Efficient Matrix Models for Relational Learning

Ajit Paul Singh

October 2009
CMU-ML-09-111

MACHINE LEARNING DEPARTMENT

Carnegie Mellon

Efficient Matrix Models for Relational Learning

Ajit Paul Singh

OCTOBER 2009
CMU-ML-09-111

Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Geoffrey J. Gordon, Chair
Tom Mitchell
Christos Faloutsos
Pedro Domingos, U. Washington

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.

This research was sponsored by the Air Force Research Laboratory under grant number FA87500810009; SRI International under grant numbers 03000211, 71000152, and 55000691; the National Science Foundation under grant numbers SBE0354420 and ACI012671; John Hopkins University under grant number 820353658; and the University of Pittsburgh under grant number 0000164. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: Relational learning; matrix factorization; Bregman divergence; stochastic optimization; Bayesian models; Metropolis-Hastings

To my parents.

Abstract

Relational learning deals with the setting where one has multiple sources of data, each describing different properties of the same set of entities. We are concerned primarily with settings where the properties are pairwise relations between entities, and attributes of entities. We want to predict the value of relations and attributes, but relations between entities violate the basic statistical assumption of exchangeable data points, or entities. Furthermore, we desire models that scale gracefully as the number of entities and relations increase.

Matrices are the simplest form of relational data; and we begin by distilling the literature on low-rank matrix factorization into a small number of modelling choices. We then frame a large class of relational learning problems as simultaneously factoring sets of related matrices: i.e., Collective Matrix Factorization. Each entity is described by a small number of parameters, and if an entity is described by more than one matrix, those parameters participate in multiple matrix factorizations. Maximum likelihood estimation of the resulting model involves a large non-convex optimization, which we reduce to cyclically solving convex optimizations over small subsets of the parameters. Each convex subproblem can be solved by Newton-Raphson, which we extend to the setting of stochastic Newton-Raphson.

To address the limitations of maximum likelihood estimation in matrix factorization models, we extend our approach to the hierarchical Bayesian setting. Here, Bayesian estimation involves computing a high-dimensional integral with no analytic form. If we resorted to standard Metropolis-Hastings techniques, slow mixing would limit the scalability of our approach to large sets of entities. We show how to accelerate Metropolis-Hastings by using our efficient solution for maximum likelihood estimation to guide the sampling process.

This thesis rests on two claims, that (i) that Collective Matrix Factorization can effectively integrate different sources of data to improve prediction; and, (ii) that training scales well as the number of entities and observations increase. We consider two real-world data sets in experimental support of these claims: augmented collaborative filtering and augmented brain imaging. In augmented collaborative filtering, we show that genre information about movies can be used to increase the predictive accuracy of user's ratings. In augmented brain imaging, we show that word co-occurrence information can be used to increase the predictive accuracy of a model of changes in brain activity to word stimuli, even in regions of the brain that were never included in the training data.

Acknowledgments

This thesis would not have been possible without the support, patience, and indulgence of Geoff Gordon, my advisor. I am forever in his debt, and in awe of his eye for detail. Thanks in particular goes to Andrew Moore, whose guidance and insightful discussions largely shaped my first two years at Carnegie Mellon.

My doctoral career would not have been possible without the tremendous support and advice I received during my undergraduate and master's at the University of Alberta. Thanks to Renée Elio, who gave me my first research job, and every opportunity to build on that work as an undergraduate. Thanks especially to Russ Greiner, whose ability as a teacher and advisor drew me towards a research career in Machine Learning.

At Carnegie Mellon, I have had the pleasure of working with such a fantastic group of people in the SELECT and Auton labs, and more broadly in the School of Computer Science. Thanks especially to my co-authors: Geoff Gordon, Carlos Guestrin, Russ Greiner, Andrew Moore, Andreas Krause, Jure Leskovec, Asela Gunawardana, Chris Meek, Arun Surendran, Jon Kleinberg, Tim van Allen, and Peter Hooper.

Most of all, thanks to my family for encouraging and supporting me, both morally and with homemade snacks. Without them, I would have never have able to accomplish this.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Prediction is the pith and marrow of machine learning, the task which defines it. Stripped of all artifice, the goal of prediction is to guess the properties of an object given information about like objects.[1] The enormous variety of prediction tasks stem from different definitions of “property” and “like”.

The most common form of prediction represents objects by a predetermined set of features, or *attributes*. Attributes are the properties of an object whose value can be ascertained. Ascertaining the value of attributes may be difficult, costly, or time-consuming, and so we wish to automate the prediction of attributes. Each object is reduced to an assignment of values to attributes, a record. A set of records is known as a data set. The paradigmatic assumption in machine learning is that these records are exchangeable draws from a fixed, unknown probability distribution over the attributes. This probability distribution provides a way of reasoning about the behaviour of new objects. Because the data forms a table whose rows are records, and whose columns are attributes, we call such data tabular or attribute-value. Attribute-value data suffers from two important representational limitations, which motivate this thesis:

1. *Entities must be of the same type*. If objects are represented by a fixed set of attributes, then all the objects must possess those attributes. For example, if all the objects are human beings, then attributes like AGE, GENDER, HEIGHT describe each person. However, if we add a teapot to the set of persons, one would be hard-pressed to determine its gender.

[1] One may ask, “What is an object?”. Since this author has no desire to traipse down that ontological rabbit hole, let us simply assume that objects are things that have properties. Evoking the literature on relational databases, we usually refer to objects as “entities”.

2. *There are no relations.* Attributes are properties of objects; there is no notion of properties of sets of objects, or *relations*. For example, if the objects are people, we would be hard pressed to encode the notion of friendship as an attribute: first, because friendship is a property of two entities; second, because each entity may have different numbers of friends.

It is not hard to see that there are many kinds of data that violate attribute-value limitations: e.g., graphs where the nodes and edges correspond to entities and relations, respectively. Another example is a relational database, where the entities correspond to records, grouped by type into tables. A non-trivial relational database must have relations, which cannot be easily fit into an attribute-value representation. In Artificial Intelligence, predicate logic has long been used to encode relational data in knowledge bases: entities correspond to constants, relations and attributes are predicates, and logical sentences express the connection between relations. For background on the use of logic for knowledge bases, we refer the reader to Levesque and Lakemeyer [66].[2]

The primary difference between attribute-value and relational data is the existence of relations, or links, between entities. Because of the relations between entities, standard statistical assumptions, such as independence of entities, are violated. Moreover, the correlations due to relations should not be ignored as they provide a source of information that can significantly improve the accuracy of common machine learning tasks over models that exploit only the attributes. Moreover, in many scenarios, relations are the properties we want to predict.

1.1 Information Integration

Relational learning is a rich representation that has found use in many applications. However, in this thesis, we focus our attention on problems involving *information integration*. The phrase "information integration" is often used to refer to the problems involved in merging different data sources: e.g., duplicate record elimination, coreference resolution, record linkage. However, in this dissertation, information integration refers to the likely motivation for the aforementioned pre-processing tasks: incorporating different sources of information about entities to improve a predictive model.

[2] From the perspective of first-order logic, the things we call relations are actually functions, i.e., a mapping from sets of entities to a value, which need not be a truth value in logic. However, the phrase "relational learning" has come to refer broadly to techniques that exploit links between entities, whether or not those links are encoded as relations in first-order logic.

While the phrase "information integration" is often used to refer to the problems involved in merging different data sources (e.g., duplicate record elimination, coreference resolution, record linkage), we are concerned with integrating information from different data sources after the these pre-processing steps have been performed.

An entity often has many different properties (attributes and relations). One source of data may describe only a small subset of all the properties of a particular entity-type. Information integration problems address the scenario where one has multiple sources of data, each containing a different subset of properties for the same entities. The data sets may be collected independently, but the entities they describe overlap. If the properties in different data sources are correlated, information from one data source may be used to improve predictions of properties in other data sources.

Information integration problems are ubiquitous, and in this section we present examples which suggest the broad range of scenarios where the work of this thesis may be useful (the ones we consider further in this thesis are marked in boldface):

▷ *Web user modeling*: There are many different kinds of information one can collect about a user: search queries, bookmarks, online purchases, and measures of social interaction among users, such as e-mail and instant messages. The belief is that knowledge of a user's behaviour in one data source is predictive of their behaviour in other data sources (e.g., friends are more likely to have similar interests than two randomly selected persons, and this similarity of interests is reflected in their search queries).

▷ *Gene function prediction*: There are different sources of information one can collect about a gene: manually annotated hierarchies, such as the Gene Ontology [125], location on the genome, sequence similarity, and interaction between expressed proteins. The data sources are often collected independent of each other, but one may wish to augment a model for prediction in one data source (e.g., Gene Ontology) using another correlated source of data (e.g., interaction between expressed proteins).

▷ Educational psychometrics: Standardized educational testing yields large quantities of data regarding student performance on questions in different areas (e.g., English, History, Algebra, Calculus). The same students are tested in the same or different subjects. If one believes that there is transfer of skills, within or across domains (e.g., ability in algebra is predictive of ability in calculus), then integrating student performance data from different subjects should improve a predictive model of student performance.

▷ **fMRI modeling**: Functional Magnetic Resonance Imaging (fMRI) is often used to measure responses in small regions of the brain (i.e., voxels) given external stimuli. Given enough experiments on a sufficiently broad range of stimuli, one can build models that predict patterns of brain activation given new stimuli [83]. Running enough experiments is costly, but we can often collect cheap side information about the stimuli. We consider an experiment where the stimulus is a word-picture pair displayed on a screen. We can collect statistics of whether the stimulus word co-occurs with other commonly used words in large, freely available text corpora. By integrating the two sources of information, related through the shared set of stimulus words, we can significantly improve a model of brain activation.

▷ **Collaborative filtering**: Recommendation systems involve using users' measured preferences for certain items to select other items the users might be interested in. A notable example of this is the Netflix Prize challenge, which involves predicting the rating a user would assign to a movie given millions of user ratings as training data. The recommendation problem involves a very simple relation: the rating a user assigns to a movie. However, we often have other sources of side information—properties of the users like their age, gender, and friendship between users; and properties of movies like the genre it belongs to and the actors in the film. These forms of side information are themselves additional relations; can we use them to improve the quality of recommendations?

As we seek to model increasingly complicated objects, the representational limitations of attribute-value learning become obvious. Objects can have many different properties (attributes and relations); we cannot measure them all at once, nor can we easily establish, a priori, which properties are most relevant to the task at hand. In contrast, a relational representation allows for a unified representation of heterogeneous data sources.

We focus on two information integration tasks in this thesis: *augmented collaborative filtering* and *augmented brain imaging*. In the first task, we have data about interactions between users and items (ratings, purchases), which is augmented with side information about the items. In the second task, we have data about how activity in small regions of the brain change in response to a stimuli, which is augmented with side information about the stimuli.

1.2 Thesis Statement

Our goal is the development of statistical techniques that can predict the value of unobserved relations, by exploiting correlations between observed relations. For any such technique, we have as our desiderata the following:

1. *A flexible representation language*: Relational data is a richer representation than its attribute-value counterpart, which can complicate modeling. Typically, there are

multiple relations, and each can take on its own value type. For example, in the augmented collaborative filtering example, ratings are on an ordinal scale, whereas the other relations are binary. Our modeling approach should be able to take into account relations with different value-types.

2. *Does not require structural knowledge*: Often, we are presented with data where we believe the relations are correlated, but the structure of these correlations is unknown. Our techniques should not rely upon the existence of prior knowledge about how information propagates across relations. In the augmented brain imaging example, one would be hard pressed to elicit rules relating word counts to activity in regions of the brain. We believe that the two data sources are correlated, but encoding structural information about how they correlate is difficult.

3. *Generalization to new entities*: Our training data will contain a fixed set of entities, but any model we learn should be able to predict relations involving entities that did not appear in the training data, but where some relations involving the new entity are observed. In the augmented collaborative filtering example, we want our model to generalize well to new users, not just the ones who were represented in the training data.

4. *Relations are sparsely observed*: One rarely observes a relation for all combinations of its arguments. While the data may contain a large number of entities, the typical number of observed relations an entity participates in may be small. In the augmented collaborative filtering example, only a small fraction of items are rated by any single user.

5. *Models reflect internal uncertainty*: Since statistical models are learned from finite training data, there is always some uncertainty in predicted values. We seek models that can quantify their own internal uncertainty about the world. This is most commonly achieved through Bayesian techniques.

6. *Simple interface*: While the model may involve complicated routines for learning and prediction, the end-user should not have to concern themselves with the internal details of a relational model.

1.3 Statistical Design Patterns

There are certain problems in machine learning which recur in a variety of different domains. For example, a vision researcher might want to infer a hierarchical categorization of images using SIFT features; while at the same time, a bioinformatics researcher is trying to infer a gene ontology from sequence features. The objects and their properties are very different, but the underlying problem is the same—inducing a hierarchical categorization of objects from recorded properties. Each researcher may independently create very similar probabilistic models for inducing hierarchies, or they might realize the similarity from a literature review, and attempt to adapt related work to their own application. The limitation of the cross-pollination approach is that significant effort may be required to separate out the domain-specific aspects in related work. The consequence of tying models to specific applications is that a lot of effort is duplicated unnecessarily.

There are well-developed subareas within machine learning, where commonly occurring modeling tasks have been abstracted away from their specific domains, allowing them to be easily reused and adapted in other settings. One of the best examples in machine learning is time-series modeling, where Hidden Markov Models are a popular approach. The first application of Hidden Markov Models (HMMs) were in speech recognition in the 1970s [100]. Over time, the salient structure of the problem was abstracted away from its original application—i.e., the salient structure is discrete-state filtering in sequential data under a Markov assumption. Today, Hidden Markov Models and their variants are one of the most popular techniques in bioinformatics: different domains; common structure. Modeling sequential data is a common problem in many fields, and we recognize HMMs as a basic approach for such problems.

Graphical models can act as an interface between the low-level details of training and inference and high-level domain-specific tasks. Using graphical models as an interface works reasonably well when there are a small number of random variables (e.g., one random variable per attribute). When dealing with complex, structured data involving many different types of entities, each with their own set of attributes and relations, the graphical-models-as-interface approach is less useful. High-level languages for defining graphical models have been proposed: Markov Logic Networks [32][3], Bayesian Logic [79], Probabilistic Inductive Logic Programming [101], Probabilistic Relational Models [39], Relational Markov Networks [124], Church [43], IBAL [96], FACTORIE [77], Infer.NET [80], DAPER [49]. These high-level languages provide macros for generating graphical models with repeated structure among the variables. The macros are usually syntactically derived from subsets of typed first-order logic, although FACTORIE and Infer.NET are based on object-oriented languages, and Church and IBAL are based on LISP and ML, respectively. From the perspective of relational learning, first-order logics are convenient: they have compact descriptors for relations, and declarative languages can hide many of

[3] Domingos and Lowd [32] makes a related argument about the need for high-level languages for graphical models.

the low-level details of constructing the underlying graphical model. Making an analogy to software engineering, graphical models are akin to the intermediate representation a compiler uses when converting C or Java code into assembly; these high-level languages are akin to programming languages like C++, LISP, or Prolog. Converting a description in the high-level language into a graphical model is akin to compilation.

While these languages are enormously powerful tools, they provide little design guidance to the modeler. What we want is the ability to convey expertise in the design of graphical models for solving particular kinds of problems. In software engineering, such expertise is often encoded in a *design pattern* [37], a reusable solution to a recurring problem in software design. We propose a similar concept for graphical models, a *statistical design pattern*, a reusable solution to a recurring modeling problem. Information integration is a statistical design pattern that appears under many guises in the literature, e.g., in applications of semi-supervised learning, transfer learning, transduction. The approach we propose, *Collective Matrix Factorization*, is an example of a statistical design pattern that addresses the information integration problem.

There are a few examples of statistical design patterns that have evolved naturally—e.g., Hidden Markov Models and their variants form a statistical design pattern. Low-rank matrix factorization is another example of a statistical design pattern. Neither example maps to a specific model, but rather a class of models that address a basic problem. There are many variants of matrix factorization, but they all address the same basic problem: predicting the value of an arity-two relation, where the rows and columns index each of the arguments, and the entries correspond to values of the relation. In Chapter 3 we provide a unified view of low-rank matrix factorization algorithms (e.g., singular value decomposition [42], non-negative matrix factorization [63]).

An advantage of statistical design patterns is that the graphical models that implement the pattern may have special structure which makes training and inference easier. Such structure is extensively exploited in training and prediction with Hidden Markov Models. Similarly, we exploit structure in the graphical models produced by collective matrix factorization to reduce the computational cost of training and prediction, even though the graphical model will grow with the number of entities.

A statistical design pattern may be encoded in a variety of high-level languages, but in this thesis we focus on collective matrix factorization as a graphical plate model, a kind of high-level language itself. Plate models are also easily expanded into graphical models. Many of our contributions focus on the details of low-level optimization, which are easier to describe as graphical models. The modeler need never be exposed to the details of the graphical model: from their view, collective matrix factorization is presented purely in terms of matrices. Figure 1.1 diagrams our world-view.

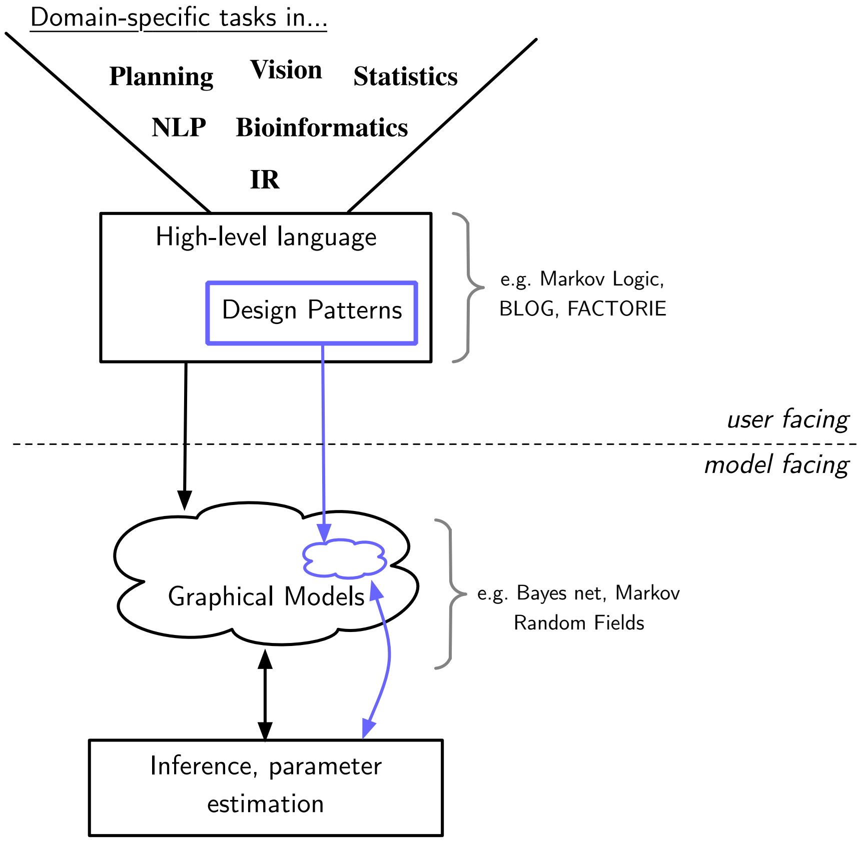


Figure 1.1: The process of developing a probabilistic model. The modeler/user begins with a domain-specific task. Once the domain is mapped onto a graphical model, either directly or using a high-level language, the details of inference and estimation can be largely hidden.

1.4 Main Contributions

This dissertation can be broken down in three parts. In the first part, we propose a unified view of matrix factorization, viewing matrices as the simplest, most common form, of relational data. In the second part, we propose representing a large class of relational data sets as collections of related matrices. Each matrix represents an arity-two relation, where the rows and columns index entities. Two matrices are related if they share dimensions; that is, they describe the same entities participating in different relations. Sets of related matrices provide a simple interface for the end modeler. Beneath this interface, we propose a statistical model based on tying parameters in the low-rank factorization of each matrix. We call this approach *Collective Matrix Factorization*. This leads to a maximum likelihood parameter estimation problem where the parameter space grows with the number of entities, and where the underlying optimization is non-convex. In the third part, we extend Collective Matrix Factorization to a fully Bayesian setting, which addresses many of the limitations of the maximum likelihood approach. Even though we are modeling relational data using relatively simple matrix factorization approaches, we show that (i) there is a surprising amount of flexibility in the types of problems one can represent using sets of related matrices; (ii) the structure within matrix factorizations allows us to develop very efficient algorithms for learning and prediction.

This thesis stands on the following contributions, many of which center on novel algorithms for the large-scale parameter estimation problems that arise from factoring sets of related matrices:

- A unified view of matrix factorization that reduces the panoply of models in the literature into six modeling choices, independent of the choice of learning algorithm. This approach subsumes dimensionality reduction techniques, clustering algorithms, and matrix co-clustering algorithms into a single framework. We believe that these choices capture the important differences between matrix factorization models.

- An efficient maximum likelihood estimation algorithm for Collective Matrix Factorization. Our approach exploits structure in matrix factorization models to reduce parameter estimation into cyclically updating parameters in a set of tied linear models. This reduction allows us to exploit not only the gradient of the objective, but partial information about the Hessian to dramatically speed up learning. Our alternating Newton-projection approach can be applied to any matrix (or collective matrix) factorization where the objective is decomposable and twice-differentiable.

- When the data matrices are densely observed (i.e., where entities are observed to participate in many relationships) we propose the use of stochastic Newton optimization to reduce the cost of maximum likelihood estimation. While stochastic optimization has long been used for linear models, we are the first to generalize this approach to matrix factorization.

- We extend Collective Matrix Factorization to the hierarchical Bayesian case, where a posterior distribution over the model parameters is computed. This Bayesian approach allows us to accurately generalize to new entities, and to account for the effect of parameter uncertainty on predictions. We present experimental evidence which illustrates the merits of the hierarchical Bayesian approach over maximum likelihood in Collective Matrix Factorization.

- ▷ Usually Bayesian techniques reduce the expressiveness of the model to make parameter estimation easier (e.g., conjugacy assumptions). We propose a Metropolis-Hastings sampler, which imposes no such restrictions. Every maximum likelihood Collective Matrix Factorization has a Bayesian analogue, which we can compute using, for example, Metropolis-Hastings. However, standard random-walk Metropolis-Hastings mixes very slowly, and so this simple version of the Bayesian approach is only practical on very small matrices. Instead, we propose an adaptive, block Metropolis-Hastings sampler where the proposal distribution is dynamically computed using the gradient and per-row Hessian. Our alternating Newton-projection technique, used for maximum likelihood estimation, provides the local gradient and per-row Hessian. Essentially, we are using an efficient algorithm for maximum likelihood inference to guide Metropolis-Hastings, reducing the cost of learning. Our adaptive approach is practical on much larger data sets, such as those found in the augmented brain imaging problem.

- ▷ Using the augmented collaborative filtering and augmented brain imaging problem, we show that integrating multiple relations leads to superior prediction. This illustrates the value of Collective Matrix Factorization on information integration problems.

1.5 Organization of the Thesis

The remainder of the thesis is organized as follows:

Chapter 2: We cover background material and notation which will be used throughout this thesis. A significant portion of the chapter deals with probabilistic graphical models.

Chapter 3: We present our unified view of matrix factorization, which captures the important modeling decisions common to all matrix factorization models.

Chapter 4: Building on a particular matrix factorization algorithm, Exponential Family PCA [26], we introduce collective matrix factorization as a technique for modeling relational data. By exploiting structure within collective matrix factorization, we develop computationally efficient techniques for learning the model parameters, even when the number of entities and the number of observed relations is large. We empirically evaluate our algorithm on an augmented collaborative filtering task.

Chapter 5: We extend collective matrix factorization to a hierarchical Bayesian model. We discuss the limitations of the maximum a posteriori inference used in the previous chapter, and how the hierarchical Bayesian model addresses these concerns. We show how the training algorithm for collective matrix factorization can be used to guide exact Markov Chain Monte Carlo inference in the hierarchical Bayesian variant of collective matrix factorization.

Chapter 6: We review the literature on statistical relational learning, discussing how this thesis relates to existing research in relational learning.

Chapter 7: We summarize the main contributions of this thesis. We finally conclude with a discussion of open problems and directions for future research.

Chapter 2

Background

In this chapter, we introduce terminology and concepts from statistics and optimization that are relevant to this thesis. A large portion of our work is concerned with efficient algorithms for maximum likelihood and Bayesian estimation in complex, high-dimensional, distributions. Section 2.2 is a brief refresher on maximum likelihood and Bayesian estimation. Maximum likelihood estimation involves optimizing the parameters with respect to a likelihood function, so we review the basic of optimization, including convexity, gradient descent, and Newton's method in Section 2.3. Collective Matrix Factorization (Chapter 4) is a probabilistic model with large numbers of variables, where the joint distribution can be expressed as the product of factors over smaller subsets of variables. Graphical models, introduced in Section 2.4, are a useful formalism for expressing factorizations of complex distributions. In Chapter 5, we propose a hierarchical Bayesian extension to Collective Matrix Factorization, where Markov Chain Monte Carlo allows us to train the model. Markov Chain Monte Carlo is reviewed in Section 2.5.

2.1 Basic Terminology

We introduce some basic notation and terminology used through this thesis:

- Vectors and scalars are denoted by lower-case Roman or Greek letters: x, y, θ. The elements of a length n vector are denoted $x = (x_1, x_2, \ldots, x_n)$. Vectors are assumed to be row-vectors, contrary to the usual convention.

- Matrices are denoted by capital Roman or Greek letters: X, Y, Θ. Rows and columns of a matrix are denoted by subscript notation: $X_{i\cdot}$, $Y_{\cdot j}$, $\Theta_{i\cdot}$. To emphasize the size of a matrix we write $X_{m \times n}$, indicating the matrix has m rows and n columns. The $n \times n$ identity matrix is denoted $I_{n \times n}$.

- ▷ The set of real numbers is denoted $\mathbb{R}$. Real coordinate spaces over n coordinates are denoted $\mathbb{R}^n$. In the case of matrices, real coordinate spaces on mn dimensions are denoted $\mathbb{R}^{m \times n}$.

- ▷ A norm is denoted $|| \cdot ||$. The Euclidean norm on $\mathbb{R}^n$ is denoted $|| \cdot ||_2$:

$$||x||_2 = \left(|x_1|^2 + \ldots + |x_n|^2\right)^{1/2}.$$

The corresponding Euclidean norm on on $\mathbb{R}^{m \times n}$, the Frobenius norm, is defined as

$$||X||_{Fro} = \left(\sum_{i=1}^{m}\sum_{j=1}^{n} X_{ij}^2\right)^{1/2}.$$

- ▷ The trace of a $m \times m$ square matrix A is

$$\text{tr}(A) = \sum_{i=1}^{m} A_{ii}.$$

- ▷ The inner product between vectors $x, y \in \mathbb{R}^n$ is denoted $\langle x, y \rangle$ where

$$\langle x, y \rangle = xy^T = \sum_{i=1}^{n} x_i y_i.$$

The inner product between matrices $X, Y \in \mathbb{R}^{m \times n}$ is denoted $\langle X, Y \rangle$ where

$$\langle X, Y \rangle = \text{tr}(X^T Y) = \sum_{i=1}^{m}\sum_{j=1}^{n} X_{ij} Y_{ij}.$$

A short hand notation for the inner product is $x \circ y$ or $X \circ Y$. The element-wise (Hadamard) product between matrices or vectors is denoted $x \odot y$, or $X \odot Y$, where the arguments are of the same dimensions.

- ▷ The gradient of a function f is denoted ∇f. The gradient of a function with respect to a subset of its variables, x, is denoted $\nabla_x f$. The Hessian of a function is likewise denoted $\nabla^2 f$ or $\nabla_x^2 f$.

2.2 Probability and Statistics

In this section, we assume the reader has a basic understanding of probability: the concept of a random variable, probability distributions and densities, expectations, conditional probability, independence, and conditional independence.

We denote random variables, vectors, and matrices using the same notation as variables (see Section 2.1). We disambiguate random variables from variables only when the difference is unclear from the context. A draw from a random variable with probability density function $\pi(\theta)$ is denoted

$$x \sim \pi(\theta).$$

A joint distribution on random variables $\{x_1, x_2, \ldots, x_n\}$, with parameters θ, is denoted

$$p(x_1, x_2, \ldots, x_n; \theta).$$

The parameters of the distribution are sometimes referred to as the model. Data $\mathcal{D}$ consists of a set of records, each of which contains a (possibly partial) assignment to the n random variables. The joint distribution also defines a likelihood function,

$$\ell(\theta) = p(\mathcal{D} \mid \theta),$$

which, for a fixed set of training data, assigns a score to each possible value of θ.

Training (a.k.a. learning, parameter estimation) is the process of selecting a model, or assigning scores to different models, given a fixed set of training data. Maximum likelihood estimation chooses the parameters which maximize the likelihood function (an optimization problem that we discuss further in Section 2.3):

$$\theta_{MLE} = \operatorname*{argmax}_{\theta} \ell(\theta). \tag{2.1}$$

Bayesian estimation is another approach to determining the behaviour of θ, where the goal is to determine the posterior distribution of the parameters given training data:

$$p(\theta \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \theta) p(\theta)}{p(\mathcal{D})}. \tag{2.2}$$

The prior distribution over parameters $p(\theta)$, is an assumption made on how likely different models are, prior to observing the data. The posterior distribution $p(\theta \mid \mathcal{D})$ is a combination of prior belief with the likelihood, which is a function of the training data. Computing the evidence,

$$p(\mathcal{D}) = \int p(\mathcal{D} \mid \theta) p(\theta), d\theta, \tag{2.3}$$

requires computing an integral over a potentially large parameter space. In many cases, the integral will not have a closed-form solution. We discuss techniques for estimating the posterior distribution, by sampling from it, in Section 2.5.

2.3 Optimization

Optimization of a continuous function plays a fundamental role in training statistical models under the principle of maximum likelihood. This section briefly reviews the basics of optimization that are relevant to this dissertation. For a thorough introduction to optimization, see Boyd and Vandenberghe [13], Nocedal and Wright [90].[1]

2.3.1 Optimization and Maximum Likelihood Estimation

Mathematical optimization (or just optimization) is at the heart of maximum likelihood parameter estimation. Given variables $\theta \in \mathbb{R}^n$ and a function over these variables, $f_0 : \mathbb{R}^n \to \mathbb{R}$, an optimization problem has the following form:

$$\begin{aligned} &\min_{\theta} && f_0(\theta) && (2.4) \\ &\text{subject to} && f_i(\theta) \leq b_i, \quad i = 1 \ldots c. && (2.5) \end{aligned}$$

The function f_0 is known as the objective. The functions $f_i : \mathbb{R}^n \to \mathbb{R}$ are inequality constraints, and the constants b_i are the bounds on the constraints. If one or more constraints are defined, the problem is known as a constrained optimization; otherwise, it is an unconstrained optimization.

Maximum likelihood estimation (Equation 2.1) is framed as a maximization problem, which can be easily converted to a minimization problem: let $f_0(\theta) = -\log p(\mathcal{D} \mid \theta)$ be the negative log-likelihood of the model. If only some values in $\mathbb{R}^n$ correspond to valid parameters, encode those constraints using f_i.

[1] We follow the notation of Boyd and Vandenberghe [13], but Nocedal and Wright [90] provides more detailed coverage of optimization algorithms.

2.3.2 Convex Optimization

When the functions $f_0, f_1, \ldots, f_c$ are convex, Equations 2.4-2.5 are known as a convex optimization. A function is convex if the following inequality holds for all $\theta_1, \theta_2 \in \mathbb{R}^n$ with $\alpha \geq 0$, $\beta \geq 0$, and $\alpha + \beta = 1$:

$$f_i(\alpha\theta_1 + \beta\theta_2) \leq \alpha f_i(\theta_1) + \beta f_i(\theta_2).$$

A large number of convex optimizations are additionally linear programs: each function $f_0, f_1, \ldots, f_c$ satisfies

$$f_i(\alpha\theta_1 + \beta\theta_2) = \alpha f_i(\theta_1) + \beta f_i(\theta_2).$$

Nonlinear and non-convex optimizations are those where the objective and/or constraints violate the linearity and/or convexity conditions above.

An optimum of the objective f_0 is a value $\hat{\theta}$ which is feasible (i.e., satisfies the constraints) and is the infimal value in an ϵ-ball around it:

$$f_0(\hat{\theta}) = \inf\{f_0(\theta) \,|\, \forall i = 1 \ldots c,\ f_i(\theta) \leq b_i,\ ||\theta - \hat{\theta}||_2 \leq \epsilon\}.$$

An optimum is local if there is another point $\hat{\theta}_2$ that is an optimum and $f_0(\hat{\theta}_2) < f_0(\hat{\theta})$. A global optimum θ^* is one where there exists no other point $\hat{\theta}_2$ such that $f_0(\hat{\theta}_2) < f_0(\theta^*)$. A desirable property of convex optimizations is that any local optimum is also a global optimum. Often, in non-convex optimization, the best guarantee one can provide is convergence to a local optimum.

2.3.3 Techniques for Unconstrained Optimization

The maximum likelihood estimation problems encountered in this thesis are most often unconstrained optimizations, where the objective f_0 is differentiable. Therefore, a necessary and sufficient condition for θ^* to be a stationary point is

$$\nabla f_0(\theta^*) = 0. \tag{2.6}$$

In some cases there is an analytic solution to Equation 2.6. In most cases a solution is found by an iterative algorithm that, starting at initial value $\theta^{(0)}$, generates a sequences of feasible iterates $\theta^{(1)}, \ldots, \theta^{(\infty)}$ such that $f_0(\theta^{(1)}), \ldots, f_0(\theta^{(\infty)})$ is a sequence that converges to $f_0(\theta^*)$. If f_0 is convex, the iterative algorithm converges to a global optimum. However, the same algorithms can be applied to non-convex optimizations, with the caveat that they may (and typically will) converge to a local optimum.

Gradient descent (a.k.a. method of steepest descent) is a common iterative approach to solving Equation 2.6:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \cdot \underbrace{\nabla f_0(\theta^{(t)})}_{-s^{(t)}}.$$

The stopping criteria checks whether $||\nabla f_0(\theta^{(t+1)})||_2 \leq \epsilon_0$ for a small positive value of ϵ_0. The step length $\eta \in [0, 1]$ is chosen to guarantee sufficient decrease of the objective. In the case of gradient descent, the step $s^{(t)}$ is the negative gradient.

A common condition to testing whether a step in the direction of $s^{(t)}$ with length η leads to sufficient decrease is the Armijo condition:

$$f_0\left(\theta^{(t)} + \eta \cdot \nabla f_0\left(\theta^{(t)}\right)\right) - f_0\left(\theta^{(t)}\right) \leq c_1 \eta \left\langle \nabla f_0, s^{(t)} \right\rangle. \tag{2.7}$$

for some constant $c_1 \in (0, 1)$. A common approach to finding a good step length involves starting with $\eta = 1$, and then testing step lengths $\{\beta\eta, \beta^2\eta, \beta^3\eta \ldots\}$, for $\beta \in (0, 1)$, till sufficient decrease is confirmed, or the step length is deemed insufficiently large.

Newton-Raphson (a.k.a. Newton's Method) is an iterative approach to solving Equation 2.6 that takes the local curvature of f_0 at each iterate $\theta^{(t)}$ into account:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \cdot \underbrace{\nabla f_0(\theta^{(t)}) \left[\nabla^2 f_0(\theta^{(t)})\right]^{-1}}_{-s^{(t)}}. \tag{2.8}$$

As with gradient descent, the Armijo condition may be used to test for sufficient decrease of the objective. We note that if $\theta \in \mathbb{R}^n$, the Hessian will be a $n \times n$ matrix. The cost of the Newton-Raphson step is usually dominated by the $O(n^3)$ cost of inverting the Hessian.

While one iteration of Newton-Raphson is more expensive than one of gradient descent, Newton-Raphson has a superior rate of convergence: fewer iterations are required to converge to an optimum. A detailed comparison of rates of convergence is beyond the scope of this section, but we can loosely characterize the rate of convergence for gradient descent as linear: $\{f_0(\theta^{(t)})\}_t$ converges to its minimum at a linear rate.[2] For Newton-Raphson, there exists a sequence of iterates sufficiently close to θ^* that $\{||\nabla f_0(\theta^{(t)})||\}_{t\,:\,t>t_0}$ converges quadratically to zero.[3]

[2] A sequence $\{x^{(t)}\}$ that converges to x^* has a linear rate of convergence in t if there is a constant $r \in (0, 1)$ such that $\frac{||x^{(t+1)} - x^*||}{||x^{(t)} - x^*||} \leq r$. The same sequence has a Q-order of convergence p, for $p > 1$ if there is a constant $M > 0$ such that $\frac{||x^{(t+1)} - x^*||}{||x^{(t)} - x^*||^p} \leq M$ for some $t > t_0$. When $p = 2$ the Q-order of convergence is commonly referred to as quadratic convergence [90, Section 2.2].

[3] The rate of convergence characterisations assume that the line search is exact: η is chose to maximize the decrease in the objective between iterations. Newton-Raphson requires additional conditions be true in

2.4 Probabilistic Graphical Models

Probabilistic graphical models are compact representations of a probability distribution over random variables $X_1, \dots, X_n$. The joint distribution of the random variables is factored into the product of potential functions over subsets of the variables:

$$\begin{aligned} p(x_1, \dots, x_n; \theta) &= \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(x_C), \text{ where,} \\ Z &= \sum_{x_1} \cdots \sum_{x_n} \prod_{C \in \mathcal{C}} \psi_C(x_C), \\ x_C &\subseteq \{x_1, x_2, \dots, x_n\}, \\ \psi_C(x_C) &\geq 0, \end{aligned}$$

We refer to the subsets of variables, $\mathcal{C}$, as cliques. The cliques are groupings of variables over which the potentials are defined, and are related to the independence structure of the graphical model. If a variable is continuous, then replace the corresponding summation with integration. The parameters θ are the set of parameters used to define the potential functions ψ_C.

2.4.1 Directed Graphical Models

In a directed graphical model, each potential function is the probability distribution of a variable given parents:

$$p(x_1, \dots, x_n; \theta) = \prod_{i=1}^{n} \underbrace{p(x_i \mid x_{\pi_i}; \theta_i)}_{\psi_i(x_i)}.$$

The parents of variable x_i are denoted x_{π_i}. The parameters of the conditional probability distribution, which determines the distribution of x_i given the value of x_{π_i}, is denoted θ_i. If we draw a graph where each x_i corresponds to a node, and each directed edge from $x_j \in x_{\pi_i}$ to x_i corresponds to a dependence, then the resulting graph must be acyclic for the model to be directed. An example of a directed acyclic graph (DAG) on five variables is presented in Figure 2.1. The parameters of the graphical model are $\theta = \{\theta_1, \dots, \theta_n\}$.

A useful property of directed graphical models is that the product of the conditional probability distributions (CPDs) is the joint distribution, and so the normalizing term $Z =$

a neighbourhood around θ^*, such as Lipschitz continuity of the Hessian. We refer to reader to [90, Section 3.3] for details.

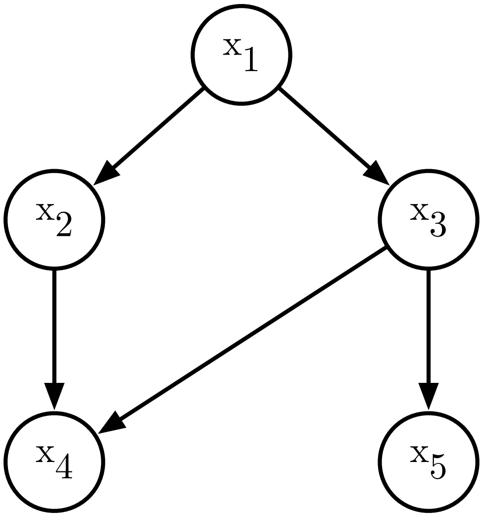


Figure 2.1: A directed acyclic graph on random variables $\{x_1, \dots, x_n\}$.

1. Since the log-normalizer does not involve a high-dimensional integration, maximum likelihood estimation is straightforward if all the variables are observed:

$$\begin{aligned} f_0(\theta) &= -\log p(x_1, \dots, x_n; \theta) \\ &= -\sum_{i=1}^{n} \log p(x_i \,|\, x_{\pi_i}; \theta_i). \end{aligned} \tag{2.9}$$

To minimize f_0, we can optimize independently over each term in the summation in Equation 2.9. The optimization is further simplified by assuming that, for each assignment to x_{π_i}, $p(x_i \,|\, x_{\pi_i}; \theta_i)$ is an exponential family distribution (Definition 2) which depends only on the value of x_{π_i} and θ_i.

2.4.2 Undirected Graphical Models

Given an arbitrary joint distribution there are often many ways to factor it into the normalized product of potentials. In some cases we are fortunate, and the factorization can be represented compactly using a directed graphical model. Undirected graphical models do not require that the graph of dependencies between variables be acyclic. The most common form of undirected graphical model uses a log-linear formulation:

$$p(x_1, \ldots, x_n) = \frac{1}{Z} \exp\left(\sum_j w_j f_j(x_1, \ldots, x_n)\right),$$

where $f_j(\cdot) \in \{0, 1\}$ is a function of the state. While we do not use undirected graphical models in this dissertation, the log-linear formulation has found use in other techniques for relational learning: e.g., Markov Logic Networks [105], and Relational Markov Networks [124].

2.4.3 Inference

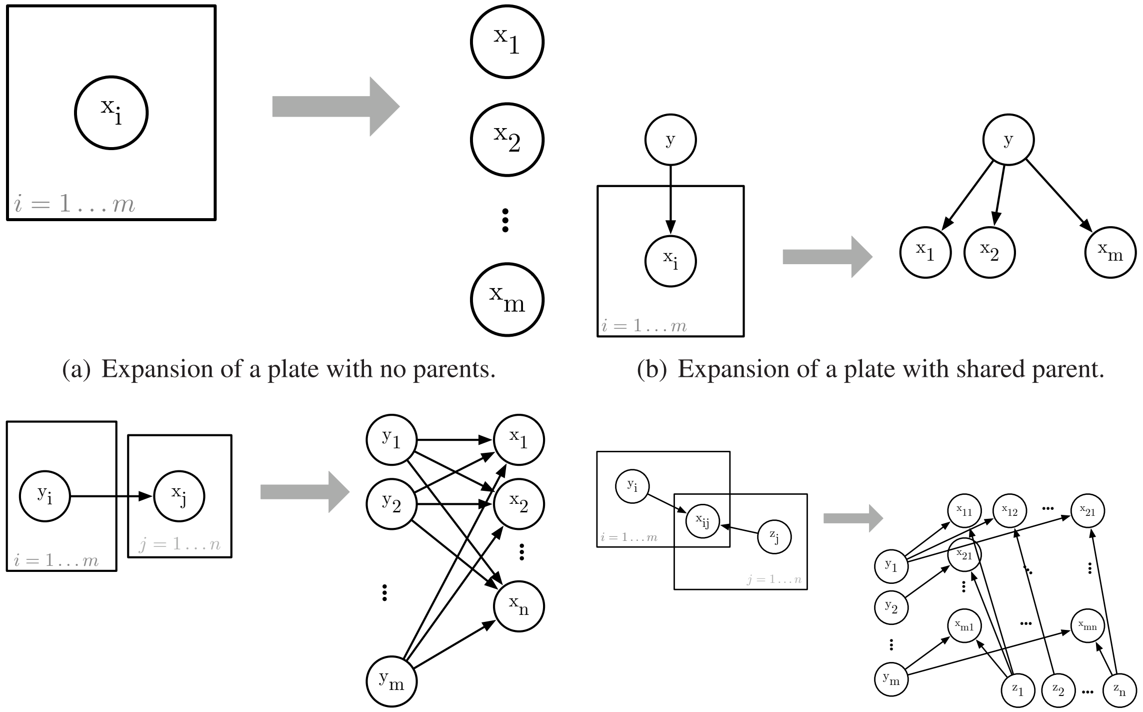
In visual representations of a graphical model, we shade nodes whose values are known, i.e., evidence. The most fundamental use of a joint distribution involves computing the probability of an event given (partial) evidence—i.e., inference. If no restrictions are placed on the graphical model, then inference on a directed graphical model, framed as a decision problem, is NP^{PP}-complete [93].[4] In the case of Collective Matrix Factorization (Chapter 4), inference is additionally complicated by the fact that one of the basic operations in inference, joining potentials and eliminating variables, may involve computing integrals with no analytic form.

2.4.4 Plate Models

Plate models are a language for defining repeated structure in directed graphical models [18, 118]. When defining the directed acyclic graph among variables, a plate is a notational short-hand for replication of a variable and the arcs that represent dependence. Four examples of plates notation and the equivalent unrolled model are presented in Figure 2.2.

Plates are usually denoted by square boxes around sets of variables. Each plate contains one or more variables, and there is no requirement that a variable exist in a plate. However, each variable in a plate is indexed by a subscript. An annotation to the plate indicates how many times the subscripted variable is repeated. A plate is, in effect, a for-loop. Like a for-loop, unrolling a plate consists of creating a variable for each iteration of the plate annotation (e.g., $i = 1 \ldots n$). If a variable has multiple subscripts, then it belongs to multiple plates: e.g., intersecting plates, Figure 2.2(d). We note that random

[4] The complexity result requires finite bit-length of the inputs: i.e., discrete variables with tabular CPDs.



(a) Expansion of a plate with no parents.

(b) Expansion of a plate with shared parent.

(c) Expansion across plates.

(d) Expansion of intersecting plates

Figure 2.2: Examples of plate models and the corresponding unrolled network.

variables in a plate model or directed acyclic model may be scalar or a vector of random variables, a notational convenience we use when defining the plate model for Collective Matrix Factorization (Chapter 4).

2.5 Bayesian Learning

In Section 2.2, we discussed the central role of the posterior distribution $p(\theta \mid \mathcal{D})$ in Bayesian learning. If we wish to use the posterior for prediction, e.g., to predict the likelihood of new test data, $\mathcal{D}^{new}$, we must average over the posterior distribution:

$$p(\mathcal{D}^{new} \mid \mathcal{D}) = \int p(\mathcal{D}^{new} \mid \theta) p(\theta \mid \mathcal{D}) \, d\theta. \tag{2.10}$$

The integration in Equation 2.10 is often intractable, and so we approximate it by the following Monte Carlo estimate:

$$
\begin{aligned}
p(\mathcal{D}^{new} \,|\, \mathcal{D}) &\approx \frac{1}{S}\sum_{s=1}^{S} p(\mathcal{D}^{new} \,|\, \theta^{(s)}), \\
\theta^{(s)} &\sim p(\theta \,|\, \mathcal{D}).
\end{aligned}
$$

The Monte Carlo estimate requires samples from the posterior distribution. The hard part of Monte Carlo estimation is generating samples from a complex, high-dimensional distribution that may not even have an analytic form. Metropolis-Hastings is a particularly flexible approach for generating the required samples.

2.5.1 Metropolis-Hastings

One of the oldest techniques that falls under the rubric of Markov Chain Monte Carlo (MCMC) is the Metropolis-Hastings (MH) algorithm. While it is often difficult to sample directly from a particular target distribution $\pi(\theta)$, we may define a Markov chain whose states are the possible values of θ, and whose steady state distribution is $\pi(\theta)$.[5] The following presentation largely follows Chib and Greenberg [23].

We have assumed that $\theta \in \mathbb{R}^n$, and so a continuous state Markov chain over $\mathbb{R}^n$ must be defined. Let $dy \subseteq \mathbb{R}^n$. We denote the probability of jumping from θ to the infinitesimal region dy as $P(\theta, dy)$.[6] Suppose that we define the transition distribution $P(\cdot, \cdot)$ as

$$
\begin{aligned}
P(\theta, dy) &= p_{MH}(\theta, y)\, dy + r(\theta)\delta_\theta(dy), & (2.11)\\
\delta_\theta(dy) &= \begin{cases} 1, & \text{if } \theta \in dy \\ 0, & \text{otherwise} \end{cases}, \\
p_{MH}(\theta, \theta) &= 0, \\
p_{MH}(\theta, y) &\geq 0, \\
1 &\geq \int p_{MH}(\theta, y)\, dy, & (2.12)\\
r(\theta) &= 1 - \int p_{MH}(\theta, x)\, dx,
\end{aligned}
$$

[5] Most commonly, $\pi(\theta)$ is the posterior distribution $p(\theta \,|\, \mathcal{D})$. However, Metropolis-Hastings can be used to sample from any probability distribution.

[6] $P(\theta, dy)$ is analogous to the transition matrix in a discrete-state Markov chain.

Informally, $p_{MH}(\theta, y)\, dy$ is a continuous density that describes the probability of the random walk moving from θ to dy where $\theta \notin dy$; $r(\theta)$ is the probability that the random walk remains at its current state, θ. Since it is possible that $r(\theta) \neq 0$, $p_{MH}(\theta, y)\, dy$ may not integrate to 1. If the distribution $p_{MH}(\theta, y)$ satisfies a condition known as *detailed balance*, then the continuous state Markov Chain defined by $P(\theta, dy)$ has a unique steady state distribution, which is $\pi(\theta)$. The detailed balance condition on $p_{MH}(\cdot, \cdot)$ is

$$\pi(\theta) p_{MH}(\theta, y) = \pi(y) p_{MH}(y, \theta).$$

In the terminology of Markov chains, $p_{MH}(\theta, y)$ is the probability of transitioning from state θ to state y.

We need a distribution p_{MH} which exhibits detailed balance, and is easy to sample from. Metropolis-Hastings gives us a procedure for creating an appropriate p_{MH} given a distribution over states, $q(\cdot, \cdot)$, which is easy to sample from; but which may not exhibit detailed balance. The easy-to-sample distribution q is often known as the *proposal distribution*.

Let us assume that the proposal distribution, q, has the same support as the distribution we wish to sample from. Next, define a correction function $\alpha(\cdot, \cdot) \in [0, 1]$, which we choose to enforce detailed balance. In Metropolis-Hastings, $p_{MH}(\theta, y) = q(\theta, y)\alpha(\theta, y)$. Therefore, the detailed balance condition is

$$\pi(\theta) p_{MH}(\theta, y) = \pi(y) p_{MH}(y, \theta) \tag{2.13}$$

$$\pi(\theta) q(\theta, y)\alpha(\theta, y) = \pi(y) q(y, \theta)\alpha(y, \theta). \tag{2.14}$$

Given detailed balance, what form should $\alpha(\cdot, \cdot)$ take? Assume that there is a violation of detailed balance where the walk does not move frequently enough from y to θ, i.e. $\pi(\theta)q(\theta, y)\alpha(\theta, y) > \pi(y)q(y, \theta)\alpha(y, \theta)$. To correct this imbalance, it makes sense to pick the largest possible value for $\alpha(y, \theta)$. Since p_{MH} is a probability distribution, $\alpha(\cdot, \cdot) \in [0, 1]$, and so the largest possible values of $\alpha(y, \theta)$ is 1. In the scenario we have described, the detailed balance condition yields the form of $\alpha(\theta, y)$:

$$\begin{aligned} \pi(\theta) q(\theta, y)\alpha(\theta, y) &= \pi(y) q(y, \theta) \underbrace{\alpha(y, \theta)}_{=1} \\ \implies \alpha(\theta, y) &= \frac{\pi(y) q(y, \theta)}{\pi(\theta) q(\theta, y)}. \end{aligned}$$

Notice that we do not even need to be able to evaluate q, just ratios of q—one need not even know the normalizing constant for the proposal. Since $\alpha(\cdot, \cdot)$ must be a probability, we define $\alpha(\theta, y)$ as

$$\alpha(\theta, y) = \min\left[1, \frac{\pi(y)q(y,\theta)}{\pi(\theta)q(\theta,y)}\right].$$

It is straightforward to check that this definition of $\alpha(\theta, y)$ satisfies detailed balance while ensuring that $0 \leq \alpha(\theta, y) \leq 1$. If we start the walk at a state where $\pi(\theta) > 0$, and the support of the proposal contains the support of $\pi(\cdot)$, then the only way $\pi(\theta)q(\theta, y) = 0$ is if $\pi(\theta) = 0$. The probability of accepting a transition into a state where $\pi(\theta) = 0$ is zero, and so $\alpha(\theta, y)$ can be set arbitrarily to 1. Distilling the discussion above, we can express the correction function as

$$\alpha(\theta, y) = \begin{cases} \min\left[1, \frac{\pi(y)q(y,\theta)}{\pi(\theta)q(\theta,y)}\right], & \text{if } \pi(\theta)q(\theta, y) > 0 \\ 1, & \text{otherwise} \end{cases}. \tag{2.15}$$

The correction function $\alpha(\cdot, \cdot)$ is also known as the acceptance probability of a transition in the Markov chain. Given our choice of p_{MH}, we can interpret the Markov chain defined by $P(\theta, dy)$ as follows: if the Markov chain is at state θ, then we transition to a state drawn from proposal $q(\theta, y)$ with probability $\alpha(\theta, y)$. Otherwise, the draw from the proposal is rejected and the Markov chain remains at state θ. Once the random-walk over states has converged to its steady state, the probability of hitting state θ is $\pi(\theta)$.

Metropolis-Hastings simply involves simulating the random-walk defined by $P(\theta, y)$ to generate approximate samples from $\pi(\theta)$. Pseudocode for simulating the random-walk is provided in Algorithm 1. The output is a collection of samples $\{\theta^{(t)}\}_{t=1}^{T}$, which are not guaranteed to be independent. Since unbiased Monte Carlo estimation requires independent samples from $\pi(\theta)$, we filter the collection of samples to minimize serial correlation. First, we throw away the first $t < t_0$ burn-in samples, where t_0 is chosen by the user. Metropolis-Hastings is a random walk computation of the steady state of a Markov chain: it takes time for the walk to converge to its steady state. Among the remaining samples, where $t \geq t_0$, there can still be correlations between $\theta^{(t)}$ and $\theta^{(t+k)}$, especially if k is small. Subsampling simply throws away all but every m^{th} sample, mitigating the effect of serial correlation.[7] After disposing of the burn-in samples, and subsampling, we are left with a smaller set of samples, which we can use to produce estimates of functionals of $\pi(\theta)$.

[7] Subsampling mitigates the effect of serial correlation, but increases the variance of Monte Carlo estimates and increases the amount of sampling. In Markov Chain Monte Carlo, the choice of bias/variance/computation tradeoff is left to the user.

Algorithm 1: Metropolis-Hastings Algorithm.

```
Input: Initial parameters $\theta^{(0)}$; Proposal distribution $p(\theta, y)$; Number of samples $T$.
Output: Collection of samples $\{\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(T)}\}$
for $t = 0 \ldots T-1$ do
    Sample $u \sim \mathcal{U}[0,1]$.
    Sample proposed parameters $\theta^* \sim p(\theta^{(t)}, \cdot)$.
    if $u < \alpha(\theta^{(t)}, \theta^*)$ (see Equation 2.15) then $\theta^{(t+1)} = \theta^*$
    else $\theta^{(t+1)} = \theta^{(t)}$
end
```

Random Walk Metropolis-Hastings

The design of a Metropolis-Hastings algorithm consists largely of the choice of $p(\theta, y)$, the *proposal distribution*. In the case of Random Walk Metropolis-Hastings, the proposal distribution is a multivariate Gaussian distribution:

$$p(\theta, y) = \mathcal{N}(\theta, v \cdot I_{n \times n}),$$

i.e., the mean is the current state in the chain, and the covariance is spherical with variance v, which must be chosen by the modeler. If v is too small, then the Markov chain will take a long time to mix; if v is too large, then $\pi(\theta^*)/\pi(\theta)$ will be small, and so the probability of accepting the proposal will be small. Essentially, we want $p(\theta^{(t)}, \cdot)$ to be a good approximation of $\pi(\theta^{(t)})$. Balancing v between these two concerns often requires substantial human intervention. Moreover, a choice of v that approximates the density around $\pi(\theta^{(0)})$ may not be a good approximation of the density around $\pi(\theta^{(t)})$, $t > 0$.[8] A poor choice of v can force the modeller to accept that the underlying chain will be slow to mix: i.e., large T, slow training; or, that the samples will not be a good representation of $\pi(\theta)$. In Chapter 5, we propose a technique for automatically creating a proposal distribution that dynamically approximates the posterior distribution in hierarchical Bayesian Collective Matrix Factorization.

Block Sampling

If $\theta \in \mathbb{R}^n$ contains many parameters, then defining a good proposal distribution may be difficult. We can partition the state into n variables: $\theta = (\theta_1, \ldots, \theta_n)$. The Metropolis-Hastings transition distribution, Equation 2.11, can be defined on each of the n variables, assuming that the others are fixed. It can be shown that cyclically sampling from

[8] Also, there may be no v that approximates the density very well if it is highly non-spherical.

$p(\theta_i \mid \theta \setminus \theta_i)$, for each $i = 1 \dots n$, yields a transition in a Markov chain whose steady state distribution is also $\pi(\theta)$. Even more generally, we can block θ into B groups of variables, sampling from the conditional distribution of each block given that the rest of the variables are fixed. Blocking reduces the problem of defining a proposal distribution over all the parameters into defining conditional proposal distributions over subsets of the parameters. It is often substantially easier to define these conditional proposal distributions, and it may be easier to sample from them as well.

Gibbs Sampling

Gibbs Sampling is an example of a block Metropolis-Hastings sampler where the conditional sampling distribution of variable θ_i (or block i) is $p(\theta_i \mid \theta \setminus \theta_i) = \pi(\theta_i \mid \theta \setminus \theta_i)$. It can be shown that with this choice of proposal distribution, the acceptance rate $\alpha(\theta, \theta^*) = 1$.

The catch of Gibbs sampling is that we must be able to compute the conditional sampling distribution. If $\pi(\theta)$ is a posterior distribution, then the conditional sampling distribution will involve an integration problem similar to the one in Equation 2.2:

$$p(\theta_i \mid \theta \setminus \theta_i, \mathcal{D}) = \frac{p(\mathcal{D}, \theta \setminus \theta_i \mid \theta_i) p(\theta_i)}{p(\mathcal{D}, \theta \setminus \theta_i)}.$$

If the prior and likelihood are conjugate, then it is straightforward to compute the conditional sampling distributions.

2.5.2 Bayesian Model Averaging

The purpose of the posterior distribution $p(\theta \mid \mathcal{D})$ is the role it plays in Equation 2.10, the posterior predictive distribution. Why should we use the posterior predictive distribution? The posterior predictive distribution is a form of Bayesian Model Averaging [102, 103, 50, 130], whose rationale we discuss in this section.[9]

A substantial source of confusion surrounding Bayesian Model Averaging stems from the name: What is a model? What quantities are being averaged? The word “model” is so overloaded in practice that we prefer the phrase *hypothesis*. A hypothesis $\mathcal{H}$ is a set of probability distributions indexed by parameters θ—each distribution is uniquely identified by a value of θ. A composite hypothesis is a set with more than one probability distribution; a simple hypothesis is a set with exactly one probability distribution. We

[9] Many of the ideas in Bayesian Model Averaging go back to the early 1960s [33]. We recommend Hoeting et al. [50] and Wasserman [130] as tutorial references.

use the standard set notation $h \in \mathcal{H}$ to refer to a particular distribution in the hypothesis; θ is used to denote a variable whose value identifies a particular distribution. The word "model" in Bayesian Model Averaging can refer both to a composite hypothesis and to a simple hypothesis. In this section, we consider the more general form of Bayesian Model Averaging, where model refers to a composite hypothesis; later, we consider another form of Bayesian Model Averaging, where model refers to the distributions within a chosen hypothesis. In Chapter 5 we use the latter form of Bayesian Model Averaging, where models correspond to parameters of a hypothesis.

A simple example of the difference between hypotheses and parameters is described in Wasserman [130]: consider a collection of independent flips of a coins, $\{Y_i\}_{i=1}^n$, $Y_i \in \{0, 1\}$. One hypothesis, $\mathcal{H}_1$, may be that the coin flips are Bernoulli distributed, $p(Y; \theta) = \theta^Y(1 - \theta)^{1-Y}$, where $\theta = 1/2$. Another hypothesis, $\mathcal{H}_2$, may be that the coin flips are Bernoulli distributed where $\theta \neq 1/2$. $\mathcal{H}_1$ is a simple hypothesis; $\mathcal{H}_2$ is a composite hypothesis. To this example we could add other, more exotic, hypotheses: e.g., $\mathcal{H}_3$, that the coin flips are discretizations of draws from a standard normal distribution.

Given a finite collection of hypotheses $\mathcal{H}_1, \ldots, \mathcal{H}_K$,[10] Bayesian Model Averaging proposes that we average our predictions on new data $\mathcal{D}^{new}$ under hypothesis $\mathcal{H}_k$ by the probability of hypothesis $\mathcal{H}_k$ given the training data, $\mathcal{D}$:

$$p(\mathcal{D}^{new} \,|\, \mathcal{D}) = \sum_{k=1}^{K} p(\mathcal{D}^{new} \,|\, \mathcal{H}_k, \mathcal{D}) p(\mathcal{H}_k \,|\, \mathcal{D}). \tag{2.16}$$

By Bayes' rule

$$p(\mathcal{H}_k \,|\, \mathcal{D}) \propto p(\mathcal{D} \,|\, \mathcal{H}_k) p(\mathcal{H}_k).$$

Every hypothesis $\mathcal{H}_k$ is indexed by parameters θ_k, whose value is uncertain. In true Bayesian fashion, we specify a prior over the parameters, $p(\theta_k \,|\, \mathcal{H}_k)$. The prior over parameters allow us to integrate them out:

$$p(\mathcal{D} \,|\, \mathcal{H}_k) = \int p(\mathcal{D} \,|\, \theta_k, \mathcal{H}_k) p(\theta_k \,|\, \mathcal{H}_k) \, d\theta_k. \tag{2.17}$$

We can view $p(\mathcal{D} \,|\, \mathcal{H}_k)$ as a hypothesis-dependent mapping from data sets to a score, the probability of that data set. Figure 2.3 illustrates this evidence mapping for two different hypotheses. Given two hypothesis under which the training data is the most probable

[10] While there are a finite set of hypotheses proposed by the modeller, each hypothesis may contain an infinite set of distributions (c.f., $\mathcal{H}_1$ in the coin flip example, above).

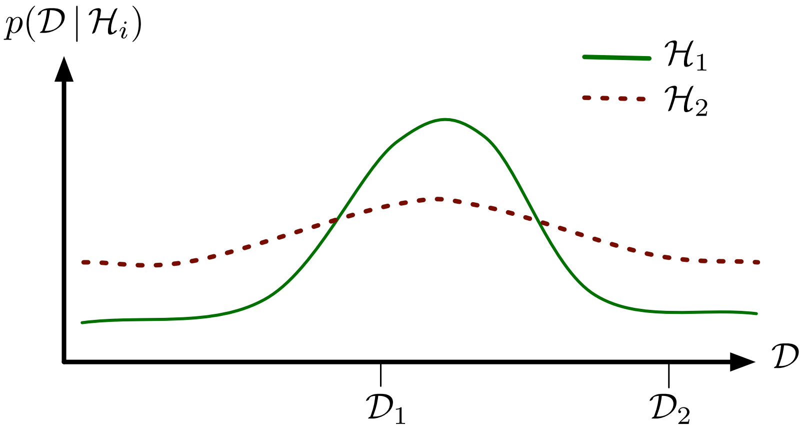


Figure 2.3: An illustration of the evidence given two different hypotheses: $\mathcal{H}_1$ and $\mathcal{H}_2$. The less flexible hypothesis, $\mathcal{H}_1$ models fewer data sets well than the more flexible hypothesis $\mathcal{H}_2$. Two data sets, $\mathcal{D}_1$ and $\mathcal{D}_2$, are marked on the x-axis.

of all possible data sets, we prefer the one that fits fewer data sets well—i.e., the less flexible hypothesis. Typically the more flexible hypothesis is one which contains many more distributions than the other. The larger hypothesis may fit the training data purely by chance, leading to a scenario where the training error is low, but the test error is high [30].

Posterior Predictive Distributions and Bayesian Model Averaging

Much of the ambiguity surrounding Bayesian Model Averaging stems from the fact that there are two averaging operations in the previous section: over hypotheses and over parameters. Colloquially, models can refer to both hypotheses and parameters. If we conflate hypotheses and parameters, then Bayesian Model Averaging is equivalent to integrating out the uncertainty over the parameters.[11]

Let us restrict our consideration to a single composite hypothesis, $\mathcal{H}$, indexed by parameters θ. In this case, the posterior predictive distribution,

$$p(\mathcal{D}^{new}\,|\,\mathcal{D},\mathcal{H}) = \int p(\mathcal{D}^{new}\,|\,\theta,\mathcal{H})p(\theta\,|\,\mathcal{D},\mathcal{H})\,d\theta, \tag{2.18}$$

[11] Conflating hypotheses and parameters has the same effect as assuming that only one hypothesis is possible in the prior.

closely resembles Equation 2.16, where the summation over hypotheses $\mathcal{H}_1, \ldots \mathcal{H}_K$ has been replaced by an integration over the parameters θ.[12] The most important difference between Equations 2.16 and 2.18 is that the former allows for composite hypotheses, which can overlap and vary in size.

To distinguish Equation 2.16 from 2.18 we refer to the latter as Hypothesis-Specific Bayesian Model Averaging. In Hypothesis-Specific Bayesian Model Averaging, the models do not overlap, and they have the same degrees of freedom. While Hypothesis-Specific Bayesian Model Averaging can only make statements vis-á-vis a particular hypothesis, it is not subject to many of the criticisms lobbed at Bayesian Model Averaging over multiple composite hypotheses, which often stem from a poor choice of $H_1, \ldots, H_K$.

In Collective Matrix Factorization, a composite hypothesis corresponds to the structure of a graphical (plate) model. The parameters of the graphical model index the hypothesis. In Chapter 5 we use Hypothesis-Specific Bayesian Model Averaging to integrate out the posterior uncertainty in the parameters of Hierarchical Bayesian Collective Matrix Factorization. We compare the Bayesian approach against the same model whose parameters are selected under maximum a posteriori. The difference between the two approaches is that of model selection (where prediction is done using the maximum a posteriori estimate) and soft model selection (where prediction is done using the posterior predictive distribution).

Model Averaging, Model Combination, and Model Selection

There are at least three procedures involving hypotheses and/or parameters that are often confused for one another:

▷ *Model Averaging*: here we average over different hypotheses, i.e., Equation 2.16; or, we average over different parameters in the same hypothesis, i.e., Equation 2.18.

▷ *Model Combination*: here we take different simple hypotheses $h_1, \ldots, h_r$, and combine them into a new simple, compound, hypothesis $h_{1:r}$. In general $h_{1:r}$ need not be a member of the composite hypotheses from which $h_1, \ldots, h_r$ were selected. The compound hypothesis $h_{1:r}$ may be substantially more complex than any of the composite hypotheses from which $h_1, \ldots, h_r$ were selected. Ensemble methods are a form of model combination where $h_1 \ldots h_r$ are the base learners and $h_{1:r}$ is the ensemble learner. For example, consider the case where $h_1, \ldots, h_r$ are linear discriminants. The weighted vote of linear discriminants, $h_{1:r}$, is not a linear discriminant.

[12] The equation for the posterior predictive distribution here is the same as Equation 2.10, except that we have made the assumption of a hypothesis explicit in the notation.

▷ *Model Selection*: Given a collection of hypotheses, $\mathcal{H}_1, \ldots, \mathcal{H}_K$, choose the most plausible hypothesis given training data. Model comparison is a special case of model selection, where one must determine if there is a statistically significant difference between two hypotheses (c.f., hypothesis testing [65], Bayes factors [56]).[13]

Linguistically, the phrase "model averaging" suggests a combination of models, perhaps by averaging them together. Ensemble methods like bagging have been explicitly compared to Bayesian Model Averaging [31]. However, ensemble methods are a form of model combination; Bayesian Model Averaging is soft model selection [81], model selection that takes the posterior uncertainty into account. Comparing ensemble methods to Bayesian Model Averaging is to draw a false equivalence between model combination and model averaging: both techniques have their merits, but they start from very different premises.[14]

Consider the posterior predictive distribution (Equation 2.18). If asymptotic consistency holds within the assumed composite hypothesis $\mathcal{H}$, then $p(\theta \,|\, \mathcal{D}, \mathcal{H})$ converges to a point mass as the number of observations, the size of $\mathcal{D}$, tends towards infinity. If the posterior distribution over the parameters is a point mass, then we have selected a single distribution. In Hypothesis-Specific Bayesian Model Averaging, a point mass posterior is precisely model selection. Uncertainty in the posterior over θ reflects the inability to distinguish between elements of $\mathcal{H}$. If asymptotic consistency does not hold, or convergence occurs at a very slow rate relative to the size of $\mathcal{D}$, then there can be a substantial advantage to Hypothesis-Specific Bayesian Model Averaging (as we shall see in Chapter 5).[15]

In contrast to Bayesian Model Averaging, ensemble methods involve combining multiple base learners, even as the training data set grows arbitrarily large. While model combination is useful (especially when the base learners are simple) our concern in this dissertation is with model averaging and model selection.

2.5.3 Alternatives to Markov Chain Monte Carlo

Ultimately Bayesian learning is the task of modeling the posterior distribution $p(\theta \,|\, \mathcal{D})$. When the posterior lacks an analytic form, asymptotically exact techniques must resort to

[13] Much of the complexity of both Frequentist and Bayesian hypothesis testing stems from the subtle problem of defining "statistical significance".

[14] In theory, one could take advantage of ensemble methods by including the compound hypothesis formed by the ensemble method into the set of composite hypotheses considered in Equation 2.16.

[15] The same arguments can be applied to Equation 2.16, where the models are composite hypotheses, if one assumes the hypotheses do not overlap.

sampling from the posterior. Markov Chain Monte Carlo is a popular approach for sampling complex, high-dimensional distributions. There are, however, alternatives that approximate the posterior distribution using simpler distributions, whose parameters can be optimized to minimize the discrepancy between the approximation and the true posterior. These techniques are known as *variational methods*. Variational Bayesian Expectation-Maximization [7] is an example of a variational technique which could be applied as an alternative to sample-based Bayesian inference. Unlike sample-based learning, variational methods are asymptotically approximate: with limited computation, they often outperform sample-based techniques; but even with infinite computation, they cannot represent every possible posterior distribution.

In light of the existence of alternatives, the reader may question why we insist on sampling in Chapter 5. When variational methods work, they can produce excellent results at relatively low computational cost. However, when a Bayesian model trained using variational inference fails, it is difficult to assign blame to the model, or to the approximate technique used to train the model. In contrast, with sufficient computation, we can generate enough samples from MCMC to yield an arbitrarily precise finite-sample approximation of the posterior. This allows us to make statements about our model, not just about our model combined with a particular approximate inference algorithm.

Chapter 3

Single Matrix Factorization

3.1 Introduction

The most ubiquitous form of relational data is a single matrix, which represents a mapping from pairs of arguments to a value: `Relation(x,y)` $\rightarrow \mathcal{S} \subset \mathbb{R}$. The rows of the matrix index values of the first argument; columns index values of the second argument. The value of a matrix entry at the row and column corresponding to entities x and y, respectively, is the value of the relation. In first-order logic the values the mapping can take are true/false, $\mathcal{S} = \{0, 1\}$, but we allow for more general mappings—e.g., matrix entries may be real, ordinal, integral, etc.

Matrix data shows up in many different domains. In topic modeling, the input is a data matrix representing the relation `Count(document,word)` $\rightarrow \{0, 1, 2, \ldots\}$, which measures how often a word occurs in a document. In recommendation systems the input is a data matrix representing the relation `Rating(user,item)` $\rightarrow \{1, 2, \ldots, R \in \mathbb{N}\}$, which measures the rating a user assigns to an item on an ordinal scale. In citation analysis, the input is a data matrix representing the relation `Cites(citing,cited)` $\rightarrow \{0, 1\}$, which measures whether a document cites another one.

The most common tasks involving matrix data are entity clustering and prediction. Entity clustering involves grouping rows together by similarity, likewise the columns. Prediction involves inferring the unseen value of a relation given observed values. The unseen value of a relation may involve entities that appeared in the training data, or entities that did not appear in the training data. The latter scenario allows for relational models that generalize to new entities.

An approach common to entity clustering and prediction is low-rank matrix factorization:

Definition 1. *Low-rank matrix factorization is a statistical model that represents an* $m \times n$ *data matrix as a function of the product of two lower-rank factor matrices: an* $m \times k$ *matrix* U*, and a* $n \times k$ *matrix* V*. That is,* $X \approx f(UV^T)$ *for an element-wise transformation* $f : \mathbb{R} \to \mathbb{R}$ *and* $k < \min\{m, n\}$*. The parameters of the model are* (U, V)*.*

Over the years a number of low-rank matrix factorization algorithms have been proposed for applications in text modeling, image analysis, social network analysis, bioinformatics, etc. As a result, we have a panoply of models and no clear notion of what the important differences are between them. This is precisely the situation where a statistical design pattern is of value. Independent of any particular application, we want a generic view of the modeling choices common to variants of low-rank matrix factorization. This chapter presents a statistical design pattern, a unified view, for single matrix factorization. We shall show that the differences between many matrix factorization algorithms can be viewed in terms of a small number of modeling choices. In particular, our unified view places dimensionality reduction methods, such as singular value decomposition [42], into the same framework as matrix co-clustering algorithms like probabilistic latent semantic indexing [52].

A limitation of matrix data is that entities can participate in only one relation. In Chapter 4 we relax this restriction, using a low-rank matrix factorization as a building block for collective matrix factorization, which allows entities to participate in more than one relation.

Main Contributions: The contributions of this chapter are descriptive: we present a statistical design pattern for low-rank matrix factorization which subsumes a wide variety of matrix models in the literature, using only a small set of modeling choices. An advantage of the statistical design pattern view is that it allows us to generalize fast training procedures to many different matrix factorization models. Another advantage is that our unified view reduces the problem of choosing one matrix factorization model from a set of dozens of possible ones to what we believe is a simpler problem: deciding on a much smaller number of modeling choices.

Relationship to Other Chapters: Low-rank matrix factorization is the building block for collective matrix factorization, which extends factoring a single matrix to factoring sets of related matrices, i.e., matrices which share dimensions. While we focus on a particular kind of matrix factorization in later chapters, namely Exponential Family PCA [26], many of the concepts are most easily explained in the context of single-matrix factorization. This chapter focuses on the modeling aspects: a more detailed discussion of algorithms for parameter estimation is deferred to Chapters 4 and 5.

3.2 Singular Value Decomposition

The singular value decomposition (SVD) is the oldest matrix factorization model, tracing its history as an approximation technique to 1907 [122]. SVD can be framed as a bilinear form, where the data matrix X is approximated by the product of low-rank factors U and V, chosen to minimize the squared-loss:

$$\operatorname*{argmin}_{(U,V)\,:\,U^TU=I,V^TV=\Lambda} \sum_{i=1}^{n}\sum_{j=1}^{m}\left(X_{ij}-U_{i\cdot}V_{j\cdot}^T\right)^2. \tag{3.1}$$

The column-orthonormality constraint, $U^TU = I$, and the orthogonality constraint $V^TV = \Lambda$ for diagonal Λ guarantee a unique non-degenerate solution, up to a permutation of the factors. Often, one drops the constraints, noting that any non-degenerate solution (U, V) can be orthogonalized by an invertible linear transform $R \in \mathbb{R}^{k\times k}$: (UR, VR^{-1}) has the same score as (U, V).

We illustrate the major design choices in low-rank matrix factorization by first considering variants of the singular value decomposition:

1. To evaluate Equation 3.1 we must know the value of all the entries of the data matrix X. This obviates the use of SVD for predicting unseen entries of X, but we can still use it for entity clustering (e.g., Latent Semantic Indexing [27]), or for generalizing to new entities. Moreover, in many of the examples of data matrices we considered, one observes only a small fraction of the entries of X: e.g., `Rating(user,item)` is usually only observed for a small fraction of all possible (user, item) pairs. Data weights allow matrix factorizations to work around these problems. Assume we are given $W_{ij} \geq 0$ for each X_{ij}. The loss for weighted SVD [36, 119] is

$$\operatorname*{argmin}_{(U,V)} \sum_{i=1}^{n}\sum_{j=1}^{m} W_{ij}\left(X_{ij}-U_{i\cdot}V_{j\cdot}^T\right)^2. \tag{3.2}$$

   When $W_{ij} = 0$, the corresponding entry of X makes no contribution to the objective, and whatever value is assigned to X_{ij} has no influence on the model. Setting data weights to zero allows for missing values in the data.

2. For SVD, $X \approx UV^T$, where the entries of U and V are arbitrary real numbers. If X_{ij} is a real number, then there is at least a prima-facie case for this model. However,

if X_{ij} contains integral counts, binary values, or ordinal scales, then the model may predict a value which is impossible given the interpretation of a value of X_{ij}. For example, if X_{ij} are true/false (1/0), the model allows for predictions like $\hat{X}_{ij} = 2$. One technique for enforcing constraints on values of a relation is to introduce a prediction link, f, which transforms UV^T into prediction in the data domain. Typically, f is a scalar function applied to each element of UV^T. An example of f is the sigmoid function, which maps the real-line onto the $[0, 1]$ interval.

3. In certain scenarios we wish to assign an interpretation to the low-rank factors. For this purpose, we may impose hard constraints on the factors. A common example is clustering or factor analysis, where each of the k columns of U (likewise V) correspond to a topic or factor, and the entries of a row of U, $(U_{i1}, \ldots, U_{ik})$, correspond to the degree an entity is described by a particular topic or factor. In clustering models, $\sum_k U_{ik} = 1 \wedge U_{ik} \geq 0$, which is a hard constraint on the factor. Such hard constraints can dramatically increase the complexity of parameter estimation.

4. Even in the absence of hard constraints we must acknowledge that matrix factorizations contain a large number of parameters: k parameters for every entity being modelled. The problem is exacerbated by missing values. Adding a regularization penalty to the objective mitigates overfitting. For example, one may choose to add ℓ_2-regularization:

$$\operatorname*{argmin}_{(U,V)} \sum_{i=1}^{n} \sum_{j=1}^{m} \left(X_{ij} - U_{i\cdot} V_{j\cdot}^T\right)^2 + \lambda_U \sum_{i=1}^{n} \sum_{\ell=1}^{k} U_{i\ell}^2 + \lambda_V \sum_{j=1}^{m} \sum_{\ell=1}^{k} V_{j\ell}^2,$$

where $\lambda_U > 0, \lambda_V > 0$ control the relative penalty for large parameters. Another approach to regularization allows k to be large, even larger than $\max\{m, n\}$, but places a penalty on a continuous proxy for the rank of UV^T, the trace-norm [120]:

$$\operatorname*{argmin}_{(U,V)} \sum_{i=1}^{n} \sum_{j=1}^{m} \left(X_{ij} - U_{i\cdot} V_{j\cdot}^T\right)^2 + \lambda \cdot \text{tr}(UV^T).$$

5. Another limitation of the singular value decomposition is the squared-loss objective itself. Even if we allow for nonlinear prediction links, $\hat{X} = f(UV^T)$, the square loss itself may be undesirable. The squared-loss is not robust to outliers: a single, sufficiently large, outlier of X can dominate the model. One approach is to minimize under ℓ_1 loss [57]:

$$\operatorname*{argmin}_{(U,V)} \sum_{i=1}^{n} \sum_{j=1}^{m} \left|X_{ij} - U_{i\cdot} V_{j\cdot}^T\right|.$$

There are many reasons for supporting a variety of measures of reconstruction error, or loss functions, between X and $\hat{X} = f(UV^T)$. We discuss the connection between loss functions and prediction links in Section 3.4.

6. All of the techniques we have discussed are point estimators that fall under the rubric of maximum likelihood or regularized maximum likelihood. A fully Bayesian perspective requires modelling the posterior distribution of the model given the data, $p(U, V \,|\, X)$. Because the majority of the literature is concerned with the maximum likelihood case, we defer a discussion of Bayesian matrix factorization to Section 3.10.

These six modeling choices are not mutually exclusive; the variety of models for low-rank matrix factorization is largely due to researchers exploring the Cartesian product of these modeling choices.

3.3 Data Weights

The great advantage of the singular value decomposition is the existence of a single global optimum, which is easily computed or approximated for large matrices [42]. However, even a seemingly innocuous change to the SVD model can lead to a significantly more difficult optimization. Nowhere is this more apparent than in weighted SVD [119], a.k.a. criss-cross regression [36].

The only difference between SVD and Weighted SVD is the addition of data weights: compare Equations 3.1 and 3.2. Adding data weights significantly complicates training: the objective can have multiple local optima, and one usually resorts to general purpose nonlinear optimization. Essentially all low-rank matrix factorizations, save SVD, can (and typically do) have multiple local optima in their objectives.

3.4 Prediction Links and Matching Losses

The singular value decomposition assumes that each entry of the data matrix is Gaussian, where the corresponding entry in $\Theta = UV^T$ is the natural parameter: $X_{ij} \sim \text{Gaussian}(\Theta_{ij})$. In this section, we discuss the close relationship between the choice of prediction link, the choice of loss function, and distribution assumptions made by low-rank matrix factorizations. We begin by noting that the Gaussian is an exponential family of distributions:

Definition 2. *A parametric family of distributions $\psi_F = \{p_F(x \mid \theta) : \theta\}$ is a regular exponential family if each density p_F can be expressed as the following canonical form:*

$$\log p_F(x \mid \theta) = \log p_0(x) + \langle \theta, x \rangle - F(\theta),$$

where θ is the vector of natural parameters for the distribution, x is the vector of minimal sufficient statistics, and $F(\theta) = \log \int p_0(x) \exp(\langle \theta, x \rangle)\, dx$ is the log-partition function. $p_0(x)$ is a base measure, independent of the parameters. A distribution in ψ_F is uniquely identified by its natural parameters.

The measure of error of a matrix factorization model is usually evaluated in the data space, i.e., using data X and estimates $\hat{X} = f(UV^T)$. Characterizing matrix factorizations as statistical models allows one to measure error in the parameter space, i.e., the likelihood of data X given model $\Theta = UV^T$. In the following section we show that, when $X_{ij} \sim \psi_F(\Theta_{ij} = U_{i\cdot} V_{j\cdot}^T)$, measuring error in data space is equivalent to measuring it in parameter space.

3.4.1 Bregman Divergence

Almost every loss function used in the literature on low-rank matrix factorization is a Bregman divergence. In this section we introduce Bregman divergences, and their relationship to exponential families.

Definition 3 (Generalized Bregman Matrix Divergence [44, 45])**.** *For a closed, proper, convex function $F : \mathbb{R}^{m \times n} \to \mathbb{R}$ the generalized Bregman divergence [44, 45] between matrices Θ and X is*

$$\mathbb{D}_F(\Theta \,||\, X) = F(\Theta) + F^*(X) - X \circ \Theta,$$

where F^* is the convex conjugate, $F^*(\mu) = \sup_{\Theta \in \operatorname{dom} F} [\Theta \circ \mu - F(\Theta)]$.[1]

Definition 4 (Generalized Weighted Bregman Divergence)**.** *For a closed, proper, convex function $F : \mathbb{R} \to \mathbb{R}$ and constant weight matrix $W \in \mathbb{R}_+^{m \times n}$, the generalized weighted Bregman divergence is*

$$\mathbb{D}_F(\Theta \,||\, X, W) = \sum_{ij} W_{ij} \left(F(\Theta_{ij}) + F^*(X_{ij}) - X_{ij}\Theta_{ij} \right).$$

[1] We remind the reader that $A \circ B$ denotes the matrix inner product $\sum_{ij} A_{ij} B_{ij}$.

Definition 5 (Bregman Divergence [15, 20])**.** *For a closed, proper, differentiable convex function $F : \mathbb{R} \to \mathbb{R}$, where the gradient $\nabla F = f$, and $W_{ij} = 1$, the Bregman divergence is*

$$D_{F^*}(X \,||\, f(\Theta)) = \sum_{ij} F^*(X_{ij}) - F^*(f(\Theta_{ij})) - \nabla F^*(f(\Theta_{ij}))(X_{ij} - f(\Theta_{ij})).$$

This definition is a special case of $\mathbb{D}_F(\Theta \,||\, X, W)$.

Each of the Bregman divergences defines a measure of dissimilarity, in some cases a weighted measure, between the parameter matrix Θ and the data matrix X. Definition 3 is included for completeness. The divergences Definition 3 defines are not commonly used in matrix factorization (e.g., von-Neumann divergence [61, 128]). Definition 5 is the standard definition presented in the literature, which assumes that the generating function F is differentiable. Definition 4 is the one we use in this thesis: F is allowed to be non-differentiable, and weighted divergences are required for matrix factorizations that use data weights.

There is a close relationship between Bregman divergences and regular exponential families:

$$\log p_F(x \,|\, \theta) = \log p_F(x) + F^*(x) - D_{F^*}(x \,||\, f(\theta)),$$

where the prediction link $f(\theta) = \nabla F(\theta)$ is known as the matching link for F [3, 5, 26, 35]. The log-partition function which defines the exponential family ψ_F, implicitly defines a prediction link, $f = \nabla F$, and a matching loss function, $D_{F^*}(x \,||\, f(\theta))$. In theory, one could pick the prediction link f such that it does not relate to the log-partition function, F. In practice, the link and the loss almost always match, since this guarantees that the resulting loss is convex in Θ. To emphasize the importance of matching links and losses, our definitions of generalized Bregman divergence force the link and the loss to match. Therefore, any one of the following modelling choices is equivalent to deciding the other two:

1. Choice of regular exponential family, $X_{ij} \sim \psi_F(\Theta_{ij})$.
2. Choice of prediction link, $f = \nabla F$, where $\hat{X} = f(UV^T)$.
3. Choice of matching loss, $\mathbb{D}_F(\Theta \,||\, X, W)$.

Conceptually, it is easiest to decide that the entries of the data matrix are drawn from a particular exponential family: e.g., Gaussian, Poisson, Bernoulli. The prediction link and matching loss follow automatically.

The relationship between matrix factorization and exponential families is made clear by viewing the data matrix as a collection of samples: $\{X_{11}, \ldots, X_{mn}\}$. Let $\Theta = UV^T$ be a matrix of natural parameters. If we assume that $X_{ij} \sim \psi_F(\Theta_{ij} = U_{i\cdot}V_{j\cdot}^T)$, then

$$(U^*, V^*) = \operatorname*{argmin}_{(U,V)} D_{F^*}(X \,||\, f(\Theta)) \tag{3.3}$$

yields the maximum likelihood solution. Collective Matrix Factorization is built on matrix factorizations where the link and loss always match, and data weights are required. Therefore, we prefer framing the maximum likelihood optimization as

$$(U^*, V^*) = \operatorname*{argmin}_{(U,V)} \mathbb{D}_F(X \,||\, UV^T, W). \tag{3.4}$$

The generalization of the singular value decomposition to non-Gaussian data distributions, using Equation 3.3, is known as Exponential Family PCA (E-PCA) [26].

Relationship to Generalized Linear Models

The relationship between SVD and E-PCA is analogous to the relationship between linear regression and generalized linear models (GLMs) [78]. In the regression case, the main difference between the two types of models is that linear regression assumes that the response is Gaussian; generalized linear models assume the response is modelled by a regular exponential family. Both regressions are linear models. Linear regression has a closed form maximum likelihood solution; iteratively-reweighted least squares yields the maximum likelihood solution for GLMs. In the matrix factorization case, the main difference between the SVD model and E-PCA model is that the former assumes the matrix entries are Gaussian; the latter assumes the matrix entries are drawn from a regular exponential family. Both matrix factorizations are bilinear models. In both matrix factorizations, the underlying optimization is non-convex. However, in the case of SVD there is a unique local optimum; in E-PCA the non-convex optimization typically has multiple local optima. While the singular value decomposition can be solved as an eigenvalue problem, even for large sparse matrices, the same cannot be said for E-PCA. Almost every variant of maximum likelihood low-rank matrix factorization, other than SVD, is trained using a general purpose nonlinear optimization (e.g., Expectation-Maximization, Gradient Descent, Newton's Method).

3.5 Parameter Estimation

To recap, all maximum likelihood matrix factorizations of $X \in \mathbb{R}^{m \times n}$ can be differentiated by the following choices:

1. Data weights $W \in \mathbb{R}_{+}^{m \times n}$.
2. Prediction link $f : \mathbb{R}^{m \times n} \to \mathbb{R}^{m \times n}$. The prediction link is usually an element-wise operator on its inputs.
3. Hard constraints on factors, $(U, V) \in \mathcal{C}$. When the constraints for each factor are independent of each other, we denote them $\mathcal{C}_U$ and $\mathcal{C}_V$.
4. Weighted divergence or loss between X and $\hat{X} = f(UV^T)$: $\mathcal{D}(X||\hat{X}, W) \geq 0$.
5. Regularization penalty, $\mathcal{R}(U, V) \geq 0$.

Learning the model $X \approx f(UV^T)$ reduces to the following optimization:

$$(U^*, V^*) = \operatorname*{argmin}_{(U,V) \in \mathcal{C}} \left\{ \mathcal{D}(X||f(UV^T), W) + \mathcal{R}(U, V) \right\}. \tag{3.5}$$

The objective in Equation 3.5 is almost always non-convex in (U, V). Moreover, with the exception of the singular value decomposition, there can be many local optima. The user is forced to resort to a variety of non-linear optimization techniques: gradient descent, conjugate gradients, and expectation-maximization. In certain special cases more esoteric solvers can be applied: non-negative matrix factorization uses a multiplicative update. Trace-norm regularization leads to a semidefinite program.

Alternating projections is an optimization technique we pay significant attention to in later chapters. Matrix factorization is (generalized) bilinear: the model $f(UV^T)$ is linear in U when V is fixed, and vice-versa.[2] This leads to classes of block coordinate descent algorithms where the optimization cycles between updates of the factors:

$$U^{(t)} = \operatorname*{argmin}_{U \in \mathcal{C}(U)} \left\{ \mathcal{D}\left(X||f\left(U\left(V^{(t-1)}\right)^T\right), W\right) + \mathcal{R}\left(U, V^{(t-1)}\right) \right\}, \tag{3.6}$$

$$V^{(t)} = \operatorname*{argmin}_{V \in \mathcal{C}(V)} \left\{ \mathcal{D}\left(X||f\left(U^{(t)}V^T\right), W\right) + \mathcal{R}\left(U^{(t)}, V\right) \right\}. \tag{3.7}$$

Each of Equation 3.6 and 3.7 is known as a projection. Choosing a good projection algorithm is critical to achieving fast convergence to a local optimum.

[2] The phrase *bilinear* refers to the form of the parameters. Even if the prediction link f is a nonlinear function, UV^T remains a bilinear form. Our use of “bilinear” is analogous to the use of “linear” in linear models, such as generalized linear models and wavelets, where linearity refers to the basis.

3.5.1 Structure in Alternating Projections

Low-rank matrix factorization contains rich independence structure among the parameters, almost none of which is taken advantage of by general purpose solvers such as gradient descent. One example of structure in matrix factorization is *decomposability* of the loss function, namely the fact that the error between X and $\hat{X} = f(UV^T)$ can be expressed as the sum of the errors between entries X_{ij} and $\hat{X}_{ij}$. All of the losses in Table 3.1 are decomposable.

The advantage of decomposable losses in alternating projection is that when the one of the factors U or V is fixed, the update over the other factor can be decomposed into parallel updates over each row of the factor. If we can evaluate the loss, the regularizer, and the hard constraints on rows of each factor, then the per-factor projections in Equations 3.6-3.7 can be reduced into parallel projections over the rows of each factor:

$$\forall i = 1 \dots n : \; U_{i\cdot}^{(t)} = \operatorname*{argmin}_{U_{i\cdot} \in \mathcal{C}(U_{i\cdot})} \left\{ \mathcal{D}(X_{i\cdot} \,||\, f\left(U_{i\cdot}\left(V^{(t-1)}\right)^T, W\right) + \mathcal{R}\left(U_{i\cdot}, V^{(t-1)}\right) \right\}, \tag{3.8}$$

$$\forall j = 1 \dots m : \; V_{j\cdot}^{(t)} = \operatorname*{argmin}_{V_{j\cdot} \in \mathcal{C}(V_{j\cdot})} \left\{ \mathcal{D}\left(X_{\cdot j} \,||\, f\left(U^{(t)} V_{j\cdot}^T, W\right)\right) + \mathcal{R}\left(U^{(t)}, V_{j\cdot}\right) \right\}. \tag{3.9}$$

These operations are parallel in the sense that the $\forall i$ and $\forall j$ statements can be evaluated as a parallel-for loop. In essence, we have broken down a projection over mk (or nk) parameters into m (or n) separate projections over k parameters. For many choices of loss $\mathcal{D}$, constraints $\mathcal{C}$, and regularizer $\mathcal{R}$, these projections correspond to widely used linear models. For example, under unweighted squared-loss, with no hard constraints, and ℓ_2 regularization the per-row projections correspond to ℓ_2-regularized linear regression where the fixed factor acts as the covariate. In many cases, Equations 3.8-3.9 are convex optimizations. ℓ_2-regularized E-PCA is an example where the projections are convex. The per-row updates can be framed as single steps of Newton-Raphson, as in Gordon [45].

3.5.2 Row-Column Exchangeability

The reader may rightly question why decomposable losses are so common in low-rank matrix factorization. Certainly, there are computational advantages to decomposability. However, there is also a case to be made for such losses independent of any computational considerations. The following presentation is based on [2, 51].

Let us view the data matrix $X_{m \times n}$ as a realization of any underlying matrix $Z_{m \times n}$ plus a noise matrix $E_{m \times n}$, where $Z = f(UV^T)$. Z is the noise-free matrix underlying X, and its entries can be viewed as random variables. We say that the random variables $\{Z_{ij}\}$ are row-column exchangeable (or, Aldous exchangeable), if the likelihood of Z is invariant to permutations of the rows and columns:

$$p(Z_{11} \dots Z_{mn} \,|\, X_{11} \dots X_{mn}) = p(Z_{\pi(1)\tilde{\pi}(1)} \dots Z_{\pi(m)\tilde{\pi}(n)} \,|\, X_{\pi(1)\tilde{\pi}(1)} \dots X_{\pi(m)\tilde{\pi}(n)}), \tag{3.10}$$

where π is a permutation of the rows of X and Z, and likewise, $\tilde{\pi}$ is an independent permutation of the columns of X and Z. Colloquially, Aldous exchangeability means that the identity of a row (or column) contains no information about Z, and that the relative position between two rows (or two columns) contains no information about Z. Aldous exchangeability is an extension of de Finetti exchangeability to matrices of random variables, and like de Finetti exchangeability, it leads to a representation theorem:

Theorem 1 (Aldous' Theorem [1])**.** *If Z is row-column exchangeable, then there exists a function g and independent uniformly distributed random variables μ, $\{u_1, \dots, u_m\}$, $\{v_1, \dots, v_n\}$, and $\{\varepsilon_{ij}\}$ such that*

$$p(Z_{ij}) = g(\mu, u_i, v_j, \varepsilon_{ij}).$$

A consequence of Theorem 1 is that any statistical model of a row-column exchangeable matrix can be parameterized by a global effect μ, a row effect u_i, a column effect v_j, and dyadic effect ε_{ij}. Moreover, the dyadic effects are independent of one other, which naturally leads to decomposable objectives for matrix factorization. It should be noted that Theorem 1 is descriptive, not prescriptive (just like de Finetti exchangeability): no particular form of g is implied, and so this theorem provides no especial justification for the bilinear form $f(UV^T)$.

If we consider some of the applications of matrix factorization, row-column exchangeability is a reasonable assumption. In text modelling, each row of the matrix is a document, each column a word, and each entry a word frequency count. The relative position of two documents, or two words, in the matrix is arbitrary. If we permute the rows of X and the rows of U in the same manner, the likelihood of the model does not change. In collaborative filtering, the relative position of users and items in the data matrix is not informative. Row-column exchangeability is less plausible when a dimension corresponds to time, or a spatially-varying quantity. The modeler may choose to ignore non-exchangeability in the data matrix to reap the computational advantages of decomposability.

3.6 Regularization

The objective in Equation 3.4, like many low-rank matrix factorization losses, will overfit. The number of parameters grows with the number of entities, and amount of data available for each entity can be bounded, e.g., when new rows can be added to the matrix, but the set of columns is fixed. As a result, there is the need for regularization. We typically use a separate regularizer for each factor, so $\mathcal{R}(U, V) = \mathcal{R}(U) + \mathcal{R}(V)$. The most common matrix regularizers are simple extensions of those used in regression, namely ℓ_p regularizers: $\mathcal{R}(U) \propto \lambda \sum_{ij} |u_{ij}|^p$, where λ controls the strength of the regularizer. From a computational perspective, ℓ_p regularizers have the advantage of preserving decomposability of the objective:

$$\mathcal{R}(U) = \sum_{ij} |U_{ij}|^p = \sum_i \sum_j |U_{ij}|^p = \sum_i \mathcal{R}(U_{i\cdot}).$$

In this thesis we use ℓ_2-regularization: $\mathcal{R}(U) = \lambda ||U||_{Fro}^2/2$. ℓ_2 regularizers have the merits of being differentiable, decomposable, and easily related to a Gaussian prior in the Bayesian matrix factorization.

In contrast, consider a non-decomposable regularizer: trace-norm tr(UV^T). Any optimization using the trace norm cannot be easily decomposed, since the trace-norm depends on the product of U and V. In fact, if we consider the difference between max-margin matrix factorization [120] and its fast analogue [104], the only difference in the models is that the fast analogue replaces trace-norm regularization with ℓ_2-regularization, allowing a hard-to-solve semidefinite program to be replaced by a straightforward nonlinear optimization.

As with linear models, one may prefer to use a sparse ℓ_1 regularizer, which can be reduced to an inequality constraint on each row of the factors: $||U_{i\cdot}^{(r)}||_1 \leq t_i$. Since the ℓ_1-regularizer is decomposable, we can use Equations 3.8-3.9, which leads to an ℓ_1-regularized regression problem. When the $\mathcal{D}$ is a Bregman divergence, the per-row projections are simply ℓ_1-regularized generalized linear models. Therefore, we can exploit a variety of approaches for ℓ_1-regularized regression in matrix factorization. We refer the reader to Schmidt et al. [113] for a survey of techniques for ℓ_1-regularized regression.

3.7 Constraints

Inequality constraints turn the projection into a constrained optimization. ℓ_1-regularization is one example of inequality constraints. Non-negative matrix factorization is another example of such constraints, where the commonly-used multiplicative updates guarantee that the solution is feasible at each iteration. Linear constraints can be used to place matrix co-clustering into the same statistical design pattern as matrix factorization. With no constraints on the factors, matrix factorization can be viewed as factor analysis: an increase in the influence of one latent variable (column of U or V) does not require a decrease in the influence of the other latents. In clustering the stochastic constraints,

$$\forall i \; \sum_j U_{ij} = 1 \wedge \forall i \forall j \; U_{ij} \geq 0,$$
$$\forall j \; \sum_i V_{ij} = 1 \wedge \forall i \forall j \; V_{ij} \geq 0,$$

imply that each row (column) of the data matrix must belong to one of k latent clusters, each corresponding to a column of U (V), where $U_{i\cdot}$ $(V_{j\cdot})$ is a distribution over cluster membership for that entity. In matrix co-clustering, stochastic constraints are placed on both factors, since the goal is to simultaneously cluster both rows and columns.

3.7.1 Bias Terms

Aldous' theorem (Theorem 1) suggests that there may be an advantage to modeling per-row and per-column behaviour. For example, in collaborative filtering, bias terms can calibrate for a user's mean rating. A straightforward way to account for bias is to append an extra column of parameters to U paired with a constant column in V: $\tilde{U} = [U \; u_{k+1}]$ and $\tilde{V} = [V \; \mathbf{1}]$. We do not regularize the bias. It is equally straightforward to allow for bias terms on both rows and columns: $\tilde{U} = [U \; u_{k+1} \; \mathbf{1}]$ and $\tilde{V} = [V \; \mathbf{1} \; v_{k+1}]$, and so $\tilde{U}\tilde{V}^T = (UV^T) + u_{k+1}\mathbf{1}^T + \mathbf{1}v_{k+1}^T$. Note that these are biases in the space of natural parameters, a special case being a margin in the hinge or logistic loss—*e.g.*, the per-row (per-user, per-rating) margins in MMMF [104] are just row biases.

3.8 Models subsumed by the unified view

The beauty of reducing maximum likelihood low-rank matrix factorization into a statistical design pattern is that we can reduce the complexity of choosing between many different models into five modeling choices. Table 3.1 lists a few common instances of low-rank matrix factorization, along with how they can be reduced into a small number of modeling choices. As we will see in Section 3.11, these dimensions of variability provide the basis for a statistical design pattern.

Table 3.1: Single matrix factorization models. $\operatorname{dom} X_{ij}$ describes the types of values allowed in the data matrix. Unweighted matrix factorizations are denoted $W_{ij} = 1$. If constraints or regularizers are not used, the entry is marked with a em-dash. $A \circ B$ denotes the matrix inner product: tr$(A^T B)$. $A \odot B$ denotes the element-wise (Hadamard) product.

Method	$\operatorname{dom} X_{ij}$	Link $f(\theta)$	Loss $\mathcal{D}(X \vert \hat{X} = f(\Theta), W)$	W_{ij}
SVD [42]	$\mathbb{R}$	θ	$\vert\vert W \odot (X - \hat{X}) \vert\vert_{Fro}^2$	1
W-SVD [36, 119]	$\mathbb{R}$	θ	$\vert\vert W \odot (X - \hat{X}) \vert\vert_{Fro}^2$	≥ 0
k-means [48]	$\mathbb{R}$	θ	$\vert\vert W \odot (X - \hat{X}) \vert\vert_{Fro}^2$	1
k-medians	$\mathbb{R}$	θ	$\sum_{ij} \vert W_{ij} \left(X_{ij} - \hat{X}_{ij} \right) \vert$	1
ℓ_1-SVD [57]	$\mathbb{R}$	θ	$\sum_{ij} \vert W_{ij} \left(X_{ij} - \hat{X}_{ij} \right) \vert$	≥ 0
pLSI [52]	$\mathbf{1} \circ X = 1$	θ	$\sum_{ij} W_{ij} \left(X_{ij} \log \frac{X_{ij}}{\hat{X}_{ij}} \right)$	1
NMF [63]	$\mathbb{R}_+$	θ	$\sum_{ij} W_{ij} \left(X_{ij} \log \frac{X_{ij}}{\Theta_{ij}} + \Theta_{ij} - X_{ij} \right)$	1
ℓ_2-NMF [91, 63]	$\mathbb{R}_+$	θ	$\vert\vert W \odot (X - \hat{X}) \vert\vert_{Fro}^2$	1
Logistic PCA [112]	$\{0, 1\}$	$(1 + e^{-\theta})^{-1}$	$\sum_{ij} W_{ij} \left(X_{ij} \log \frac{X_{ij}}{\hat{X}_{ij}} + (1 - X_{ij}) \log \frac{1 - X_{ij}}{1 - \hat{X}_{ij}} \right)$	1
E-PCA [26]	many	many	decomposable Bregman (D_F)	1
G^{2}L^{2}M [45]	many	many	decomposable Bregman $(\mathbb{D}_F)$	1
MMMF [120]	$\{0, \ldots, R\}$	min-loss	$\sum_{r=1}^{R-1} \sum_{ij: X_{ij} \neq 0} W_{ij} \cdot h(\Theta_{ij} - B_{ir})$	1
Fast-MMMF [104]	$\{0, \ldots, R\}$	min-loss	$\sum_{r=1}^{R-1} \sum_{ij: X_{ij} \neq 0} W_{ij} \cdot h_\gamma(\Theta_{ij} - B_{ir})$	1

Method	Constraints U	Constraints V	Regularizer $\mathcal{R}(U, V)$	Algorithm(s)
SVD	$U^T U = I$ can be applied post-hoc	$V^T V = \Lambda$ can be applied post-hoc	—	Gaussian Elimination, Power Method
W-SVD	—	—	—	Gradient, EM
k-means	—	$V^T V = I$ $V_{ij} \geq 0$	—	EM
k-medians	—	$V^T V = I$ $V_{ij} \geq 0$	—	Alternating
ℓ_1-SVD	—	—	—	Alternating
pLSI	$\mathbf{1}^T U \mathbf{1} = 1$ $U_{ij} \geq 0$	$\mathbf{1}^T V = \mathbf{1}$ $V_{ij} \geq 0$	—	EM
NMF	$U_{ij} \geq 0$	$V_{ij} \geq 0$	—	Multiplicative
ℓ_2-NMF	$U_{ij} \geq 0$	$V_{ij} \geq 0$	—	Multiplicative, Alternating
Logistic PCA	—	—	—	EM
E-PCA	—	—	—	Alternating
G^{2}L^{2}M	—	—	decomposable Bregman $\mathbb{D}_F(\cdot \vert\vert U) + \mathbb{D}_F(\cdot \vert\vert V)$	Alternating (Subgradient, Newton)
MMMF	—	—	$tr(UV^T)$	Semidefinite Program
Fast-MMMF	—	—	$\frac{1}{2}(\vert\vert U \vert\vert_{Fro}^2 + \vert\vert V \vert\vert_{Fro}^2)$	Conjugate Gradient

In light of Equations 3.8-3.9, a great many matrix factorizations can be seen as extensions of linear models to bilinear forms:

- ▷ SVD is the matrix analogue of linear regression.
- ▷ E-PCA is the matrix analogue of generalized linear models.
- ▷ G^{2}L^{2}M is the matrix analogue of a regularized generalized linear model.
- ▷ MMMF is the matrix analogue of ordinal regression under hinge loss.
- ▷ Fast-MMMF is the matrix analogue of ordinal regression under a smooth alternative to hinge loss, h_γ (e.g., a smoothed hinge loss, or logistic loss).
- ▷ k-medians is the matrix analogue of quantile regression [58], where the quantile of interest is the median.
- ▷ Robust SVD [69] is the matrix analogue of the LASSO [126].

The key difference between a regression/clustering algorithm and its matrix analogue is that the exchangeability (or strong iid) assumption is replaced by row-column exchangeability.

Another useful aspect of our unified view is that it is relatively easy to capture similarities between matrix factorizations. For example, the relationship between pLSI and NMF, first derived by [28], becomes straightforward when one compares the two algorithms in Table 3.1. The prediction link is the identity function in pLSI and NMF. The loss for pLSI is KL-divergence, but unnormalized KL-divergence for NMF. The two losses differ by an additive $\sum_{ij} \Theta_{ij} - X_{ij}$ factor. In NMF, if we constrain the entries of X to sum to one, and likewise constrain the entries of Θ to sum to one, then NMF is the same as pLSI. Adding an orthogonality constraint to ℓ_2-NMF yields a soft clustering variant of k-means.

The careful application of constraints can be used to fit clustering algorithms into our framework: Orthogonality of the column factors $V^TV = I$ along with integrality of V_{ij} corresponds to hard clustering the columns of X, as at most one entry in $V_{i\cdot}$ can be nonzero. In the k-means algorithm, U contains the cluster memberships, and V is a normalized version of the cluster centroids [29]. The rank of the decomposition and the number of clusters in k-means is k. Alternating projections corresponds to the classical approach to clustering: updating U corresponds to assigning points to clusters; updating V corresponds to updating the cluster centroids.

Max-margin matrix factorization is one of the more elaborate models: ordinal ratings $\{1, \ldots, R\}$[3] are modeled using $R-1$ parallel separating hyperplanes, corresponding to the binary decisions $X_{ij} \leq 1, X_{ij} \leq 2, X_{ij} \leq 3, \ldots, X_{ij} \leq R-1$. The per-row bias term B_{ir} allows the distance between hyperplanes to differ for each row. Since this technique was conceived for user-item matrices, the biases capture differences in each user. Predictions are made by choosing the value which minimizes the loss of the $R-1$ decision boundaries, which yields a number in $\{1, \ldots, R\}$ instead of $\mathbb{R}$.

3.8.1 Extensions of the Unified View

Our unified view assumes a bilinear form, UV^T, which is common in the literature on matrix factorization. However, there is little reason why we cannot extend our consideration of low-rank matrix factorization to multilinear forms: e.g., UAV^T, where U and V can have different numbers of factors, and A is a transformation between low-rank subspaces. Moving from a bilinear to a multilinear form is essentially a change in the structure of the underlying probabilistic graphical model: the same five choices can be used to differentiate models that share a particular multilinear form.

Moving towards a multilinear form naturally leads one to consider tensor factorization: e.g., UAV^T is a special case of Tucker decomposition [129] on a 2D-tensor, a matrix. Our five modeling choices can also be used to differentiate tensor factorizations, but the choices may be subtler for tensors than for matrices. For example, sparsity constraints on the bilinear form are a generalization of sparse regression regularizers; sparsity constraints in a multilinear form are often more complex (c.f., CUR-decomposition [34]: for the multilinear form UAV^T, U and V are sparse matrices, but A is dense).

3.9 Limitations of Maximum Likelihood Approach

All the matrix factorizations we have focused on are estimated under (regularized) maximum likelihood. The standard Bayesian objections to maximum likelihood estimation apply. There are many plausible models, where plausibility is measured by a model's probability under $p(U, V \mid X, W)$. Predictions should be made by considering the posterior predictive distribution, the expectation of the quantity we wish to predict under $p(U, V \mid X, W)$. The posterior predictive distribution is an example of model averaging, where every model, weighted by its posterior probability, contributes to the prediction.

[3] Zeros in the matrix are considered missing values, and are assigned zero weight.

Since maximum likelihood chooses a single point (U^*, V^*), there is no model averaging, which leads to higher variance in prediction. Moreover, point estimators ignore uncertainty in the model parameters (U, V).

In the case of low-rank matrix factorization there is a more subtle argument against maximum likelihood solutions: they define a generative distribution for entries of a matrix, but not for rows or columns of a matrix. As a result, maximum likelihood matrix factorizations are statistically ill-defined when the goal is to predict on new rows or columns of the data matrix, which did not appear in the training data. This is particularly troubling when rows and columns of the matrix correspond to grounding of a relation: maximum likelihood matrix factorization can incorrectly define the probability of a new entity participating in the relationship. Following the conventions of the literature on collaborative filtering, we refer to this problem as *strong generalization*.

The theoretical limits of maximum likelihood matrix factorization for strong generalization were first pointed out by Welling et al. [131]. We present a detailed explanation of the argument of Welling et al. [131], applying it to matrix factorizations not considered in that paper. Our goal is to derive the form for common maximum likelihood factorizations from a statistical viewpoint, beginning with the likelihood of the data.

The most elementary premise we can make is that there is a row-column exchangeable data matrix X, the factors are U and V, and there are no data weights. Any row-column exchangeable matrix can be viewed as a set of exchangeable rows $\{X_{i\cdot}\}_i$, or as a set of exchangeable columns $\{X_{\cdot j}\}_j$. We consider the case of exchangeable rows; the derivation for exchangeable columns is similar.

Below, we work through the derivation for the maximum likelihood method for predicting the distribution of a new row of X. As will be seen, this derivation contains a critical flaw, rendering predictions on this new row suspect. The flaw in the derivation may explain why matrix factorization models tend to have poor predictive power on strong generalization tasks. While there may exist other, less problematic, derivations for strong generalization on matrix models, none are currently known.

Let $\nu = \{V, \mu_V, \Sigma_V\}$ contain the factor V and the parameters of the prior over V:

$$p(V) = \prod_{j=1}^{n} p(V_{j\cdot} \mid \mu_V, \Sigma_V),$$

where (μ_V, Σ_V) are the parameters of a k-dimensional Gaussian prior over each row of V. Let $\gamma = \{\mu_U, \Sigma_U\}$ contain the prior parameters of a similar k-dimensional Gaussian prior over each row of U. Let $\phi = \{\nu, \gamma\}$ be the set of parameters. Ultimately, we will optimize a function of ϕ. Using the notation we have introduced:

$$\begin{aligned} p_\phi(X, U) &= \prod_{i=1}^{m} p_\phi(X_{i\cdot}, U_{i\cdot}), \\ &= \prod_{i=1}^{m} p_\nu(X_{i\cdot} \,|\, U_{i\cdot}) p_\gamma(U_{i\cdot}). \end{aligned}$$

There is still a large gap between $p_\phi(X, U)$ and anything recognizable as a matrix factorization. The following variational EM derivation bridges the gap. Consider the log-likelihood of a row of the data matrix, $X_{i\cdot}$:

$$\begin{aligned} \log p_\phi(X_{i\cdot}) &= \log \int p_\phi(X_{i\cdot}, U_{i\cdot}) \, dU_{i\cdot}, \\ &= \log \int_{U_{i\cdot}} q(U_{i\cdot} \,|\, X_{i\cdot}) \frac{p_\phi(X_{i\cdot}, U_{i\cdot})}{q(U_{i\cdot} \,|\, X_{i\cdot})}, \\ &\geq \int_{U_{i\cdot}} \log q(U_{i\cdot}|X_{i\cdot}) \frac{p_\phi(X_{i\cdot}, U_{i\cdot})}{q(U_{i\cdot} \,|\, X_{i\cdot})}, \qquad (3.11) \\ &= \underbrace{\int_{U_{i\cdot}} q(U_{i\cdot}|X_{i\cdot}) \log p_\phi(X_{i\cdot}, U_{i\cdot})}_{B(\phi)} - \underbrace{\int_{U_{i\cdot}} q(U_{i\cdot}|X_{i\cdot}) \log q(U_{i\cdot}|X_{i\cdot})}_{-H(q)}. \qquad (3.12) \end{aligned}$$

Equation 3.11 follows from Jensen's inequality. The value of the entropy of the variational distribution $H(q)$ is critical to this argument.

The variational distribution q is an approximation to the true posterior over latents, $p(U_{i\cdot} \,|\, X_{i\cdot})$. We are free to set q to whatever we choose. For reasons which will become clear below, we choose q to be

$$\begin{aligned} q(U_{i\cdot} \,|\, X_{i\cdot}) &= \prod_{\ell=1}^{k} \delta(U_{ik} - \hat{U}_{ik}), \qquad (3.13) \\ \delta(u) &= \begin{cases} 1, & u \neq 0, \\ 0, & \text{otherwise} \end{cases}, \end{aligned}$$

where $\{\hat{U}_{ik}\}_{i,k}$ are variational parameters, which must be optimized to achieve a tight lower bound on the data likelihood $\log p_\phi(X_{i\cdot})$. We shall need to consider U_{ik} under the assumption that it is discrete, and later that it is continuous. To simplify our equations we denote both summation and integration over U using $\sum$. Likewise we use δ to denote both a 0-1 indicator and a Dirac delta. If U_{ik} were discrete, then $H(q)$ would be zero. For now, let us pretend that $H(q)$ is also zero when U_{ik} is continuous. Under our assumption that $H(q)$ is zero, $B(\phi)$ is a lower bound on the log-likelihood. Substituting 3.13 into $B(\phi)$:

$$\begin{aligned}\log p_\phi(X_{i\cdot}) &\geq B(\phi)\\ &= \sum_{U_{i\cdot}} \prod_{\ell=1}^{k} \delta(U_{i\ell} - \hat{U}_{i\ell}) \log p_\phi(X_{i\cdot} \,|\, U_{i\cdot})\\ &= \sum_{U_{i\cdot}} \prod_{\ell=1}^{k} \delta(U_{i\ell} - \hat{U}_{i\ell}) \log \left[p_\nu(X_{i\cdot} \,|\, U_{i\cdot}) p_\gamma(U_{i\cdot})\right]\\ &= \log p_\nu(X_{i\cdot} \,|\, \hat{U}_{i\cdot}) + \log p_\gamma(\hat{U}_{i\cdot}). \qquad (3.14)\end{aligned}$$

Summing Equation 3.14 over all the rows of X yields the following lower bound:

$$\log p(X) \geq \mathcal{O}(\phi, \hat{U}) = \sum_i \log \left[p_\nu(X_{i\cdot} \,|\, \hat{U}_{i\cdot})\right] + \sum_i \log \left[p_\gamma(\hat{U}_{i\cdot})\right]. \qquad (3.15)$$

The reason for our choice of q becomes clear: optimizing the variational lower bound reduces to alternating optimization over the factors. Alternating optimization consists of optimizing $\mathcal{O}$ by coordinate descent: cyclically optimize over $\hat{U}$ (a.k.a. the variational E-step) and ϕ (a.k.a. the variational M-step) till convergence. The variational parameters $\hat{U}$ are a point estimate of the U factor, and ϕ contains the point estimate of the V factor. The first summation in $\mathcal{O}$ can be viewed as a loss function, with the second summation acting as a regularizer over the U factor.

The defect in the derivation of $\mathcal{O}$ from $p(X)$ is that it assumes the entropy of the variational distribution $H(q) = 0$. While this is true when the latent variables U are discrete, this is not true when the latent variables are continuous. It can be shown that if $U_{i\ell}$ is continuous and unbounded, or continuous with the stochastic constraint $\sum_\ell U_{i\ell} = 1 \wedge U_{i\ell} \geq 0$, then $H(q) = -\infty$. Many of the matrix factorizations in Table 3.1 are ones where $H(q) = -\infty$. When $H(q)$ is not finite, Equation 3.15 is false, and there is no reason to believe that optimizing the lower bound will yield a good low-rank representation for a new row of X.

What we have provided is an argument that indicates how maximum likelihood estimation can go wrong on folding-in. The severity of the problem depends on the matrix factorization model and the data. As we shall see in the next chapter, a Bayesian matrix factorization is never subject to the folding-in problem.

3.10 Bayesian Matrix Factorization

The discussion thus far has centered on regularized maximum likelihood estimation, which suffers from the limitations discussed in Section 3.9. Bayesian Matrix Factorizations do not suffer from the aforementioned limitations. In this section, we briefly discuss how our unified view of matrix factorization relates to Bayesian models of matrix data. We explore Bayesian matrix factorization in further detail in Chapter 5.

The signature difference of Bayesian matrix factorization is that we model the entire posterior distribution of the parameters:

$$(U, V) \sim p(U, V \mid X, W), \tag{3.16}$$

$$p(U, V \mid X, W) = \frac{p(X \mid U, V, W)p(U, V \mid W)}{\int p(X \mid U, V, W)p(U, V \mid W)\, dU dV}. \tag{3.17}$$

The posterior distribution models uncertainty in our parameter estimates. In the previous section, we discussed how maximum likelihood matrix factorizations can lead to predictions inconsistent with the joint distribution of the parameters and data. In contrast, the posterior distribution, Equation 3.17, is consistent with the joint distribution. The criticism of Welling et al. [131] does not apply to Bayesian matrix factorization, which correctly defines a generative distribution over rows and columns.

At first glance, Equation 3.17 bears little semblance to the objective we optimize in maximum likelihood matrix factorization: Equation 3.5. However, consider the problem of finding the posterior mode:

$$(U^*, V^*) = \operatorname*{argmax}_{(U,V)} \left(\log p(X \mid U, V, W) + \log p(U, V \mid W) + c \right),$$

$$c = \log \int p(X \mid U, V, W)p(U, V \mid W)\, dU dV,$$

where c is a constant that does not vary with the parameters. Recalling the relationship between regular exponential families and Bregman divergences, it is easy to show that $\log p(X \mid U, V, W)$ is equivalent to the losses we consider in maximum likelihood matrix factorization. Moreover, if $p(U, V \mid W) = p(U)p(V)$ then the log-prior acts as a regularizer over each factor. For example, if $p(U) = \prod_i p(U_{i\cdot})$, and $U_{i\cdot}$ is a multivariate Gaussian with diagonal covariance, then $\log p(U)$ is equivalent to ℓ_2-regularization of U. In many cases, the optimization Equation 3.5 corresponds to finding the mode of the posterior distribution in Equation 3.17. All the matrix factorizations in Table 3.1 can be generalized to the Bayesian case, but there is no guarantee that sampling from the posterior is easy.

A common approach for sampling from the posterior is Markov Chain Monte Carlo (MCMC), where one cyclically samples over parameters in such a manner that the distribution of samples is, eventually, the posterior. To efficiently sample over subsets of the parameters, one often has to restrict the permissible combinations of likelihood and prior. For example, Bayesian Probabilistic Matrix Factorization [109] assumes that the likelihood and prior are Gaussian. Discrete Component Analysis [19] also makes restrictive conjugacy assumptions on the combination of likelihood and prior. One of the contributions of our Bayesian extension to Collective Matrix Factorization is an adaptive MCMC algorithm that is computationally efficient without restricting the permissible combinations of likelihood and prior. In the single-matrix scenario, another approach that does MCMC for Bayesian Matrix Factorization without conjugacy is Bayesian Exponential Family PCA [84]. Our approach uses both the gradient and partial Hessian of the objective; Bayesian Exponential Family PCA uses only the gradient.

Related to Bayesian inference in matrix factorization is the matter of hierarchical priors. Just as de Finetti exchangeability motivates hierarchical Bayesian modeling, row-column exchangeability (esp. Aldous' theorem) provides a theoretical foundation for hierarchical priors on U and V. Hierarchical priors have the effect of pooling information across the rows and columns. Pooling information across rows can greatly alleviate sparsity in the training data. If a matrix is row-column exchangeable, then the distribution of an entry depends on global effects, row effects, column effects, and dyadic effects. A hierarchical prior on U allows one to learn how the global effect influences the prediction for a particular entry (likewise V). We further discuss the use of hierarchical priors in Chapter 5.

3.11 Matrix Factorization as a Statistical Design Pattern

A significant part of software design patterns is the formalization of the descriptive process. Gamma et al. [37] lists the components of a design pattern; we discuss how they relate to a statistical design pattern, using low-rank matrix factorization as our example.

- ▷ **Pattern name and classification**: "The pattern's name conveys the essence of the pattern succinctly". Low-rank matrix factorization conveys the essence of the problem, and is a terminology widely used in the statistics and machine learning community.

- ▷ **Intent**: "What does the design do? What is its rationale and intent? What particular design problems does it address?". Low rank-matrix factorization represents an $m \times n$ data matrix X as a function of the product of two lower-rank factors $U_{m\times k}$ and $V_{n\times k}$.

That is, $X \approx f(UV^T)$. The purposes of this pattern are (i) predicting unobserved entries of the data matrix X (hold-out prediction); (ii) predicting unobserved entries of a new row or column appended to the data matrix (fold-in prediction); (iii) clustering the rows and/or columns of the matrix. Low-rank matrix factorization addresses the choice of parameterization issue. There are many choices for models of X; this is one that has been successfully used in many domains, and respects our theoretical understanding of the problem (cf, Aldous' theorem).

- ▷ **Also Known As**: "Other well-known names for the pattern, if any". Variants of low-rank matrix factorization go by many names in the literature: principal components analysis, matrix co-clustering, matrix factorization, two-layer latent variable models, Bayesian matrix factorization. See Table 3.1.

- ▷ **Motivation**: "A scenario that illustrates a design problem...and how the pattern solve[s] the problem". The examples discussed in Section 3.1 serve as motivating examples.

- ▷ **Applicability**: "What are the situations in which the design pattern can be applied? What are examples of poor design that the pattern can address? How can you recognize these situations?" Low-rank matrix factorization can be applied when the data can be set up as a matrix, where we believe the value in entry X_{ij} depends in some fashion on the identity of the entities indexed at row i and column j. In essence, the statistical design pattern is applicable when one has a row-column exchangeable matrix. An example of a poor design that matrix factorization resolves are techniques that work on matrix data, but treat them as exchangeable or iid rows of data (i.e., confusing exchangeability and row-column exchangeability).

- ▷ **Structure**: "A graphical representation of the classes in the pattern...". The closest analogue in graphical models would be a visual description, such as the plate model in Figure 3.1.

- ▷ **Participants**: "The classes and/or objects participating in the design pattern and their responsibilities." The closest analogue in a statistical design pattern is the training objective and the optimization technique, which abstracts the statistical model into a form that can be readily implemented.

- ▷ **Collaborations**: "What are the trade-offs and results of using the pattern? What aspects of the system structure does it let you vary independently?". There are trade-offs in the use of matrix factorization: latent-variable models are difficult to interpret, there is no intensional definition of a topic/cluster/factor; the parameter space grows with the number of rows and columns, defeating any theoretical arguments based on asymptotic

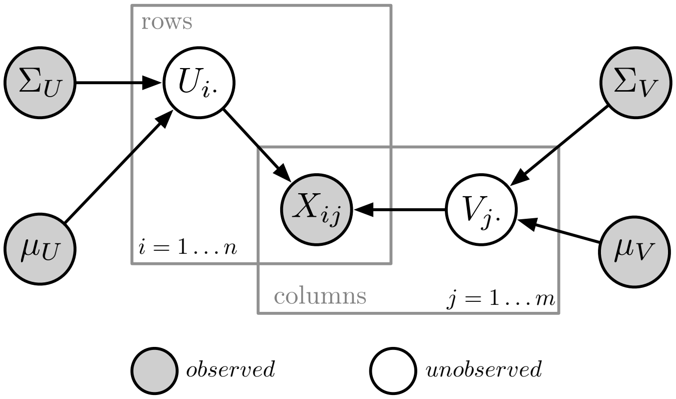


Figure 3.1: Plate model representation of maximum likelihood single matrix factorization. The intersection of the plate is interpreted as "for each combination of row i and column j". Shaded nodes refer to known or observed variables. Unshaded nodes refer to unknown (unobserved) variables.

consistency; training involves large-parameter optimizations over non-linear objectives, with many local optima. With respect to varying system structure, the six modeling choices we have described in this chapter can be, and in the literature are, independently varied.

▷ **Implementation**: "What pitfalls, hints, or techniques should you be aware of when implementing this pattern?" We briefly discussed some of the algorithmic issues in training, which are more extensively explored in the context of collective matrix factorization, which subsumes low-rank matrix factorization (Chapter 4).

▷ **Sample Code**: "Code fragments that illustrate how you might implement this pattern". The publications underlying this thesis include a MATLAB toolkit for low-rank matrix factorization (as part of collective matrix factorization).[4]

▷ **Known Uses**: A few of the examples of low-rank matrix factorization include document clustering (e.g., pLSI), collaborative filtering [41], and predicting links in social networks [51]. .

▷ **Related Patterns**: "What design patterns are closely related to this one? What are the important differences?" Collective Matrix Factorization is a generalization of matrix factorization to sets of related matrices, which we propose as an approach to information

[4] `http://www.cs.cmu.edu/~ajit/cmf`

integration. Semi-supervised matrix factorization deals the scenario where either the rows or columns are labelled, and the goal is to label new rows or columns.

Statistical design patterns formalize expertise in the development and application of graphical models in much the same way that software design patterns formalize the development and application of large software systems.

Chapter 4

Collective Matrix Factorization

4.1 Introduction

In the last chapter, we considered the simplest example of a relational data set—an arity two relation where the rows and columns index entities. A limitation of matrix factorization is that we only consider one relation; but, going back to the introduction, our interest is in data sets where entities can participate in many different relations (properties). In this chapter, we generalize matrix factorization to allow an entity to participate in more than one arity-two relation.

We build upon low-rank matrix factorization, but as we have seen there are many potential models we could build on. Since our goal is predicting unobserved values of a relation, we shall not consider clustering and matrix co-clustering methods in further detail. Factor analysis models lack stochastic (clustering) constraints on the factors, which significantly complicate training. As importantly, relations can take on many different types of values; we desire flexibility in modeling the response type. If we consider the models in Table 3.1, these desiderata leave us with Exponential Family PCA and its regularized extension, G^2L^2M.

We begin by discussing the basic idea of parameter tying in low-rank matrix factorization, which is the foundation of collective matrix factorization. Then, we formally develop collective matrix factorization as a probabilistic model (Section 4.3). We discuss how alternating projections, first introduced in Section 3.5, are particularly useful in training our models. We also detail the Newton projection, which is what makes training relatively efficient (Section 4.3.1). For clarity, we focus on a simple example of collective matrix factorization, involving two related matrices. In Section 4.3.3 we show how the approach is readily generalized to more than two related matrices. To further illustrate the power of the alternating projections approach, we show how a technique for stochastic approximation in regression models can be generalized to low-rank matrix factorization (Section 4.4). Returning to our overarching goal, information integration, we consider the problem of augmented collaborative filtering. In our experiments (Section 4.5) we augment a relation that represents users' ratings of movies with side information about the movies, namely which genres describe the movie. We show that collective matrix factorization can use genre information to improve the quality of ratings prediction, and vice versa.

Main Contributions: While we do not claim ownership of the idea of tying low-rank parameters in matrix factorization, we are the first to apply low-rank parameter tying to Exponential Family PCA. Since E-PCA is so flexible in the choice of data distributions, we can deal with relations of differing response type. The most significant contributions are in the learning algorithm, where we extend alternating Newton-projection on a single matrix to sets of related matrices. The resulting algorithm is memory-efficient, easy to parallelize, and can work even with a large number of entities (the per-iteration cost is linear in the number of entities). Moreover, we consider the case where the matrices are both large and densely observed. Here, we develop a stochastic Newton projection that allows one to trade between the computational cost of training and predictive accuracy of the resulting model. Empirically, we have found that the stochastic Newton projection leads to significantly faster convergence during the first few iterations, making it particularly suitable for training models where CPU resources are limited. Our stochastic Newton approach can be applied anywhere alternating Newton-projections can be used, but is most appropriate for large, densely observed matrices.

4.2 Relational Schemas

A relational schema defines the structure among a set of relations. Abstractly, we view a relational schema as a collection of t entity-types $\mathcal{E}_1 \dots \mathcal{E}_t$ along with a list of relations, each represented by a matrix of data. Each matrix involves two entity-types, one for the rows and another for the columns. There are n_i entities of type i, denoted $\{x_e^{(i)}\}_{e=1}^{n_i}$. A relation between two types is $\mathcal{E}_i \sim_u \mathcal{E}_j$; index $u \in \mathbb{N}$ allows us to distinguish multiple relations between the same types, and is omitted when there is no ambiguity. The matrix containing the values for $\mathcal{E}_i \sim_u \mathcal{E}_j$ has n_i rows, n_j columns, and is denoted $X^{(ij,u)}$. If we have not observed all possible values of a relation, we fill in unobserved entries with 0 (so that $X^{(ij,u)}$ is a sparse matrix), and assign them zero weight when learning parameters. By convention, for each relation $\mathcal{E}_i \sim \mathcal{E}_j$, we assume $i \leq j$.

Since all the relations are arity-two, we represent the schema as an undirected graph where the nodes are entity types, and the edges are abstract relations between them: $E = \{(i,j) : \mathcal{E}_i \sim \mathcal{E}_j \ \wedge \ i \leq j\}$.[1] Without loss of generality, we assume that the schema-as-graph is fully connected. If not, we can fit each connected component in the schema independently. Our definition of a relational schema is similar to an entity-relationship model [22].

We fit each relation matrix as the product of latent factors, $X^{(ij)} \approx f^{(ij)}(U^{(i)}(U^{(j)})^T)$, where $U^{(i)} \in \mathbb{R}^{n_i \times k}$ and $U^{(j)} \in \mathbb{R}^{n_j \times k}$ for $k \in \{1, 2, \ldots\}$. If $\mathcal{E}_j$ participates in more than one relation, we allow our model to reserve columns of $U^{(j)}$ for modelling a specific relation. This flexibility allows us, for example, to have relations with different latent dimensions, or to have more than one relation between $\mathcal{E}_i$ and $\mathcal{E}_j$ without forcing ourselves to predict the same value for each relation. In an implementation, we would store a list of participating column indices from each factor for each relation; but to avoid clutter, we ignore this possibility in our notation. Unless otherwise noted, the prediction link $f^{(ij)}$ is an element-wise function on matrices.

4.3 Collective Factorization

To avoid an excess of notation, we introduce Collective Matrix Factorization on a three entity-type problem, corresponding to the schema $\mathcal{E}_1 \sim \mathcal{E}_2 \sim \mathcal{E}_3$. We generalize the algorithm to three or more related matrices in Section 4.3.3. The factorizations corresponding to the three entity-type schema are

$$X^{(12)} \approx f^{(12)}\left(U^{(1)}\left(U^{(2)}\right)^T\right), \tag{4.1}$$

$$X^{(23)} \approx f^{(23)}\left(U^{(2)}\left(U^{(3)}\right)^T\right). \tag{4.2}$$

Let $X = X^{(12)}$, $Y = X^{(23)}$, $f_1 = f^{(12)}$, $f_2 = f^{(23)}$, $m = n_1$, $n = n_2$, $r = n_3$, $U = U^{(1)}$, $V = U^{(2)}$, and $Z = U^{(3)}$. We assume that the latent dimension is $k = k_{12} = k_{23}$. We can rewrite Equations 4.1-4.2 as

$$X \approx f_1(UV^T), \tag{4.3}$$

$$Y \approx f_2(VZ^T), \tag{4.4}$$

[1] An abstract relation is one where the arguments are variables: e.g., $\texttt{Rating}(\mathit{user}, \mathit{movie})$. Observed relations are sets of relations where the arguments are grounded: e.g., $\{\texttt{Rating}(\mathit{user}_1, \mathit{movie}_{12}) = 1, \texttt{Rating}(\mathit{user}_3, \mathit{movie}_{12})\}$.

In our experiments, augmented collaborative filtering is an example of a problem that fits into the three entity-type schema: $\mathcal{E}_1$ are users, $\mathcal{E}_2$ are movies, and $\mathcal{E}_3$ are genres. X is a matrix of observed ratings, and Y indicates which genres a movie belongs to (each column corresponds to a genre, and movies can belong to multiple genres).

From Equations 4.3-4.4 we choose a probabilistic model with parameters $\mathcal{F} = \{U, V, Z\}$, and data $\mathcal{D} = \{X, Y\}$. The likelihood of each matrix is

$$p(X \mid U, V, W) = \prod_{i=1}^{m} \prod_{j=1}^{n} \left[p_X \left(X_{ij} \mid U_{i\cdot} V_{j\cdot}^T \right) \right]^{W_{ij}} , \tag{4.5}$$

$$p(Y \mid V, Z, \tilde{W}) = \prod_{j=1}^{n} \prod_{r=1}^{r} \left[p_Y \left(Y_{jr} \mid V_{j\cdot} Z_{r\cdot}^T \right) \right]^{\tilde{W}_{jr}} . \tag{4.6}$$

The per-entry distributions p_X and p_Y are one-parameter exponential families with natural parameter $U_{i\cdot} V_{j\cdot}^T$ and $V_{j\cdot} Z_{r\cdot}^T$, respectively. The modeler is free to choose p_X and p_Y, and they need not be from the same exponential family; this allows us to integrate relations with different response types: e.g., `Co-occurs` may be well-modelled by the Binomial distribution, but `Response` (brain voxel activity) is best modelled by a Gaussian. The weights $W_{ij} \geq 0$ and $\tilde{W}_{jr} \geq 0$ allow us to handle missing data: we set an indicator to zero when the corresponding value in the data matrix is unobserved. We can learn the factors in Equation 4.5 and 4.6 using maximum likelihood or Bayesian methods. If we maximize Equation 4.5 when X is fully observed under maximum likelihood, we get back Exponential Family PCA (see Table 3.1 under E-PCA).

Maximizing the product of Equations 4.5 and 4.6 with respect to the factors $\mathcal{F}$ is an example of collective matrix factorization [116]. We place a multivariate Gaussian prior on each row of U:

$$p(U \mid \Theta_U) = \prod_{i=1}^{m} \mathcal{N}(U_{i\cdot} \mid \mu_U, \Sigma_U), \tag{4.7}$$

where $\mathcal{N}(\cdot \mid \mu_U, \Sigma_U)$ is a Gaussian with mean vector μ_U and covariance matrix Σ_U. We assume that $\mu_U = 0$ and Σ_U is spherical, so that the covariance can be simplified to regularization coefficients λ_U, λ_V, and λ_Z. The priors over V and Z are defined similarly. In log-space, a zero-mean spherical covariance prior can also be encoded using Bregman divergences:

$$\begin{aligned} \mathbb{D}_G(0 \,||\, U) &= G^*(U), \\ \mathbb{D}_H(0 \,||\, V) &= H^*(V), \\ \mathbb{D}_I(0 \,||\, Z) &= I^*(Z), \end{aligned}$$

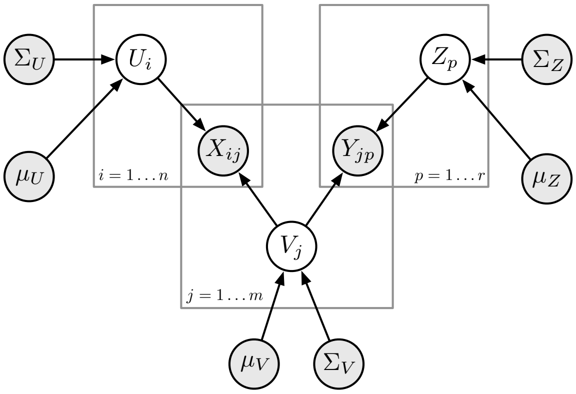


Figure 4.1: Plate representation of collective matrix factorization. Shaded nodes indicate observed data or quantities which must be selected, e.g. μ_U and Σ_U. Indicates W and $\tilde{W}$ are elided.

where $G(u) = \lambda_U u^2/2$, $H(v) = \lambda_V v^2/2$, and $I(z) = \lambda_Z z^2/2$. Appendix A provides a complete derivation of the gradient and Hessian, along with a more detailed discussion of the Bregman form for the regularizers. A plate model for collective matrix factorization is provided in Figure 4.1.

Taking together, Equations 4.5–4.7 define the posterior distribution of the parameters given the data:

$$p(U, V, Z \mid X, Y, \Theta) = c \cdot p(X \mid U, V, W)p(Y \mid V, Z, \tilde{W})p(U \mid \Theta)p(V \mid \Theta)p(Z \mid \Theta),$$

where c is a normalizing constant. Maximum a posteriori inference (a.k.a. regularized maximum likelihood) involves searching for the parameters which minimize the negative log-posterior

$$\mathcal{L} = -\log p(U, V, Z \mid X, Y, W, \tilde{W}, \Theta). \tag{4.8}$$

Since we assume that X_{ij} and Y_{jr} are draws from an exponential family distribution, then maximizing the likelihood of $p_X(X_{ij} \mid z_{ij} = U_{i\cdot}V_{j\cdot}^T)$ is equivalent to minimizing the Bregman divergence $\mathbb{D}(z_{ij} \,||\, X_{ij})$. Therefore, we can rewrite $\mathcal{L}$ using Bregman divergences:

$$\mathcal{L} = \underbrace{\mathbb{D}_{F_1}\left(UV^T \,||\, X, W\right)}_{\text{from } p(X \mid U,V,W)} + \underbrace{\mathbb{D}_{F_2}\left(VZ^T \,||\, Y, \tilde{W}\right)}_{\text{from } p(Y \mid V,Z,\tilde{W})} + \underbrace{\mathcal{R}(U)}_{\text{from } p(U \mid \Theta_U)} + \underbrace{\mathcal{R}(V)}_{\text{from } p(V \mid \Theta_V)} + \underbrace{\mathcal{R}(Z)}_{\text{from } p(Z \mid \Theta_Z)}, \tag{4.9}$$

where $\mathcal{R}(\cdot)$ is the ℓ_2-regularizer for the factor. We can make a coarse adjustment for the relative importance of the two data matrices involves scaling $\mathbb{D}_{F_1}$ by $\alpha \in [0, 1]$ and $\mathbb{D}_{F_2}$ by $(1 - \alpha)$. We consider the effect of varying α in Section 4.5.

4.3.1 Parameter Estimation

Equation 4.8 is non-convex in its parameters (U, V, Z). If we considered $\mathcal{L}$ without relating it to Bregman divergence, it would appear that our only recourse is to apply a generic nonlinear optimizer. However, Equation 4.9 reveals structure in the problem that we can exploit. First $\mathcal{L}$ is the sum of five terms, each of which is convex in its arguments. The problem is that arguments to the Bregman divergences are the product of factors, UV^T and VZ^T, instead of (U, V) and (V, Z). However, it can be shown that $\mathbb{D}_{F_1}(UV^T \,||\, X, W)$ is convex in U when V is fixed, and likewise convex in V when U is fixed [45]. Similarly, $\mathbb{D}_{F_2}$ is convex in V when Z is fixed, and convex in Z when V is fixed. So while $\mathcal{L}$ is not convex, it is *componentwise convex*: i.e., convex in one low-rank factor when the others are fixed.

Componentwise convexity of $\mathcal{L}$ leads naturally to alternating projections: cyclically optimize $\mathcal{L}$ with respect to U, then V, then Z. Since $\mathcal{L}$ is convex in a single low-rank factor, each projection is a convex optimization. Alternating projections is block coordinate descent.

The question that remains is how to implement the projection. For a convex optimization, the ideal solution would be to use Newton-Raphson, which incorporates both the gradient and the Hessian. Below, we outline the derivation of the Newton projection. A more detailed derivation, starting from the basics of matrix calculus, is available in Appendix A.

First, differentiate $\mathcal{L}$ with respect to each factor:[2]

$$\nabla_U \mathcal{L} = \alpha \left(W \odot \left(f_1(UV^T) - X \right) \right) V + \nabla G^*(U), \tag{4.10}$$

$$\begin{aligned} \nabla_V \mathcal{L} = {} & \alpha \left(W \odot \left(f_1(UV^T) - X \right) \right)^T U + \\ & (1-\alpha) \left(\tilde{W} \odot \left(f_2(VZ^T) - Y \right) \right) Z + \nabla H^*(V), \end{aligned} \tag{4.11}$$

$$\nabla_Z \mathcal{L} = (1-\alpha) \left(\tilde{W} \odot \left(f_2(VZ^T) - Y \right) \right)^T V + \nabla I^*(Z). \tag{4.12}$$

For ℓ_2 regularization on a factor U, $G(U) = \lambda ||U||_2^2/2$, therefore $\nabla G^*(U) = U/\lambda$. Setting these gradients equal to zero yields the exact minimum. We can use gradient descent to find a root of the above equations.

To improve performance over gradient decent, a natural idea is to try a Newton method for solving Equation 4.10-4.12. A cursory inspection of Equations 4.10-4.12 suggests that a Newton step is infeasible. For example, the Hessian with respect to U would involve

[2] We remind the reader that $A \odot B$ refers to the Hadamard (element-wise) product.

nk parameters, and thus inversion of a $nk \times nk$ matrix, which would require $O(n^3k^3)$ time and $O(n^2k^2)$ memory. In our augmented collaborative filtering experiments, n is the number of users; spending $O(n^3k^3)$ time is out of the question.

The cursory argument assumes that the Hessian is an arbitrary matrix. However, we can show that when only one low-rank factor can vary, most of the second derivatives of $\mathcal{L}$ with respect to that factor are zero, and the Hessian has special structure that makes inverting it easy. For the subclass of models where Equations 4.10-4.12 are differentiable and the loss is decomposable, define

$$\begin{aligned}
q(U_{i\cdot}) &= \alpha \left(W_{i\cdot} \odot \left(f_1(U_{i\cdot}V^T) - X_{i\cdot}\right)\right) V + \nabla G^*(U_{i\cdot}), \\
q(V_{i\cdot}) &= \alpha \left(W_{\cdot i} \odot \left(f_1(UV_{i\cdot}^T) - X_{\cdot i}\right)\right)^T U + \\
&\qquad (1-\alpha) \left(\tilde{W}_{i\cdot} \odot \left(f_2(V_{i\cdot}Z^T) - Y_{i\cdot}\right)\right) Z + \nabla H^*(V_{i\cdot}), \\
q(Z_{i\cdot}) &= (1-\alpha) \left(\tilde{W}_{\cdot i} \odot \left(f_2(VZ_{i\cdot}^T) - Y_{\cdot i}\right)\right)^T V + \nabla I^*(Z_{i\cdot}).
\end{aligned}$$

Since all but one factor is fixed, consider the derivatives of $q(U_{i\cdot})$ with respect to any scalar parameter in U: $\nabla_{U_{js}} q(U_{i\cdot})$. Because U_{js} only appears in $q(U_{i\cdot})$ when $j = i$, the derivative equals zero when $j \neq i$. Therefore the Hessian $\nabla_U^2 \mathcal{L}$ is block-diagonal, where each non-zero block corresponds to a row of U. The inverse of a block-diagonal matrix is the inverse of each block, and so the Newton direction for U, $[\nabla_U \mathcal{L}][\nabla_U^2 \mathcal{L}]^{-1}$, can be reduced to updating each row $U_{i\cdot}$ using the direction $[q(U_{i\cdot})][q'(U_{i\cdot})]^{-1}$.

The above derivation shows that we can reduce the projection over the factor U into a set of parallel row-wise projections over $\{U_{i\cdot}\}$. By exploiting structure within the Hessian, we have reduced the cost of using the Hessian from $O(n^3k^3)$ to $O(nk^3)$. We can further reduce the cost of Newton projection by replacing the exact computation of the step with an approximation, e.g., using a fixed number of iterations of conjugate gradient to find the step direction. Using an approximate step reduces the cost of the Newton projection to $O(nk^2)$ time and $O(nk^2)$ memory.[3]

The block diagonal Hessian may give one the impression that there are no interactions between different rows of a factor, which is not true. Different rows of a factor can influence each other, but only indirectly through the other factor in the bilinear form. Two rows of U interact, indirectly, through their effect on V. However, since we fixed V in the projection over U, the inter-row effects are masked.

By fixing all but one factor and varying each factor in turn, we get a block-diagonal approximation to the true Hessian across all factors. Approximating the Hessian is also at

[3] The memory requirements can be reduced to $O(k^2)$ if the per-row projections are done sequentially.

the heart of quasi-Newton methods, like BFGS [90], which use a low-rank approximation of the Hessian. We get a perfect model of the Hessian within each row by ignoring weaker, indirect, inter-row effects. Low-rank approximations of the Hessian approximate the inter-row effects, as well as the intra-row effects. Combining the two approaches to improve our approximation of the per-factor Hessians $\nabla^2_U \mathcal{L}$, $\nabla^2_V \mathcal{L}$, and $\nabla^2_Z \mathcal{L}$, as well as the Hessian with respect to all the factors, $\nabla \mathcal{L}$, remains a topic of future work.

It remains to be shown how we can compute the non-zero blocks in the Hessian. Any (local) optimum of the loss $\mathcal{L}$ corresponds to roots of the equations $\{q(U_{i\cdot}) = 0\}_{i=1}^m$, $\{q(V_{i\cdot}) = 0\}_{i=1}^n$, and $\{q(Z_{i\cdot}) = 0\}_{i=1}^r$.

To find the roots of these equations, we use Newton's method. For example, the Newton update for $U_{i\cdot}$ is

$$U_{i\cdot}^{\text{new}} = U_{i\cdot} - q(U_{i\cdot})[q'(U_{i\cdot})]^{-1}. \tag{4.13}$$

To concisely describe the Hessian we introduce terms for the contribution of the regularizer,

$$G_i \equiv \operatorname{diag}(\nabla^2 G^*(U_{i\cdot})), \quad H_i \equiv \operatorname{diag}(\nabla^2 H^*(V_{i\cdot})), \quad I_i \equiv \operatorname{diag}(\nabla^2 I^*(Z_{i\cdot})),$$

and terms for the contribution of the reconstruction error,

$$\begin{aligned} D_{1,i} &\equiv \operatorname{diag}(W_{i\cdot} \odot f_1'(U_{i\cdot} V^T)), \quad & D_{2,i} &\equiv \operatorname{diag}(W_{\cdot i} \odot f_1'(U V_{i\cdot}^T)), \\ D_{3,i} &\equiv \operatorname{diag}(\tilde{W}_{i\cdot} \odot f_2'(V_{i\cdot}^T Z)), \quad & D_{4,i} &\equiv \operatorname{diag}(\tilde{W}_{\cdot i} \odot f_2'(V Z_{i\cdot}^T)). \end{aligned}$$

The Hessians with respect to the loss $\mathcal{L}$ are

$$\begin{aligned} q'(U_{i\cdot}) &\equiv \nabla q(U_{i\cdot}) = \alpha V^T D_{1,i} V + G_i, \\ q'(Z_{i\cdot}) &\equiv \nabla q(Z_{i\cdot}) = (1-\alpha) V^T D_{4,i} V + I_i, \\ q'(V_{i\cdot}) &\equiv \nabla q(V_{i\cdot}) = \alpha U^T D_{2,i} U + (1-\alpha) Z^T D_{3,i} Z + H_i. \end{aligned}$$

Each update of U, V, and Z reduces at least one term in Equation 4.9. Therefore, iteratively cycling through the updates leads to a local optimum. The Newton step may be scaled by a step length $\eta \in (0, 1]$, which we select using the Armijo criterion [90]. In practice, we simplify the projection by taking one Newton step, instead of running to convergence. Even taking one step, we are guaranteed to reduce the value of at least one term in Equation 4.9, guaranteeing convergence to a local optimum.

Adjusted Dependent Variate Projection

Since the per-row updates are identical to an iteration of iteratively-reweighted least squares for a GLM, we mirror the literature by rewriting the update in adjusted dependent variate form [78]. The adjusted dependent variate form represents each update to a factor row as the solution to a weighted regression problem. Let $A_{i\cdot} = U_{i\cdot}/\lambda_U$, $B_{i\cdot} = V_{i\cdot}/\lambda_V$, and $C_{i\cdot} = Z_{i\cdot}/\lambda_Z$ account for the regularizer terms in the gradient. Let η be the step length. Rearranging terms in Equation 4.13,

$$U_{i\cdot}^{\text{new}} q'(U_{i\cdot}) = \alpha \left(U_{i\cdot} V^T + \eta \left(W_{i\cdot} \odot \left(X_{i\cdot} - f(U_{i\cdot} V^T) \right) \right) D_{1,i}^{-1} \right) D_{1,i} V + U_{i\cdot} G_i - \eta A_{i\cdot}. \tag{4.14}$$

Likewise for $Z_{i\cdot}$

$$Z_{i\cdot}^{\text{new}} q'(Z_{i\cdot}) = (1-\alpha) \left(Z_{i\cdot} V^T + \eta \left(\tilde{W}_{\cdot i} \odot \left(Y_{\cdot i} - f(V Z_{i\cdot}^T) \right) \right)^T D_{4,i}^{-1} \right) D_{4,i} V + Z_{i\cdot} I_i - \eta C_{i\cdot}. \tag{4.15}$$

The update for $V_{i\cdot}$, which ties the data matrices together, has a similar derivation since $L(U, V, Z)$ is the sum of per-matrix losses and the differential operator is linear:

$$\begin{aligned} V_{i\cdot}^{\text{new}} q'(V_{i\cdot}) = \alpha & \left\{ \left(V_{i\cdot} U^T + \eta \left(W_{\cdot i} \odot \left(X_{\cdot i} - f(U V_{i\cdot}^T) \right) \right)^T D_{2.i}^{-1} \right) D_{2,i} U \right\} + \\ & (1-\alpha) \left\{ \left(V_{i\cdot} Z^T + \eta \left(\tilde{W}_{i\cdot} \odot \left(Y_{i\cdot} - f(V_{i\cdot} Z^T) \right) \right) D_{3,i}^{-1} \right) D_{3,i} Z \right\} + \\ & V_{i\cdot} H_i - \eta B_{i\cdot} \end{aligned} \tag{4.16}$$

4.3.2 Imbalanced Matrices

The objective $\mathcal{L}$ is the sum of two per-matrix reconstruction errors. If one matrix is larger than the other, it will dominate the objective. We use data weights, or equivalently α, to turn $\mathcal{L}$ into a per-element objective by scaling each element of X by $(nm)^{-1}$ and each element of Y by $(nr)^{-1}$. This heuristic ensures that larger matrices do not dominate the model simply because they are larger. Data weights can also be used to correct for differences in the scale of $\mathbb{D}_{F_1}$ and $\mathbb{D}_{F_2}$, which can occur when one of the Bregman divergences is not regular. Since we restrict our consideration to Bregman divergences that correspond to probability distributions, we do not need to correct for differences in scale: probabilities are all on the same $[0, 1]$ scale. If the Bregman divergences are not regular, computing

$$D_{F_1}(UV^T \,||\, X, W)/D_{F_2}(VZ^T \,||\, Y, \tilde{W}),$$

averaged over uniform random parameters U, V, and Z, provides an adequate estimate of the relative scale of the two losses.

4.3.3 Generalizing to Arbitrary Schemas

We have focused on the three entity-type (two matrix) case, but our approach can be applied to any relational schema E containing only binary relations. The objective is

$$\mathcal{L}_{general} = \sum_{(i,j)\in E \vee (j,i)\in E} \alpha^{(ij)} \left(\mathbb{D}_{F^{(ij)}}(U^{(i)}(U^{(j)})^T \,||\, X^{(ij)}, W^{(ij)})\right) + \\ \sum_{i=1}^{t} \left(\sum_{j:(i,j)\in E \vee (j,i)\in E} \alpha^{(ij)} \right) \mathcal{R}\left(U^{(i)}\right).$$

The relative weights $\alpha^{(ij)} \geq 0$ measure the importance of each matrix in the reconstruction. We scale the regularization of a factor $U^{(i)}$ by its relative importance in the model: the degree to which we are concerned about overfitting in factor should be proportional to that factor's importance in prediction. Since the loss is a linear function of individual losses, and the differential operator is linear, both gradient and Newton updates can be derived in a manner analogous to Section 4.3.1, taking care to distinguish when $U^{(i)}$ acts as a column factor as opposed to a row factor.

4.4 Stochastic Approximation

In optimizing a collective factorization model, we are in the unusual situation that our primary concern is not the cost of inverting the Hessian, but rather the cost of computing the gradient itself: if k is the largest embedding dimension, then the cost of a gradient update for a row $U_r^{(i)}$ is $O(k \sum_{j:\mathcal{E}_i \sim \mathcal{E}_j} n_j)$, while the cost of a Newton update for the same row is $O(k^3 + k^2 \sum_{j:\mathcal{E}_i \sim \mathcal{E}_j} n_j)$. Typically k is much smaller than the number of entities, and so the Newton update costs only a factor of k more. (The above calculations assume fully observed matrices; for sparsely-observed relations, we can replace n_j by the number of entities of type $\mathcal{E}_j$ which are related to entity $x_r^{(i)}$, but the conclusion remains the same.)

The expensive part of the gradient calculation for $U_r^{(i)}$ is to compute the predicted value for each observed relation that entity $x_r^{(i)}$ participates in, so that we can sum all of the weighted prediction errors. One approach to reducing this cost is to compute errors only on a subset of observed relations, picked randomly at each iteration. This technique is known as stochastic approximation [9]. The best-known stochastic approximation algorithm is stochastic gradient descent; but, since inverting the Hessian is not a significant part of our computational cost, we will recommend a stochastic Newton's method instead.

Consider the update for $U_{i\cdot}$ in the three factor model. This update can be viewed as a regression where the data are $X_{i\cdot}$ and the features are the columns of V. If we denote a sample of the data as $s \subseteq \{1, \ldots, n\}$, then the sample gradient at iteration τ is

$$\hat{q}_\tau(U_{i\cdot}) = \alpha \left(W_{is} \odot \left(f(U_{i\cdot}V_{s\cdot}^T) - X_{is}\right)\right) V_{s\cdot} + \nabla G^*(U_{i\cdot}),$$

Similarly, given subsets $p \subseteq \{1, \ldots, m\}$ and $q \subseteq \{1, \ldots, r\}$, the sample gradients for the other factors are

$$\begin{aligned}\hat{q}_\tau(V_{i\cdot}) &= \alpha \left(W_{pi} \odot \left(f(U_{p\cdot}V_{i\cdot}^T) - X_{pi}\right)\right)^T U_{p\cdot} + \\ &\qquad (1-\alpha)\left(\tilde{W}_{iq} \odot \left(f(V_{i\cdot}Z_{q\cdot}^T) - Y_{iq}\right)\right) Z_{q\cdot} + \nabla H^*(V_{i\cdot}), \\ \hat{q}_\tau(Z_{i\cdot}) &= (1-\alpha)\left(\tilde{W}_{si} \odot \left(f(V_{s\cdot}Z_{i\cdot}^T) - Y_{si}\right)\right)^T V_{s\cdot} + \nabla I^*(Z_{i\cdot}).\end{aligned}$$

Given the sample gradient, the stochastic gradient update for U at iteration τ is

$$\begin{aligned}U_{i\cdot}^{\tau+1} &= U_{i\cdot}^{\tau} - \eta_\tau \hat{q}_\tau(U_{i\cdot}), \\ \eta_\tau &= 1/\tau.\end{aligned}$$

and similarly for the other factors. Note that we use a fixed, decaying sequence of learning rates instead of a line search: sample estimates of the gradient are not always descent directions, and the fixed schedule for step lengths is required to guarantee convergence of the projection. An added advantage of the fixed schedule over line search, in the context of a fast approximate update, is that the latter is computationally expensive.

We sample data non-uniformly, without replacement, from the distribution induced by the data weights. That is, for a row $U_{i\cdot}$, the probability of drawing X_{ij} is $W_{ij}/\sum_j W_{ij}$. This sampling distribution provides a compelling relational interpretation: to update the latent factors of $x_r^{(i)}$, we sample only observed relations involving $x_r^{(i)}$. For example, to update a user's latent factors, we sample only movies that the user rated. We use a separate sample for each row of U: this way, errors are independent from row to row, and their effects tend to cancel. In practice, this means that our actual training loss decreases at almost every iteration.

With sampling, the cost of the gradient update no longer grows linearly in the number of entities related to $x_r^{(i)}$, but only in the number of entities sampled. Another advantage of this approach is that when we sample one entity at a time, $|s| = |p| = |q| = 1$, stochastic gradient yields an online algorithm, where observed relations are processed as they appear, and are not stored across iterations.

As mentioned above, we can often improve the rate of convergence by moving from stochastic gradient descent to stochastic Newton-Raphson updates [9, 11]. For the three-factor model the sample Hessians are

$$\begin{aligned}
\hat{q}'_\tau(U_{i\cdot}) &= \alpha V_{s\cdot}^T \hat{D}_{1,i} V_{s\cdot} + G_i,\\
\hat{q}'_\tau(Z_{i\cdot}) &= (1-\alpha) V_{s\cdot}^T \hat{D}_{4,i} V_{s\cdot} + I_i,\\
\hat{q}'_\tau(V_{i\cdot}) &= \alpha U_{p\cdot}^T \hat{D}_{2,i} U_{p\cdot} + (1-\alpha) Z_{q\cdot}^T \hat{D}_{3,i} Z_{q\cdot} + H_i.
\end{aligned}$$

where

$$\begin{aligned}
\hat{D}_{1,i} &\equiv \operatorname{diag}(W_{is} \odot f_1'(U_{i\cdot} V_{s\cdot}^T)), \quad & \hat{D}_{2,i} &\equiv \operatorname{diag}(W_{pi} \odot f_1'(U_{p\cdot} V_{i\cdot}^T)),\\
\hat{D}_{3,i} &\equiv \operatorname{diag}(\tilde{W}_{iq} \odot f_2'(V_{i\cdot}^T Z_{q\cdot})), \quad & \hat{D}_{4,i} &\equiv \operatorname{diag}(\tilde{W}_{si} \odot f_2'(V_{s\cdot} Z_{i\cdot}^T)).
\end{aligned}$$

To satisfy convergence conditions, which will be discussed in Section 4.4.1, we use an exponentially weighted moving average of the Hessian:

$$\bar{q}_{\tau+1}(\cdot) = \left(1 - \frac{2}{\tau+1}\right) \bar{q}_\tau(\cdot) + \frac{2}{\tau+1} \hat{q}'_{\tau+1}(\cdot). \tag{4.17}$$

When the sample at each step is small compared to the embedding dimension, the Sherman-Morrison-Woodbury lemma (*e.g.*, [9]) can be used for efficiency. The stochastic Newton update is analogous to Equation 2.8, except that the step length follows a fixed schedule, the gradient is replaced by its sample estimate $\hat{q}$, and the Hessian is replaced by its sample estimate $\bar{q}$.

4.4.1 Convergence

The objective function $\mathcal{L}$ (Equation 4.9) is the training error, whose form is determined by a fixed set of observations: the observed entries of the data matrices, which we denote as

$$\begin{aligned}
\mathcal{X} &= \{X_{ij} \,|\, W_{ij} > 0, \forall i, j\},\\
\mathcal{Y} &= \{Y_{jr} \,|\, \tilde{W}_{jr} > 0, \forall j, r\}.
\end{aligned}$$

Minimizing the training error, which is determined from a fixed batch of observations, is known as *batch learning*. In contrast, *online learning* assumes that the observations are drawn from a fixed distribution over observations: $p(\mathcal{X}, \mathcal{Y})$, also known as the sampling distribution. As discussed in the previous section, the weights determine the sampling distribution; if the weights are all either zero or one, then we sample the data uniformly.

At each iteration of the training algorithm, a data point is drawn from the sampling distribution, used to update the parameters, and then discarded. Samples are drawn independently across iterations, so the same matrix entry may be sampled repeatedly. Stochastic optimization is akin to online learning, except that we use a batch of samples at each iteration.[4]

While using less data at each iteration is computationally advantageous, there is also a compelling statistical argument to stochastic optimization. The training error $\mathcal{L}$ can be expressed as the (weighted) average of reconstruction errors, plus regularization terms independent of the data. The expected loss $E[\mathcal{L}]$ can similarly be defined using the sampling distribution:

$$E[\mathcal{L}] = \sum_{(X \in \mathcal{X}, Y \in \mathcal{Y})} \left[\mathbb{D}_{F_1}(UV^T \,||\, X) + \mathbb{D}_{F_2}(VZ^T \,||\, Y) \right] p_{(\mathcal{X},\mathcal{Y})}(X, Y) + \sum_{F \in \{U,V,Z\}} \mathcal{R}(F).$$

The linearity of expectations allows for the sampling over matrices to be reduced to sampling in the innermost loop, where $E[\mathcal{L}]$ is minimized with respect to a factor row. The stochastic optimization in the previous section is analogous to alternating Newton-projection, except that it minimizes $E[\mathcal{L}]$. Since we are sampling a fixed set of observations according to their weights, $E[\mathcal{L}] = \mathcal{L}$.

The convergence argument for alternating Newton-projection is straightforward. Each projection strictly decreases the objective $\mathcal{L}$, which is bounded below. However, if the projection is implemented as a step of stochastic Newton, there is no guarantee of that the objective will decrease after each projection. We can show that stochastic Newton is a convergent procedure, if a sufficient number of iterations are run.

Theorem 2 (Convergence of stochastic Newton projection)**.** *Let $C(F_{i\cdot})$ correspond to $E[\mathcal{L}]$ where all but one row of a low-rank factors $F \in \{U, V, Z\}$ is fixed.*[5] *Let*

- ▷ *$F_{i\cdot}^*$ denote the local optimum,*
- ▷ *$F_{i\cdot}^{(0)}$ the initial value of the factor row, and, $\{F_{i\cdot}^{(t)}\}_{t=1}^{\infty}$ a sequence of stochastic Newton iterates on $F_{i\cdot}$.*

[4] Online learning typically assumes that the data generating distribution is unknown, but that an oracle is available to generate samples. In particular, there are an infinite number of possible samples in online optimization; in our use of stochastic optimization, there are only a finite number of possible samples.

[5] It does not matter if $F_{j\cdot}, j \neq i$ is fixed or not, since row updates are independent in the projection.

Consider the final phase of convergence, where the parameters are in a region around the optimum $F_{i\cdot}^*$ where general convexity holds:

$$\forall \epsilon > 0, \inf_{||F_{i\cdot} - F_{i\cdot}^*||_2^2 > \epsilon} \langle F_{i\cdot} - F_{i\cdot}^*, \nabla C(F_{i\cdot}) \rangle > 0.^6 \tag{4.18}$$

If we assume that the data and parameters are uniformly bounded, then the sequence of stochastic Newton iterates converges almost surely,

$$\lim_{t \to \infty} F_{i\cdot}^{(t)} = F_{i\cdot}^*,$$

with probability one.

Proof. We use results from [11, 10, 12], which provide the following sufficient conditions for convergence of stochastic Newton optimization. These conditions must hold in a region around $F_{i\cdot}^*$ where Equation 4.18 holds:

1. **Decreasing step lengths**: The sequence of step length η_τ must satisfy the following conditions: $\sum_{\tau=1}^{\infty} \eta_\tau = \infty$, $\sum_{\tau=1}^{\infty} \eta_\tau^2 < \infty$.

2. **Bounded stochastic gradient**: For a single data sample (X_{ij}, Y_{jr}) the gradient is bounded:

$$\forall F_{i\cdot} \, \exists A_{F,i}, B_{F,i} \geq 0, \;\; E[\hat{q}_\tau(F_{i\cdot})] < A_{F,i} + B_{F,i}(F_{i\cdot} - F_{i\cdot}^*)^2.$$

3. **Bound curvature on the Hessian**: The eigenvalues of the Hessian $\bar{q}_\tau(F_{i\cdot})$ are bounded by constants $\lambda_{max} > \lambda_{min} > 0$, for all τ with probability one.

4. **Convergence of the Hessian**: The perturbation of the sample Hessian from its mean is bounded. Let $\mathcal{P}_{\tau-1}$ consist of the history of the stochastic Newton iterations: the data samples and the parameters for the first $\tau - 1$ iterations. Let $g_\tau = o_s(f_\tau)$ denote an almost uniformly bounded stochastic order of magnitude.[7] The convergence condition on the Hessian is a concentration of measure statement:

$$E[\bar{q}_\tau | \mathcal{P}_{\tau-1}] = \bar{q}_\tau + o_s(1/\tau).$$

[7] The stochastic o-notation is similar to regular o-notation, except that we are allowed to ignore measure-zero events.

The objective $C(F_{i\cdot})$ is strictly convex, due to the ℓ_2-regularizer, so general convexity holds over the domain of the objective. Our choice of step length schedule $\eta_\tau = 1/\tau$ is motivated by the decreasing step lengths condition. Because the parameters and data are uniformly bounded in an interval, any smooth continuous function of these quantities will also be uniformly bounded. The gradient and Hessian are continuous functions of the parameters and the data, and so the curvature conditions on the stochastic gradient and moving average Hessian are met. The ℓ_2-regularizer guarantees that the sample Hessians are invertible, and therefore $\bar{q}$ is also invertible, so its eigenvalues are lower bounded away from zero. The convergence condition on the Hessian motivates Equation 4.17:

$$E[\bar{q}_\tau | \mathcal{P}_{\tau-1}] = \left(1 - \frac{2}{\tau}\right) \bar{q}_{\tau-1} + \frac{2}{\tau} E[\hat{q}'_\tau | \mathcal{P}_{\tau-1}],$$

since $\mathcal{P}_{\tau-1}$ contains $\bar{q}_{\tau-1}$. Any perturbation from the mean is due to the second term. We assumed the parameters are uniformly bounded, and so the elements of $E[\hat{q}_\tau | \mathcal{P}_{\tau-1}]$ are uniformly bounded; since this term has bounded elements and is scaled by $2/\tau$, the perturbation is $o_s(1/\tau)$. □

The above proof also serves as a proof for the convergence of stochastic gradient descent: all the conditions for convergence hold by setting the Hessian to the identity: $\bar{q}(\cdot) = I_{k \times k}$.

Theorem 2 provides for convergence of a projection: if we run enough steps of stochastic Newton within a projection step, the loss is guaranteed to decrease, and the overall convergence of alternating projections is preserved. In practice, we run one step of stochastic Newton to update each row factor, and we have not observed any ill-effects for convergence.[8]

4.5 Experiments

In this chapter, we focus on the augmented collaborative filtering task: can we improve predictions for unobserved values of a rating relation $\texttt{Rating}(\textit{user}, \textit{movie}) \in \{1, \ldots, 5\}$ using side information about the genres, $\texttt{HasGenre}(\textit{movie}, \textit{genre}) \in \{0, 1\}$? Like

[8] It should be noted that while stochastic Newton projection can still yield a convergent algorithm, the quadratic rate of convergence guarantees for Newton projection do not hold for stochastic Newton projection. The advantage of stochastic Newton over stochastic gradient is a better sublinear rate of convergence. However, since we are wrapping the second-order step inside a coordinate descent algorithm, the difference is moot with respect to rate of convergence guarantees on $\mathcal{L}$.

many information integration tasks, the relations come from two different data sources, each providing a view into properties of a shared entity. User ratings are drawn from the Netflix Prize data [89]. Genres were added from the Internet Movie Database [54]. We briefly consider a more complicated schema, where in addition to ratings and genres we include information of which actors are in a particular movie: $\texttt{HasRole}(\mathit{actor}, \mathit{movie})$.

To yield a harder data set for stochastic optimization, we binarize the ratings:

$$\texttt{IsRated}(\mathit{user}, \mathit{movie}) \in \{0, 1\}.$$

IsRated tells us whether a user was interested enough in a movie to watch and rate it. While ratings are sparsely observed, IsRated is densely observed. In schema notation $\mathcal{E}_1$ corresponds to users, $\mathcal{E}_2$ corresponds to movies, $\mathcal{E}_3$ corresponds to genres, and $\mathcal{E}_4$ corresponds to actors.

The experiments provide evidence for three claims: (i) that integrating different relationships can improve predictions (see Section 4.5.1); (ii) that stochastic Newton projections can be effectively used to reduce the cost of training (see Section 4.5.2); and (iii) that while factor analysis is a more general representation than co-clustering, this does not imply superior generalization (see Section 4.5.3).

Model and Optimization Parameters

For consistency, we control many of the model and optimization parameters across the experiments. When the ratings relation is IsRated, we use a logistic model: sigmoid link with the matching log-loss. When the ratings are ordinal, we use a Poisson model: exponential link with the matching loss, unnormalized KL-divergence. In either setting, we evaluate the test error using mean absolute error (MAE). Since the data for IsRated is highly imbalanced in favour of movies not being rated, and since positive predictions are more important than negative ones, we scale the weight of the negative entries down by the fraction of observed entries where the relation is true. Unless otherwise stated the regularizers are all $G^*(\cdot) = 10^{-5}|| \cdot ||_F^2/2$, normalized to a per-element regularizer, as suggested in Section 4.3.2. In Newton steps, we use an Armijo line search, rejecting updates with step length smaller than $\eta = 2^{-4}$. Using Newton steps, we run till the change in training loss falls below 5% of the objective; using stochastic Newton steps, we run for a fixed number of iterations.

4.5.1 Relations Improve Predictions

Claim: Collective Matrix Factorization can use `HasGenre` to improve the prediction of `IsRated`, and vice-versa.

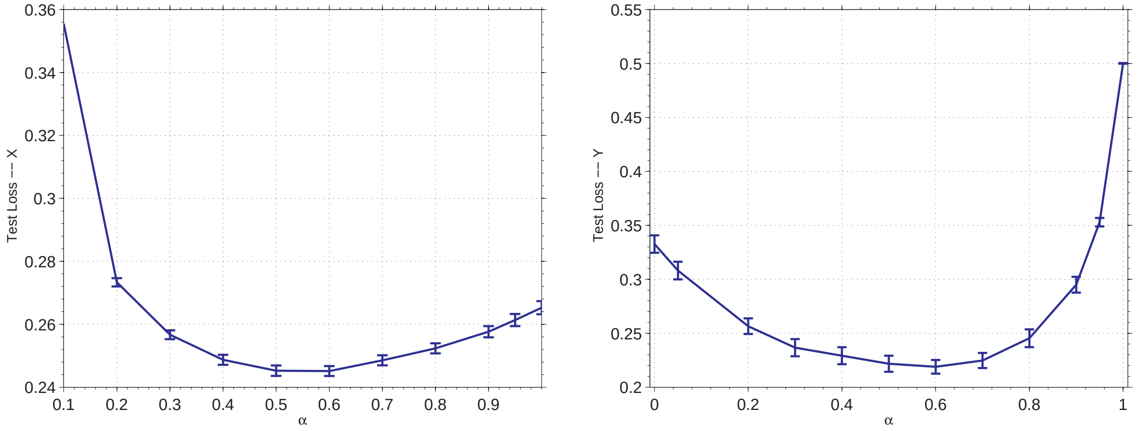
Our evidence for the claim consists of experiments on two relatively small augmented collaborative filtering tasks, to allow for repeated trials. Since we are using the three entity-type model there is a single mixing parameter, α, in Equation 4.9. We measure how the predictive accuracy of Collective Matrix Factorization varies with α, using Newton steps. We learn a model for several values of α, starting from the same initial random parameters, using full Newton steps. The performance on a test set, held-out entries sampled from the matrices according to the test weights, is measured at each α. Each trial is repeated ten times to provide 1-standard deviation bars.

Two scenarios are considered. In the first scenario, users and movies were sampled uniformly at random; all genres that occur in more than 1% of the movies are retained. We only use the users' ratings on the sampled movies. In the second scenario, we only sample users that rated at most 40 movies, which greatly reduces the number of ratings for each user and each movie. In the first case, the median number of ratings per user is 60 (the mean, 127); in the second case, the median number of ratings per user is 9 (the mean, 10). In the first case, the median number of ratings per movie is 9 (the mean, 21); in the second case, the median number of ratings per movie is 2 (the mean, 8). In the first case we have $n_1 = 500$ users and $n_2 = 3000$ movies and in the second case we have $n_1 = 750$ users and $n_2 = 1000$ movies. We use a $k = 20$ embedding dimension for both matrices.

The first rating scenario, Figure 4.2, shows that collective matrix factorization improves both prediction tasks—whether a user rated a movie, and which genres a movie belongs to—thus verifying the claim at the beginning of the section. When $\alpha = 1$ the model uses only rating information; when $\alpha = 0$ it uses only genre information.

In the second rating scenario, Figure 4.3, there is far less information in the ratings matrix. Half the movies are rated by only one or two users. Because there is so little information between users, the extra genre information is more valuable. As can be seen by comparing unaugmented genre prediction ($\alpha = 0$) to intermediate values of α, there is no guarantee that mixing information will improve predictive accuracy.

We hypothesized that adding in the roles of popular actors, in addition to genres, would further improve performance. By symmetry the update equation for the actor factor is analogous to the update for the genre factor. Since there are over 100,000 actors in our data, most of which appear in only one or two movies, we selected 500 popular actors (those that appeared in more than ten movies). Under a wide variety of settings for the



(a) Ratings

(b) Genres

Figure 4.2: Test errors (MAE) for predicting whether a movie was rated, and the genre, on the dense rating example. Error bars are 2-standard deviations wide.

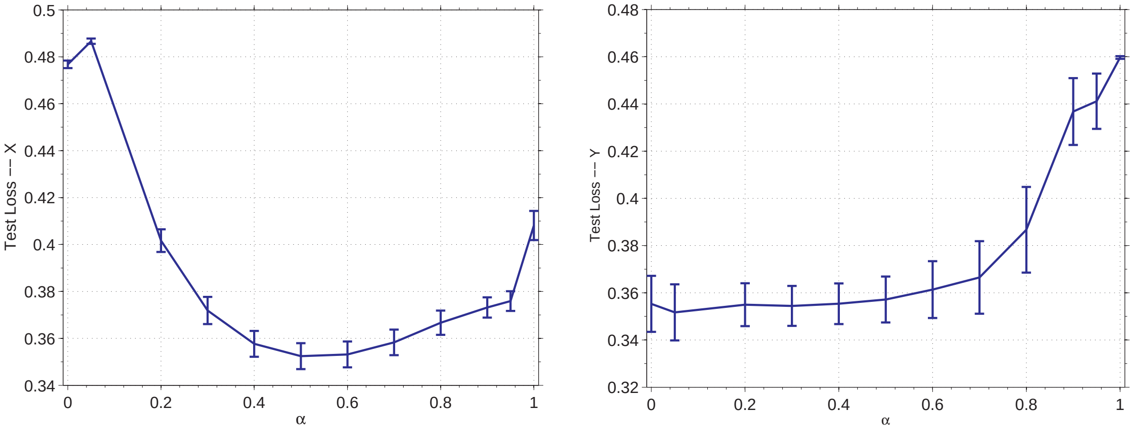
mixing parameters $\{\alpha^{(12)}, \alpha^{(23)}, \alpha^{(24)}\}$ there was no statistically significant improvement, regardless of whether `IsRated` or `Rating` was used.

4.5.2 Stochastic Approximation

Claim: On the augmented collaborative filtering data, stochastic Newton projection can be used to train a model with test error comparable to that of a model trained using Newton projection, while using substantially less computation.

Our claim regarding stochastic optimization is that it provides an efficient alternative to Newton updates in the alternating projections algorithm. Since our interest is in the case with a large number of observed entries in each relation, we use the `IsRated` relation. There are $n_1 = 10000$ users, $n_2 = 2000$ movies, and $n_3 = 22$ of the most common genres in the data set. The mixing coefficient is set close to the optimum value in Figures 4.2(a) and 4.3(a), namely $\alpha = 0.5$. We set the embedding dimension of both factorizations to $k = 30$.

On this three factor problem we learn a collective matrix factorization using both Newton and stochastic Newton methods with batch sizes of 25, 75, and 100 samples per row. The batch size is larger than the number of genres, and so they are all used. Our primary concern is sampling the larger user-movie matrix. Using Newton steps, ten cycles of alternating projection are used; using stochastic Newton steps thirty cycles are used. After



(a) Ratings

(b) Genres

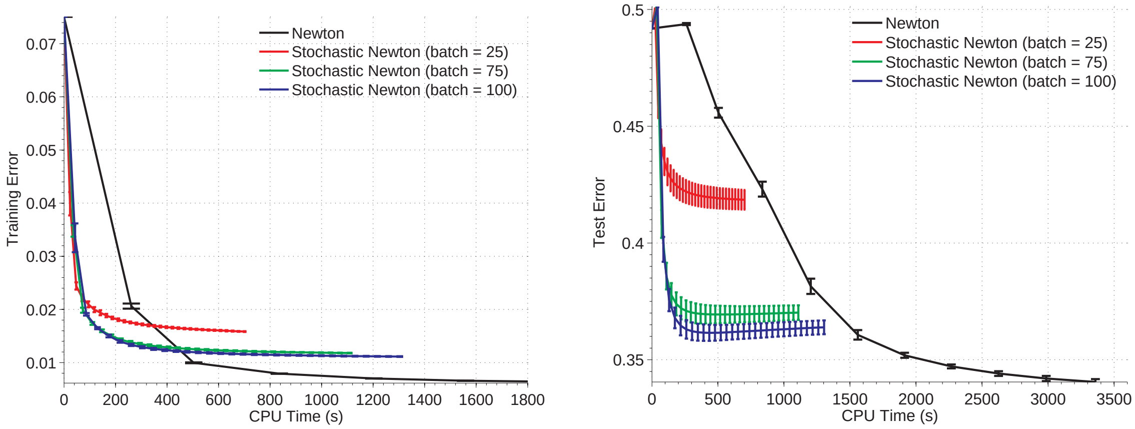
Figure 4.3: Test errors (MAE) for predicting whether a movie was rated, and the genre, on sparse rating example. Error bars are 2-standard deviations wide.

each cycle, we measure the training loss (log-loss) and the test error (mean absolute error), which are plotted against the CPU time required to reach the given cycle in Figure 4.4. The error bars are 2-standard deviations, and are computed by repeating the experiment five times.

Using only a small fraction of the data we achieve results comparable to what full Newton achieves after five iterations. At batch size 100, we are sampling 1% of the users and 5% of the movies; yet our performance on test data is the same as a full Newton step given 8x longer to run. Diminishing returns with respect to batch size suggests that using very large batches is unnecessary.[9]

It should be noted that `rating` is a computationally simpler problem than `israted`. On a three factor problem with $n_1 = 100000$ users, $n_2 = 5000$ movies, and $n_3 = 21$ genres, with over 1.3M observed ratings, alternating projection with full Newton steps runs to convergence in 32 minutes on a single 1.6 GHz CPU. We use a small embedding dimension, $k = 20$, but one can exploit common tricks for large Hessians. We used the Poisson link for ratings, and the logistic for genres; convergence is typically faster under the identity link, although predictive performance suffers.

[9] Even if the batch size were equal to max$\{n_1, n_2, n_3\}$ stochastic Newton would not return the same result as full Newton due to the $1/\tau$ damping factor on the sample Hessian.



(a) Training Loss (Log-loss)

(b) Test Error (MAE)

Figure 4.4: Behaviour of Newton vs. Stochastic Newton on a three-factor model.

4.5.3 Comparison to pLSI-pHITS

Caveat: Collective Matrix Factorization makes no guarantees with respect to generalization error.

In this section we provide an example where the additional flexibility of collective matrix factorization leads to better results; and another where a co-clustering model, pLSI-pHITS, has the advantage.

pLSI-pHITS [24] is a relational clustering technique which is specific to the three entity-type schema $\mathcal{E}_1 \sim \mathcal{E}_2 \sim \mathcal{E}_3$: $\mathcal{E}_1 =$ words and $\mathcal{E}_2 = \mathcal{E}_3 =$ documents, but it is trivial to allow $\mathcal{E}_2 \neq \mathcal{E}_3$.[10] The data matrix $X^{(12)}$ contains frequency of word occurrence in documents; data matrix $X^{(23)}$ contains indicators of whether or not a document cites another. The likelihood maximized by pLSI-pHITS is

$$\mathcal{L} = \alpha \left(X^{(12)} \circ \log \left(UV^T \right) \right) + (1 - \alpha) \left(X^{(23)} \circ \log \left(VZ^T \right) \right), \tag{4.19}$$

where the parameters U, V, and Z contain probabilities:

$$\begin{aligned} u_{i\ell} &= p(x_i^{(1)} \mid h_\ell), \\ v_{i\ell} &= p(h_\ell \mid x_i^{(2)}), \\ z_{i\ell} &= p(x_i^{(3)} \mid h_\ell), \end{aligned}$$

[10] We do not require the citing and cited documents be the same. The two sets in general, need not be the same. Moreover, the words used in the citing documents need not cover the words in the cited documents.

for clusters $\{h_1, \ldots, h_k\}$. Probability constraints require that each column of U, V^T, and Z must sum to one, which induces a clustering of entities. Since different entities can participate in different numbers of relations (*e.g.,* some words are more common than others) the data matrices $X^{(12)}$ and $X^{(23)}$ are usually normalized; we can encode this normalization using weight matrices. The objective, Equation 4.19, is the weighted average of two probabilistic LSI [52] models with shared latent factors h_k. pLSI is one of the single matrix models we describe in Table 3.1.

We sample two instances of `IsRated`, controlling for the number of ratings each movie has. The sampling is as follows: sample a chosen number of users, among those users who have rated at least one movie. Then, retain only the movies rated by at least one selected user. We created a dense data set by sampling 1000 users, In the dense data set, the median number of ratings per movie (user) is 11 (76); in the sparse data set, the median number of ratings per movie (user) is 2 (4). In both cases there are 1000 randomly selected users, and 4975 randomly selected movies, all the movies in the dense data set.

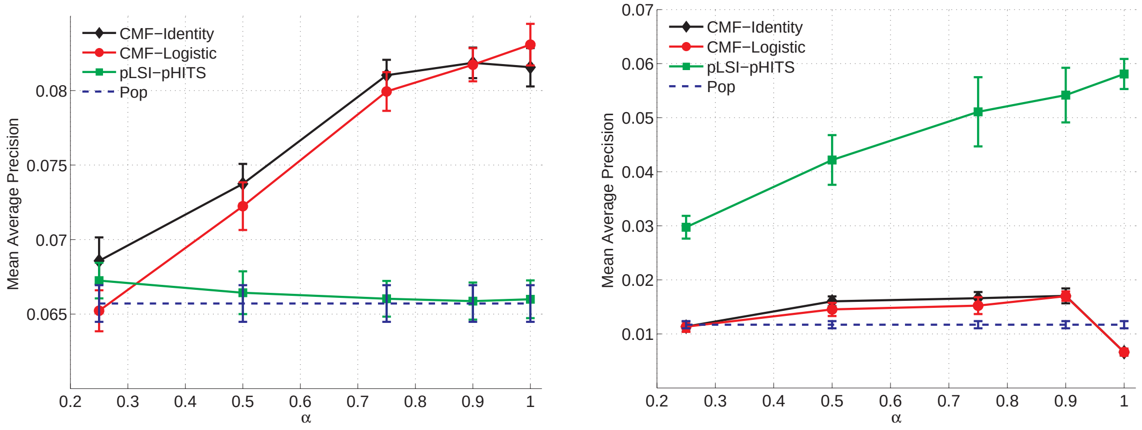
Since pLSI-pHITS is a co-clustering method, and our collective matrix factorization model is a relation prediction method, we choose a measure that favours neither inherently: ranking. We induce a ranking of movies for each user, measuring the quality of the ranking using mean average precision [47]: queries correspond to user's requests for ratings, "relevant" items are the movies of the held-out links, we use only the top 200 movies in each ranking[11], and the averaging is over users. Most movies are unrated by any given user, and so relevance is available only for a fraction of the items: the absolute mean average precision values will be small, but relative differences are meaningful. We compare four different models for generating rankings of movies for users:

- ▷ CMF-IDENTITY: Collective matrix factorization using identity prediction links, $f_1(\theta) = f_2(\theta) = \theta$ and squared loss. Full Newton steps are used. The regularization and optimization parameters are the same as those described in Section 4.5, except that the smallest step length is $\eta = 2^{-5}$. The ranking of movies for user i is induced by $U_{i\cdot}V^T$.

- ▷ CMF-LOGISTIC: Like CMF-Identity, except that the matching link and loss correspond to a Bernoulli distribution, as in logistic regression: $f_1(\theta) = f_2(\theta) = 1/(1 + \exp^{-\theta})$.

- ▷ PLSI-PHITS: Described above. We give the regularization advantage to pLSI-pHITS: the amount of regularization $\beta \in [0, 1]$ is chosen at each iteration using tempered EM. The smaller β is, the stronger the parameter smoothing towards the uniform distribution. We are also more careful about setting β than Cohn et. al. [24], using a decay rate of 0.95 and minimum β of 0.7. To have a consistent interpretation of iterations between

[11] The relations between the curves in Figure 4.5 are the same if the rankings are not truncated.



(a) Dense

(b) Sparse

Figure 4.5: Ranking movies for users on a data set where each movie has many ratings (dense) or only a handful (sparse). The methods are described in Section 4.5.3. Error bars are 1-standard deviation wide.

this method and CMF, we use tempering to choose the amount of regularization, and then fit the parameters from a random starting point with the best choice of β. Movie rankings are generated using the predicted $p(movie \mid user) = p(x^{(2)} \mid x^{(1)})$.

▷ Pop: A baseline method that ignores the genre information. It generates a single ranking of movies, in order of how frequently they are rated, for all users.

In each case the models, save popularity ranking, have embedding dimension $k = 30$ and run for at most 10 iterations. We compare on a variety of values of α, but we make no claim that mixing information improves the quality of rankings. Since α is a free parameter we want to confirm the relative performance of these methods at several values. In Figure 4.5, collective matrix factorization significantly outperforms pLSI-pHITS on the dense data set; the converse is true on the sparse data set. Ratings do not benefit from mixing information in any of the approaches, on either data set.[12] While the flexibility of collective matrix factorization has its advantages, especially computational ones, we do not claim unequivocal superiority over relational models based on matrix co-clustering.

[12] We note that, using a cluster quality criterion, on a different data set, Cohn and Hofmann [24] was able to show that mixing improves prediction on pLSI-pHITS.

Chapter 5

Hierarchical Bayesian Collective Matrix Factorization

5.1 Introduction

In the previous chapter, we defined a fully probabilistic model for sets of related matrices: Collective Matrix Factorization. The basic approach involves simultaneously finding a low-rank representation for a set of related matrices, where shared dimensions in the matrices correspond to shared low-rank factors. We reduced the parameter estimation problem to finding the posterior mode of the parameters, which is referred to as regularized maximum likelihood or maximum a posteriori inference.

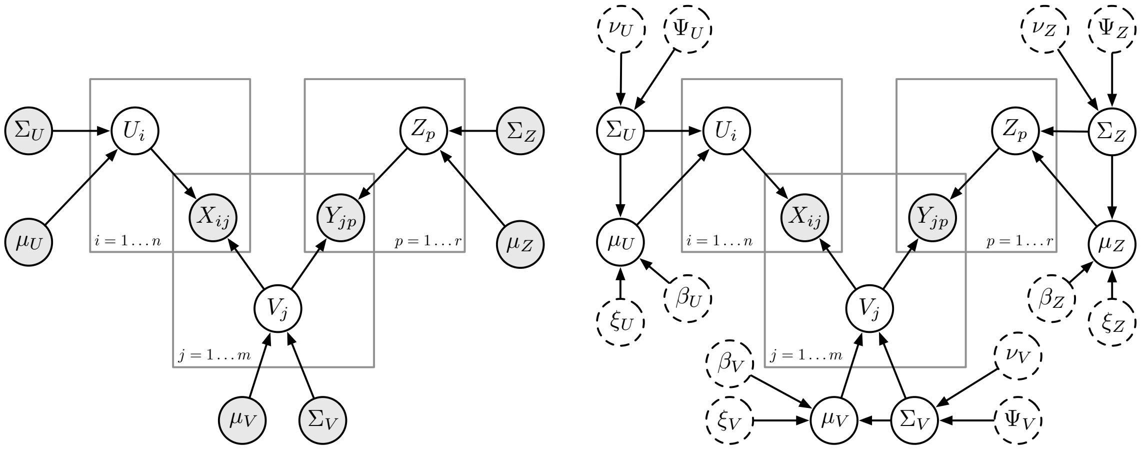
Collective Matrix Factorization inherits both the benefits and some of the limitations of maximum likelihood single matrix factorization: (i) the model is a point estimate that ignores uncertainty in the posterior distribution over parameters. Since data matrices are often sparse, and the standard consistency arguments in favour of maximum likelihood do not apply, the posterior may assign significant mass to many models; (ii) strong generalization (prediction involving new entities) is not statistically well defined (Section 3.9); (iii) the Gaussian prior over factor rows, proposed by Gordon [45], for E-PCA, requires that the value of the hyperparameters Θ be assigned before parameter estimation can begin. If we want to go beyond the naïve zero-mean assumption, we are forced to search over at least a $(k+1)$-dimensional space for Θ (i.e., k parameters for the mean and a spherical covariance matrix). Each point in the grid search over hyperparameter values involves learning a Collective Matrix Factorization on held-out data, which is computationally expensive.

A subtle criticism of Collective Matrix Factorization is that information cannot be easily pooled across rows, or across columns. Since the prior distribution over factor rows is fixed, the only way two rows of a factor can influence each other is indirectly: e.g., two rows $U_{i\cdot}$ and $U_{j\cdot}$, $i \neq j$, can only influence each other through their effect on V. Learning the prior automatically, through a hierarchical prior, allows for information to be pooled across the rows of a factor.

A subtler criticism of Collective Matrix Factorization centers on what sort of information is transferred. Take the three entity-type example, where the posterior distribution is $p(U, V, Z \,|\, X, Y, \Theta)$. Consider the analogous posterior involving only the X data matrix: $p(U, V \,|\, X, \Theta_U, \Theta_V)$. In each case we can compute the marginal distribution of an element of a factor, say $U_{i\ell} \in U$. If we compare the posterior distribution over $U_{i\ell}$ in the single and two-matrix cases, it is clear that the two distributions $p(U_{i\ell} \,|\, X, Y, \Theta)$ and $p(U_{i\ell} \,|\, X, \Theta_U, \Theta_V)$ can, and usually will, differ. Under maximum a posteriori inference, the only difference we see between the two distributions is a difference in the mode: changes in the variance, skew, and other properties of the distribution are not accounted for. The only way Y can affect the prediction of entries in X is in the effect it has on V and U. The only way X can affect the prediction of entries in Y is in the effect it has on V and Z. Changes in the posterior mode account for some of the effect of information sharing between matrices; changes in the posterior distribution account for all of the effect.

A motivating example. Functional Magnetic Resonance Imaging (fMRI) is often used to measure responses in small regions of the brain (i.e., voxels) given external stimuli. Given enough experiments on a sufficiently broad range of stimuli, one can build models that predict patterns of brain activation given new stimuli [83]. Running enough subjects through an fMRI scanner, on a wide variety of stimuli, is costly. However, we can often collect cheap side information about the stimuli. In this chapter, we consider an experiment where the stimulus is a word-picture pair displayed on a screen. We can collect statistics of whether the stimulus word co-occurs with other commonly used words in large, freely available text corpora. By integrating the two sources of information, related through the shared set of stimulus words, we hope to improve the quality of predictive models of brain activation.

Main Contributions: In this chapter, we propose a hierarchical Bayesian variant of Collective Matrix Factorization, which addresses the four criticisms of the maximum likelihood approach, described above. We design a Metropolis-Hastings algorithm for sampling from the posterior by cyclically sampling from Bayesian Generalized Linear Models. Were we to use a naïve random-walk proposal in a factored Metropolis-Hastings, the algorithm would be a standard block MCMC sampler; the Markov Chain would also be slow to mix (i.e., slow training). Using what we already know about efficiently computing the gradient and per-row Hessians of the regularized likelihood, we develop an *adaptive*



(a) Collective Matrix Factorization

(b) Hierarchical (and Hierarchical Bayesian) Collective Matrix Factorization

Figure 5.1: Plate representations of collective matrix factorization (Chapter 4) and its analogue with hierarchical priors (Section 5.2). Shaded nodes indicate observed data or quantities which must be selected. Dashed nodes are fixed parameters. Weight matrices W and $\tilde{W}$ are elided.

proposal distribution for Metropolis-Hastings. The resulting algorithm (i) avoids the need for tedious hand tuning of a large number of proposal distributions; (ii) is fast enough to handle $>$20,000 entities in a few minutes. Unlike many comparable techniques, we make no limiting conjugacy assumptions on our model: every maximum likelihood Collective Matrix Factorization has a Bayesian analogue, which we can train using our adaptive block Metropolis-Hastings sampler. In our experiments, we show that this approach can be used to augment costly fMRI data with inexpensive word co-occurrence data, improving predictions of brain activity even when the voxel being tested never appeared in the training set.

5.2 Hierarchical Collective Matrix Factorization

In this section, we extend Collective Matrix Factorization (Figure 5.1(a)) to include hierarchical priors (Figure 5.1(b)).[1]

[1] We remind the reader that U_i, V_j, and Z_p are factor rows, k-dimensional vectors of random variables; and that μ is a vector of normal mean parameters, and Σ a covariance matrix.

Collective Matrix Factorization requires choosing a good fixed value for the hyperparameters $\Theta = \{\Theta_U, \Theta_V, \Theta_Z\}$, where $\forall F \in \{U, V, Z\}, \Theta_F = \{\mu_F, \Sigma_F\}$. Finding a good value for Θ is a difficult task: even if we assume that the prior means are zero and the prior covariances spherical, cross-validation involves searching over the scales of the three covariance matrices.

Another issue with Collective Matrix Factorization is that information between rows of a factor can only be shared indirectly, through another factor. Consider the plate model in Figure 5.1(a). Using d-separation [94], it is easy to deduce that if only one factor is free (say U) then the rows of that factor are independent of one other. Independence between factor rows allows us to reduce the projection over a large factor matrix into parallel optimizations over each row of that factor. Computationally, the independence of rows in the free factor is useful. Statistically, we want a more direct way of pooling shared behaviour across rows or columns of a matrix.

Since we do not usually have any reason to know the value of the Gaussian hyperparameters Θ, a sensible approach is to treat them as quantities to be learned, placing an appropriate weak prior on them: i.e, a hierarchical model. Hierarchical priors address both the problem of selecting Θ, and the desire for pooling information within a matrix. Moreover, we can design the hierarchical priors so that independence of factor rows is preserved under alternating projections.

The natural approach is to place hierarchical priors on Θ_U, Θ_V, and Θ_Z separately. This approach makes it easy to train the resulting model under alternating projections. Moreover, our approach follows naturally from the nature of row-column exchangeable matrices (Section 3.5.2). Any row-column exchangeable matrix can be viewed either as a collection of exchangeable rows, or as a collection of exchangeable columns. If we consider the matrix factorization $X \approx f(UV^T)$, exchangeable rows justify the use of a hierarchical prior on U; exchangeable columns justify the use of a hierarchical prior on V.[2]

We elect to use the conjugate prior for (μ, Σ): the normal-Inverse-Wishart distribution [38]. The normal-Inverse-Wishart prior on $\Theta_F = \{\mu_F, \Sigma_F\}$ is defined by first sampling the covariance from a Wishart distribution, $\mathcal{W}$, then conditionally sampling the mean from a Gaussian distribution, $\mathcal{N}$:

$$\begin{aligned}\Sigma_F^{-1} &\sim \mathcal{W}(\nu_F, \Psi_F),\\ \mu_F \,|\, \Sigma_F &\sim \mathcal{N}(\xi_F, \Sigma_F/\beta_0),\end{aligned}$$

[2] de Finetti's representation theorem for infinitely exchangeable sequences of random variables is commonly used to motivate hierarchical priors (see Gelman et al. [38], Chapter 5).

The fixed hyperprior parameters $\nu_F > k$, $\Psi_F \in \mathbb{R}_+^{k \times k}$, $\xi_F \in \mathbb{R}^k$, $\beta_0 > 0$ are chosen by the modeler. Our choice of hierarchical priors for matrix factorization mirrors that of Salakhutdinov and Mnih [108], though we do not make the simplifying assumption that the likelihood is conjugate to the hierarchical prior. Choosing other priors that exploit sparsity, block structure, or other properties remains a topic of future work.[3]

In addition to treating things that are unknown as unknown, the hierarchical prior acts as a shrinkage estimator for the rows of a factor. Shrinkage may be especially useful when some of the entities are associated with only a few observations: in the absence of much data we let the low-rank representation of an entity tend towards the population mean. Shrinkage estimators pool information about rows or columns of a matrix, indirectly, through the latent hyperparameters.

It is not hard to generalize our maximum likelihood decomposition approach to the hierarchical case (Algorithm 2). Simply alternate between optimizing the parameters given fixed hyperparameters (Section 4.3.1), and optimizing the hyperparameters given fixed parameters. Given fixed $F \in \mathcal{F}$, the most probable value of (μ_F, Σ_F) is the mode of a normal-Inverse-Wishart distribution with the following parameters:

$$\begin{gathered}\bar{F} = \frac{1}{n_F}\sum_{i=1}^{n_F} F_{i\cdot}, \quad S_F = \sum_{i=1}^{n_F}(F_{i\cdot} - \bar{F})^T(F_{i\cdot} - \bar{F}), \quad \xi_F^* = \frac{n_F}{\beta_F + n_F}\bar{F} + \frac{\beta_F \xi_F}{\beta_F + n_F}, \\ \Psi_F^* = \Psi_F^{-1} + S_F + \frac{\beta_F n_F}{\beta_F + n_F}(\bar{F} - \xi_F)^T(\bar{F} - \xi_F), \quad \nu_F^* = \nu_F + n_F, \quad \beta_F^* = \beta_F + n_F.\end{gathered} \tag{5.1}$$

As discussed earlier in this thesis, this decomposition approach is known as alternating projections in the maximum likelihood case. Below, in Section 5.3.1, we use a similar decomposition of the model parameters to develop a block Metropolis-Hastings sampler.

To derive a hierarchical Bayesian extension to Collective Matrix Factorization, we first consider the hierarchical prior, whose addition to the plate representation of Collective Matrix Factorization is depicted in Figure 5.1(b). In the hierarchical plate model, there are several quantities to consider:

▷ The data variables, $\mathcal{D} = \{X, Y\}$.

▷ The data weights on X and Y: W and $\tilde{W}$, respectively.

▷ The parameters $\mathcal{F} = \{U, V, Z\}$

[3] While we have focused on ℓ_2-regularization, all variants of Collective Matrix Factorization work with any prior whose probability density function is twice-differentiable and decomposable.

- ▷ The hyperparameters, parameters of the prior on $\mathcal{F}$: $\Theta = \{\mu_U, \Sigma_U, \mu_V, \Sigma_V, \mu_Z, \Sigma_Z\}$.
- ▷ Hyperprior parameters, the parameters of the prior on Θ:

  $$\Theta_0 = \{\nu_U, \Psi_U, \xi_U, \nu_V, \Psi_V, \xi_V, \nu_Z, \Psi_Z, \xi_Z, \beta_0\}.$$

  Θ_0 is fixed, chosen by the user. In all our experiments, for each factor F, $\nu_F = k$ is the embedding dimension, Ψ_F is a $k \times k$ identity matrix, $\xi_U = 0$, and $\beta_0 = 1$. This choice of hyperprior parameters encodes a very weak preference for the prior over factor rows to be a k-dimensional Gaussian with mean 0 and identity covariance. Since this is a prior over factor rows, and the number of rows depends on the size of the data matrices, our choice of Θ_0 becomes less consequential as X and Y get larger.

Factoring the posterior distribution over $\{\mathcal{F}, \Theta\}$ according to the hierarchical model in Figure 5.1(b):

$$p(\mathcal{F}, \Theta \,|\, X, Y, W, \tilde{W}, \Theta_0) = p(U, V \,|\, X, W) p(V, Z \,|\, Y, \tilde{W}) \prod_{F \in \{U,V,Z\}} p(F \,|\, \Theta_F) p(\Theta_F \,|\, \Theta_0). \tag{5.2}$$

We shall often need to work in log-space, where Equation 5.2 is equal to

$$\mathcal{O} \;=\; \mathcal{L} + \sum_{F \in \mathcal{F}} \log p(\Theta_F \,|\, \Theta_0). \tag{5.3}$$

The objective for non-hierarchical Collective Matrix Factorization, $\mathcal{L}$, is defined in Equation 4.9. The distribution over Θ_F is the normal-Inverse-Wishart, defined above. Given that we have already derived the gradient and Hessian of $\mathcal{L}$, deriving the equations for the gradient and Hessian of $\mathcal{O}$ are straightforward.

5.2.1 Generalization to an Arbitrary Number of Relations

As with the previous chapter, we focus on the two matrix case. However, Algorithms 2-4 are presented for the general case, which can involve any number of related matrices. Here, we remind the reader of the general notation. Entity-types, the different kinds of relation arguments, are indexed by $i = 1 \ldots t$. The number of entities of type i in the training set is denoted n_i. A matrix corresponding to a relation between entity-types i and j is denoted $X^{(ij)}$. Each relation matrix is represented as the product of low-rank factors:

$$X^{(ij)} \approx f^{(ij)} \left(U^{(i)} \left(U^{(j)} \right)^T \right),$$

where $f^{(ij)}$ is the element-wise link function that transforms the low-rank latent representation into predictions. Each factor $U^{(i)}$ has its own Gaussian prior, defined over factor rows. In the hierarchical case, each of these priors is assigned a normal-Inverse-Wishart hyperprior.

5.3 Bayesian Inference for the Hierarchical Model

A limitation of both collective matrix factorization and its hierarchical extension is that only point estimates are possible. There may be substantial uncertainty in the parameter estimates, especially when each entity only participates in a few relationships. Failing to model this uncertainty often leads to worse accuracy at prediction. Unlike most debates on the merits of maximum likelihood vs. Bayesian estimation, the former cannot invoke asymptotic consistency here: even in the limit of infinite data, the number of parameters grows with the number of entities.

Another limitation of maximum likelihood Collective Matrix Factorization is that it is of limited value in predicting behaviour of a new entity, one not present in the training data. The standard approach to "folding-in" a new row in a data matrix involves finding the maximum likelihood solution for the low-dimensional representation of the new row, fixing the previously trained parameters and hyperparameters (Section 5.3.3). The bilinear nature of the model makes fixing one factor to its maximum likelihood value problematic. Consider folding-in a new row of a data matrix $X_{new,\cdot}$. A particular value of V defines a distribution over the k factors for each column of X. If we fit $U_{new,\cdot}$ conditioned on a particular fixed value of V, then we are assuming that the chosen value of V is the true distribution over factors for each column entity. In practice, there may be many plausible values of V that differ enough to affect the predictions $\hat{X}_{new,\cdot} = f_1(U_{new,\cdot}V^T)$. Fixing V to a single value ignores the possibility of other plausible models that may better fit the behaviour of a particular row entity.[4]

To address these concerns, we propose a fully Bayesian approach to training the hierarchical model introduced in Section 5.2. Instead of approximating the posterior in Equation 5.2 by its mode, we approximate it using multiple samples drawn from it.

[4] This argument is akin to the one that is used to explain why Latent Dirichlet Allocation is better at prediction on new documents than pLSI. That latter assumes a fixed distribution over topics; the former integrates over the topic simplex, a distribution over distributions of topics.

5.3.1 Block Metropolis-Hastings

In finding the maximum a posteriori solution for hierarchical CMF, we exploited structure in the large number of unknowns $\{\mathcal{F}, \Theta\}$ to make the optimization tractable. First we alternated between optimizing over $\mathcal{F}$ and Θ. When optimizing over $\mathcal{F}$, we alternated between optimizing each factor $F \in \mathcal{F}$. When optimizing over a factor F, we reduce the computation to parallel updates over each row of F, and so the hardest part of inference is optimizing a convex function over $k \ll \min\{m, n, r\}$ variables. In Newton-projection over a factor row, $F_{i\cdot}$, the core operation involves finding a step towards the most likely value of $F_{i\cdot}$, given that the other parameters are fixed.

In a block Metropolis-Hastings sampler (Section 2.5.1) we might instead consider sampling from the conditional distribution of $F_{i\cdot}$, given that the other parameters are fixed. If it is feasible to sample from this distribution, or to sample from an approximation of this distribution, we have arrived at a block Markov Chain Monte Carlo sampler.

Let $\Omega = \{\mathcal{F}, \Theta\}$ denote the parameters we wish to sample over. We choose the following conditional sampling distributions for block Metropolis-Hastings[5],

$$\forall F \in \{U, V, Z\}\ \forall i = 1 \dots n_F :\ F_{i\cdot} \sim p(F_i \,|\, \Omega - F_{i\cdot}), \tag{5.4}$$

$$\forall F \in \{U, V, Z\} :\ \Theta_F \sim p(\Theta_F \,|\, \Omega - \Theta_F). \tag{5.5}$$

Sampling directly from the distribution in Equation 5.5 is straightforward: the distribution of Θ_F is normal-Inverse-Wishart, so the conditional sampling distribution has a closed form (Section 5.2). In contrast, the form of the conditional sampling distribution in Equation 5.4 is unknown, unless we assume that X and Y are Gaussian: i.e., the data distributions and factor priors are mutually conjugate.

5.3.2 Hessian Metropolis-Hastings

If we assume that the elements of X and Y are Gaussian, then Metropolis-Hastings can be reduced to Gibbs sampling, leading to a multiple-matrix generalization of Bayesian Probabilistic Matrix Factorization [109].

However, in almost all the examples of information integration we consider, the entries of different data matrices have different response types: e.g., in the fMRI domain, word co-occurrence is binary, but voxel activation is continuous. We do not wish to sacrifice the flexibility of our model, which supports multiple response types, to reap the

[5] We note that the way parameters are grouped in block Metropolis-Hastings is similar to the way they are grouped in alternating Newton projections.

benefits of Bayesian inference. Therefore, we consider more general Metropolis-Hastings approaches.

In Metropolis-Hastings, one often resorts to sampling from a Gaussian proposal distribution whose mean is the sample at time t, $F_{i\cdot}^{(t)}$, with covariance matrix $v_i \cdot I$. This forces the user to choose v_i for each row in a way that causes the underlying Markov chain to mix quickly. Tuning one proposal distribution is tedious; tuning a proposal distribution for each entity is masochistic. If the Hessian of the target distribution, with respect to $F_{i\cdot}^{(t)}$, is far from spherical, then the rate at which the underlying Markov chain mixes can be slow, regardless of how well v_i is tuned.

The distribution in Equation 5.4 may not be easy to sample from; but given a point, namely $F_{i\cdot}^{(t)}$, we can easily compute the local gradient and Hessian of the distribution with respect to $F_{i\cdot}$. By using the gradient and Hessian, we can create a proposal distribution that better approximates $p(F_{i\cdot} \mid \Omega - F_{i\cdot})$.

In the case of Metropolis-Hastings for a Bayesian Generalized Linear Model, one technique that uses both the gradient and Hessian to dynamically construct the proposal is Hessian Metropolis-Hastings (HMH) [98]. To define a Metropolis-Hastings sampler, we need to define the forward sampling distribution, from which a proposal value $F_{i\cdot}^{(*)}$ is drawn given the previous value in the chain, $F_{i\cdot}^{(t)}$. The choice of a forward sampling distribution leads to a corresponding backward sampling distribution, which defines the probability of returning to $F_{i\cdot}^{(t)}$ from the proposal value $F_{i\cdot}^{(*)}$. The forward sampling distribution in HMH is a Gaussian whose mean is determined by taking one Newton step from $F_{i\cdot}^{(t)}$, and whose covariance is derived from the Hessian used to take the Newton step. The backward sampling distribution in HMH is a Gaussian whose mean is determined by taking one Newton step from $F_{i\cdot}^{(*)}$, and whose covariance is derived from the Hessian used to take the Newton step. Instead of searching over step lengths η, we sample from a fixed distribution over step lengths (here, uniformly at random). Algorithms 3 and 4 describe the Hessian Metropolis-Hastings sampler for Equation 5.4.

Random-walk Metropolis-Hastings takes a dim view of what one can do with a sampling distribution: it assumes only that we can efficiently evaluate probabilities, up to a normalizing constant. If we assume we know nothing about the structure of the sampling distribution, then there is little we can do with the Gaussian forward and backward sampling distributions other than pick a distribution with appropriate variance, centered at the required point. However, since we have a efficient Newton-projection for finding the mode of $p(F_{i\cdot} \mid \Omega - F)$, we should use it to pick a proposal that is closer to the mode. After all, for any smooth distribution on $F_{i\cdot}$, the mode is in a region of high probability.

5.3.3 Folding-in

Training only provides us with a low-rank representation of entities in the training set. To deal with new entities we must "fold-in" a low-rank representation of that entity. In maximum likelihood techniques, we have a fixed set of parameters and hyperparameters, $\{\mathcal{F}, \Theta\}$, which makes the process straightforward. Assume we are adding a row to X, $X_{new,\cdot}$, and that $U_{new,\cdot}$ has been initialized[6]:

1. Fix the value of $\{\mathcal{F}, \Theta\}$ to the value which was learned during training. The value of $U_{new,\cdot}$ remains unfixed.
2. Using $X_{new,\cdot}$ and the fixed $\{\mathcal{F}, \Theta\}$, estimate $U_{new,\cdot}$. This can be done using several steps of Newton projection. Essentially, we are training a GLM where $U_{new,\cdot}$ are the weights.

The process is similar when Bayesian inference is used, except that instead of maximizing the value of $U_{new,\cdot}$, we sample several times from the posterior distribution

$$p(U_{new,\cdot} \,|\, \mathcal{D}, \mathcal{F}, \Theta, X_{new,\cdot}),$$

and average the predictions from each sample.

5.3.4 Bayesian Prediction

Once we have the low-rank representation for an entity (either from training for hold-out, or by folding-in for new entities) we want to predict the value of relations that entity participates in. We consider predicting entry X_{ij}. For a point estimate the prediction is simply $\hat{X} = f_1(U_{i\cdot} V_{j\cdot}^T)$. Given a posterior distribution over many models, we must average over the models:

$$p(X_{ij} \,|\, \mathcal{D}) = \int p(X_{ij} \,|\, \mathcal{F}, \Theta) p(\mathcal{F}, \Theta \,|\, X, Y, W, \tilde{W}) \, d\{\mathcal{F}, \Theta\}. \tag{5.6}$$

Equation 5.6 is known as the posterior predictive distribution. Since we only possess samples from the posterior distribution, we use a Monte Carlo approximation to the posterior predictive:

$$p(X_{ij} \,|\, \mathcal{D}) = \frac{1}{S} \sum_{s=1}^{S} p(X_{ij} \,|\, \mathcal{F}^{(s)}, \Theta^{(s)}), \tag{5.7}$$

[6] We initialize factor rows by drawing from the prior distribution over the factor's rows.

where $\{\mathcal{F}^{(s)}, \Theta^{(s)}\}$ is a sample from the posterior distribution, generating using Metropolis-Hastings. We control the number of samples used, S. In our experiments, $S = 10$ on hold-out experiments; $S = 5$ on fold-in experiments. The predictive performance was not significantly improved by using more samples.

5.3.5 Hypothesis-Specific Bayesian Model Averaging

Given the substantial effort involved in designing hierarchical Bayesian Collective Matrix Factorization and the adaptive Metropolis-Hastings algorithm, the reader may question whether the prima facie case for Bayesianism is sufficiently compelling. The case for Bayesianism rests on two points:

▷ That the posterior distribution over the unknowns $\{\mathcal{F}, \Theta\}$ is highly uncertain, assigning significant mass to many different models.[7]

▷ That generalizing to new entities using the posterior distribution is not subject to Welling's criticism of matrix factorization models that generalize to new entities using only the posterior mode (Section 3.9).

The posterior predictive distribution, discussed in the previous section, is the standard Bayesian solution for dealing with posterior uncertainty. Using the posterior predictive distribution is precisely Hypothesis-Specific Bayesian Model Averaging (Section 2.5.2) where the composite hypothesis, $\mathcal{H}$, is the set of distributions defined by the plate model in Figure 5.1(b). The unknowns $\{\mathcal{F}, \Theta\}$ index the distributions within $\mathcal{H}$.

Hypothesis-Specific Bayesian Model Averaging (HS-BMA) is a way of improving the quality of predictions given $\mathcal{H}$.[8] HS-BMA can be viewed as a form of soft model selection, picking the best element of $\mathcal{H}$ subject to uncertainty in the unknowns. In most applications of HS-BMA, $\mathcal{H}$ is fixed with respect to the data, and consistency often holds. In Bayesian matrix factorization models, such as the ones discussed in this chapter, the number of parameters required to describe an element of $\mathcal{H}$ depends on the size of the training data.

All low-rank matrix factorizations can suffer from substantial uncertainty in their unknowns; it is the price we pay to model entity-specific interactions. At first blush, a ma-

[7] The posterior for a matrix factorization can assign mass to many different models that differ only in a permutation of the columns of $\mathcal{F}$. Our concern with model uncertainty is the existence of multiple high-scoring models that are not identical up to a permutation of the factors.

[8] If one wanted to be pedantic, $\mathcal{H}$ could be added as a conditioning event to all of the distributions discussed in this chapter. Likewise, in Chapter 4, a composite hypothesis for Collective Matrix Factorization could be added as a conditioning event to each distribution involving that model.

trix looks very much like a table of attribute-value data, which consists of a fixed set of attributes (columns) and an exchangeable collection of data points (rows). The critical difference between an attribute-value model and its matrix analogue (e.g., generalized linear models and Exponential Family PCA) is that the former assigns parameters to attributes; the latter assigns parameters to entity-attribute interactions. A consequence of this difference is that while attribute-value models can often provide consistency guarantees, matrix factorizations do not. The advantage of matrix factorization models is that they can represent dyadic entity-attribute interactions; attribute-value models cannot.

The intuitive meaning of a consistency guarantee in a Bayesian procedure is that the posterior distribution over unknowns converges to a point mass on the true generating model, given sufficient data.[9] While consistency holds for many probabilistic attribute-value models, it does not hold for matrix factorization models: $\mathcal{H}$ is dependent on the training data. The observations in a matrix factorization are entries in the data matrix, and so the only way the number of observations can tend to infinity is if (i) one dimension is fixed and the other grows; (ii) both rows and columns grow without bound. In case (i), let the columns be fixed. Even though the columns are fixed, they are not attributes: matrix factorizations assume that each new row has a unique dyadic interaction with each of the (fixed) columns. In case (ii), it is even clearer that the parameter space is growing with the number of observations. Under these circumstances, one should expect substantial uncertainty, at least over $\mathcal{F}$. We do not know whether consistency holds for the posterior over Θ; the analysis is complicated by the fact that the parameters model row-column exchangeable observations (matrices) instead of exchangeable observations (data points).

The case for Hypothesis-Specific Bayesian Model Averaging is particularly strong in low-rank matrix factorization. While we could do model selection using the posterior mode, there will always be significant posterior uncertainty over the low-rank factors, regardless of how large the data matrices get. The problem of posterior uncertainty can be even worse if the data matrices are sparsely observed. We want to do model selection over $\{\mathcal{F}, \Theta\}$, but we also want to be mindful of posterior uncertainty in the unknowns, especially since we cannot eliminate such uncertainty by throwing more data at the problem. Using the posterior predictive distribution allows for a compromise, i.e., soft model selection. In our experiments (Section 5.4) we find that soft model selection yields better predictions than model selection over our choice of $\mathcal{H}$: the Bayesian solution outperforms the maximum a posteriori solution.

[9] We assume that $\mathcal{H}$ contains the true generating model, and that the true generating model is assigned non-zero probability in the prior.

Variants of Hypothesis-Specific Bayesian Model Averaging

We choose to integrate out the uncertainty in both low-rank factors $\mathcal{F}$ and hyperparameters Θ in the posterior predictive distribution (Equation 5.6). One may choose to treat the unknowns differently: e.g., integrate out the uncertainty over $\mathcal{F}$; but use only the posterior mode for Θ. In our setting, the update for Θ is an analytic Gibbs step. There are variants of the model we consider where the update for Θ is not analytic, and using posterior mode may have computational advantages: e.g., if the low-rank factors were subject to a stochastic or non-negative constraint, with the same Gaussian prior over factor rows.

5.3.6 Exploiting Decomposition

There is a strong similarity between alternating Newton-projection for maximum likelihood inference in hierarchical Collective Matrix Factorization, and the adaptive block Metropolis-Hastings algorithm we propose for hierarchical Bayesian Collective Matrix Factorization. The underlying graphical model for both approaches is the same (Figure 5.1(b)).

One way of viewing the decomposition is that learning a bilinear form, UV^T and VZ^T, is reduced into cyclically learning linear forms: e.g., $U_{i\cdot}$ when V is fixed. Learning a linear model is a well-studied problem in machine learning, and we take advantage of that in this chapter. Once the decomposition is established, we create a Bayesian algorithm for collective matrix factorization by simply replacing a ℓ_2-regularized Generalized Linear Model (GLM) [78] with its Bayesian analogue. Hessian Metropolis-Hastings was first proposed for solving Bayesian GLMs; we extend its use to Bayesian collective matrix factorization.

We note that there are other ways that we may productively use ideas from linear models. In MAP inference we have found that sample-based estimates of the gradient and Hessian increase the initial rate of convergence during training [116]. We speculate that using stochastic gradients and Hessians in HMH may be useful in accelerating training, in much the same manner that mini-batches are used to accelerate other MCMC techniques. Moreover, the lack of a guarantee on decrease in the objective at each iteration of stochastic projection does not affect the convergence guarantees of Metropolis-Hastings: a bad step can affect the acceptance rate, but not convergence guarantees.

Algorithm 2: Decomposition Algorithm for MAP and Bayesian Inference

```
Input: Data matrices, one per relation $\{X^{(ee')}\}$. The model (embedding dimension $k$, the choice of
    exponential family for the entries of each matrix, and the hyperparameters if they are fixed in
    advanced).
Output: Low rank factors for each entity-type: $U^{(1)} \dots U^{(E)}$.
for $e = 1 \dots E$ do
    Initialize $U^{(e,0)}$ using Algorithm 3 with prior mean $\mu_e = 0$ and $\Sigma_e = I$. This initialization
    works well for either MAP or Bayesian inference.
end
while not converged do
    for $e = 1 \dots E$ do
        foreach row of $U^{(e,t)}$ : $U_{i\cdot}^{(e,t)}$ do
            Update $U_{i\cdot}^{(e,t+1)}$ using $U_{i\cdot}^{(e,t)}$ and all the observations involving the current entity: i.e.,
            $\forall e', e, X_{i\cdot}^{(ee')}$. For MAP inference the update is a convex optimization, implemented as
            a Newton step; for Bayesian inference we sample from the conditional sampling
            distribution (Equation 5.4) using Hessian Metropolis-Hastings (Algorithm 4).
        end
        If the hyperparameters for factor $U^{(e)}$, i.e., $(\mu_e, \Sigma_e)$, are not fixed then compute the
        normal-Inverse-Wishart posterior with parameters defined in Equation 5.1. Use the
        posterior mode for MAP, or a Gibbs sample for Bayesian inference.
    end
    $t = t + 1$
end
```

5.4 Experiments

Mapping fMRI data into related matrices: In this section, we show that our Bayesian approach is significantly better than the MAP alternative, using the fMRI+word problem outlined in the introduction. The data collection protocol is described in detail in Mitchell et al. [83]. The stimuli, the shared entities, consist of word-picture pairs flashed on a screen (e.g., bear, barn, pliers). The stimuli are chosen to be representative of categories (e.g., animals, buildings, tools). Nine subjects are presented with each of sixty stimuli. Each patient was presented with each stimulus six times. For each stimulus word we generated the average response of each voxel over the 9×6 trials.[10] This procedure results in the Response$(stimulus, voxel)$ relation. There are initially >20,000 voxels, which we restrict to a set of 500 voxels deemed to be most stable [83]. The Co-occurs$(words, stimulus)$ relation is collected by measuring whether or not the

[10] We acknowledge, but do not correct for, registration discrepancies in fMRI imaging. Each brain has a different shape, which must be mapped onto a canonical coordinate system (canonical brain map). The fMRI images we average over have been registered (for details, refer to Mitchell et al. [83]).

```
Algorithm 3: Initialization for Hessian Metropolis-Hastings
  Input: Number of entities of type $e$, $n_e$. Prior mean $\mu_e$ and covariance $\Sigma_e$.
  Output: Initial value for low-rank factor $U^{(e,0)}$. Mean and negative precision matrix for the first
          forward sampling distribution: $\bar{U}_{i\cdot}^{(e,0)}$ and $\nabla^2 \mathcal{O}\left(U_{i\cdot}^{(e,0)}\right)$.
  for $i = 1 \dots n_t$ do
      Sample from the prior: $U_{i\cdot}^{(e,0)} \sim \mathcal{N}\left(\mu_e, \Sigma_e\right)$.
      Choose a random step length: $\eta \sim \mathcal{U}[0,1]$.
      Compute gradient and Hessian. Estimate posterior mean using one Newton step:
        $\bar{U}_{i\cdot}^{(e,0)} = U_{i\cdot}^{(e,0)} + \eta \cdot \left[\nabla \mathcal{O}\left(U_{i\cdot}^{(e,0)}\right)\right] \left[\nabla^2 \mathcal{O}\left(U_{i\cdot}^{(e,0)}\right)\right]^{-1}$.
  end
```

```
Algorithm 4: Hessian Metropolis-Hastings
  Input: Previous sample from the Markov chain, $U_{i\cdot}^{(e,t)}$. Observations involving entity $i$.
  Output: Next sample: $U_{i\cdot}^{(e,t+1)}$. Mean and negative precision of the next proposal.
  Sample the proposal: $U_{i\cdot}^{(e,*)} \sim \mathcal{N}\left(\bar{U}_{i\cdot}^{(e,t)}, \left[-\nabla^2 \mathcal{O}\left(U_{i\cdot}^{(e,t)}\right)\right]^{-1}\right)$.
  Compute the gradient $[\nabla \mathcal{O}(U_{i\cdot})]$ and Hessian $\left[\nabla^2 \mathcal{O}(U_{i\cdot})\right]$ at $U_{i\cdot}^{(e,*)}$.
  Estimate the posterior mean using one Newton step with random step length:
    $\bar{U}_{i\cdot}^{(e,*)} = U_{i\cdot}^{(e,*)} + \eta \cdot \left[\nabla \mathcal{O}\left(U_{i\cdot}^{(e,*)}\right)\right] \left[\nabla^2 \mathcal{O}\left(U_{i\cdot}^{(e,*)}\right)\right]^{-1}$, where $\eta \sim \mathcal{U}[0,1]$.
  Compute the acceptance probability
    $\alpha = \min \left\{ 1, \frac{p\left(U_{i\cdot}^{(e,*)} | \mathcal{D}, F^{(t)}, \Theta^{(t)}\right)}{p\left(U_{i\cdot}^{(e,t)} \,|\, \mathcal{D}, \mathcal{F}^{(t)}, \Theta^{(t)}\right)} \times \frac{\mathcal{N}\left(U_{i\cdot}^{(e,t)} | \bar{U}_{i\cdot}^{(e,*)}, \left[-\nabla^2 \mathcal{O}\left(U_{i\cdot}^{(e,*)}\right)\right]^{-1}\right)}{\mathcal{N}\left(U_{i\cdot}^{(e,*)} | \bar{U}_{i\cdot}^{(e,t)}, \left[-\nabla^2 \mathcal{O}\left(U_{i\cdot}^{(e,t)}\right)\right]^{-1}\right)} \right\}$.
  if $r \sim \mathcal{U}[0,1] \leq \alpha$ then $k = *$ else $k = t$.
  Update: $U_{i\cdot}^{(e,t+1)} = U_{i\cdot}^{(e,k)}$, $\bar{U}_{i\cdot}^{(e,t+1)} = \bar{U}_{i\cdot}^{(e,k)}$, $\nabla^2 \mathcal{O}\left(U_{i\cdot}^{(e,t+1)}\right) = \nabla^2 \mathcal{O}\left(U_{i\cdot}^{(e,k)}\right)$.
```

stimulus word occurs within five tokens of a word found in the Google Teraword corpus [14]. Of these words, we select 20,000 at random. The relations map into two matrices: X = `Co-occurs` and Y = `Response`. Throughout our experiments we assume that X_{ij} is Bernoulli distributed, and that Y_{ij} is Gaussian. The embedding dimension is $k = 25$ throughout.

Evaluation Criteria: The first type of experiment is hold-out prediction: the entities are fixed, and we hold-out entries of X and Y. After learning the model $\mathcal{F}$ or $\{\mathcal{F}, \Theta\}$, we use it to predict the held-out entries of X and Y. In the Bayesian case, we average the response from several sampled models. The second type of experiment is fold-in prediction: instead of holding out entries, we hold-out entire words and voxels. After learning a model using the remaining rows of X and columns of Y, we "fold-in" the held-out entities. Since we held-out entities, we need to find their low-rank representation under each model. The observations involving the held-out entity are split: two-thirds are used to fold the entity into the model; the rest are used to estimate test error. In both hold-out and fold-in experiments, a measure of test error is required: we use mean squared error, a natural choice given that the voxel responses are modelled as Gaussians.

Models Compared: We compare three variants of Collective Matrix Factorization:

▷ Collective Matrix Factorization (CMF), where the model is the most likely value of $\mathcal{F}$, given fixed Θ. We discuss how Θ is chosen below.

▷ Hierarchical Collective Matrix Factorization (H-CMF), where the model is the maximizer of Equation 5.3 over $\{\mathcal{F}, \Theta\}$.

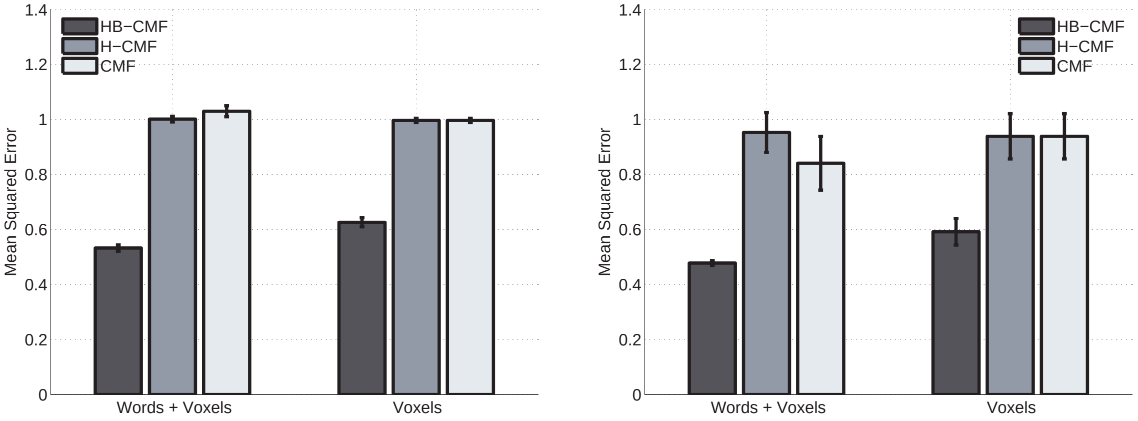
▷ Hierarchical Bayesian Collective Matrix Factorization (HB-CMF), where the model is the posterior distribution over $\{\mathcal{F}, \Theta\}$: Equation 5.2. Since the posterior lacks a closed form, we cannot directly compute the posterior predictive distribution. Instead, predictions are made using samples of $\mathcal{F}$ and a Monte Carlo estimate of the posterior predictive distribution.

The difference between CMF and H-CMF is the addition of an automatically learned hierarchical prior. The difference between H-CMF and HB-CMF is the difference between maximum likelihood and Bayesian inference on the same underlying model (Figure 5.1(b)). In each of the three approaches, folding-in consists using the mode or samples from the posterior over new factors to make predictions.

For CMF, we must choose a diagonal covariance matrix for each factor F, Σ_F. We propose a psychic initialization of the covariance: let the j^{th} diagonal element of Σ_F be the variance of the j^{th} column of the estimate of F produced by H-CMF.[11] The variants of Collective Matrix Factorization are generalizations of ones we will discuss in Chapter 6 (see Table 6.1): on the Voxel+Word data CMF = MRMF = SoRec, where the regularizer is carefully tuned; on the Voxel-only data HB-CMF $\approx$ BXPCA: the likelihoods are the same, but we do not assume a conjugate prior over V, and the Gaussian priors have diagonal covariance; and CMF = PMF, except that we use full covariance matrices for the Gaussian prior.

The results for the hold-out experiment (Figure 5.2(a)) indicate the value of the Bayesian approach. The only difference between HB-CMF and H-CMF is that the former is learned

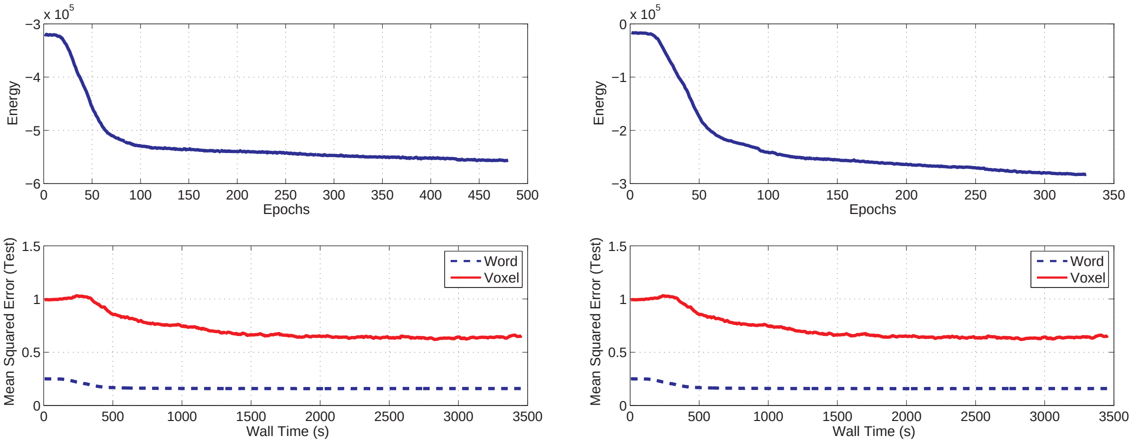
[11] CMF requires a good fixed prior for a fair comparison. A computationally expensive approach for finding a fixed prior would be cross-validation. Our approach is computationally simpler: use an empirical estimate of the prior parameters.



(a) Hold-out

(b) Fold-in

Figure 5.2: Performance on predicting `Response(stimulus,voxel)` using just the voxel response (Voxel), and augmenting it with word co-occurrence data (Words + Voxels). The bars represent algorithms discussed: CMF (Chapter 4), H-CMF (Section 5.2), HB-CMF (Section 5.3). The error bars are 2-standard deviations.



(a) Hold-out

(b) Fold-in

Figure 5.3: Mixing behavior of Algorithm 4. The slowest mixing instance of the hold-out and fold-in experiments are shown. Each point on the energy vs. epochs plots (top) measures the loss of a sample. Each point on the test error vs. time plots (bottom) measures the test error of a sample on predicting word co-occurrence or voxel activation.

using Bayesian inference: handling posterior uncertainty leads to vastly better predictions. In addition, a statistically significant improvement is achieved by augmenting with the word data. The results for the fold-in experiment (Figure 5.2(b)) show an even larger difference between HB-CMF and its competitors. Perhaps surprisingly, we are able to achieve high prediction accuracy when testing on voxels in the brain that never appeared in the training set. Our Bayesian fold-in procedure was able to infer the value of unobserved voxels simply by looking at their correlations with existing voxels. The training and test voxels are drawn from both hemispheres, with weak spatial correlation—voxels from regions associated with visual processing tended to be more stable. Again, there is a statistically significant improvement in augmenting the data with word co-occurrence.

As a baseline measure, we can consider the mean over all the voxel responses (entries in Y) as a default predictor. Since the entries in the Y matrix have been standardized to zero mean Gaussians, the mean predictor is $Y_{ij} = 0$, which yields a mean squared error equal to the variance, i.e., 1. Comparing HB-CMF to H-CMF, where the only difference is learning the posterior distribution vs. learning the posterior mode, there is a clear advantage to being Bayesian.

The reader may rightly question whether the use of non-Gaussian data distributions is worth the additional complexity of Metropolis-Hastings over Gibbs sampling. If we take HB-CMF, the best performing algorithm, and replace the non-Gaussian prediction links with Gaussian ones, prediction accuracy is 26% worse on hold-out prediction, and 39% worse on fold-in prediction. Thus, there is a statistically significant benefit to using non-Gaussian data distributions on the fMRI+word data.

Performance: The hierarchical Bayesian method is vastly better at the predictive task than its non-Bayesian analogues; it also scales well, for a Bayesian method. Sampling the parameters and hyperparameters in HB-CMF takes, on average, $7.2s$ for the hold-out experiment using a parallel implementation on four processors.[12] Our implementation was developed in MATLAB, using the Distributed Computing Environment toolkit and its implementation of parallel-for. An analogous iteration of H-CMF takes $< 10s$. That said, H-CMF converges in less than 20 iterations; HB-CMF takes over 100 iterations to converge. The practicality of the Bayesian approach depends largely on how many iterations are required (Figure 5.3). The energy vs. epochs plots suggest that the underlying Markov chain over states mixes quickly.

The premise of this dissertation is that the world abounds with entities that are related, but not by any readily apparent set of logical rules. Our matrix factorization approach allows for information integration in such scenarios. However, we are limited by the use of

[12] The four processors are cores on an AMD Opteron 2384 CPU, operating at 2.7GHz.

point estimators. A Bayesian alternative is desirable, but we need an approach that scales to large matrices, without having to choose between artificial conjugacy assumptions or painstaking tuning of a proposal distribution. By decomposing the problem into cyclic updates over linear models, and using what we know about efficient MAP estimation, we can be Bayesian without compromise: the model need not be simplified; the operator need never tune a proposal; and the approach scales linearly with the number of entities, with almost no random walk behaviour in sampler.

Chapter 6

Literature Survey

6.1 A (very) brief overview of relational learning

Relational learning is too vast in scope, and too varied in its goals, to survey to more than a modicum of completeness. Instead, we begin by briefly discussing approaches to relational data that we believe stand apart from the focus of this thesis:

▷ **Graph-based techniques**: Graphs can be used to encode relational data: nodes correspond to entities; links correspond to binary relations. Such representations deal with data where there is one arity-two relation involving one entity-type. Even in the single matrix case, we allow for relations involving two different entity-types (one indexed by rows, another indexed by columns). In collective matrix factorization, we allow for multiple types of relations, involving potentially many different entity-types. From the perspective of flexibility of relational representation, the closest graph-based approaches are those that deal with attributed graphs (e.g., Tong et al. [127], Minkov and Cohen [82], Singh et al. [117]). These works draw much of their inspiration from PageRank [92], which uses a random walk over a graph to induce rankings of nodes. Even in attributed graph mining the data is assumed to be static: observed edges are a fixed set, and nodes do not appear over time. Our matrix-based models allow for relations/edges to appear during training (i.e., online learning). Moreover, hierarchical Bayesian collective matrix factorization defines a fully generative model of entities, allowing new entities/nodes to be easily added to the data set.

▷ **Propositionalization**: Refers to approaches that first convert a relational data set into a tabular, or attribute-value, data set (e.g., Kramer et al. [59]). The transformed data set is then given to a standard attribute-value learning algorithm for prediction or clustering: e.g., logistic regression. Some techniques, like structural logistic regression [97], treat propositionalization as part of the learning problem. In logic-based representations of the data, propositions may take the form of rules; the value of the corresponding attribute is true or false. In graph-based representations of the data, propositions may take the form of structural properties of the graph: e.g., the number of neighbouring nodes that satisfy a given property, or the hitting time of a random walk on the graph. This thesis is primarily concerned with predicting the value of a relation. Propositionalization removes all relations from the data set. Ablating relations may be acceptable if our goal is to predict the properties of entities using relations; ablating relations is not acceptable if our goal is to predict properties of sets of entities, i.e., relations.

A common form of propositionalization in databases is denormalization, where multiple tables in a database are joined into a single one. If any of the relationships in the original schema are one-to-many or many-to-many, then records are repeated many times in the join, which can skew the marginal distribution of an attribute. In the augmented collaborative filtering example, $\texttt{Rating}(user, movie)$ and $\texttt{HasGenre}(movie, genre)$ are many-to-many relations. Were the relations joined, the frequency of a particular genre would depend on how often movies of a particular genre are rated. If many of the rated movies are action films, then the marginal distribution of genres would be skewed towards them, regardless of how common action films are among the set of movies. This problem is known as relational autocorrelation [55].

▷ **Inductive Logic Programming**: First-order logic (FOL) provides a particularly expressive language for relational models. The earliest approach that uses FOL for relational learning is inductive logic programming [86, 62], which is a non-probabilistic approach to relational learning. Probabilistic extensions of ILP exist (e.g., [101]), and are examples of statistical relational models.

Inductive logic programming (ILP) is the task of inducing first-order rules from observed relations, encoded as statements in first-order predicate logic, a logic program. Once a logic program, usually a set of Horn clauses, is inferred one can deduce new facts about entities using existing rules and facts. There are a variety of techniques for ILP based both on generalization from clauses in a knowledge base [110, 111, 85, 86, 121], and on specialization techniques, which build larger logic programs in much the same fashion as decision trees build larger classification rules [115, 99].

Logic programs can only generalize to statements that can be deduced. Our concern is with predicting statements that are probable, but not easily deducible. Hence our interest in statistical relational models (see below).

▷ **Statistical Relational Models**: More recently, a number of researchers have explored probabilistic extensions of logical representations, using these extensions for what we have come to know as statistical relational learning (cf., [40]). The most high-level description of a statistical relational model is based on the possible worlds interpretation in first-order logic. First-order logic defines a (possibly infinite) set of possible worlds, each of which is assigned a truth value. A statistical relational model instead defines a distribution over possible worlds, typically using a graphical model with repeated structure. Various subsets of logic provide the macro language for defining repeated structure in these graphical models. For example, Probabilistic Relational Models [39] can be viewed as a generalization of Bayesian networks to relational databases, where frame-based logic provides the macro language for defining the Bayesian network. Markov Logic Networks [105] can be viewed as a generalization of Markov Networks to relational domains, where a subset of first-order logic provides the macro language for defined repeated structure within the distribution over possible worlds. Bayesian Logic [79] defines a declarative language for defining contingent Bayesian networks, allowing one to reason about unknown objects using an infinite set of possible worlds.

A recurring theme in these approaches is the desire to represent an enormous variety of graphical models. For example, Probabilistic Graphical Models subsume discrete-variable Bayesian networks. Markov Logic Networks can represent, with sufficiently many variables, any discrete-variable probability distribution.[1] While we laud attempts to create high-level languages for graphical models, the focus of this thesis is on providing a statistical design pattern that (i) provides a simple, restricted interface to the modeler—i.e., sets of related matrices; (ii) can help provide guidance to modelers in defining repeated structure, especially in domains where defining repeated structure using expert knowledge is difficult (e.g., consider the brain imaging example in Chapter 5, where one would be hard pressed to encode relationships between words and regions of the brain); (iii) allow for scalable training and inference algorithms whose implementation need not be exposed to the modeler.

The closest areas of related work are in the literature on matrix factorization. As such, we review the connections between collective matrix factorization and other low-rank matrix factorizations below.

[1] Markov Logic Networks can also represent distributions over real-valued variables, provided that there is a fixed-bit representation of reals.

6.2 Connections to single-matrix factorization

There are two ways in which this thesis relates to the (vast) literature on low-rank matrix factorization. The first is descriptive, centering on our unified view of matrix factorization in Chapter 3. The second is algorithmic, centering on our work on efficient algorithms for maximum likelihood and Bayesian inference matrix factorization models.

Our unified view of matrix factorizations places a wide variety of low-rank matrix factorizations into the same conceptual framework, reducing each matrix factorization to a small number of modeling choices. We subsume both factor analysis, clustering, and matrix co-clustering techniques into the same formalism. We believe that this is a significant contribution to understanding the literature on single-matrix factorization. Moreover, if the end-user of matrix factorization models is to have any hope of choosing from the variety of models, they must first understand what makes the models different. Ultimately, the goal of a statistical design pattern is to formalize the expertise that goes into the design and implementation of a class of graphical models. Our hope is that this formalization of low-rank matrix factorization models will facilitate their use, and facilitate transfer of expertise across their domains of use.

Our algorithmic contributions center on the idea that matrix factorizations are bilinear models, or sets of tied bilinear models in collective matrix factorization. Bilinear models are closely related to linear models, and we can exploit decomposability of the loss to reduce inference into cyclic updates over tied linear models. Reducing inference into cyclic updates allows us to exploit componentwise convexity. In contrast, other techniques for solving low-rank matrix factorization usually treat training as a nonlinear optimization with little special structure: e.g., gradient descent, conjugate gradient, expectation-maximization, multiplicative updates.[2] In the case of max-margin matrix factorization, the trace-norm constraint makes the loss non-decomposable. The exceptions among solvers that exploit structure in low-rank matrix factorization are specialized techniques for SVD (e.g., Gaussian Elimination), and the alternating Bregman projection approach for E-PCA [26, 45].

The idea of reducing the parameter space to cyclic updates over subsets of the parameters has been used in Bayesian matrix factorizations, but there has always has always been a trade-off between richness of the models, and the computational cost of training:

▷ **Block Gibbs samplers** require that the conditional distribution of each block can be

[2] Multiplicative updates are largely focused on non-negative matrix factorization, and its variants. In the case of NMF, it can be shown that the multiplicative update can be derived from gradient descent with a particular, parameter dependent, scaling of the gradient [63].

sampled from exactly. Most frequently, we require that the likelihood and prior are conjugate, as is the case with Salakhutdinov and Mnih [109] (Gaussian likelihood and prior) and Buntine and Jakulin [19] (Poisson likelihood, Gamma prior; Poisson likelihood, conditional Gamma prior; Multinomial likelihood, Dirichlet prior). In certain special cases, there are non-conjugate pairs of distributions whose product can be sampled from exactly: e.g., the product of a Gaussian and an Exponential distribution is a rectified Normal distribution, which allows for a Bayesian extension of non-negative matrix factorization [114].

The requirement of exact conditional distributions on each block becomes more severe when dealing with multiple matrices, each with their own data distribution. To update the shared factor with Gibbs, the data likelihoods, as well as the priors, must be mutually conjugate.

▷ **Hamiltonian Monte Carlo**: Hamiltonian Monte Carlo (a.k.a. Hybrid Monte Carlo) [88, 74] is a technique that utilizes the gradient of the distribution being modelled to improve the mixing rate. HMC is used in a Bayesian generalization of E-PCA [84]. In this Bayesian version of E-PCA, the prior over rows of U is, like our work, multivariate Gaussian; and the prior over rows of V is chosen to be conjugate to the likelihood. A hierarchical prior is placed over rows of U, but not V. The hyperprior over the multivariate Gaussian is a restricted form of the Inverse-Wishart: the covariance matrix must be diagonal.

The use of Hamiltonian Monte Carlo requires that the variables must be unconstrained, which requires careful use of change-of-variable transforms to handle covariance matrices. An advantage of our block adaptive MCMC approach is that we can exploit partial Hessian structure, namely, the per-row Hessians. Unlike Mohamed et al. [84], we use partial Hessian information in addition to the gradient.

▷ **Variational Bayes** is a form of approximate inference, whereas the adaptive Metropolis-Hastings sampler we propose is asymptotically exact. Variational inference can make claims about runtime independent of the data presented; but the quality of the approximation depends on the data matrix X. In contrast, our adaptive Metropolis-Hastings approach cannot make claims about runtime independent of the data; but we can make guarantees with respect to the quality of the approximation which can be achieved: asymptotically, our approach is exact. When a variational method performs poorly, it is hard to disambiguate whether this is due to the model or the quality of the approximation. With MCMC, we can test whether the chain has mixed (although this is non-trivial to do in practice). Once we are confident that the chain has mixed, and that we have enough samples from the posterior, poor results can be readily attributed to the model.

The broader objection to variational methods is that every time the likelihood and prior change, a new lower bound must be derived. More choices of models equals more work for the modeler, both in deriving the bound and writing code to optimize the bound.

The lower-bound used in variational inference can be designed to take advantage of decomposability of the loss, reducing training to updates over rows of factor matrices. For example, VBSVD [67], is a Bayesian analogue to weighted SVD. Their variational Bayesian solution assumes that the posterior distribution over each row of U and V is Gaussian: the update equations are over each row of each factor. The quality of the approximations depends, in part, on how close the per-row posteriors are to Gaussian.

If we consider each of these Bayesian methods and their maximum likelihood analogue, it becomes clear that there is a tradeoff between (i) representational flexibility of the model, (ii) asymptotically exact vs. approximate inference, and (iii) human effort in designing the training algorithm. In constrast, our adaptive Metropolis-Hastings approach (i) is representationally flexible: every maximum likelihood variant of CMF we discuss has a Bayesian analogue which can be trained using our approach; (ii) scales to relatively large data sets, such as the fMRI problem we discuss in Chapter 5; (iii) our algorithm works for a variety of models: one implementation allows us to train on many different combinations of likelihood and prior. These properties are especially desirable when we consider sets of related matrices. Conjugacy in the multiple-matrix setting for HB-CMF requires mutual conjugacy between the priors and the likelihoods, restricting the combinations of distributions we can use to model X_{ij} and Y_{jr}. If an entity-type participates in many different relations, the mutual conjugacy problem can be highly restrictive.

Discrete Component Analysis (DCA) [19] is an attempt to unify Bayesian matrix factorization models into a unified view, much as we did for maximum likelihood low-rank matrix factorizations in Chapter 3. DCA assumes that the prior and likelihood are conjugate, but they consider the possibility that rows of the data matrix are draws from a multinomial distribution:

$$X_{i\cdot} \sim \text{Multinomial}(U_{i\cdot}V^T).$$

The multinomial rows assumption is shared with Latent Dirichlet Allocation [8]. In addition to variational approximation and Gibbs sampling, the use of conjugate priors makes Rao-Blackwellization within Gibbs practical. The use of conjugate priors allows for Rao-Blackwellization within Gibbs. Rao-Blackwellization within Gibbs is based on the idea that the unknown variables (in the case of HB-CMF, $\{\mathcal{F}, \Theta\}$) can be divided into two groups, such that sampling from one given the other is analytically tractable. In the case of HB-CMF, either $p(\mathcal{F} \mid \Theta)$ or $p(\Theta \mid \mathcal{F})$ would have to be analytic. We are not aware of an analytic form for either distribution.

Hierarchical priors are common to many approaches to Bayesian matrix factorization. They are essential to the design of a Bayesian analogue of ICA (c.f., Buntine and Jakulin [16]). Latent Dirichlet Allocation [8] and Multinomial PCA [17] utilize hierarchical priors to pool information across documents. H-CMF, which is hierarchical but estimated under maximum likelihood, generalizes unconstrained Probabilistic Matrix Factorization [108] to non-Gaussian prediction links. The difference in our approach is that we make no conjugacy assumptions, which allows HB-CMF to be easily generalized to include more complex types of priors (e.g., Laplace priors, which encourage sparsity in the same manner as ℓ_1-regularization).

6.3 Connections to multiple-matrix factorization

We do not claim to be the first to proposed parameter tying in matrix factorization, though we believe generalizing E-PCA to multiple matrices is far more flexible than techniques that generalize a matrix factorization which assumes a particular response type. We contend that Collective Matrix Factorization and its variants provide a flexible way of modelling sets of related matrices, in a manner that is useful for information integration.

To be useful for information integration, we need to be able to scale our models to handle a relatively large number of entities, which poses an algorithmic problem. A great deal of this thesis is concerned with algorithmic contributions that make Collective Matrix Factorization and its variants practical on data sets with large numbers of entities.

We begin by disambiguating our work from relational co-clustering, which assumes that our primary concern is with clustering entities. Later, we consider multiple-matrix factorization models that we consider more closely related to Collective Matrix Factorization.

6.3.1 Relational Co-Clustering

Relational (co-)clustering refers to models on sets of related matrices or tensors where the underlying factorization involves clustering or co-clustering of the rows and columns (or higher dimensions, in the case of tensors). This thesis focuses on predicting the value of a relation, not clustering entities. There is no *prima facie* reason to favour clustering over factor analysis in prediction; but there are computational reasons to favour factor analysis over clustering. We view the relational co-clustering techniques below as complementary to Collective Matrix Factorization (i.e., focused on entity clustering in relational data, as opposed to relation prediction):

▷ **Multi-type relational clustering**: A symmetric block model $X^{(ij)} \approx C_i A^{(ij)} C_j^T$ where $C_i \in \{0,1\}^{n_i \times k}$ and $C_j \in \{0,1\}^{n_k \times k}$ are cluster indicator matrices, and $A^{(ij)} \in \mathbb{R}^{k \times k}$ contains the predicted output for each combination of row and column clusters.

Early work on this model uses a spectral relaxation specific to squared loss [70], while later generalizations to regular exponential families [72] use EM. An equivalent formulation in terms of regular Bregman divergences [71] uses iterative majorization [64, 123] as the inner loop of alternating projection.

▷ **Relational multi-way clustering [6]**: A generalization of Bregman co-clustering of a matrix to sets of related tensors. Each entity-type has a low-rank representation, and an alternating optimization approach is used to determine a hard of soft clustering for each entity-type, cyclically till convergence. Instead of a Bregman projection, the basic operation is a clustering of the entities under *Bregman information*, the expected Bregman divergence. Essentially, the clustering of entities is a generalization of hard or soft k-means clustering, where Euclidean distance can be replaced by a variety of information theoretic quantities, such as the mutual information between entities and cluster exemplars. The entity clustering is a point-estimate. In our work, the basic projection operation is a convex optimization instead of a cluster assignment problem. Unlike our work, Banerjee et al. [6] considers sets of tied tensors, allowing for higher-arity relations.

▷ **Extensions of pLSI and LDA**: In the text modeling literature, multiple matrix factorizations have centered on the assumption that there is, at the core, a matrix representing the relationship $\texttt{Count}(document, word)$, where each row of the corresponding data matrix is modelled as a multinomial distribution over words. A multinomial row assumption is true of both probabilistic Latent Semantic Indexing [52] and Latent Dirichlet Allocation [8]. pLSI-pHITS [24] augments the `Count` relationship with a citation relation between documents, $\texttt{Cites}(document, document)$. The data reduces to a pair of related matrices, each of which is modelled by pLSI: the distribution of topics for each document is shared. We compared pLSI-pHITS against our approach in Section 4.5.3.

pLSI and its multi-matrix extensions suffer from the fold-in problem (Section 3.9). Bayesian alternatives like LDA do not suffer from the fold-in problem. Recently, there have been several extensions of Latent Dirichlet Allocation which augmented the bag-of-words representation with side information, either by tying hyperparameters together (e.g., [87]), or by tying parameters to a matrix factorization like E-PCA [21]. All of these techniques assume that the primary source of data is the `Count` relation, which is well-modelled by the multinomial link; we do not.

Table 6.1: Comparison of single- and multiple- matrix factorization models which model each matrix entry as a draw from an regular exponential family. We refer the reader to Section 6.3.2 for a description of the columns.

Algorithm	#M	Bayes?	Prior/Reg	Hier	Exp	Optimizer
PMF [108]	1	MLE	Gaussian (SC)	✓	✓	Gradient descent
E-PCA [26]	1	MLE	None	✗	✓	Alt. Bregman
BMF [109]	1	Bayes	Gaussian	✓	✗	Gibbs sampling
BXPCA [84]	1	Bayes	Multiple	✓	✓	Hybrid MC
VBSVD [67]	1	Bayes	Gaussian (DC)	✓	✗	Variational Bayes
SoRec [73]	2	MLE	Gaussian (SC)	✗	✗	Gradient descent
PMDC [132]	2+	Bayes	Gaussian (DC)	✓	✗	Variational Bayes
MRMF [68]	2+	MLE	Gaussian (SC)	✗	✗	Gradient descent
CMF [116]	2+	MLE	Gaussian (SC)	✗	✓	Alternating Newton
H-CMF	2+	MLE	Gaussian	✓	✓	Alternating Newton
HB-CMF	2+	Bayes	Gaussian	✓	✓	Adaptive MCMC

6.3.2 Related Factor Analysis Models

The most closely related single- and multiple- matrix factorizations models are those that do not make a clustering or co-clustering assumption, but do assume that each entry of the data matrix can be modelled as a draw from an exponential family distribution. We summarize the most closely related single- and multiple- matrix factorization models in Table 6.1. We compare the number of matrices involved (#M), whether or not the method is Bayesian (Bayes?), the type of prior (Prior/Reg), whether the prior is learned automatically (Hier), whether or not the entries can be modelled using any one-parameter exponential family (Exp), and the optimization techniques used (Optimizer). If the prior is Gaussian, we note whether the covariance is spherical (SC) or diagonal (DC). If neither SC or DC is annotated, the form of the prior covariance is unconstrained. It should be noted that even small differences in the features we consider can dramatically increase the difficulty of learning.

The three variants of collective matrix factorization are the most flexible. We cover both maximum likelihood and Bayesian inference, automatically estimated hierarchical priors and flat ones, as well as Gaussian and non-Gaussian data distributions. The only other technique which takes advantage of row-column exchangeability to compute per-row Hessians is variational Bayes (used by VBSVD, PMDC). In VBSVD and PMDC, the quality of the variational Bayes approximation to the posterior becomes better as the posterior over each factor row becomes more Gaussian, and as the factor rows become more independent.

The closest competing approach to our Bayesian model is PMDC, which is restricted to modeling entries of a matrix with a Gaussian. The variational Bayesian solution is based on approximating the posterior over each row of a factor as a multivariate Gaussian. Our approach similarly benefits from using a Gaussian distribution (as a proposal distribution) without introducing the biases that variational methods can introduce. Moreover, whenever the exponential family used to model matrix entries changes, a new variational bound must be derived. Our approach applies to a wide variety of exponential families without change.

Our decomposition approach is greatly informed by BMF: we sample the hyperparameters using the same Gibbs step. However, we can model sets of related matrices without requiring that the predictions be Gaussian. The Hybrid Monte Carlo sampler in BXPCA uses the gradient of the likelihood; we use the gradient and the per-row Hessians.

The three-factor schema $\mathcal{E}_1 \sim \mathcal{E}_2 \sim \mathcal{E}_3$ also includes supervised matrix factorization. In this problem, the goal is to classify entities of type $\mathcal{E}_2$: matrix $X^{(12)}$ contains class labels according to one or more related concepts (one concept per row), while $X^{(23)}$ lists the features of each entity. An example of a supervised matrix factorization algorithm is the support vector decomposition machine [95]: in SVDMs, the features $X^{(23)}$ are factored under squared loss, while the labels $X^{(12)}$ are factored under Hinge loss. Supervised dimensionality reduction [106] is replaces the hinge and squared losses with Bregman divergences, training the model parameters using EM. A similar model was proposed by Zhu et al. [135], using a once-differentiable variant of the Hinge loss. Another example is supervised LSI [133], which factors both the data and label matrices under squared loss, with an orthogonality constraint on the shared factors. Principal components analysis, which factors a doubly centered matrix under squared loss, has also been extended to the three-factor schema [134].

Recursive Attribute Factoring [25] is unique in its approach to multiple-matrix factorization (where one matrix consists of document-word counts, and another contains document-document links). Instead of simultaneously finding a low-rank representing documents and words, they recursively factor the matrices: each low-rank model of documents is appended to the document-word counts, and the augmented matrix becomes the data for another round of factorization. Unlike our work, they factor exclusively under squared-loss.

In Chapter 3 we discussed max-margin matrix factorization (MMMF) as a single-matrix factorization, where the matrix is the $\texttt{Rating}(user, movie)$ relation. However, it can also be viewed as a kind of max-margin matrix factorization. The key intuition behind MMMF is that it treats an rating as an ordinal value: $\texttt{Rating} \in \{1, 2, \ldots, R\}$ and $1 < 2 < \ldots < R$. Fast MMMF reduces the prediction of an entry to a set of binary threshold problems: namely, if $r_{ij} = \texttt{Rating}(user_i, movie_j)$, predicting $r_{ij} > 1, r_{ij} > 2, \ldots, r_{ij} > R - 1$. If we use a smooth loss (such as log-loss) for each of these binary predictions and add the losses together, the result is equivalent to a collective matrix factorization where $\mathcal{E}_1$ are users, $\mathcal{E}_2$ are movies, and $\mathcal{E}_1 \sim_u \mathcal{E}_2$ for $u = 1 \ldots R - 1$ are the binary rating prediction data: $X_{pq}^{(ij,r)} = 1$ iff user p assigns a rating greater than r to movie q. A single matrix of data weights can be used to mask missing values in all $R - 1$ relations.

In order to predict different values for the $R - 1$ different relations, we need to allow the latent factors $U^{(1)}$ and $U^{(2)}$ to contain some untied columns, *i.e.*, columns which are not shared among relations. For example, the MMMF authors have suggested adding a bias term for each rating level or for each (user, rating level) pair. To get a bias for each (user, rating level) pair, we can append $R - 1$ untied columns to $U^{(1)}$, and have each of these columns multiply a fixed column of ones in $U^{(2)}$. To get a shared bias for each rating level, we can do the same, but constrain each of the untied columns in $U^{(1)}$ to be a multiple of the all-ones vector.

6.4 Conclusions

The role we envision for collective matrix factorization is that of a statistical design pattern for information integration, a class of graphical models where the complexities of training and inference are hidden, exposing the user only to sets of related matrices, with a small number of tunable parameters (such as k, the number of embedding dimensions).

Other approaches to statistical relational learning provide a high-level interface for creating much larger classes of graphical models, albeit at the cost of losing computationally useful structure. That, combined with the fact that we build upon matrix factorization, makes it natural to compare our work to other matrix factorization methods. Our basic claim is that even though sets of related matrices are a simple representation, and tying low-rank matrix factorizations is a straightforward idea, the combination of these ideas cover a fairly rich class of successful models in the machine learning literature.

Chapter 7

Future Work and Discussion

We put before the reader the claim that Collective Matrix Factorization, and its variants, are a statistical design pattern for information integration problems. In Section 7.1 we describe potential extensions to Collective Matrix Factorization, that speak to the limitations and ambitions felt most acutely during our research in this area. We have already described the main contributions of this thesis in Chapter 1, and so instead close with a discussion of why we believe these contributions are of significance.

7.1 Future Work

7.1.1 Higher Arity Relations

The most obvious limitation of our approach is that each relation must have exactly two arguments, which allows for the encoding of attributes and typed links, but not relations with three or more arguments. Banerjee et al. [6] has considered the problem in the context of relational clustering, though using point estimation. Alternating projection algorithms has long been used in tensor factorization, and we believe that extending our work to tensors would be relatively straightforward.

7.1.2 Distributed Models

We have discussed the parallel implementation of a projection, but Collective Matrix Factorization could also be reasonably adapted to the distributed setting, where the data and parameters are split up across multiple machines.[1] All we require to update the low-rank parameters for an entity are the current values of the low-rank parameters, the observed relations the entity is involved in, and the factors that represent the other argument in relations the entity is involved in.[2] Entities, and their low-rank representations, can be distributed across multiple machines, along with copies of the necessary rows and/or columns of the data matrices. All one need ever communicate between machines are factor rows. It remains to be shown, though, how one can efficiently distribute data and parameters across multiple machines, so that the computational and communication costs of training are not focused on a small subset of the machines.

7.1.3 Conditional Models

We have concerned ourselves with *generative* models, where we model the joint distribution of the parameters, hyperparameters, and all data matrices. If we are interested only in predicting the value of one relation, then introducing parameters to model the other relations can be both statistically and computationally inefficient. One way of addressing these concerns may be through the use of *discriminative* models: instead of modeling the joint distribution of the inputs and outputs, model the conditional distribution of the outputs given the inputs. In the two matrix case, the task would involve modelling $p(X \mid Y)$ or $p(Y \mid X)$ instead of $p(X, Y)$. The application domains we have considered lend themselves to discriminative models. In the fMRI domain, we may be most interested in predicting voxel activation given word co-occurrence data. In the movie rating domain, we may be most interested in predicting user's ratings given side information about movies.

The closest related work on discriminative models is based on Canonical Correlation Analysis (CCA) [53], which views the data matrix as a collection of exchangeable rows or columns. Given two collections of related data points (rows or columns), we can imagine a regression model, for each data set, that projects each set on points onto an unknown scalar quantity, which we call canonical variates. Canonical Correlation Analysis involves finding regression parameters for each data set such that the correlation between the canonical variates is maximized. If we draw out CCA as a graphical model, as in Bach and Jordan [4], it becomes clear that the regression weights are a latent representation for exchangeable data, and that the regression weights are tied in a specific fashion. A generalization

[1] We note that timing experiments for the hierarchical Bayesian model (Chapter 5) are based on a parallel implementation using four processors in a shared memory configuration. However, we have not considered the distributed setting, where communication constraints between processors are much more of a concern.

[2] Technically, all that is required are the rows of the other factor that correspond to observed entries in a data matrix.

of CCA to multiple related matrices has been used to integrate data from multiple fMRI studies, improving the quality of a linear regression model of brain activation [107].

The multiple CCA approach assumes a star schema: a single common entity type E_1, which is related to the other entity types: $E_1 \sim E_i, \forall i > 1$. When the schema is not a star, conditional relational models become problematic. Conditional distributions usually only include parameters between the output (relation we want to predict) and the inputs (other relations). Parameters are not included to model the relations between the input relations. One potential direction for defining a conditional Collective Matrix Factorization model would be to keep the existing generative structure, but learn the parameters discriminatively (e.g., by maximizing a conditional likelihood instead of the likelihood). However, maximizing conditional likelihood does not have a straightforward Bayesian analogue.

Another problematic aspect of conditional models for matrix data is that they assume exchangeability of each matrix, but not row-column exchangeability. Developing conditional models on row-column exchangeable models remains a topic for future work.

7.1.4 Relational Active Learning

Information integration involves aggregating disparate sources of information, some of which are easier to acquire than others. In the augmented collaborative filtering problem, collecting side information about movies is easy and inexpensive; collecting ratings from people is time consuming. Large-scale rating data, such as the Netflix data set [89], is the side product of a company's business. In the augmented brain imaging problem, word co-occurrence data is cheap to collect; brain imaging data is expensive to collect. fMRI data collection requires human subjects, access to a fMRI device, and domain experts to process the data. While Collective Matrix Factorization can improve predictions on models of expensive data, can we use our models to *guide* the process of collecting expensive data?

Active learning is a formalism where the learning algorithm can select training observations, instead of passively using data drawn from a fixed, unknown, probability distribution (a.k.a. passive learning). The advantage of active learning over passive learning, which has been the focus of this thesis, is that the learning algorithm can select observations which are more likely to improve predictive accuracy. Active learning is usually framed as a value of information problem. The learning algorithm has a function which can assign a value to every possible subset of observations.[3] The goal is to find the best

[3] Often, value of information functions are based on an information theoretic property, such as mutual information (c.f., Guestrin et al. [46], Krause et al. [60]). Since Collective Matrix Factorization has been framed as a probabilistic graphical model, such information theoretic functions are easy to define.

possible subset of observations.

There are a number of questions that arise when one considers active learning within a relational model: Can we exploit relations between expensive and inexpensive data sources to reduce the cost of data collection, while still producing highly predictive models? How do relations guide the process of selecting maximally informative objects? What is the appropriate tradeoff between observing properties of an object and the relations it participates in? What is the relative difficulty of this problem in the adaptive scenario, where the model incorporates data as it is selected, versus the one-shot scenario, where we must choose all observations to make *before* seeing any of their results?

7.1.5 Temporal-Relational Models

Collective Matrix Factorization assumes that observations are scalars:

$$\texttt{Relation}(x, y) \in \mathbb{R}.$$

In many cases, observations are better represented as time-varying quantities:

$$\texttt{Relation}(x, y) = (v_t)_{t=1}^{T},\ v_t \in \mathbb{R}.$$

For example, while we have treated fMRI measures of voxel activation as scalar quantities, there are other techniques for measuring brain activity at high frequency, such as magnetoencephalography (MEG). The voxel response relation for MEG data would be better described by $\texttt{Response}_t(stimulus, voxel)$: each observation is a time series.

Were one to replace each random variable that models a scalar observation in Collective Matrix Factorization with a time series, similar to a Hidden Markov Model or Kalman Filter, the end result would be to tie the hidden state and/or dynamics of time series models together. Predictions of an observed relation (time series) would depend not only on its own past, but also on the past of other, correlated, temporal observations.

One could imagine the utility of alternating optimization approaches, where instead of projection the basic operation is training a time series model. Such an approach would also provide an alternative to modeling time as a dimension in matrix factorization, which violates the row-column exchangeability assumption.

7.2 Why this thesis matters

Statistical relational learning is a vast topic, perhaps too vast. Rarely does one see researchers talking about propositional or attribute-value learning as a monolithic field of study. Rather, we typically refine our discussion of attribute-value learning along different axes: by task abstraction (clustering, prediction); by learning abstraction (batch, online, active); and by application domains (in natural language processing, in robotics). It is our belief that a similar refinement of statistical relational learning is valuable; information integration is a refinement of relational learning by task abstraction. Refining our discussion to focus on information integration is valuable because (i) such problems appear in many different domains; (ii) we have developed easy-to-use techniques that address information integration, or provide the basis for a more elaborate solution; (iii) while easy-to-use, our techniques are backed by sophisticated, scalable algorithms for learning and inference.

From a statistical perspective, the goal of relational learning is ambitious. The vast majority of statistics and machine learning holds exchangeability of data points as a paradigmatic assumption. However, the whole purpose of relational learning is to exploit correlations due to violations of exchangeability. We have shown that one can have a relational model that exploits exchangeability within a relation and correlations between relations.

Even though relational data violates some of the most basic assumptions of common statistical models, we can build elaborate models of relational data from very simple, very familiar, building blocks. Sets of related matrices can represent a large swath of relational data sets, and this representation is not especially complicated. Given a set of related matrices, tying low-rank parameters together is not an especially complicated idea. Given a large, non-convex optimization, alternating projections (a.k.a. block coordinate descent) is not an especially elaborate solver. Given a projection, recognizing independence among subsets of random variables is not unheard of. And finally, we have in our hands the basic operation: train a linear model, in our case, a generalized linear model, or its Bayesian analogue. Now, if there is one class of models we understand well, and know how to scale, it is linear models.

While the individual steps above are simple, we believe that our combination leads to an approach more powerful than is suggested by the sum of parts. Keeping things simple does have it advantages.

Appendix A

Newton Update for the Three Entity-Type Model

A.1 Derivation of the 3-entity-type model

The 3-entity-type model consists of two relationship matrices, a $m \times n$ matrix X and a $n \times r$ matrix Y. In the spirit of generalized linear models we define the low rank representations of the relationship matrices to be $X \approx f_1(UV^T)$ and $Y \approx f_2(VZ^T)$ where $f_1 : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ and $f_2 : \mathbb{R}^{n\times r} \to \mathbb{R}^{n\times r}$ are the prediction links[1], and $U \in \mathbb{R}^{m\times k}$, $V \in \mathbb{R}^{n\times k}$, and $Z \in \mathbb{R}^{r\times k}$ are the parameters of the model for $k \in \mathbb{Z}_{++}$, a positive integer. We further define functions $G : \mathbb{R}^{m\times k} \to \mathbb{R}^{m\times k}$, $H : \mathbb{R}^{n\times k} \to \mathbb{R}^{n\times k}$, $I : \mathbb{R}^{r\times k} \to \mathbb{R}^{r\times k}$, to model prior knowledge about our parameters, *e.g.*, regularizers. We additionally require the convex conjugate,

$$G^*(U) = \sup_{U \in \mathrm{dom}(A)} \left[U \circ A - G(A) \right],$$

where $U \circ A = tr(U^T A) = \sum_{ij} U_{ij} A_{ij}$ is the matrix dot product. The overall loss function for our model is

$$L(U, V, Z | W, \tilde{W}) = \alpha L_1(U, V|W) + (1-\alpha) L_2(V, Z|\tilde{W}) \tag{A.1}$$

[1] Throughout this appendix we assume that prediction links f and the corresponding loss defined by F are decomposable, $i.e.$, $F : \mathbb{R} \to \mathbb{R}$ and $f : \mathbb{R} \to \mathbb{R}$ are applied element-wise when their arguments are matrices.

where we introduce fixed weight matrices for the observations, $W \in \mathbb{R}^{m \times n}$ and $\tilde{W} \in \mathbb{R}^{n \times r}$. The individual objectives on the reconstruction of X and Y are, respectively,

$$\begin{aligned}
L_1(U, V|W) &= \left(W \circ F_1(UV^T) - (W \odot X) \circ UV^T\right) + G^*(U) + H^*(V) \qquad \text{(A.2)}\\
&= \sum_{ij} W_{ij}\left(F_1(U_{i\cdot}V_{j\cdot}^T) - X_{ij} \cdot U_{i\cdot}V_{j\cdot}^T\right) + \frac{a}{2}\sum_{rs} U_{rs}^2 + \frac{b}{2}\sum_{pq} V_{pq}^2,\\
L_2(V, Z|\tilde{W}) &= \left(\tilde{W} \circ F_2(VZ^T) - (\tilde{W} \odot Y) \circ VZ^T\right) + H^*(V) + I^*(Z) \qquad \text{(A.3)}\\
&= \sum_{ij} \tilde{W}_{ij}\left(F_2(V_{i\cdot}Z_{j\cdot}^T) - Y_{ij} \cdot V_{i\cdot}Z_{j\cdot}^T\right) + \frac{b}{2}\sum_{pq} V_{pq}^2 + \frac{c}{2}\sum_{rs} Z_{rs}^2,
\end{aligned}$$

where F_1 and F_2 are element-wise functions over matrices, *i.e.*, L_1 and L_2 are each decomposable. The scalars $a, b, c \geq 0$ control the strength of regularization on the factors (the larger the value, the weaker the regularizer). The objective $L(U, V, Z|W, \tilde{W})$ is convex in any one of its arguments, but is in general non-convex in all its arguments. As such, we use an alternating minimization scheme that optimizes one factor U, V, or Z at a time. This appendix describes the derivation of both gradient and Newton update rules for U, V, and Z. For completeness, Section A.1.1 reviews useful definitions from matrix calculus. The gradient of the objective, with respect to each argument is derived in Section A.1.2. Finally, by assuming the loss is decomposable, we derive the Newton update in Section A.1.3, whose additional cost over its gradient analogue is essentially a factor of k times more expensive.

A.1.1 Matrix Calculus

For the sake of completeness we define matrix derivatives, which generalize both scalar and vector derivatives. Using this definition of matrix derivatives, we also generalize the scalar chain and product rules to matrices. The discussion herein is based on Magnus et. al. [75, 76].

Let M be an $n \times q$ matrix of variables, where $m_{\cdot j}$ denotes the j-th column of M. The vec-operator yields a $nq \times 1$ matrix that stacks the columns of M:

$$\operatorname{vec} M = \begin{pmatrix} m_{\cdot 1} \\ m_{\cdot 2} \\ \vdots \\ m_{\cdot q} \end{pmatrix}.$$

While there are several common (and incompatible) definitions of matrix derivatives, the derivative of a $n \times 1$ vector f with respect to a $m \times 1$ vector x is almost universally defined as

$$Df(x) \equiv \frac{\partial f}{\partial x^T} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_1}{\partial x_m} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \dots & \frac{\partial f_n}{\partial x_m} \end{pmatrix}.$$

The matrix derivative of an $m \times p$ matrix function φ of an $n \times q$ matrix of variables M contains $mnpq$ partial derivatives, and the matrix derivative arranges these partial derivatives into a matrix. We define the matrix derivative by coercing matrices into vectors, and using the above definition of the vector derivative:

$$D\varphi(M) \equiv \frac{\partial \operatorname{vec} \varphi(M)}{\partial (\operatorname{vec} M)^T} = \begin{pmatrix} \frac{\partial [\varphi(M)]_{11}}{\partial m_{11}} & \dots & \frac{\partial [\varphi(M)]_{11}}{\partial m_{nq}} \\ \vdots & & \vdots \\ \frac{\partial [\varphi(M)]_{mp}}{\partial m_{11}} & \dots & \frac{\partial [\varphi(M)]_{mp}}{\partial m_{nq}} \end{pmatrix},$$

which is an $mp \times nq$ matrix of partial derivatives. This definition encompasses vector and scalar derivatives as special cases. The advantages of this formulation include (i) unambiguous definitions for the product and chain rules, and (ii) we can easily convert $D\varphi(M)$ to the more common definition of the matrix derivative, $\frac{\partial \varphi(M)}{\partial M}$, via the first identification theorem [76, ch. 9]. We additionally require the Kronecker product, and the matrix chain [76, pg. 121] and matrix product [75] rules:

Definition 6 (Kronecker Product)**.** *Given a $m \times p$ matrix Q and a $n \times q$ matrix M the Kronecker product, $Q \otimes M$, is a $mn \times pq$ matrix:*

$$Q \otimes M = \begin{pmatrix} Q_{11}M & \dots & Q_{1p}M \\ \vdots & \ddots & \vdots \\ Q_{m1}M & \dots & Q_{mp}M \end{pmatrix}.$$

Definition 7 (Matrix Chain Rule)**.** *Given functions $\varphi : \mathbb{R}^{n\times q} \to \mathbb{R}^{m\times p}$, $\varphi_1 : \mathbb{R}^{\ell\times r} \to \mathbb{R}^{m\times p}$, and $\varphi_2 : \mathbb{R}^{n\times q} \to \mathbb{R}^{\ell\times r}$ the derivative of $\varphi(M) = \varphi_1(Y)$, where $Y = \varphi_2(M)$, is*

$$D\varphi(M) = D\varphi_1(Y) \times D\varphi_2(M).$$

where $A \times B$ (or equivalently, AB) is the matrix product.

Definition 8 (Matrix Product Rule)**.** *Given an $m \times p$ matrix $\varphi_1(M)$, a $p \times r$ matrix $\varphi_2(M)$, and a $\ell \times s$ matrix M, the derivative of $\varphi_1(M) \times \varphi_2(M)$ is*

$$D(\varphi_1 \times \varphi_2)(M) = (\varphi_2(M)^T \otimes I_m) \times D\varphi_1(M) + (I_r \otimes \varphi_1(M)) \times D\varphi_2(M),$$

where $A \otimes B$ is the Kronecker product.

A.1.2 Computing the Gradient

To compute the derivative of Equation A.1 with respect to U, V, and Z we require the following three lemmas:

Lemma 1. *For any differentiable function $F : \mathbb{R} \to \mathbb{R}$*

$$\frac{\partial \left(W \circ F(UV^T)\right)}{\partial U} = \left(W \odot f(UV^T)\right) V, \quad \frac{\partial \left(W \circ F(UV^T)\right)}{\partial V} = \left(W \odot f(UV^T)\right)^T U.$$

where $f = \nabla F$.

Proof. We only derive the result for $\varphi(U) = F(UV^T)$, the proof is similar for the other case. $\varphi(U)$ can be expressed as the composition of functions: $\varphi(U) = F(Y), Y = \varphi_2(U)$, $\varphi_2(U) = UV^T$. We note that

$$\begin{aligned} DF(Y) &= \frac{\partial \operatorname{vec} W \circ F(UV^T)}{\partial (\operatorname{vec} UV^T)^T} \\ &= \begin{pmatrix} \frac{\partial \sum_{ij} W_{ij} F(Y_{ij})}{\partial Y_{11}} & \dots & \frac{\partial \sum_{ij} W_{ij} F(Y_{ij})}{\partial Y_{mn}} \end{pmatrix} \\ &= \begin{pmatrix} W_{11} f(U_{1\cdot} V_{1\cdot}^T) & \dots & W_{mn} f(U_{m\cdot} V_{n\cdot}^T) \end{pmatrix} \\ &= \left(\operatorname{vec} W \odot f(UV^T)\right)^T . \end{aligned} \tag{A.4}$$

Furthermore, using the matrix product rule

$$\begin{aligned} D\varphi_2(U) &= (V \otimes I_m) \times \frac{\partial \operatorname{vec} U}{\partial (\operatorname{vec} U)^T} + (I_n \otimes U) \times \frac{\partial \operatorname{vec} V^T}{\partial (\operatorname{vec} U)^T} \\ &= (V \otimes I_m). \end{aligned} \tag{A.5}$$

Combining equations A.4 and A.5 using the matrix chain rule and the identity $(\operatorname{vec} A)^T \times (B \otimes I) = (\operatorname{vec} AB)^T$ yields $D\varphi(U) = DF(Y) \times D\varphi_2(U) = (\operatorname{vec} \left(W \odot f(UV^T)\right) V)^T$. The first identification theorem [76, ch. 9], which formally states the rules for rearranging entries of $D\varphi(U)$ to get $\frac{\partial (W \odot F(UV^T))}{\partial U}$, completes the proof. □

Lemma 2. *For fixed matrices W and X and variable matrices U and V*

$$\frac{\partial ((W \odot X) \circ UV^T)}{\partial U} = (W \odot X)V, \quad \frac{\partial ((W \odot X) \circ UV^T)}{\partial V} = (W \odot X)^T U.$$

Proof. We derive the result for $\partial(X \circ UV^T)/\partial V$, the other derivation is similar. To avoid a long digression into matrix differentials, we prove the result by element-wise differentiation. Noting that

$$X \circ UV^T = \sum_i \sum_j w_{ji} x_{ji} \sum_k u_{jk} v_{ik},$$

we compute the derivative with respect to v_{pq}:

$$\begin{aligned} \frac{\partial((W \odot X) \circ UV^T)}{\partial v_{pq}} &= \sum_i \sum_j w_{ji} x_{ji} \frac{\partial}{\partial v_{pq}} \sum_k u_{jk} v_{ik} \\ &= \sum_j w_{jp} x_{jp} u_{jq} = \left((W \odot X)^T U\right)_{pq}. \end{aligned}$$

Where the final step follows from the fact that the derivative is zero unless $(i,j) = (p,q)$. Since the result holds for all $p \in \{1, \dots, n\}$ and $q \in \{1, \dots, k\}$ it follows that $\partial(X \circ UV^T)/\partial V = X^T U$. □

Lemma 3. *For the ℓ_2 regularizers where $a, b, c > 0$ controls the strength of regularizers (larger values $\implies$ weaker regularization):*

$$G(U) = \frac{a||U||^2_{Fro}}{2}, \quad H(V) = \frac{b||V||^2_{Fro}}{2}, \quad I(Z) = \frac{c||Z||^2_{Fro}}{2},$$

the derivatives for the convex conjugates are

$$\frac{\partial G^*(U)}{\partial U} = \frac{U}{a} = A, \quad \frac{\partial H^*(V)}{\partial V} = \frac{V}{b} = B, \quad \frac{\partial I^*(Z)}{\partial Z} = \frac{Z}{c} = C.$$

Proof. The result is easily proven by finding the convex conjugate and differentiating it. □

Combining Lemmas 1,2, and 3, and denoting the Hadamard (element-wise) product of matrices $Q \odot M$, we have that for $f_1 = \nabla F_1$ and $f_2 = \nabla F_2$

$$\frac{\partial L(U,V,Z)}{\partial U} = \alpha \left(W \odot \left(f_1(UV^T) - X\right)\right) V + A, \tag{A.6}$$

$$\frac{\partial L(U,V,Z)}{\partial V} = \alpha \left(W \odot \left(f_1(UV^T) - X\right)\right)^T U + (1-\alpha) \left(\tilde{W} \odot \left(f_2(VZ^T) - Y\right)\right) Z + B, \tag{A.7}$$

$$\frac{\partial L(U,V,Z)}{\partial Z} = (1-\alpha) \left(\tilde{W} \odot \left(f_2(VZ^T) - Y\right)\right)^T V + C. \tag{A.8}$$

Setting the gradient equal to zero yields update equations either for A, B, C or for U, V, Z. An advantage of using a gradient update is that we can relax the requirement that the links are differentiable, replacing gradients with subgradients in the prequel.

A.1.3 Computing the Newton Update

One may be satisfied with a gradient step. However, the assumption of a decomposable loss means that most second derivatives are set to zero, and a Newton update can be done efficiently, reducing to row-wise optimization of U, V, and Z. For the subclass of models where Equations A.6-A.8 are differentiable and the loss is decomposable define,

$$q(U_{i\cdot}) = \alpha \left(W_{i\cdot} \odot \left(f_1(U_{i\cdot}V^T) - X_{i\cdot}\right)\right) V + A_{i\cdot},$$
$$q(V_{i\cdot}) = \alpha \left(W_{i\cdot} \odot \left(f_1(UV_{i\cdot}^T) - X_{i\cdot}\right)\right)^T U + (1-\alpha) \left(\tilde{W}_{i\cdot} \odot \left(f_2(V_{i\cdot}Z^T) - Y_{i\cdot}\right)\right) Z + B_{i\cdot},$$
$$q(Z_{i\cdot}) = (1-\alpha) \left(\tilde{W}_{\cdot i} \odot \left(f_2(VZ_{i\cdot}^T) - Y_{\cdot i}\right)\right)^T V + C_{i\cdot}.$$

Any local optimum of the loss corresponds to a simultaneous root of the functions $\{q(U_{i\cdot})\}_{i=1}^m$, $\{q(V_{i\cdot})\}_{i=1}^n$, and $\{q(Z_{i\cdot})\}_{i=1}^r$. Using a Newton step, the update for $U_{i\cdot}$ is

$$U_{i\cdot}^{\text{new}} = U_{i\cdot} - \eta \left[q(U_{i\cdot})\right] \left[q'(U_{i\cdot})\right]^{-1}, \tag{A.9}$$

where $\eta \in (0, 1]$ is the step length, chosen using line search with the Armijo criterion [90, ch. 3]. The Newton steps for $V_{i\cdot}$ and $Z_{i\cdot}$ are analogous. To describe the Hessians of the loss, q', we introduce the following notation for the Hessians of G^*, H^* and I^*:

$$\begin{aligned} G_i &\equiv \operatorname{diag}(\nabla^2 G^*(U_{i\cdot})) = \operatorname{diag}(a^{-1}\mathbf{1}), \\ H_i &\equiv \operatorname{diag}(\nabla^2 H^*(V_{i\cdot})) = \operatorname{diag}(b^{-1}\mathbf{1}), \\ I_i &\equiv \operatorname{diag}(\nabla^2 I^*(Z_{i\cdot})) = \operatorname{diag}(c^{-1}\mathbf{1}). \end{aligned}$$

For conciseness we also introduce the following terms:

$$D_{1,i} \equiv \operatorname{diag}(W_{i\cdot} \odot f_1'(U_{i\cdot}V^T)), \quad D_{2,i} \equiv \operatorname{diag}(W_{\cdot i} \odot f_1'(UV_{i\cdot}^T)),$$
$$D_{3,i} \equiv \operatorname{diag}(\tilde{W}_{i\cdot} \odot f_2'(V_{i\cdot}^T Z)), \quad D_{4,i} \equiv \operatorname{diag}(\tilde{W}_{\cdot i} \odot f_2'(VZ_{i\cdot}^T)).$$

This allows us to describe the Hessians of Equation A.1 with respect to each row of the parameter matrices.

Lemma 4. *The Hessians of Equation A.1 with respect to* $U_{i\cdot}$, $V_{i\cdot}$, *and* $Z_{i\cdot}$ *are*

$$\begin{aligned} q'(U_{i\cdot}) &\equiv \frac{\partial q(U_{i\cdot})}{\partial U_{i\cdot}} = \alpha V^T D_{1,i} V + G_i, \\ q'(Z_{i\cdot}) &\equiv \frac{\partial q(Z_{i\cdot})}{\partial Z_{i\cdot}} = (1-\alpha) V^T D_{4,i} V + I_i, \\ q'(V_{i\cdot}) &\equiv \frac{\partial q(V_{i\cdot})}{\partial V_{i\cdot}} = \alpha U^T D_{2,i} U + (1-\alpha) Z^T D_{3,i} Z + H_i. \end{aligned}$$

Proof. We prove the result for $q'(U_{i\cdot})$, noting that the other derivations are similar. Since $q(\cdot)$ and its argument $U_{i\cdot}$ are both vectors, $Dq(U_{i\cdot})$ is identical to $\partial q(U_{i\cdot})/\partial U_{i\cdot}$. Ignoring the terms that do not vary with $U_{i\cdot}$,

$$\begin{aligned} Dq(U_{i\cdot}) &= D\left[\alpha(W_{i\cdot} \odot f(U_{i\cdot}V^T))V + A_{i\cdot}\right] \\ &= \alpha D\left[(W_{i\cdot} \odot f(U_{i\cdot}V^T))V\right] + DA_{i\cdot} \\ &= \alpha \left\{ (V^T \otimes 1) \times \frac{\partial \operatorname{vec}(W_{i\cdot} \odot f(U_{i\cdot}V^T))}{\partial (\operatorname{vec} U_{i\cdot})^T} + \left(I_k \otimes (W_{i\cdot} \odot f(U_{i\cdot}V^T))\right) \times \frac{\partial \operatorname{vec} V}{\partial (\operatorname{vec} U_{i\cdot})^T} \right\} \\ &\quad + \frac{\partial \operatorname{vec} \nabla G^*(U_{i\cdot})}{\partial (\operatorname{vec} U_{i\cdot})^T} \\ &= \alpha \left\{ (V^T \otimes 1) \times D_{1,i} V + \left(I_k \otimes (W_{i\cdot} \odot f(U_{i\cdot}V^T))\right) \times 0 \right\} + G_i \\ &= \alpha V^T D_{1,i} V + G_i. \end{aligned}$$

The introduction of $D_{1,i}V$ above follows from observing that

$$\left(\frac{\partial \operatorname{vec}(W_{i\cdot} \odot f(U_{i\cdot}V^T))}{\partial \operatorname{vec} U_{i\cdot}} \right)_{p,q} = \frac{\partial \left(W_{ip} \odot f(U_{i\cdot}V_{p\cdot}^T)\right)}{\partial U_{iq}} = \underbrace{W_{ip} f'(U_{i\cdot}V_{p\cdot}^T)}_{\omega_{ip}} V_{pq},$$

which is simply the scaling of the p^{th} row of V by ω_{ip}. □

An alternate form for the updates, known as the adjusted dependent variates form, can be derived by plugging the gradient $q(U_{i\cdot})$ and the Hessian $q'(U_{i\cdot})$ into Equation A.9:

$$\begin{aligned} U_{i\cdot}^{\text{new}} q'(U_{i\cdot}) &= U_{i\cdot}(\alpha V^T D_{1,i} V + G_i) + \alpha\eta\left(W_{i\cdot} \odot \left(X_{i\cdot} - f(U_{i\cdot}V^T)\right)\right) V - \eta A_{i\cdot} \quad \text{(A.10)} \\ &= \alpha U_{i\cdot} V^T D_{1,i} V + \alpha\eta\left(W_{i\cdot} \odot \left(X_{i\cdot} - f(U_{i\cdot}V^T)\right)\right) D_{1,i}^{-1} D_{1,i} V + U_{i\cdot} G_i - \eta A_{i\cdot} \\ &= \alpha\left(U_{i\cdot}V^T + \eta\left(W_{i\cdot} \odot \left(X_{i\cdot} - f(U_{i\cdot}V^T)\right)\right) D_{1,i}^{-1}\right) D_{1,i} V + U_{i\cdot} G_i - \eta A_{i\cdot} \end{aligned}$$

Likewise for $Z_{i\cdot}$,

$$Z_{i\cdot}^{\text{new}} q'(Z_{i\cdot}) = (1-\alpha)\left(Z_{i\cdot}V^T + \eta\left(\tilde{W}_{\cdot i} \odot \left(Y_{\cdot i} - f(VZ_{i\cdot}^T)\right)\right)^T D_{4,i}^{-1}\right) D_{4,i}V + Z_{i\cdot}I_i - \eta C_{i\cdot} \tag{A.11}$$

The derivation of the update for $V_{i\cdot}$ is similar, since $L(U, V, Z|W, \tilde{W})$ is a linear combination and the differential operator is linear:

$$\begin{aligned} V_{i\cdot}^{\text{new}} q'(V_{i\cdot}) = \alpha &\left\{\left(V_{i\cdot}U^T + \eta\left(W_{\cdot i} \odot \left(X_{\cdot i} - f(UV_{i\cdot}^T)\right)\right)^T D_{2.i}^{-1}\right) D_{2,i}U\right\} + \\ &(1-\alpha)\left\{\left(V_{i\cdot}Z^T + \eta\left(\tilde{W}_{i\cdot} \odot \left(Y_{i\cdot} - f(V_{i\cdot}Z^T)\right)\right) D_{3,i}^{-1}\right) D_{3,i}Z\right\} + \\ &V_{i\cdot}H_i - \eta B_{i\cdot} \end{aligned} \tag{A.12}$$

While $D \in \{D_{j,i}\}_{j=1}^4$ may not be invertible, *i.e.*, a diagonal entry is zero when the corresponding weight is zero, the form of the update equations shows that this does not matter. If a diagonal entry in D is zero, replacing its corresponding entry in D^{-1} by any nonzero value does not change the result of Equations A.10, A.11, and A.12, as the zero weight cancels it out.

A.2 Generalized and Standard Bregman Divergence

We prove that the Generalized Bregman divergence between $\theta, x \in \mathbb{R}$,

$$\mathbb{D}_F(\theta \,||\, x) = F(\theta) + F^*(x) - \theta x,$$

is equivalent to the standard definition [15, 20],

$$D_{F^*}(x \,||\, f(\theta)) = F^*(x) - F^*(f(\theta)) - \nabla F^*(f(\theta))(x - f(\theta)),$$

when F is convex, twice-differentiable, closed and $f = \nabla F$ is invertible. By the definition of the convex conjugate,

$$\mathbb{D}_F(\theta \,||\, x) = F(\theta) + \sup_{\phi}\{\phi x - F(\phi)\} - \theta x,$$

where at the supremum $\theta = f^{-1}(x)$. Therefore,

$$\begin{aligned} \mathbb{D}_F(\theta \,||\, x) &= F(\theta) + f^{-1}(x) \cdot x - F(f^{-1}(x)) - \theta x \\ &= F(\theta) - F(f^{-1}(x)) - x(\theta - f^{-1}(x)) \\ &= F(\theta) - F(f^{-1}(x)) - \nabla F(f^{-1}(x))(\theta - f^{-1}(x)) \\ &= D_F(\theta \,||\, f^{-1}(x)). \end{aligned}$$

For the above choice of F, $D_F(\theta \,||\, f^{-1}(x)) = D_{F^*}(x \,||\, f(\theta))$. When F is the log-partition function of a regular exponential family, it satisfies the properties required for the equivalence between generalized and standard Bregman divergences.

Bibliography

[1] David J. Aldous. Representations for partially exchangeable arrays of random variables. *Journal of Multivariate Analysis*, 11(4):581–598, 1981.

[2] David J. Aldous. *Exchangeability and related topics*, volume 1117/1985 of *Lecture Notes in Mathematics*, chapter 1, pages 1–198. Springer Berlin / Heidelberg, 1985.

[3] Katy S. Azoury and M.K. Warmuth. Relative loss bounds for on-line density estimation with the exponential family of distributions. *Machine Learning*, 43:211–246, 2001.

[4] Francis R. Bach and Michael I. Jordan. A probabilistic interpretation of canonical correlation analysis. Technical Report 688, Department of Statistics, University of California Berkeley, 2005.

[5] Arindam Banerjee, Srujana Merugu, Inderjit S. Dhillon, and Joydeep Ghosh. Clustering with Bregman divergences. *Journal of Machine Learning Research*, 6:1705–1749, 2005.

[6] Arindam Banerjee, Sugato Basu, and Srujana Merugu. Multi-way clustering on relation graphs. In *Proceedings of the Seventh International Conference on Data Mining (SDM)*. SIAM, 2007.

[7] Matthew J. Beal. *Variational Algorithms for Approximate Bayesian Inference*. PhD thesis, Gatsby Computational Neuroscience Unit, University College London, 2003.

[8] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[9] Léon Bottou. Online algorithms and stochastic approximations. In David Saad, editor, *Online Learning and Neural Networks*. Cambridge UP, 1998.

[10] Léon Bottou. Stochastic learning. *Advanced Lectures on Machine Learning, Lecture Notes in Artificial Intelligence (LNAI)*, 3176:146–168, 2004.

[11] Léon Bottou and Yann LeCun. Large scale online learning. In Sebastian Thrun, Lawrence K. Saul, and Bernhard Schölkopf, editors, *NIPS*. MIT Press, 2003.

[12] Léon Bottou and Yann LeCun. On-line learning for very large datasets. *Applied Stochastic Models in Business and Industry*, 21(2):137–151, 2005.

[13] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge UP, 2004.

[14] Thorsten Brants and Alex Franz. Web 1T 5-gram corpus, version 1. Electronic, September 2006.

[15] Lev M. Bregman. The relaxation method of finding the common points of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7:200–217, 1967.

[16] Wray Buntine and Aleks Jakulin. Applying discrete PCA in data analysis. In David M. Chickering and Joseph Y. Halpern, editors, *Proceedings of the Twentieth Conference in Uncertainty in Artificial Intelligence (UAI)*, pages 59–66, Arlington, Virginia, United States, 2004. AUAI Press.

[17] Wray Buntine and Sami Pettru. Is multinomial PCA multi-faceted clustering or dimensionality reduction? In Christopher M. Bishop and Brendan J. Frey, editors, *Artificial Intelligence and Statistics (AISTATS)*, pages 300–307, 2003.

[18] Wray L. Buntine. Operations for learning with graphical models. *Journal of Artificial Intelligence Research*, 2:159–225, 1994.

[19] Wray L. Buntine and Aleks Jakulin. Discrete component analysis. In Craig Saunders, Marko Grobelnik, Steve R. Gunn, and John Shawe-Taylor, editors, *Subspace, Latent Structure and Feature Selection, Statistical and Optimization, Perspectives Workshop (SLSFS)*, volume 3940 of *Lecture Notes in Computer Science*, pages 1–33. Springer, 2005.

[20] Yair Censor and Stavros A. Zenios. *Parallel Optimization: Theory, Algorithms, and Applications*. Oxford UP, 1997.

[21] Jonathan Chang and David Blei. Relational topic models for document networks. In *Artificial Intelligence and Statistics (AISTATS)*, 2009.

[22] Peter P. Chen. The entity-relationship model: Toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, March 1976.

[23] Siddhartha Chib and Edward Greenberg. Understanding the Metropolis-Hastings algorithm. *The American Statistician*, 49(4):327–335, November 1995.

[24] David Cohn and Thomas Hofmann. The missing link–a probabilistic model of document content and hypertext connectivity. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems (NIPS) 13*. MIT Press, 2000.

[25] David Cohn, Deepak Verma, and Karl Pfleger. Recursive attribute factoring. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems (NIPS) 19*, Cambridge, MA, 2006. MIT Press.

[26] Michael Collins, Sanjoy Dasgupta, and Robert E. Schapire. A generalization of principal component analysis to the exponential family. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems (NIPS) 13*. MIT Press, 2001.

[27] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[28] Chris H. Q. Ding, Tao Li, and Wei Peng. Nonnegative matrix factorization and probabilistic latent semantic indexing: Equivalence chi-square statistic, and a hybrid method. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI)*. AAAI Press, 2006.

[29] Chris H. Q. Ding, Tao Li, Wei Peng, and Haesun Park. Orthogonal nonnegative matrix tri-factorizations for clustering. In Tina Eliassi-Rad, Lyle H. Ungar, Mark Craven, and Dimitrios Gunopulos, editors, *Proceedings of the Twelfth International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 126–135. ACM, 2006.

[30] Pedro Domingos. The role of Occam's Razor in knowledge discovery. *Data Mining and Knowledge Discovery*, 3(4):409–425, 1999.

[31] Pedro Domingos. Bayesian averaging of classifiers and the overfitting problem. In Pat Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, pages 223–230. Morgan Kaufmann, 2000.

[32] Pedro Domingos and Daniel Lowd. *Markov Logic: An Interface Layer for Artificial Intelligence*. Morgan & Claypool, 2009.

[33] David Draper. Assesment and propagation of model uncertainty. *Journal of the Royal Statistical Society B*, 57(1):45–97, 1995.

[34] Petros Drineas, Michael W. Mahoney, and S. Muthukrishnan. Relative-error CUR matrix decompositions. *SIAM Journal on Matrix Analysis and Applications*, 30: 844–881, 2006.

[35] Jürgen Forster and Manfred K. Warmuth. Relative expected instantaneous loss bounds. In Nicolò Cesa-Bianchi and Sally A. Goldman, editors, *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory (COLT)*, pages 90–99. Morgan Kaufmann, 2000.

[36] K. Ruben Gabriel and S. Zamir. Lower rank approximation of matrices by least squares with any choice of weights. *Technometrics*, 21(4):489–498, 1979.

[37] Erich Gamma, Richard Helm, Ralph Johnson, and John M. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

[38] Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian Data Analysis*. CRC Press, 2nd edition, 2004.

[39] L. Getoor, N. Friedman, D. Koller, and A. Pfeffer. Learning probabilistic relational models. *Relational Data Mining*, pages 307–335, 2001.

[40] Lise Getoor and Ben Taskar, editors. *Introduction to Statistical Relational Learning*. MIT Press, 2008.

[41] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval Journal*, 4(2): 133–151, 2001.

[42] Gene H. Golub and Charles F. Van Loan. *Matrix Computions*. John Hopkins UP, 3rd edition, 1996.

[43] Noah Goodman, Vikash K. Mansinghka, Daniel M. Roy, Keith Bonawitz, and Joshua B. Tenenbaum. Church: a language for generative models. In David A. McAllester and Petri Myllymäki, editors, *Proceedings of the Twenty-Fourth Conference in Uncertainty in Artificial Intelligence (UAI)*, pages 220–229. AUAI Press, 2008.

[44] Geoffrey J. Gordon. *Approximate Solutions to Markov Decision Processes*. PhD thesis, Carnegie Mellon University, 1999.

[45] Geoffrey J. Gordon. Generalized^2 linear^2 models. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems (NIPS) 15*. MIT Press, 2002.

[46] Carlos Guestrin, Andreas Krause, and Ajit Paul Singh. Near-optimal sensor placements in Gaussian processes. In Luc De Raedt and Stefan Wrobel, editors, *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML)*, pages 265–272. ACM, 2005.

[47] Donna Harman. Overview of the second text retrieval conference (TREC-2). *Information Processing and Management*, 31(3):271–289, 1995.

[48] J.A. Hartigan. *Clustering Algorithms*. Wiley, 1975.

[49] David Heckerman, Chris Meek, and Daphne Koller. Probabilistic entity-relationship models, PRMs, and plate models. In Getoor and Taskar [40], chapter 7, pages 201–238.

[50] Jennifer A. Hoeting, David Madigan, Adrian E. Raftery, and Chris T. Volinsky. Bayesian model averaging: A tutorial. *Statistical Science*, 14(4):382–417, 1999.

[51] Peter Hoff. Multiplicative latent factor models for description and prediction of social networks. *Computational and Mathematical Organization Theory*, 2008.

[52] Thomas Hofmann. Probabilistic latent semantic indexing. In *SIGIR*, pages 50–57. ACM, 1999.

[53] H. Hotelling. Relations between two sets of variates. *Biometrika*, 28(3-4):321–377, 1936.

[54] Internet Movie Database Inc. IMDB alternate interfaces. `http://www.imdb.com/interfaces`, January 2007.

[55] David Jensen and Jennifer Neville. Linkage and autocorrelation cause feature selection bias in relational learning. In Claude Sammut and Achim G. Hoffmann, editors, *Proceedings of the Nineteenth International Conference on Machine Learning (ICML)*, pages 259–266. Morgan Kaufmann Publishers Inc., 2002.

[56] Robert E. Kass and Adrian E. Raftery. Bayes factors. *Journal of the American Statistical Association*, 90(430), 1995.

[57] Qifa Ke and Takeo Kanade. Robust ℓ_1 norm factorization in the presence of outliers and missing data by alternative convex programming. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 739–746. IEEE Computer Society, 2005.

[58] Roger Koenker and Jr. Bassett, Gilbert. Regression quantiles. *Econometrica*, 46(1): 33–50, 1978.

[59] Stefan Kramer, Nada Lavrac, and Peter Flach. *Propositionalization approaches to relational data mining*, chapter 11, pages 262–286. Springer-Verlag New York, Inc., 2000.

[60] Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9:235–284, 2008.

[61] Brian Kulis, Mátyás A. Sustik, and Inderjit S. Dhillon. Low-rank kernel learning with Bregman matrix divergences. *Journal of Machine Learning Research*, 10: 341–376, February 2009.

[62] Nada Lavrac and Saso Dzeroski. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, New York, 1994.

[63] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *NIPS*. MIT Press, 2001.

[64] J. De Leeuw. Block relaxation algorithms in statistics. In H.H. Bock, W. Lenski, and M.M. Richter, editors, *Information Systems and Data Analysis*. Springer, 1994.

[65] Erich Leo Lehmann and Joseph P. Romano. *Testing Statistical Hypotheses*. Springer, 3rd edition, 2008.

[66] Hector J. Levesque and Gerhard Lakemeyer. *The Logic of Knowledge Bases*. MIT Press, 2001.

[67] Yew Jin Lim and Yee Whye Teh. Variational Bayesian approach to movie rating prediction. In *KDD-Cup Workshop*, 2007.

[68] Christoph Lippert, Stefan Hagen Weber, Yi Huang, Volker Tresp, Mathhias Schubert, and Hans-Peter Kriegel. Relation prediction in multi-relational domains using matrix factorization. In *NIPS 2008 Workshop: Structured Input–Structured Output*, 2008.

[69] Li Liu, Douglas M. Hawkins, Suhoy Ghosh, and S. Stanley Young. Robust singular value decomposition analysis of microarray dataset. *Proceedings of the National Academy of Sciences (PNAS)*, 100(23):13167–13172, November 2003.

[70] Bo Long, Zhongfei (Mark) Zhang, Xiaoyun Wú;, and Philip S. Yu. Spectral clustering for multi-type relational data. In William W. Cohen and Andrew Moore, editors, *ICML*, pages 585–592, New York, NY, USA, 2006. ACM Press.

[71] Bo Long, Zhongfei (Mark) Zhang, Xiaoyun Wu, and Philip S. Yu. Relational clustering by symmetric convex coding. In Zoubin Ghahramani, editor, *ICML*, volume 227 of *ACM International Conference Proceeding Series*, pages 569–576, New York, NY, USA, 2007. ACM.

[72] Bo Long, Zhongfei Mark Zhang, and Philip S. Yu. A probabilistic framework for relational clustering. In Pavel Berkhin, Rich Caruana, and Xindong Wu, editors, *KDD*, pages 470–479, New York, NY, USA, 2007. ACM.

[73] Hao Ma, Haixuan Yang, Michael R. Lyu, and Irwin King. SoRec: Social recommendation using probabilistic matrix factorization. In James G. Shanahan, Sihem Amer-Yahia, Ioana Manolescu, Yi Zhang, David A. Evans, Aleksander Kolcz, Key-Sun Choi, and Abdur Chowdhury, editors, *Proceedings of the Seventeenth Conference on Information and Knowledge Management (CIKM)*, pages 931–940. ACM, 2008.

[74] David MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge UP, 2003.

[75] Jan R. Magnus and Karim M. Abadir. On some definitions in matrix algebra. Working Paper CIRJE-F-426, University of Tokyo, February 2007.

[76] Jan R. Magnus and Heinz Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. John Wiley, 2007.

[77] Andrew McCallum, Khashayar Rohanemanesh, Michael Wick, Karl Schultz, and Sameer Singh. FACTORIE: Efficient probabilistic programming for relational factor graphs via imperative declarations of structure, inference and learning. In *NIPS Workshop on Probabilistic Programming*, 2008.

[78] P. McCullagh and J.A. Nelder. *Generalized Linear Models*. Chapman and Hall: London., 1989.

[79] Brian Christopher Milch. *Probabilistic Models with Unknown Objects*. PhD thesis, University of California, Berkeley, 2006. Also Technical Report EECS-2006-174, Dept. of Electrical Engineering and Computer Sciences, U.C. Berkeley, December 2006.

[80] T. Minka, J.M. Winn, J.P. Guiver, and A. Kannan. Infer.NET 2.2, 2009. Microsoft Research Cambridge. `http://research.microsoft.com/infernet`.

[81] Thomas P. Minka. Bayesian model averaging is not model combination. Technical report, MIT Media Lab, 2000.

[82] Einat Minkov and William W. Cohen. Learning to rank typed graph walks: Local and global approaches. In *WebKDD and SNA-KDD Joint Workshop*, 2007.

[83] Tom M. Mitchell, Svetlana V. Shinkareva, Andrew Carlson, Kai-Min Chang, Vicente L. Malave, Robert A. Mason, and Marcel A. Just. Predicting human brain activity associated with the meanings of nouns. *Science*, 2008.

[84] Shakir Mohamed, Katherine A. Heller, and Zoubin Ghahramani. Bayesian exponential family PCA. In Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Léon Bottou, editors, *Advances in Neural Information Processing Systems (NIPS) 21*. MIT Press, 2008.

[85] Stephen Muggleton and Wray L. Buntine. Machine invention of first order predicates by inverting resolution. In *Proceedings of the Fifth International Conference on Machine Learning (ML)*, pages 339–352, 1988.

[86] Stephen Muggleton and Cao Feng. Efficient induction of logic programs. In *First International Workshop on Algorithmic Learning Theory (ALT)*, pages 368–381, 1990.

[87] Ramesh Nallapati, Amr Ahmed, Eric P. Xing, and William W. Cohen. Joint latent topic models for text and citations. In Ying Li, Bing Liu, and Sunita Sarawagi, editors, *Proceedings of the Fourteenth Intenational Conference on Knowledge Discovery and Data Mining (KDD)*, pages 542–550. ACM, 2008.

[88] Radford M. Neal. Probabilistic inference using Markov Chain Monte Carlo methods. Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto, 1993.

[89] Netflix. Netflix prize dataset. `http://www.netflixprize.com`, January 2007.

[90] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer, 1999.

[91] P. Paatero and U. Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics,*, 5:111–126, 1994.

[92] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford University, 1998.

[93] James D. Park. Map complexity results and approximation methods. In Adnan Darwiche and Nir Friedman, editors, *Proceedings of the Eighteenth Conference in Uncertainty in Artificial Intelligence (UAI)*, pages 388–396. Morgan Kaufmann Publishers, Inc, 2002.

[94] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[95] Francisco Pereira and Geoffrey Gordon. The support vector decomposition machine. In William W. Cohen and Andrew Moore, editors, *Proceedings of the Twenty-Third International Conference on Machine Learning (ICML)*, pages 689–696. ACM Press, 2006.

[96] Avi Pfeffer. IBAL: A probabilistic rational programming language. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 733–740. Morgan Kaufmann Publishers, 2001.

[97] A. Popescul and L. Ungar. Structural logistic regression for link analysis. KDD Workshop on Multi-Relational Data Mining, 2003.

[98] Yuan Qi and Thomas P. Minka. Hessian-based Markov Chain Monte Carlo algorithms. In *First Cape Cod Workshop on Monte Carlo Methods, Cape Cod, Massachusetts*. 2002.

[99] J. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5(3): 239–266, 1990.

[100] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, February 1989.

[101] L. De Raedt and K. Kersting. Probabilistic inductive logic programming. In *Proceedings of the Fifteenth International Conference on Algorithmic Learning Theory (ALT)*, 2004.

[102] Adrian E. Raftery. Bayesian model selection in structural equation models. In *Testing Stuctural Equation Models*. Sage Publications Inc., 1993.

[103] Adrian E. Raftery, David Madigan, and Chris T. Volinsky. Accounting for model uncertainty in survival analysis improves predictive performance. *Bayesian Statistics*, 5:323–349, 1995.

[104] Jason D. M. Rennie and Nathan Srebro. Fast maximum margin matrix factorization for collaborative prediction. In Luc De Raedt and Stefan Wrobel, editors, *Proceedings of the Twenty-Second International Conference on Machine Learning (ICML)*, pages 713–719. ACM Press, 2005.

[105] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.

[106] Irina Rish, Genady Grabarnik, Guillermo Cecchi, Francisco Pereira, and Geoffrey J. Gordon. Closed-form supervised dimensionality reduction with generalized linear models. In William W. Cohen, Andrew McCallum, and Sam T. Roweis, editors, *Proceedings of the Twenty-Fifth International Conference on Machine Learning (ICML)*, volume 307, pages 832–839. ACM, 2008.

[107] Indrayana Rustandi, Marcel Adam Just, and Tom M. Mitchell. Integrating multiple-study multiple-subject fMRI datasets using canonical correlation analysis. In *Proceedings of the MICCAI Workshop: Statistical modeling and detection issues in intra- and inter-subject functional MRI data analysis*, 2009.

[108] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis, editors, *Advances in Neural Information Processing Systems (NIPS) 20*. MIT Press, 2007.

[109] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using MCMC. In William W. Cohen, Andrew McCallum, and Sam T. Roweis, editors, *Proceedings of the Twenty-Fifth International Conference on Machine Learning (ICML)*, volume 307. ACM, 2008.

[110] C. Sammut. *Learning concepts by performing experiments*. PhD thesis, University of New South Wales, Australia, 1981.

[111] C. Sammut and R. Banerji. *Learning concepts by asking questions*, pages 167–192. Morgan Kaufman, 1986.

[112] Andrew I. Schein, Lawrence K. Saul, and Lyle H. Ungar. A generalized linear model for principal component analysis of binary data. In Christopher M. Bishop and Brendan J. Frey, editors, *Artificial Intelligence and Statistics (AISTATS)*, January 2003.

[113] Mark Schmidt, Glenn Fung, and Rómer Rosales. Fast optimization methods for L1 regularization: A comparative study and two new approaches. In Joost N. Kok, Jacek Koronacki, Ramon López de Mántaras, Stan Matwin, Dunja Mladenic, and Andrzej Skowron, editors, *Proceedings of the Eighteenth European Conference on Machine Learning (ECML)*, volume 4701 of *Lecture Notes in Computer Science*, pages 286–297. Springer, 2007.

[114] Mikkel N. Schmidt, Ole Winther, and Lars Kai Hansen. Bayesian non-negative matrix factorization. In *International Conference on Independent Component Analysis and Signal Separation*, 2009.

[115] E. Shapiro. *Algorithmic Program Debugging*. MIT Press, 1983.

[116] Ajit P. Singh and Geoffrey J. Gordon. Relational learning via collective matrix factorization. In *KDD*, 2008.

[117] Ajit P. Singh, Asela Gunawardana, Chris Meek, and Arun C. Surendran. Recommendations using absorbing random walks. In *Proceedings of the North East Student Colloquium on Artificial Intelligence (NESCAI)*, 2007.

[118] David J. Spiegelhalter. Bayesian graphical modelling: a case-study in monitoring health outcomes. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 47(1):115–133, 2002.

[119] Nathan Srebro and Tommi Jaakola. Weighted low-rank approximations. In Tom Fawcett and Nina Mishra, editors, *Proceedings of the Twentieth International Conference on Machine Learning (ICML)*. AAAI Press, 2003.

[120] Nathan Srebro, Jason D.M. Rennie, and Tommi S. Jaakkola. Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems (NIPS) 17*, 2004.

[121] A. Srinivasan. The Aleph manual. Technical report, Computing Laboratory, Oxford, 2000. `http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/`.

[122] G.W. Stewart. On the early history of the singular value decomposition. Technical report, University of Maryland, Department of Computer Science, 1992.

[123] Petre Stoica and Yngve Selen. Cyclic minimizers, majorization techniques, and the expectation-maximization algorithm: a refresher. *Signal Processing Magazine, IEEE*, 21(1):112–114, Jan 2004.

[124] Ben Taskar, Abbeel Pieter, and Daphne Koller. Discriminative probabilistic models for relational data. In Adnan Darwiche and Nir Friedman, editors, *Proceedings of the Eighteenth Conference in Uncertainty in Artificial Intelligence (UAI)*, pages 485–49, San Francisco, CA, 2002. Morgan Kaufmann.

[125] The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29, 2000.

[126] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society B*, 58(1):267–288, 1996.

[127] Hanghang Tong, Christos Faloutsos, Brian Gallagher, and Tina Eliassi-Rad. Fast best-effort pattern matching in large attributed graphs. In Pavel Berkhin, Rich Caruana, and Xindong Wu, editors, *Proceedings of the Thirteenth International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 737–746. ACM, 2007.

[128] Koji Tsuda. Machine learning with quantum relative entropy. *Journal of Physics: Conference Series*, 143:012021 (9pp), 2009.

[129] Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, September 1966.

[130] Larry Wasserman. Bayesian model selection and model averaging. *Journal of Mathematical Psychology*, 44(1):92–107, March 2000. ISSN 0022-2496.

[131] Max Welling, Chaitanya Chemudugunta, and Nathan Sutter. Deterministic latent variable models and their pitfalls. In *Proceedings of the Eighth International Conference on Data Mining (SDM)*. SIAM, 2008.

[132] Sinead Williamson and Zoubin Ghahramani. Probabilistic models for data combination in recommender systems. In *NIPS 2008 Workshop: Learning from Multiple Sources*, 2008.

[133] Kai Yu, Shipeng Yu, and Volker Tresp. Multi-label informed latent semantic indexing. In Ricardo A. Baeza-Yates, Nivio Ziviani, Gary Marchionini, Alistair Moffat, and John Tait, editors, *Proceedings of the Twenty-Eighth International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 258–265. ACM, 2005.

[134] Shipeng Yu, Kai Yu, Volker Tresp, Hans-Peter Kriegel, and Mingrui Wu. Supervised probabilistic principal component analysis. In Tina Eliassi-Rad, Lyle H. Ungar, Mark Craven, and Dimitrios Gunopulos, editors, *Proceedings of the Twelfth International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 464–473. ACM, 2006.

[135] Shenghuo Zhu, Kai Yu, Yun Chi, and Yihong Gong. Combining content and link for classification using matrix factorization. In Wessel Kraaij, Arjen P. de Vries, Charles L. A. Clarke, Norbert Fuhr, and Noriko Kando, editors, *Proceedings of the Thirtieth International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 487–494. ACM Press, 2007.

ML

MACHINE LEARNING
DEPARTMENT

Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213

Carnegie Mellon