\text{rank}(u, i)$ as the predicted rank of item i for user u: $\text{rank}(u, i) = 0$ if item i is ranked first for user u and $\text{rank}(u, i) = I - 1$ if ranked last. For items within a set I_u,

$$\text{MAR}_u = \frac{1}{|I_u|} \sum_{i \in I_u} \text{rank}(u, i).$$

In our studies, I_u is the item set in the heldout set with $y_{ui} = 1$, i.e., the items that user u rated positively or the papers that user u downloaded after looking at the abstract. Since the value of MAR depends on the size of the item set I, we report the normalized MAR percentile instead as MAR_u / I. This also corresponds to the expected percentile ranking proposed in Hu et al. (2008) with binary feedback data. The interpretation of MAR is on average at what percentile a heldout item will be ranked (smaller is better). The reported MAR averages over all users.

6.4.4 Results

We report our studies on all data. We evaluate both the exposure model alone and the recommender model, which uses the exposure model to improve its recommendations.

Evaluating the exposure model. We first compare two different exposure models used in this chapter: Poisson factorization (PF) and the popularity model (Pop). We use the training set created in Section 6.4.1 to train the model (for PF, we use the validation set to monitor

	ML-1M		**ML-10M**		**Yahoo-R3**		**ArXiv**	
	REG	SKEW	REG	SKEW	REG	SKEW	REG	SKEW
Pop	-1.39	-2.07	-1.64	-2.76	-1.81	-2.74	-3.83	-3.95
PF	-0.97	-1.51	-1.08	-2.06	-1.58	-2.35	-2.71	-2.80

Table 6.2: Heldout predictive log-likelihood for Poisson factorization (PF) exposure model and popularity exposure model (Pop). PF outperforms Pop across datasets. The predictive log-likelihood is generally lower on SKEW than REG.

convergence). We randomly sample the same number of entries with $a_{ui} = 0$ as those with $a_{ui} = 1$ and report the average heldout predictive log-likelihood in Table 6.2.

PF always outperforms Pop. Further, the predictive log-likelihood is always lower on skewed test sets than on regular test sets. This is expected because skewed test sets follow a different exposure distribution from the training and validation sets. This makes it harder for the exposure model to correctly predict its values.

Evaluating the recommender model. We summarize the log probability (PLP) and mean average rank (MAR) (described in Section 6.4.3) in Table 6.3a and Table 6.3b, respectively. The table reports eight different model configurations based on which exposure model is used, how the model is fit, and how predictions are formed.[6]

From Table 6.3, we make the following observations.

1. Poisson factorization (PF) gives better performance in terms of both PLP and MAR than the popularity exposure model (Pop). (Pop configurations are on the top half of each table; PF configurations are on the bottom half.)

[6]There are seven distinct configurations, as the ones that are trained observationally (OBS) with conditional prediction (Cond) will not depend on the exposure model. We keep all eight configurations in Table 6.3 for easy comparison.

			ML-1M		**ML-10M**		**Yahoo-R3**		**ArXiv**	
			REG	SKEW	REG	SKEW	REG	SKEW	REG	SKEW
Pop	Cond	OBS	-1.50	-2.07	-1.62	-2.59	-1.58	-1.75	-1.61	-1.65
		CAU	-1.61	-1.95	-1.67	-1.89	-1.51	-1.56	-1.74	-1.76
	Mar	OBS	-3.17	-4.29	-3.56	-5.63	-2.98	-3.53	-3.93	-4.21
		CAU	-3.21	-4.25	-3.60	-5.24	-2.84	-3.53	-3.94	-4.15
PF	Cond	OBS	-1.50	-2.07	-1.62	-2.59	-1.58	-1.75	-1.61	-1.65
		CAU	-1.48	-1.84	-1.51	-1.96	-1.49	-1.55	-1.60	-1.62
	Mar	OBS	-2.62	-3.87	-2.69	-4.61	-2.71	-3.40	-3.05	-3.32
		CAU	-2.60	-3.57	-2.69	-4.42	-2.59	-3.14	-3.04	-3.33

(a) Predictive log tail probability (bigger is better)

			ML-1M		**ML-10M**		**Yahoo-R3**		**ArXiv**	
			REG	SKEW	REG	SKEW	REG	SKEW	REG	SKEW
Pop	Cond	OBS	13.0%	25.6%	5.4%	18.3%	15.1%	36.1%	18.4%	23.6%
		CAU	17.3%	27.1%	8.0%	18.4%	21.5%	31.7%	32.1%	35.7%
	Mar	OBS	11.8%	26.6%	5.1%	18.9%	15.6%	36.9%	22.5%	33.8%
		CAU	12.3%	26.9%	5.3%	18.7%	15.8%	36.6%	30.0%	42.9%
PF	Cond	OBS	13.0%	25.6%	5.4%	18.3%	15.1%	36.1%	18.4%	23.6%
		CAU	16.9%	26.6%	7.8%	17.1%	16.6%	29.2%	30.7%	33.9%
	Mar	OBS	6.9%	19.1%	2.9%	14.2%	10.2%	28.9%	7.5%	13.0%
		CAU	6.9%	18.4%	3.1%	14.2%	9.9%	25.9%	11.2%	13.1%

(b) Mean average rank (smaller is better)

Table 6.3: Predictive log tail probability (PLP) and mean average rank (MAR) for the recommendation model on different datasets. The results are organized by the exposure model (Pop or PF), how to fit the model (OBS or CAU), and how to make prediction (Cond or Mar). The OBS-Cond models correspond to the classical matrix factorization (Salakhutdinov and Mnih, 2008). The CAU-Cond models correspond to Schnabel et al. (2016). See main text for detailed analysis.

2. If the test set exposure comes from the same distribution as the training set (regular split), training the model observationally or causally does not make a difference in terms of PLP. As for MAR, we can make the same observation (with marginal prediction), but ArXiv is an exception.

   On the other hand, if the test set exposure distribution is different (the skewed split), training the model causally gives more robust generalization performance. Even on ArXiv, we can see that moving from regular to skewed severely degrades the performance of observationally-trained models, as opposed to causally-trained models, where the degradation is comparably weaker.

   Furthermore, we computed the mean average rank of heldout rare items, those only rated by few users. We found the percentile of held-out rare items are much smaller with causally trained models. This indicates that fitting the model causally corrects for the popularity bias induced by the exposure process. (These numbers not reported.)

3. Marginal prediction gives the best overall performance in terms of both metrics. When we predict whether a user will like an item, we should consider her preference as well as how likely she is to seek out the item.

4. In Schnabel et al. (2016), the authors use a naive Bayes propensity score estimator. Our results show that a more flexible propensity model (e.g., Poisson factorization) tends to give better recommendation performance.

5. We notice that the results with causally-trained models (CAU) on ArXiv are less stable than those from the other three datasets. ArXiv is more than one order of magnitude sparser than the other datasets and less popularity-biased—even considering abstract views, most of the papers are only viewed and downloaded by a small number of users. Therefore, the estimated propensity score could contain extreme values, a common problem for methods

involving propensity score (Morgan and Winship, 2014). As part of the future work, we will investigate different propensity score smoothing techniques.

6.5 Summary

In this chapter, we develop a causal inference approach to recommendation with explicit data. We separately model two sources of information: the *exposure* data (which items each user decided to look at) and *click* data (which of those items each user liked). Exposure data introduces bias when we estimate parameters of a recommendation model from the click data, as rare items do not get as much exposure as popular ones. We use inverse propensity weighting to correct for this bias. Through extensive empirical study, we demonstrate that this causal approach to recommender systems leads to improved generalization to new data.

As future work, we can develop similar methodology for implicit data. The main difficulty in implicit data is that we do not know which items a user has been exposed to. The ExpoMF model and its variations we developed in Chapter 5 could help with that.

Chapter 7

Conclusion

In this dissertation, we apply the tools of probabilistic latent variable models and try to understand complex real-world data about music semantics and user behavior.

Scalable music tagging with Poisson factorization. We develop scalable solution to automatic music tagging – inferring the semantic tags (e.g., "jazz", "piano", "happy", etc.) from the audio features. We treat music tagging as a matrix completion problem and apply the Poisson matrix factorization model jointly on the vector-quantized audio features and a "bag-of-tags" representation. This approach exploits the shared latent structure between semantic tags and acoustic codewords. The experimental results on the Million Song Dataset for both annotation and retrieval tasks demonstrate the steady improvement in performance as more data is used. Furthermore, we can look at the highly probable tags for each learned latent factor to understand what portion of the acoustic codeword space is being captured, and whether it is musically coherent.

Content-aware collaborative music recommendation. We address the fundamental cold-start problem of collaborative filtering: it cannot recommend new songs that no one has listened to. We train a multi-layered neural network on semantic tagging information as a content model and use it as a prior in a collaborative filtering model. The model is able to balance between the user feedback and the content features, allowing the data to "speak for itself". The proposed system is evaluated on the Million Song Dataset and shows comparably better result than the collaborative filtering approaches, in addition to the favorable performance in the cold-start case.

Modeling user exposure in recommendation. We develop a probabilistic matrix factorization model ExpoMF to capture the latent user exposure (whether or not a user is exposed to an item) in implicit feedback data. In doing so, we recover one of the most successful state-of-the-art approaches WMF as a special case of our model (Hu et al., 2008), and provide a plug-in method for conditioning exposure on various forms of exposure covariates (e.g., topics in text, venue locations). We show that our scalable inference algorithm outperforms existing benchmarks in four different domains both with and without exposure covariates. We further demonstrate the versatility of ExpoMF by incorporating other sources of exposure: 1) the authors of a paper; and 2) the friends in a social network.

Causal inference for recommendation. In the language of causal analysis (Imbens and Rubin, 2015), user exposure has close connection to the assignment mechanism. We leverage this connection for explicit data and develop a causal inference approach to recommender systems. Observational recommendation data contains two sources of information: exposure data (which items each user decided to look at) and click data (which of those items each user liked). Exposure data introduces bias when we estimate parameters of a recommendation model from the click data, as rare items do not get as much exposure as popular ones. We use inverse propensity weighting to correct for this bias. Through extensive empirical study, we demonstrate that this causal approach to recommender systems leads to improved generalization to new data.

7.1 Future directions

We present immediate next steps at the end of each chapter that can extend the work. In this section, we present some long-term directions that can be explored.

Generic inference algorithm for probabilistic latent variables models. Exact posterior inference is generally intractable for latent variables models. We develop specific inference procedures to tractably analyze the large-scale data throughout this dissertation. However, this whole process of deriving problem-specific inference algorithm can be tedious and it requires a lot of modification once the model is revised. Black-box variational inference (Ranganath et al., 2014; Kucukelbir et al., 2015) and stochastic gradient variational Bayes (Kingma and Welling, 2013; Rezende et al., 2014) are two promising avenues for generic inference algorithm that is applicable to a wide variety of models. These black-box approaches will also enable us to build models with more complex structures beyond the simple bi-linear factors in this dissertation without worrying (too much) about fitting the model.

With generic inference algorithm, we would like to automate the Box's loop of model development (Box, 1976; Blei, 2014): build the model, fit the model, criticize (evaluate) the model, and revise the model (if necessary). *Edward* (Tran et al., 2016) is a software framework that is currently under active development with this goal in mind.

Stochastic optimization for sparse user feedback data. The models presented in this dissertation are mostly bi-linear factor models, which have limited modeling capacity. To leverage the advances of more powerful models (e.g., deep neural networks), the convenient closed-form coordinate updates are generally unavailable and it is normal to resort to stochastic optimization for model inference. User feedback data, whether explicit or implicit, is often very sparse (for implicit data, it is common to have $> 99.9\%$ of 0's). This presents additional challenge for stochastic optimization, as naively subsampling random user-item interaction will very likely over-emphasize the 0's in the data.[1] This problem is also closely related to generic inference algorithm mentioned above, as most of these black-box approaches heavily reply on stochastic optimization.

Gopalan and Blei (2013) address the similar sparsity issue with network data by subsampling 0's in a biased way, down-weighting their influence, then correcting the introduced bias to make sure the noisy stochastic objective matches the true objective in expectation. Rendle and Freudenthaler (2014) propose to subsample 0's more intelligently by choosing the 0's with bigger gradient (i.e., the negative examples which the model is more uncertain about) so that the learning procedure can make progress more rapidly. Similar idea has also been explored in fast inference for network data (Raftery et al., 2012).

The recent success of work embedding models (e.g., skip-gram word2vec (Mikolov et al., 2013)) demonstrates the effectiveness of negative sampling. One can view negative sampling as a way to battle the overwhelming negative examples in sparse data. It also has intimate connection with the exposure and propensity weights presented in Chapter 5 and Chapter 6, respectively: both of them are down-weighting the gradients of the negative examples in a principled way. Exploring the deeper connection among these work and developing general-purpose stochastic subsampling schemes for sparse user feedback data would be a valuable future direction.

Bridging Bayesian inference and causal inference for recommender systems. Our attempt at developing a causal inference approach to recommendation with Bayesian inference in Chapter 6 is only a small step of bringing these two fields together. There are still some fundamental problems that need to be theoretically justified, e.g., how to use inverse propensity weighting with Bayesian inference. Our explanation of "seeing each data point multiple

[1]Training the model with stochastic gradient descent by uniformly subsampling user-item interactions will usually lead to a weaker baseline.

times" suffices for doing a point estimate. But more rigorous formulation is required if we want a fully Bayesian treatment. Rubin (1978) formulate the problem of estimating the causal effect as a Bayesian inference problem—given observed data, specify a joint model over all the random variables (observed and latent), compute the predictive density for different potential outcomes—and present conditions under which the inference is "valid" with only observed data. It would be worthwhile to follow the similar procedure but we should also be cautious about validating the necessary assumptions.

Bibliography

Airoldi, E. M., Blei, D. M., Fienberg, S. E., and Xing, E. P. (2008). Mixed membership stochastic blockmodels. *Journal of Machine Learning Research*, 9:1981–2014.

Angelino, E., Johnson, M. J., and Adams, R. P. (2016). Patterns of scalable bayesian inference. *arXiv preprint arXiv:1602.05221*.

Beal, M. J. (2003). *Variational algorithms for approximate Bayesian inference*. University of London London.

Bertin-Mahieux, T., Ellis, D., Whitman, B., and Lamere, P. (2011). The Million Song Dataset. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 591–596.

Bertsekas, D. P. (1999). Nonlinear programming.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc.

Blei, D. M. (2014). Build, compute, critique, repeat: Data analysis with latent variable models. *Annual Review of Statistics and Its Application*, 1:203–232.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2016). Variational inference: A review for Statisticians. *arXiv preprint arXiv:1601.00670*.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent Dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Bottou, L. (1998). Online learning and stochastic approximations. *On-line learning in neural networks*, 17(9).

Bottou, L., Peters, J., Quiñonero Candela, J., Charles, D. X., Chickering, D. M., Portugaly, E., Ray, D., Simard, P., and Snelson, E. (2013). Counterfactual reasoning and learning systems:

The example of computational advertising. *Journal of Machine Learning Research*, 14:3207–3260.

Box, G. E. (1976). Science and statistics. *Journal of the American Statistical Association*, 71(356):791–799.

Cemgil, A. T. (2009). Bayesian inference for nonnegative matrix factorisation models. *Computational Intelligence and Neuroscience*.

Chaney, A. J., Blei, D. M., and Eliassi-Rad, T. (2015). A probabilistic model for using social networks in personalized item recommendation. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 43–50. ACM.

Charlin, L., Ranganath, R., McInerney, J., and Blei, D. M. (2015). Dynamic Poisson factorization. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 155–162.

Cho, E., Myers, S. A., and Leskovec, J. (2011). Friendship and mobility: user movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1082–1090. ACM.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Eck, D., Lamere, P., Bertin-Mahieux, T., and Green, S. (2007). Automatic generation of social tags for music recommendation. In *Advances in Neural Information Processing Systems*, pages 385–392.

Ellis, K., Coviello, E., Chan, A., and Lanckriet, G. (2013). A bag of systems representation for music auto-tagging. *Audio, Speech, and Language Processing, IEEE Transactions on*, 21(12):2554–2569.

Gelman, A. and Hill, J. (2006). *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press.

Gopalan, P., Hofman, J., and Blei, D. (2015). Scalable recommendation with hierarchical Poisson factorization. In *Uncertainty in Artificial Intelligence*.

Gopalan, P. K. and Blei, D. M. (2013). Efficient discovery of overlapping communities in massive networks. *Proceedings of the National Academy of Sciences*, 110(36):14534–14539.

Gopalan, P. K., Charlin, L., and Blei, D. (2014). Content-based recommendations with Poisson factorization. In *Advances in Neural Information Processing Systems 27*, pages 3176–3184.

Guo, G., Zhang, J., and Yorke-Smith, N. (2015). Trustsvd: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings. In *AAAI*, pages 123–129.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning*. Springer Series in Statistics.

Hernández-Lobato, J. M., Houlsby, N., and Ghahramani, Z. (2014). Probabilistic matrix factorization with non-random missing data. In *Proceedings of the 31th International Conference on Machine Learning, ICML*, pages 1512–1520.

Hoffman, M., Blei, D., and Cook, P. (2009). Easy as CBA: A simple probabilistic model for tagging music. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 369–374.

Hoffman, M., Blei, D., and Cook, P. (2010). Bayesian nonparametric matrix factorization for recorded music. In *Proceedings of the 27th Annual International Conference on Machine Learning*, pages 439–446.

Hoffman, M., Blei, D., Wang, C., and Paisley, J. (2013). Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347.

Hu, Y., Koren, Y., and Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 263–272. IEEE.

Imbens, G. W. and Rubin, D. B. (2015). *Causal Inference in Statistics, Social, and Biomedical Sciences*. Cambridge University Press.

Ishwaran, H. and Rao, J. S. (2005). Spike and slab variable selection: frequentist and Bayesian strategies. *Annals of Statistics*, pages 730–773.

Järvelin, K. and Kekäläinen, J. (2002). Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1999). An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*.

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, (8):30–37.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Kucukelbir, A., Ranganath, R., Gelman, A., and Blei, D. (2015). Automatic variational inference in Stan. In *Advances in Neural Information Processing Systems*, pages 568–576.

Lambert, D. (1992). Zero-inflated Poisson regression, with an application to defects in manufacturing. *Technometrics*, 34(1):1–14.

Lee, D. D. and Seung, H. S. (2001). Algorithms for non-negative matrix factorization. *Advances in Neural Information Processing Systems*, 13:556–562.

Li, L., Chen, S., Kleban, J., and Gupta, A. (2015). Counterfactual estimation and optimization of click metrics in search engines: A case study. In *Proceedings of the 24th International World Wide Web Conference (WWW'14), Companion Volume*.

Li, L., Chu, W., Langford, J., and Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 661–670, New York, NY, USA.

Liang, D., Charlin, L., and Blei, D. M. (2016a). Causal inference for recommendation. *in submission*.

Liang, D., Charlin, L., McInerney, J., and Blei, D. M. (2016b). Modeling user exposure in recommendation. In *Proceedings of the 25th International Conference on World Wide Web*, pages 951–961.

Liang, D., Hoffman, M., and Ellis, D. (2013). Beta process sparse nonnegative matrix factorization for music. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 375–380.

Liang, D., Paisley, J., and Ellis, D. P. W. (2014). Codebook-based scalable music tagging with Poisson matrix factorization. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 167–172.

Liang, D., Zhan, M., and Ellis, D. P. W. (2015). Content-aware collaborative music recommendation using pre-trained neural networks. In *Proceedings of the 16th International Society for Music Information Retrieval Conference*, pages 295–301.

Ling, G., Yang, H., Lyu, M. R., and King, I. (2012). Response aware model-based collaborative filtering. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence, Catalina Island, CA, USA, August 14-18, 2012*, pages 501–510.

Little, R. J. A. and Rubin, D. B. (1986). *Statistical Analysis with Missing Data*. John Wiley & Sons, Inc., New York, NY, USA.

Ma, H., King, I., and Lyu, M. R. (2009). Learning to recommend with social trust ensemble. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 203–210. ACM.

Ma, H., Yang, H., Lyu, M. R., and King, I. (2008). SoRec: Social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, pages 931–940. ACM.

Ma, H., Zhou, D., Liu, C., Lyu, M. R., and King, I. (2011). Recommender systems with social regularization. In *Proceedings of the fourth ACM International Conference on Web Search and Data Mining*, pages 287–296. ACM.

Marlin, B. M. and Zemel, R. S. (2009). Collaborative prediction and ranking with non-random missing data. In *Proceedings of the Third ACM Conference on Recommender Systems*, pages 5–12.

Marlin, B. M., Zemel, R. S., Roweis, S. T., and Slaney, M. (2007). Collaborative filtering and the missing at random assumption. In *UAI 2007, Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence, Vancouver, BC, Canada, July 19-22, 2007*, pages 267–275.

McFee, B., Barrington, L., and Lanckriet, G. (2010). Learning similarity from collaborative filters. In *ISMIR*, pages 345–350.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Morgan, S. L. and Winship, C. (2014). *Counterfactuals and Causal Inference*. Cambridge University Press.

Neal, R. M. (1993). Probabilistic inference using Markov chain Monte Carlo methods.

Neal, R. M. and Hinton, G. E. (1998). A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*, pages 355–368. Springer.

Paisley, J., Blei, D., and Jordan, M. (2015). Bayesian nonnegative matrix factorization with stochastic variational inference. In Airoldi, E., Blei, D., Erosheva, E., and Fienberg, S., editors, *Handbook of Mixed Membership Models and Their Applications*. Chapman and Hall/CRC.

Pan, R., Zhou, Y., Cao, B., Liu, N. N., Lukose, R., Scholz, M., and Yang, Q. (2008). One-class collaborative filtering. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 502–511. IEEE.

Paquet, U. and Koenigstein, N. (2013). One-class collaborative filtering with random graphs. In *Proceedings of the 22nd international conference on World Wide Web*, pages 999–1008.

Pearl, J. (2009). *Causality: Models, Reasoning and Inference*. Cambridge University Press, New York, NY, USA, 2nd edition.

Raftery, A. E., Niu, X., Hoff, P. D., and Yeung, K. Y. (2012). Fast inference for the latent space network model using a case-control approximate likelihood. *Journal of Computational and Graphical Statistics*, 21(4):901–919.

Ranganath, R., Gerrish, S., and Blei, D. M. (2014). Black box variational inference. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, pages 814–822.

Ranganath, R., Wang, C., Blei, D., and Xing, E. (2013). An adaptive learning rate for stochastic variational inference. In *Proceedings of The 30th International Conference on Machine Learning*, pages 298–306.

Rendle, S. (2010). Factorization machines. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, pages 995–1000.

Rendle, S. and Freudenthaler, C. (2014). Improving pairwise learning for item recommendation from implicit feedback. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, pages 273–282. ACM.

Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. (2009). BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 452–461.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of The 31st International Conference on Machine Learning*, pages 1278–1286.

Robert, C. and Casella, G. (2013). *Monte Carlo statistical methods*. Springer Science & Business Media.

Rubin, D. (1974). Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of Educational Psychology*, 66(5):688–701.

Rubin, D. B. (1978). Bayesian inference for causal effects: The role of randomization. *The Annals of statistics*, pages 34–58.

Salakhutdinov, R. and Mnih, A. (2008). Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*, pages 1257–1264.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM.

Schnabel, T., Swaminathan, A., Singh, A., Chandak, N., and Joachims, T. (2016). Recommendations as treatments: Debiasing learning and evaluation. In *International Conference on Machine Learning (ICML)*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Swaminathan, A. and Joachims, T. (2015). Counterfactual risk minimization. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, pages 939–941.

Tingle, D., Kim, Y., and Turnbull, D. (2010). Exploring automatic music annotation with acoustically-objective tags. In *Proceedings of the international conference on Multimedia information retrieval*, pages 55–62. ACM.

Tran, D., Blei, D. M., Kucukelbir, A., Dieng, A., Rudolph, M., and Liang, D. (2016). Edward: A library for probabilistic modeling, inference, and criticism.

Turnbull, D., Barrington, L., Torres, D., and Lanckriet, G. (2008). Semantic annotation and retrieval of music and sound effects. *IEEE Transactions on Audio, Speech and Language Processing*, 16(2):467–476.

van den Oord, A., Dieleman, S., and Schrauwen, B. (2013). Deep content-based music recommendation. In *Advances in Neural Information Processing Systems*, pages 2643–2651.

Vanchinathan, H., Nikolic, I., De Bona, F., and Krause, A. (2014). Explore-exploit in top-n recommender systems via Gaussian processes. In *Proc. ACM Recommender Systems Conference (RecSys)*.

Wainwright, M. J. and Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305.

Wang, C. and Blei, D. M. (2011). Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 448–456. ACM.

Wang, X. and Wang, Y. (2014). Improving content-based and hybrid music recommendation using deep learning. In *Proceedings of the 22nd ACM International Conference on Multimedia*. ACM Press.

Weston, J., Bengio, S., and Usunier, N. (2011). WSABIE: Scaling up to large vocabulary image annotation. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI*, pages 2764–2770.

Xie, B., Bian, W., Tao, D., and Chordia, P. (2011). Music tagging with regularized logistic regression. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 711–716.

Yoshii, K., Goto, M., Komatani, K., Ogata, T., and Okuno, H. G. (2006). Hybrid collaborative and content-based music recommendation using probabilistic model with latent user preferences. In *ISMIR*.

Zhao, X., Zhang, W., and Wang, J. (2013). Interactive collaborative filtering. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*, pages 1411–1420.