Collaborative Filtering: A Machine Learning Perspective

by

Benjamin Marlin

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

Abstract

Collaborative Filtering: A Machine Learning Perspective

Benjamin Marlin

Master of Science

Graduate Department of Computer Science

University of Toronto

2004

Collaborative filtering was initially proposed as a framework for filtering information based on the preferences of users, and has since been refined in many different ways. This thesis is a comprehensive study of rating-based, pure, non-sequential collaborative filtering. We analyze existing methods for the task of rating prediction from a machine learning perspective. We show that many existing methods proposed for this task are simple applications or modifications of one or more standard machine learning methods for classification, regression, clustering, dimensionality reduction, and density estimation. We introduce new prediction methods in all of these classes. We introduce a new experimental procedure for testing stronger forms of generalization than has been used previously. We implement a total of nine prediction methods, and conduct large scale prediction accuracy experiments. We show interesting new results on the relative performance of these methods.

Acknowledgements

I would like to begin by thanking my supervisor Richard Zemel for introducing me to the field of collaborative filtering, for numerous helpful discussions about a multitude of models and methods, and for many constructive comments about this thesis itself.

I would like to thank my second reader Sam Roweis for his thorough review of this thesis, as well as for many interesting discussions of this and other research. I would like to thank Matthew Beal for knowledgeably and enthusiastically answering more than a few queries about graphical models and variational methods. I would like to thank David Blei for our discussion of LDA and URP during his visit to the University this fall. I would also like to thank all the members of the machine learning group at the University of Toronto for comments on several presentations relating to collaborative filtering.

Empirical research into collaborative filtering methods is not possible without suitable data sets, and large amounts of computer time. The empirical results presented in this thesis are based on the EachMovie and MovieLens data sets that have been generously made available for research purposes. I would like to thank the Compaq Computer Corporation for making the EachMovie data set available, and the GroupLens Research Project at the University of Minnesota for use of the MovieLens data set. I would like to thank Geoff Hinton for keeping us all well supplied with computing power.

On a personal note, I would also like to thank my good friends Horst, Jenn, Josh, Kevin, Liam, and Rama for many entertaining lunches and dinners, and for making the AI lab an enjoyable place to work. I would like to thank my parents who have taught me much, but above all the value of hard work. Finally, I would like to thank my fiancee Krisztina who has given me boundless support, encouragement, and motivation. She has graciously agreed to share me with the University while I persue doctoral studies, and I thank her for that as well.

Contents

1 Introduction **1**

2 Formulations **4**

2.1 A Space of Formulations 4

2.1.1 Preference Indicators 5

2.1.2 Additional Features 6

2.1.3 Preference Dynamics 7

2.2 Pure, Non-Sequential, Rating-Based Formulation 7

2.2.1 Formal Definition 8

2.2.2 Associated Tasks 8

3 Fundamentals **10**

3.1 Probability and Statistics 11

3.2 Complexity Analysis 13

3.3 Experimentation 14

3.3.1 Experimental Protocols 14

3.3.2 Error Measures 15

3.3.3 Data Sets 16

3.3.4 The Missing at Random Assumption 18

4 Classification and Regression **20**

4.1 K-Nearest Neighbor Classifier 21
4.1.1 Neighborhood-Based Rating Prediction 22
4.1.2 Complexity 24
4.1.3 Results 25
4.2 Naive Bayes Classifier 26
4.2.1 Naive Bayes Rating Prediction 28
4.2.2 Complexity 30
4.2.3 Results 30
4.3 Other Classification and Regression Techniques 31

5 Clustering **32**
5.1 Standard Clustering 33
5.1.1 Rating Prediction 34
5.1.2 Complexity 35
5.1.3 Results 36
5.2 Hierarchical Clustering 36
5.2.1 Rating Prediction 38

6 Dimensionality Reduction **39**
6.1 Singular Value Decomposition 39
6.1.1 Weighted Low Rank Approximations 40
6.1.2 Learning with Weighted SVD 42
6.1.3 Rating Prediction with Weighted SVD 43
6.1.4 Complexity 44
6.1.5 Results 45
6.2 Principal Components Analysis 45
6.2.1 Rating Prediction with PCA 46
6.3 Factor Analysis and Probabilistic PCA 47

6.3.1 Rating Prediction with Probabilistic PCA 49

7 Probabilistic Rating Models **51**

7.1 The Multinomial Model 52

7.1.1 Learning 53

7.1.2 Rating Prediction 53

7.1.3 Complexity 54

7.1.4 Results 55

7.2 Mixture of Multinomials Model 55

7.2.1 Learning 57

7.2.2 Rating Prediction 60

7.2.3 Complexity 61

7.2.4 Results 61

7.3 The Aspect Model 62

7.3.1 Learning 65

7.3.2 Rating Prediction 68

7.3.3 Complexity 69

7.3.4 Results 69

7.4 The User Rating Profile Model 70

7.4.1 Variational Approximation and Free Energy 72

7.4.2 Learning Variational Parameters 75

7.4.3 Learning Model Parameters 76

7.4.4 An Equivalence Between The Aspect Model and URP 78

7.4.5 URP Rating Prediction 80

7.4.6 Complexity 81

7.4.7 Results 82

7.5 The Attitude Model 82

7.5.1 Variational Approximation and Free Energy 85

7.5.2 Learning 86
7.5.3 Rating Prediction 87
7.5.4 Binary Attitude Model 88
7.5.5 Integer Attitude Model 92
7.5.6 Complexity 94
7.5.7 Results 96

8 Comparison of Methods **98**

8.1 Complexity 98
8.2 Prediction Accuracy 102

9 Conclusions **109**

9.1 Summary 109
9.2 Future Work 111
9.2.1 Existing Models 111
9.2.2 The Missing at Random Assumption 113
9.2.3 Extensions to Additional Formulations 114
9.2.4 Generalizations to Additional Applications 116
9.3 The Last Word 116

Bibliography **119**

List of Tables

3.1 Each Movie and MovieLens Data Set Statistics 17

4.1 PKNN-Predict: EachMovie Results 26

4.2 PKNN-Predict: MovieLens Results 26

4.3 NBClass-Predict: EachMovie Results 31

4.4 NBClass-Predict: MovieLens Results 31

5.1 K-Medians Clustering: EachMovie Results 36

5.2 K-Medians Clustering: MovieLens Results 36

6.1 wSVD-Predict: EachMovie Results 45

6.2 wSVD-Predict: MovieLens Results 45

7.1 MixMulti-Predict: EachMovie Results 61

7.2 MixMulti-Predict: MovieLens Results 61

7.3 Aspect-Predict: EachMovie Results 70

7.4 Aspect-Predict: MovieLens Results 70

7.5 URP: EachMovie Results 82

7.6 URP: MovieLens Results 82

7.7 AttBin-Predict: EachMovie Results 97

7.8 AttBin-Predict: MovieLens Results 97

8.1 Computational Complexity Of Learning and Prediction Methods 99

8.2 Space Complexity of Learned Representation 101
8.3 EachMovie: Prediction Results 103
8.4 MovieLens: Prediction Results 103

List of Figures

3.1 EachMovie Rating Distributions 18

3.2 MovieLens Rating Distributions 18

4.1 Naive Bayes Classifier 26

4.2 Naive Bayes classifier for rating prediction. 28

6.1 Factor Analysis and Probabilistic PCA Graphical Model. 48

7.1 Multinomial Model 53

7.2 Mixture of Multinomials Model. 56

7.3 Dyadic aspect model. 62

7.4 Triadic aspect model. 62

7.5 Vector aspect model. 64

7.6 LDA model. 71

7.7 URP model. 71

7.8 The attitude model. 84

8.1 Comparison of EachMovie weak generalization performance. 107

8.2 Comparison of EachMovie strong generalization performance. 107

8.3 Comparison of MovieLens weak generalization performance. 108

8.4 Comparison of MovieLens strong generalization performance. 108

List of Algorithms

4.1 PKNN-Predict 24

4.2 NBClass-Learn 29

4.3 NBClass-Predict 29

5.1 KMedians-Learn 35

5.2 KMedians-Predict 35

6.1 wSVD-Learn 43

6.2 wSVD-Predict 44

7.1 Multi-Learn 54

7.2 Multi-Predict 54

7.3 MixMulti-Learn 59

7.4 MixMulti-Predict 60

7.5 Aspect-Learn 68

7.6 Aspect-Predict 69

7.7 URP-VarInf 76

7.8 URP-AlphaUpdate 78

7.9 URP-Learn 79

7.10 URP-Predict 81

7.11 AttBin-VarInf 90

7.12 AttBin-Learn 91

7.13 AttBin-Predict 92

7.14 AttInt-VarInf 94
7.15 AttInt-Learn 95
7.16 AttInt-Predict 96

Chapter 1

Introduction

The problem of information overload was identified as early as 1982 in an ACM President's Letter by Peter J. Denning aptly titled *Electronic Junk* [17]. Denning argued that the deployment of office information systems technology coupled with a quickly increasing use of electronic mail was sure to overwhelm computer users. Since that time many new sources of information have become available through the Internet including a vast archive of hundreds of millions of Usenet news articles, and an immense collection of billions of web pages. In addition, mainstream media continue to produce new books, movies, and music at a staggering pace.

The response to the accelerating problem of information overload by the computer science community was the founding of a new research area called *information filtering*. Work in this area has largely focused on filtering text documents based on representations of their content. However, Goldberg, Nichols, Oki, and Terry founded an orthogonal research direction termed *collaborative filtering* based on filtering arbitrary information items according to user preferences [23, p. 61]. In the current literature, collaborative filtering is most often thought of as the problem of *recommendation*, the filtering-in of information items that a particular individual will like or find usefull.

However, it is incorrect to think of collaborative filtering as a single problem. Rather,

the field of collaborative filtering consists of a collection of collaborative filtering problems, which differ according to the type of input information that is assumed. Only a fraction of these formulations has been studied in depth. In order to properly situate the work which appears in this thesis, we begin by describing a space of formulations of collaborative filtering problems in chapter 2.

We focus on a pure, non-sequential, rating-based formulation of collaborative filtering as detailed in section 2.2. This formulation is the one most often associated with collaborative filtering, and is the subject of the majority of the collaborative filtering literature. Qualitatively, this formulation has many nice properties. In particular, the recommendation task decomposes into the task of *rating prediction*, and the task of producing recommendations from a set of rating predictions. The latter is trivially accomplished by sorting information items according to their predicted ratings, and thus the rating prediction task will be our primary interest.

In chapter 3 we introduce the fundamental statistical, computational, and experimental techniques that are needed to derive, and analyze rating prediction methods within the pure, non-sequential, rating-based formulation of collaborative filtering. We describe optimization and learning methods, give a brief overview of complexity analysis for rating prediction methods, and describe experimental protocols and error measures for empirical evaluation.

As we will see in the following chapters, a great deal of research has been performed within the pure, non-sequential, rating-based formulation of collaborative filtering. While early work focused on the neighborhood methods introduced by Resnick et al. [49], new and inventive techniques have been introduced from a wide variety of disciplines including artificial intelligence, human factors, knowledge discovery, information filtering and retrieval, machine learning, and text modeling.

Regardless of their origins, many rating prediction methods can be seen as modifications or applications of standard machine learning methods. Thus, machine learning

offers a unifying perspective from which to study existing collaborative filtering research. In chapter 4 we describe rating prediction methods based on classification and regression. We present the well known class of neighborhood methods and show how they can be derived from standard K nearest neighbor classification and regression [49]. We introduce a new rating prediction method based on learning a set of naive Bayes classifiers. In chapter 5 we describe applications of clustering methods to rating prediction and introduce a new rating prediction method based on K-medians clustering. In chapter 6 we present rating prediction methods based on dimensionality reduction techniques including weighted singular value decomposition [53], principal components analysis [24], and probabilistic principal components analysis [13]. We introduce a new rating prediction algorithm that extends the existing work on weighted singular value decomposition. In chapter 7 we describe a number of methods based on density estimation in probabilistic models including a multinomial model, a mixture of multinomials model, the aspect model [29], and the user rating profile model [38]. We introduce a new family of models called the Attitude model family.

We implement a total of nine rating prediction methods and perform large scale prediction accuracy experiments. In chapter 8 we present a comparison of these methods in terms of learning complexity, prediction complexity, space complexity of learned representation, and prediction accuracy.

Chapter 2

Formulations

The original Information Tapestry system proposed by Goldberg et al. allowed users to express their opinions in the form of text annotations that were associated with particular electronic mail messages and documents. Other Tapestry users were able to specify filters for incoming documents in the form of SQL-like expressions based on the document's content, the content of the annotations, the number of annotations, and the identity of the authors of the annotations associated with each document [23].

The field of collaborative filtering research consists of a large number of information filtering problems, and this collection of formulations is highly structured. In this chapter we introduce a space of collaborative filtering problem formulations, and accurately situate the current work.

2.1 A Space of Formulations

In this section we structure the space of formulations according to three independent characteristics: the type of preference indicators used, the inclusion of additional features, and the treatment of preference dynamics. A choice for each of these three characteristics yields a particular formulation. The proposed structure covers all formulations of collaborative filtering currently under study, and many that are not.

2.1.1 Preference Indicators

The main types of preference indicators used for collaborative filtering are numerical ratings triplets, numerical rating vectors, co-occurrence pairs, and count vectors. A rating triplet has the form (u, y, r) where u is a user index, y is an item index, and r is a rating value. The triplet represents the fact that user u assigned rating r to item y. The rating values may be ordinal or continuous. A numerical rating vector has the form $\mathbf{r}^u = (r_1^u, ..., r_M^u)$ where r_y^u is the rating assigned by user u to item y. The components of the vector $\mathbf{r}^u$ are either all ordinal or all continuous values. Any component of the vector may be assigned the value $\perp$, indicating the rating for the corresponding item is unknown.

Co-occurrence pairs have the form (u, y) where u is a user index and y is an item index. The relation implied by observing a pair (u, y) is that user u viewed, accessed, or purchased item y. However, it could also indicate that user u *likes* item y. A count vector $\mathbf{n}^u = (n_1^u, ..., n_M^u)$ results when multiple co-occurrence pairs can be observed for the same user and item. In this case n_y^u may represent the number of times user u viewed item y.

These preference indicators are not completely distinct. Any rating vector can be represented as a set of rating triplets. The reverse is not true unless we assume there is at most one rating specified for every user-item pair. Count vectors and sets of co-occurrence pairs are always interchangeable. The preference indicators based on ratings are semantically different from those based on co-occurrences, and there is no straight forward transformation between the two.

A further distinction drawn between preference indicators is whether they are explicitly provided by the user, or implicitly collected while the user performs a primary task such a browsing an Internet site. Claypool et al. present an interesting comparison between implicit preference indicators and explicit ratings [15]. Requiring a user to supply explicit ratings results in a cognitive burden not present when implicit preference

indicators are collected. Claypool et al. argue that the perceived benefit of supplying explicit ratings must exceed the added cognitive burden or users will tend to rate items sparsely, or stop rating items altogether. On the other hand, Claypool et al. argue that because implicit indicators can be gathered without burdening the user, every user interaction with the collaborative filtering system results in the collection of new preference indicators.

2.1.2 Additional Features

Another decision that has important consequences in collaborative filtering is whether to only use preference indicators, or allow the use of additional features. In a pure approach users are described by their preferences for items, and items are described by the preferences users have for them. When additional content-based features are included the formulation is sometimes called hybrid collaborative filtering. Additional features can include information about users such as age and gender, and information about items such as an author and title for books, an artist and genre for music, a director, genre, and cast for movies, and a representation of content for web pages.

Pure formulations of collaborative filtering are simpler and more widely used for research than hybrid formulations. However, recent research has seen the proposal of several new algorithms that incorporate additional features. See for example the work of Basu, Hirsh, and Cohen [2], Melville, Mooney, and Nagarajan [39], as well as Schein, Popescul, and Ungar [51].

The hybrid approach purportedly reduces the effect of two well known problems with collaborative filtering systems: the cold start problem, and the new user problem. The cold start problem occurs when there are few entries recorded in the rating database. In this case more accurate recommendations can me made by recommending items according to similarities in their content-based features. The new user problem occurs in an established collaborative filtering system when recommendations must be made for a

user on the basis of few recorded ratings. In this case better recommendations may be achieved by considering similarities between users based on additional user features.

2.1.3 Preference Dynamics

Very few formulations of collaborative filtering take into account the sequence in which preference indicators are collected. Instead preference indicators are viewed as a static set of values representing a "snapshot" of the user's preferences. However, over long periods of time, or in domains where user preferences are highly dynamic, older preference indicators may become inaccurate. In certain domains a non-sequential formulation risks predictions that decrease in accuracy as a user's profile becomes filled with out of date information. This problem is especially acute when implicit preference indicators are used because the user can not directly update past preference indicator values.

Recently Pavlov and Pennock, and Girolami and Kabán have introduced methods for dealing with dynamic user profiles. The advantage of this approach is that it can deal naturally with user preferences changing over time. The disadvantage is that sequential formulations requires more complex models and prediction methods than non-sequential formulations. Pavlov and Pennock adopt a maximum entropy approach to prediction within the sequential framework. Their method performs favorably on a document recommendation task when compared to content-based methods currently in use [46]. Girolami and Kabán introduce a method for learning dynamic user profiles based on simplicial mixtures of first order Markov chains. They apply their method to a variety of data sets including a web browsing prediction task [21].

2.2 Pure, Non-Sequential, Rating-Based Formulation

Throughout this work we assume a pure, non-sequential, rating-based formulation of collaborative filtering. In this formulation users and items are described only by preference

indicators. Preference indicators are assumed to be numerical rating vectors with ordinal values. No additional features are included. Preference dynamics are ignored resulting in a non-sequential treatment of preference indicator values. This formulation was popularized by Resnick, Iacovou, Suchak, Bergstorm, and Riedl through their work on the GroupLens system [49]. We select this particular formulation because it has been the subject of the greatest amount of previous research. It is appealing due to its simplicity, and the fact that it easily accommodates objective performance evaluation. We give a detailed definition of this formulation in sub-section 2.2.1. As we describe in sub-section 2.2.2, the two tasks performed under this formulation are recommendation, and rating prediction.

2.2.1 Formal Definition

We assume that there are M items $1, ..., M$ in a collection that can be of mixed types: email messages, news articles, web pages, books, songs, movies, etc. We assume there is a set of N users $1, ..., N$. A user u can provide an opinion about an item y by assigning it a numerical rating r_y^u from the ordinal scale $1, ..., V$. Each user can supply at most one rating for each item, but we do not assume that all users supply ratings for all items. We associate a rating vector also called a user rating profile $\mathbf{r}^u \in \{1, ..., V, \perp\}^M$ with each user u. Recall that the symbol $\perp$ is used to indicate an unknown rating value.

2.2.2 Associated Tasks

The main task in any formalation of collaborative filtering is recomendation. given the rating vectors $\mathbf{r}^u$ of the N users, and the rating vector $\mathbf{r}^a$ of a particular active user a, we wish to recommend a set of items that the active user might like or find useful. As we have already noted, in a rating based formulation the task of recommendation reduces to the task of rating prediction and the task of producing recommendations from a set of predictions. In rating prediction we are given the rating vectors $\mathbf{r}^u$ of the N users, and

the rating vector $\mathbf{r}^a$ of a particular active user a. We wish to predict rating values $\hat{r}_y^a$ for all items that have not yet been assigned ratings by the active user.

Given a method for predicting the ratings of unrated items, a method for recommendation can easily be constructed by first computing predictions $\hat{r}_y^a$ for all unrated items, sorting the predicted ratings, and recommending the top T items. Therefore, the focus of research within the pure, non-sequential, rating-based formulation is developing highly accurate rating prediction methods.

Chapter 3

Fundamentals

In this chapter we introduce the background material that is needed for developing rating prediction methods, analyzing their complexity, and performing experiments in the collaborative filtering domain. We approach collaborative filtering from a machine learning standpoint, which means we draw heavily from the fields of optimization, and probability and statistics. Familiarity with the basics of non-linear optimization is assumed. The collaborative filtering problems we consider are too large for higher order optimization procedures to be feasible, so we resort to gradient descent and its variants in most cases.

Familiarity with probability and statistics and Bayesian belief networks is also assumed. In chapters 6 and 7 we introduce probabilistic models for collaborative filtering containing latent variables, variables whose values are never observed. Learning these models requires the use of an *expectation maximization* procedure. In this chapter we review the Expectation Maximization algorithm of Dempster, Laird and, Rubin [16]. We also introduce the more recent free energy interpretation of standard EM due to Neal and Hinton [44]. We follow the free energy approach of Neal and Hinton for the development of all models in chapter 7.

As we will see in the following chapters, different learning and prediction methods differ greatly in terms of computational complexity, and the complexity of the models

they construct. We introduce the basic elements of the complexity analysis we apply.

Lastly, we describe the experimental protocols used to obtain rating prediction performance results. We discuss the various error measures that are commonly used in collaborative filtering research. We also introduce the EachMovie and MovieLens data sets, and describe their main properties.

3.1 Probability and Statistics

In chapters 6 and 7 we introduce methods for rating prediction based on learning probabilistic models of rating profiles. The power of these models comes from the fact that they include latent variables as well as rating variables. The rating variables correspond to the rating of each item, and are observed or unobserved depending on a particular user's rating profile. The latent variables, which are always unobserved, facilitate the modeling of complex dependencies between the rating variables.

Let $\mathbf{x}$ be a vector of observed variables, $\mathbf{z}$ be a vector of latent variables, and θ be the model parameters. Let $\mathbf{y} = (\mathbf{x}, \mathbf{z})$ be a vector of all variables in the model. If $\mathbf{y}$ were completely observed we could apply standard maximum likelihood estimation to obtain $\theta^* = \text{argmax}_\theta \log P(\mathbf{y}|\theta)$. However, with the $\mathbf{z}$ unobserved, $\mathbf{y}$ becomes a random variable and we must apply the Expectation Maximization algorithm of Dempster et al. [16].

The Expectation Maximization algorithm is an iterative procedure for maximum likelihood estimation in the presence of unobserved variables. The algorithm begins by randomly initializing the parameters. The initial guess at the parameter values is denoted $\hat{\theta}^0$. In the expectation step the expected value of the log likelihood of the complete data $\mathbf{y}$ is estimated. The expectation is taken with respect to a distribution over $\mathbf{y}$ computed using the observed data $\mathbf{x}$ and the current estimate of θ, $\hat{\theta}^t$. This expression is called the Q-function and is written $Q(\theta|\hat{\theta}^t) = E[\log P(\mathbf{y}|\theta)|\mathbf{x}, \hat{\theta}^t]$ to indicate the dependence on the current estimate of the parameter vector θ. In the maximization step $\hat{\theta}^{t+1}$ is set

to the value which maximizes the expected complete log likelihood, $Q(\theta|\hat{\theta}^t)$. These two updates are iterated as shown below until the likelihood converges.

$$\begin{aligned} \text{E-Step:} \quad & Q(\theta|\hat{\theta}^t) \leftarrow E[\log P(\mathbf{y}|\theta)|\mathbf{x}, \hat{\theta}^t) \\ \text{M-Step:} \quad & \hat{\theta}^{t+1} \leftarrow \arg\max_{\theta} Q(\theta|\hat{\theta}^t) \end{aligned}$$

Neal and Hinton view the standard EM algorithm in a slightly different fashion. They describe the expectation step as computing a distribution $q^t(\mathbf{z}) = P(\mathbf{z}|\mathbf{x}, \hat{\theta}^t)$ over the range of $\mathbf{z}$. In the maximization step $\hat{\theta}^{t+1}$ is set to the value of θ which maximizes $E_{q^t}[\log P(\mathbf{y}|\theta)]$, the expected complete log likelihood under the q-distribution computed during the previous expectation step.

$$\begin{aligned} \text{E-Step:} \quad & q^{t+1}(\mathbf{z}) \leftarrow P(\mathbf{z}|\mathbf{x}, \hat{\theta}^t) \\ \text{M-Step:} \quad & \hat{\theta}^{t+1} \leftarrow \arg\max_{\theta} E_{q^t}[\log P(\mathbf{y}|\theta)] \end{aligned}$$

For more complex models where the parameters of the q-distribution or the parameters of the model can not be found analytically, the free energy approach of Neal and Hinton leads to more flexible model fitting procedures than standard EM [44]. As Neal and Hinton show, standard EM is equivalent to performing coordinate ascent on the free energy function $F[q, \theta] = E_q[\log P(\mathbf{x}, \mathbf{z}|\theta)] + H[q]$, where $H[q] = -E_q[\log q(\mathbf{z})]$. The q-distribution may be the exact posterior over $\mathbf{z}$ or an approximation. The free energy $F[q, \theta]$ is related to the Kullback-Liebler divergence between the q-distribution $q(\mathbf{z}|\mathbf{x})$ and the true posterior $p(\mathbf{z}|\mathbf{x})$ as follows: $\log P(\mathbf{x}, \mathbf{z}|\theta) = F[q, \theta] + D(q(\mathbf{z}||\mathbf{x})||p(\mathbf{z}|\mathbf{x}))$. The Kullback-Liebler divergence is a measure of the difference between probability distributions. It is zero if the distributions are equal, and positive otherwise, thus the free energy $F[q, \theta]$ is a lower bound on the complete data log likelihood. The EM algorithm can then be expressed as follows:

$$\begin{aligned} \text{E-Step:} \quad & q^{t+1} \leftarrow \arg\max_{q} F[q, \hat{\theta}^t] \\ \text{M-Step:} \quad & \hat{\theta}^{t+1} \leftarrow \arg\max_{\theta} F[q^{t+1}, \theta] \end{aligned}$$

Since both the E-step and the M-step maximize the same objective function $F[q, \theta]$, fitting procedures other than standard EM can be justified. In the case where the parameters of q^{t+1} or $\hat{\theta}^{t+1}$ must be found iteratively, different interleavings of the iterative updates can be used and the free energy $F[q, \theta]$ is still guaranteed to converge. However, a local maximum of the free energy will only correspond to a local maximum of the expected complete log likelihood when q^* is a true maximizer of $F[q, \hat{\theta}^*]$.

3.2 Complexity Analysis

Achieving higher prediction accuracy on rating prediction tasks often comes at the cost of higher computational or space complexity. We analyze the complexity of all learning and prediction methods which we implement to assess this fundamental tradeoff.

Many of the methods we will describe, such as those based on expectation maximization, involve iterating a set of update rules until convergence. When the computational complexity of a learning or prediction method depends on iterating a set of operations until the convergence of an objective function is obtained, we introduce the notation I to indicate this dependence. For most learning and prediction algorithms I will be a function of the number of users N, the number of items M, the number of vote values V, and a model size parameter K. We provide average case estimates of the number of iterations needed to obtain good prediction performance for data sets tested.

The space complexity of the representations found by most methods will be a function of the number of items M, the number of vote values V, and a model size parameter K. For instance based methods and certain degenerate models, the space complexity of the learned representation will also depend on the number of users N.

3.3 Experimentation

In this section we describe the experimental methodology followed in this thesis. We review different experimental protocols that have been proposed in the literature for evaluating the empirical performance of collaborative filtering methods. We also discuss the various data sets that are commonly used in rating experiments, and describe some of their important properties.

3.3.1 Experimental Protocols

Most rating prediction experiments found in the literature follow a protocol popularized by Breese, Heckerman, and Kadie [10]. In these experiments the available ratings for each user are split into an observed set, and a held out set. The observed ratings are used for training, and the held out ratings are used for testing the performance of the method. The training set may be further split if a validation set is needed. However, this protocol only measures the ability of a method to generalize to other items rated by the same users who were used for training the method. We call this *weak generalization*.

A more important type of generalization, and one overlooked in the existing collaborative filtering literature, is generalization to completely novel user profiles. We call this *strong generalization*. In a strong generalization protocol the set of users is first divided into training users and test users. Learning is performed with all available ratings from the training users. A validation set may be extracted from the training set if needed. To test the resulting method, the ratings of each test user are split into an observed set and a held out set. The method is shown the observed ratings, and is used to predict the held out ratings. A crucial point in this discussion is that some collaborative filtering methods are not designed for use with novel user profiles. In this case only the weak generalization properties of the method can be evaluated.

In both forms of generalization, testing is done by partitioning each user's ratings

into a set of observed items, and a set of held out items. This can be done in a variety of ways. If K items are observed and the rest are held out, the resulting protocol is called *Given-K*. When all of a user's ratings are observed except for one, the protocol is often referred to as *all-but-1*. Since the number of observed ratings varies naturally in the data sets used for empirical evaluations, we adopt an all-but-1 protocol for both weak and strong generalization. Note that in all cases the error rates we report are taken over sets of held out ratings used for testing, not the set of observed ratings for for training.

Collaborative filtering data sets are normally quite large, and the error estimates produced by the weak and strong generalization protocols seem to exhibit relatively low variance. Nevertheless, cross validation is used to average error rates across multiple randomly selected training, and testing sets, as well as observed and unobserved rating sets. We report the mean test error rate and standard error of the mean for all experiments.

3.3.2 Error Measures

Two principal forms of error measure have been used for evaluating the performance of collaborative filtering methods. The first form attempts to directly evaluate recommendation performance. Such evaluation methods have been studied by Breese et al. [10], but are not commonly used. The lack of sufficiently dense rating data sets renders recommendation accuracy estimates unreliable.

The second form of error measure is used to evaluate the prediction accuracy of a collaborative filtering method. Several popular instances of this form of error measure are mean squared error (MSE), mean absolute error (MAE), and mean prediction error (MPE). The definitions of all three error measures can be found below assuming N users, and one test item per user as in an all-but-1 protocol.

$$MSE = \frac{1}{N}\sum_{u=1}^{N}(\hat{r}^u_{y^u} - r^u_{y^u})^2 \tag{3.1}$$

$$MAE = \frac{1}{N}\sum_{u=1}^{N}|\hat{r}^u_{y^u} - r^u_{y^u}| \tag{3.2}$$

$$MPE = \frac{1}{N}\sum_{u=1}^{N}[\hat{r}^{u}_{y^u} \neq r_{y^u}] \tag{3.3}$$

Since we will be experimenting with data sets having different numbers of rating values we adopt a normalized mean absolute error, which enables comparison across data sets. We define our NMAE error measure to be $MAE/E[MAE]$ where $E[MAE]$ denotes the expected value of the MAE assuming uniformly distributed observed and predicted rating values. An NMAE error of less than one means a method is doing better than random, while an NMAE value of greater than one means the method is performing worse than random. Note that this is a different definition of NAME than proposed previously by Goldberg et al. [24]. In the definition of Goldberg et al. the normalizing value is taken to be $r_{max} - r_{min}$, the difference between the largest and smallest rating values. However, a large portion of the resulting error scale is not used because it corresponds to errors that are far worse than a method which makes uniformly random predictions. For example, on a scale from one to five $r_{max} - r_{min} = 4$ while $E[MAE] = 1.6$.

3.3.3 Data Sets

The single most important aspect of empirical research on rating prediction algorithms is the availability of large, dense data sets. Currently only two ordinal rating data sets are freely available for use in research. These are the EachMovie (EM) data set and the MovieLens (ML) data set. EachMovie is a movie rating data set collected by the Compaq Systems Research Center over an 18 month period beginning in 1997. The base data set contains 72916 users, 1628 movies and 2811983 ratings. Ratings are on a scale from 1 to 6. The base data set is 97.6% sparse. MovieLens is also a movie rating data set. It was collected through the on going MovieLens project, and is distributed by GroupLens Research at the University of Minnesota. MovieLens contains 6040 users, 3900 movies, and 1000209 ratings collected from users who joined the MovieLens recommendation service in 2000. Ratings are on a scale from 1 to 5. The base data set is 95.7% sparse. A

Table 3.1: Each Movie and MovieLens Data Set Statistics

Data Set	EM Ratings	EM Sparsity (%)	ML Ratings	ML Sparsity (%)
Base	2811983	97.6	1000209	95.7
Weak 1	2119898	95.6	829824	95.4
Weak 2	2116856	95.6	822259	95.4
Weak 3	2118560	95.6	825302	95.4
Strong 1	348414	95.7	164045	95.4
Strong 2	348177	95.7	172344	95.2
Strong 3	347581	95.7	166870	95.4

third data set often used in collaborative filtering research is the freely available Jester Online Joke data set collected by Goldberg et al. [24]. Jester differs from EachMovie and MovieLens in that the ratings are continuous and not ordinal. The data set is also much smaller containing 70000 users, but only 100 jokes. Since this work focuses on a formulation with ordinal ratings, the Jester data set is not used.

For the purpose of experimentation we apply further pruning to the base data sets by eliminating users and items with low numbers of observed ratings. We require a minimum of twenty ratings per user. In the case of EachMoive, this leaves about 35000 users and 1600 items from which we randomly select 30000 users for the weak generalization set, and 5000 users for the strong generalization set. Filtering the MovieLens data set leaves just over 6000 users and 3500 movies from which we randomly select 5000 users for the weak generalization set and 1000 users for the strong generalization set. For both EachMovie and MovieLens, the selection of users for weak and strong generalization is performed randomly three times creating a total of twelve data sets. Table 3.1 indicates that the filtering and sampling methods used to extract the various data sets from the base EachMove and MovieLens data sets largely preserve rating sparsity levels. Figures 3.1 and 3.2 show that the rating distributions are also largely preserved. Each bar chart gives the empirical distribution over rating values for a single data set. The horizontal axis is ordered from lowest to highest rating value.

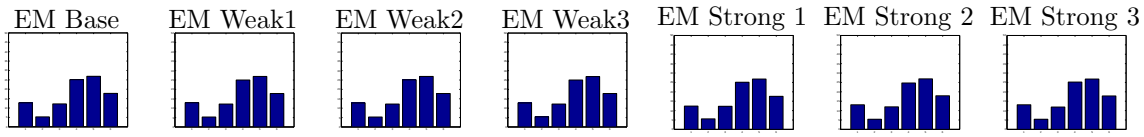

Figure 3.1: EachMovie Rating Distributions

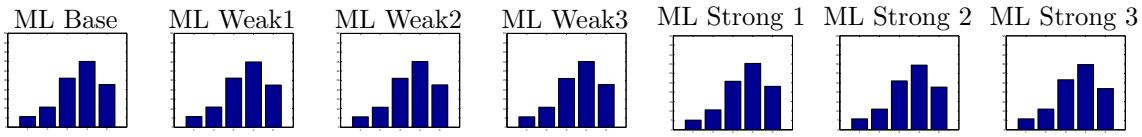

Figure 3.2: MovieLens Rating Distributions

3.3.4 The Missing at Random Assumption

One important consideration when dealing with data sets that contain large amounts of missing data is the process that causes the data to be missing. This process is referred to as the *missing data mechanism*. If the probability of having a missing value for a certain variable is unrelated to the value of the variable, then the ratings are said to be *missing completely at random*. If the probability that a variable is unobserved given the values all variables is equal to the the probability that a variable is unobserved given the values of just the observed variables, then the data is said to be *missing at random* [36]. If the data is missing completely at random or simply missing at random then the missing data mechanism can be ignored. If the data is not missing at random then ignoring the missing data mechanism can bias maximum likelihood estimates computed from the data [36].

Given a data set with missing values, it is impossible to determine whether the missing values are missing at random because their values are unknown. However, we can hypothesize based on prior knowledge of the process that generated the data. For instance, we might believe that a user is likely to only see movies that they believe they will like, and only rate movies that they have seen. In this case the probability of observing a rating value *will* depend on the user's estimate of their rating for the item. Thus the data is not missing at random, and ignoring the missing data mechanism may result in biased learning procedures.

Taking account of the missing data mechanism can be fairly easy when the goal is to find maximum likelihood estimates for simple statistics like the mean of the data. In the collaborative filtering case we are interested in optimizing the parameters of probabilistic models using maximum likelihood methods. This is a more complicated task, and the problem of incorporating missing data mechanisms into generative models has not been studied at all in the collaborative filtering literature. All existing research explicitly or implicitly makes the assumption that ratings are missing at random. While this is a very interesting issue, it is beyond the scope of the present research. In this thesis we assume that all missing ratings are missing at random, but acknowledge that the bias introduced into learning and prediction may be significant.

Chapter 4

Classification and Regression

Given an M dimensional input vector $\mathbf{x}_i$, the goal of classification or regression is to accurately predict the corresponding output value c_i. In the case of classification the outputs c take values from a finite set referred to as the set of class labels. In the case of regression, the outputs c are real valued. Each component x_{ij} of input vector $\mathbf{x}_i$ may may be categorical or numerical. Classification and regression share a common learning framework: a set of training instances $\{\mathbf{x}_i, c_i\}$ is given, and the mapping from input vectors to output values must be learned.

Rating prediction for collaborative filtering is rarely thought of in terms of classification or regression, despite the fact that some of the most well known methods fall under this heading. To see that classification offers a useful framework for personalized rating prediction consider constructing a different classifier for every item. The classifier for item y classifies users according to their rating for item y. The input features consist of ratings for items other than item y. We learn the set of classifiers independently. Some users will not have recorded a rating for item y, but it suffices to discard those users from the training set when learning the classifier for item y. Collaborative filtering can be performed as regression in a precisely analogous fashion.

Billsus and Pazzani propose an alternate framework for performing rating prediction

as classification or regression [5]. They begin by re-encode ordinal rating values on a scale of 1 to V using a binary 1-of-V encoding scheme. This is necessary when using certain classifiers that can not be applied in the presence of missing data, but is not necessary in general. The framework of Billsus and Pazzani also obscures the true relationship between standard classification techniques from machine learning, and the set of methods popularized by Resnick et al. [49], Shardanand and Maes [52], and Herlocker et al [27]. These algorithms have been called memory-based [10], similarity-based, and neighborhood-based [27] in the literature. As we show in the following section, neighborhood-based collaborative filtering methods can be interpreted as modifications of the well known K-Nearest Neighbor classifier [42].

While not explored to date, the use of other standard classifiers for rating prediction is also possible. We detail the application of the naive Bayes classifier, and briefly discuss the use of other classifiers such as decision trees and artificial neural networks.

4.1 K-Nearest Neighbor Classifier

The K-Nearest Neighbor (KNN) classifier is one of the classical examples of a memory-based, or instance-based machine learning method. A KNN classifier learns by simply storing all the training instances that are passed to it. To classify a new query vector $\mathbf{x}_q$ given the stored training set $\{\mathbf{x}_i, c_i\}$, a distance $d_{qi} = d(\mathbf{x}_q, \mathbf{x}_i)$ is computed for all i. Let $\mathbf{x}_{n_1}, ..., \mathbf{x}_{n_k}$ be the K nearest neighbors of $\mathbf{x}_q$, and $c_{n_1}, ..., c_{n_K}$ be the corresponding outputs. The output for $\mathbf{x}_q$ is then calculated as an aggregate of the class labels $c_{n_1}, ..., c_{n_K}$ [42, p. 230-231].

In the standard case where the input vectors consist of real numbers and the outputs are discrete classes, the distance function $d()$ is taken to be euclidean distance given by equation 4.1. The predicted output value is taken to be the class of the majority of $\mathbf{x}_q$'s K nearest neighbors as seen in equation 4.2. If the outputs are continuous, then the

predicted output is computed as the mean of the outputs of $\mathbf{x}_q$'s k nearest neighbors as seen in equation 4.3. This yields K-Nearest Neighbor regression.

$$d(\mathbf{x}_q, \mathbf{x}_i) = \sqrt{\sum_{j=1}^{M}(x_{nj} - x_{ij})^2} \tag{4.1}$$

$$c_q = \arg\max_{c \in C} \sum_{k=1}^{K} \delta(c, c_{n_k}) \tag{4.2}$$

$$c_q = \frac{1}{K} \sum_{k=1}^{K} c_{n_k} \tag{4.3}$$

One standard extension to KNN that can increase accuracy is to incorporate a similarity weight w_{qi}. The similarity weights is calculated as the inverse of the distance $w_{qi} = 1/d_{qi}$. This technique is applicable to both the classification, and regression cases. The modified classification and regression rules are given in equations 4.4 and 4.5 [42, p. 233-234]. An additional benefit of incorporating similarity weights is that the number of neighbors K can be set to the number of training cases N, and the presence of the weights automatically discounts the contribution of training vectors that are distant from the query vector.

$$c_q = \arg\max_{c \in C} \sum_{k=1}^{K} w_{qn_j} \delta(c, c_{n_k}) \tag{4.4}$$

$$c_q = \frac{\sum_{k=1}^{K} w_{qn_k} c_{n_k}}{\sum_{k=1}^{k} w_{qn_k}} \tag{4.5}$$

4.1.1 Neighborhood-Based Rating Prediction

Neighborhood-based rating prediction algorithms are a specialization of standard KNN regression to collaborative filtering. To make a prediction about an item y, recall that the input feature vector consists of all items in the dataset other than y. Some users will not have rated some items so the distance metric can only be calculated over the items that the active user a and each user u in the data set have rated in common.

Many specialized distance and similarity metrics have been proposed. The survey

by Herlocker et al. mentions Pearson correlation, Spearman rank correlation, vector similarity, entropy, and mean squared difference [27]. The Pearson correlation similarity metric is shown in equation 4.6. Pearson correlation was used by Resnick et al. in the GroupLens system [49], and a slight variation was used by Shardanand and Maes in the Ringo music recommender [52].

$$w_{au}^{P} = \frac{\sum_{\{y|r_y^a, r_y^u \neq \perp\}} (r_y^a - \bar{r}^a)(r_y^u - \bar{r}^u)}{\sqrt{\sum_{\{y|r_y^a, r_y^u \neq \perp\}} (r_y^a - \bar{r}^a)^2 \sum_{\{y|r_y^a, r_y^u \neq \perp\}} (r_y^u - \bar{r}^u)^2}} \tag{4.6}$$

A straight forward application of the KNN classification and regression rules to rating prediction results in the rules shown in equations 4.4 and 4.8. We have accounted for negative weights which do not occur when euclidean distance is used. We assume the active user's K nearest neighbors are given by $u_1, ..., u_K$.

$$\hat{r}_y^a = \arg\max_{v \in V} \sum_{k=1}^{K} w_{au_k} \delta(v, r_y^{u_k}) \tag{4.7}$$

$$\hat{r}_y^a = \frac{\sum_{k=1}^{K} w_{au_k} r_i^{u_k}}{\sum_{k=1}^{k} |w_{au_k}|} \tag{4.8}$$

When using certain similarity metrics including Pearson correlation and vector similarity, Resnick et al. [49] and Breese et al. [10] advocate a slight modification of the prediction rules given above. Since Pearson correlation is computed using centered ratings (ratings with the user mean subtracted), Resnick et al. compute centered ratings in the GroupLens algorithm, and then add the mean rating of the active user back in. Breese et al. do the same for vector similarity. Equation 4.9 shows the exact from of this prediction method, algorithm 4.1 gives the complete prediction algorithm.

$$\hat{r}_y^a = \bar{r^a} + \frac{\sum_{k=1}^{K} w_{au_k}^{P} (r_y^{u_k} - \bar{r}^{u_k})}{\sum_{k=1}^{K} |w_{au_k}^{P}|} \tag{4.9}$$

While the early work by Resnick et al. used all users to compute predictions [49], Shardanand and Maes include only those users whose correlation with the active user exceeds a given threshold [52]. Gokhale and Claypool explored the use of correlation thresholds, as well as thresholds on the actual number of rated items common to the

Input: $\mathbf{r}^a$, $\mathbf{r}$, K
Output: $\hat{\mathbf{r}}^a$

for $(u = 1$ to $N)$ **do**
$w_{au} \leftarrow \frac{\sum_{\{y|r_y^a, r_y^u \neq \perp\}} (r_y^a - \bar{r}^a)(r_y^u - \bar{r}^u)}{\sqrt{\sum_{\{y|r_y^a, r_y^u \neq \perp\}} (r_y^a - \bar{r}^a)^2 \sum_{\{y|r_y^a, r_y^u \neq \perp\}} (r_y^u - \bar{r}^u)^2}}$
end for
Sort w_{au}
for $k = 1$ to K **do**
$u_k \leftarrow k^{th}$ closest neighbor to a
end for
for $y = 1$ to M **do**
$\hat{r}_y^a \leftarrow \bar{r^a} + \frac{\sum_{k=1}^{K} w_{au_k} (r_y^{u_k} - \bar{r}^{u_k})}{\sum_{k=1}^{K} |w_{au_k}|}$
end for

Algorithm 4.1: PKNN-Predict

active user and each user from the data set. The later were termed *history thresholds* [22]. Herlocker et al. perform experiments using similar thresholds, as well as a *Best-K neighbors* method that is most similar to standard KNN classification. The general result of this work is that using a subset of all neighbors computed using a threshold or other technique tends to result in higher prediction accuracy than when no restrictions are placed on neighborhood size. However, a precision/recall tradeoff exists when using thresholds due to the sparsity of data. Essentially, the number of ratings that can be predicted for any user decreases as threshold values are increased. A true KNN approach does not suffer from this problem, but incurs an added computational cost.

4.1.2 Complexity

As an instance based learning method, the training for any neighborhood-based rating prediction algorithm consists of simply storing the profiles of all training users. These profiles must be kept in memory at prediction time, which is a major drawback of these methods. However, if sparse matrix storage techniques are used, the storage space needed depends only on the total number of observed ratings. Computing all rating predictions

for the active user a with a neighborhood method requires $O(N)$ similarity weight calculations each taking at most $O(M)$ time for a total of $O(NM)$. If a method is used to select neighbors with weights above a given threshold, then an additional $O(N)$ time is needed. If a method is used to select the K nearest neighbors, then an additional time complexity of $O(N \log N)$ is needed. Finally, computing all rating predictions for the active user takes $O(NM)$ in general or $O(KM)$ if only the K nearest neighbors are used.

In all of these variations the contribution of the weight calculation is $O(NM)$, and prediction time scales linearly with the number of users in the database. With a realistic number of users this becomes quite prohibitive. Note that certain definitions of prediction allow for the similarity weights to be computed as a preprocessing step; however, we assume that the active user may be a novel user so that its profile is not known before prediction time.

4.1.3 Results

A weighted nearest neighbor rating prediction algorithm using the Pearson correlation similarity metric was implemented. We elected to use the active user's K nearest neighbors to compute predictions. We call this method *PKNN-Predict*, and list the pseudo code in algorithm 4.1. PKNN-Predict has total time complexity $O(NM+N \log N+KM)$. Note that if we let $K = N$ we recover the original GroupLens method.

We tested the predictive accuracy of PKNN-Predict using three neighborhood sizes $K = \{1, 10, 50\}$. Both weak generalization and strong generalization experiments were performed using the EachMovie and MovieLens data sets. The mean error values are reported in terms of NMAE, along with the corresponding standard error values.

From these results we see that across all data sets and experimental protocols, the lowest mean error rates are obtained using one nearest neighbor to predict rating values. However, there is little variation in the results for different settings of K in the range tested.

Table 4.1: PKNN-Predict: EachMovie Results

Data Set	$K = 1$	$K = 10$	$K = 50$
Weak	0.4886 ± 0.0014	0.4890 ± 0.0014	0.4898 ± 0.0014
Strong	0.4933 ± 0.0006	0.4936 ± 0.0006	0.4943 ± 0.0006

Table 4.2: PKNN-Predict: MovieLens Results

Data Set	$K = 1$	$K = 10$	$K = 50$
Weak	0.4539 ± 0.0030	0.4549 ± 0.0030	0.4569 ± 0.0031
Strong	0.4621 ± 0.0022	0.4630 ± 0.0023	0.4646 ± 0.0023

4.2 Naive Bayes Classifier

The Naive Bayes classifier is robust with respect to missing feature values, which may make it well suited to the task of rating prediction. The naive Bayes classifier can be compactly represented as a Bayesian network as shown in figure 4.1. The nodes represent random variables corresponding to the class label C, and the components of the input vector $X_1, ..., X_M$. The Bayesian network in figure 4.1 reveals the primary modeling assumption present in the naive Bayes classifier: the input attributes X_j are independent given the value of the class label C. This is referred to as the *naive Bayes assumption* from which the name of the classifier is derived.

Training a naive Bayes classifier requires learning values for $P(C = c)$, the prior probability that the class label C takes value c; and $P(X_j = x|C = c)$, the probability that input feature X_j takes value x given the value of the class label is $C = c$. These

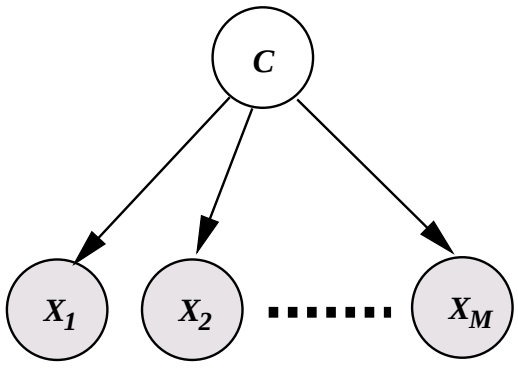

Figure 4.1: Naive Bayes Classifier

probabilities can be estimated using frequencies computed from the training data as seen in equations 4.10 and 4.11. Given a new input pattern $\mathbf{x}_q$, we classify it according to the rule shown in equation 4.12.

$$P(C = c) = \frac{1}{N} \sum_{i=1}^{N} \delta(c_i, c) \tag{4.10}$$

$$P(X_j = x | C = c) = \frac{\sum_{i=1}^{N} \delta(x_{ij}, x)\delta(c_i, c)}{\sum_x \sum_{i=1}^{N} \delta(x_{ij}, x)\delta(c_i, c)} \tag{4.11}$$

$$c_q = \arg\max_c P(C = c) \prod_j P(X_j = x_{qj} | C = c) \tag{4.12}$$

When applying a classifier to domains with attributes of unknown quality, feature selection is often used to pick a subset of the given features to use for classification. In a filter approach to feature selection, a set of features is selected as a preprocessing step, ignoring the effect of the selected features on classifier accuracy [33]. In a wrapper approach to feature selection, classification accuracy is used to guide a search through the space of feature subsets [33].

One feature selection filter often used with the naive Bayes classifier is based on the empirical mutual information between the class variable and each attribute variable. The empirical mutual information score is computed for each attribute, and the attributes are sorted with respect to their scores. The K attributes with the highest score are retained as features. In the present case where all variables are discrete, the empirical mutual information can be easily computed based on the distributions found when learning the classifier. The formula is given in equation 4.13. The mutual information can also be computed during the learning process learning.

$$MI(X_j, C) = \sum_x \sum_c P(X_j = x, C = c) \log \frac{P(X_j = x, C = c)}{P(X_j = x)P(C = c)} \tag{4.13}$$

One issue with the use of mutual information as a feature selection filter is that it may select redundant features. For example, if a model contained multiple copies of the same feature variable, and that feature variable had maximal mutual information with

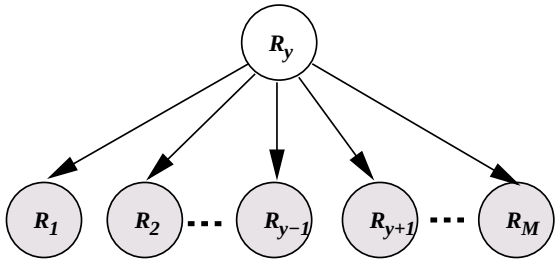

Figure 4.2: Naive Bayes classifier for rating prediction.

the class variable, the mutual information feature selection filter would select as many redundant copies of that feature variable as possible. When selecting a small number of features, this can be very problematic.

4.2.1 Naive Bayes Rating Prediction

To apply the naive Bayes classifier to rating prediction we independently learn one classifier for each item y. We train the classifier for item y using all users u in the data set who have supplied a rating for item y. The input vectors used to construct the classifier for item y consist of ratings for all items other than item y. We will refer to item y as the class item, and the remaining items as feature items. We can express this naive Bayes classifier for item y in terms of a Bayesian network as seen in figure 4.2.

To learn the naive Bayes rating predictor we must estimate $P(R_y = v)$ and $P(R_j = w|R_y = v)$. The naive Bayes learning rules given in equations 4.10 and 4.11. can be applied without modification, but we smooth the probabilities by adding prior counts to avoid zero probabilities. Training rules that include smoothing are shown in equations 4.14, and 4.15. The complete learning procedure is given in algorithm 4.2 where θ_{yv} encodes $P(R_y = v)$, and β_{yvjw} encodes $P(R_j = w|R_y = v)$.

$$P(R_y = v) = \frac{1}{N+V}(1 + \sum_{u=1}^{N} \delta(r_y^u, v)) \tag{4.14}$$

$$P(R_j = w|R_y = v) = \frac{1 + \sum_{u=1}^{N} \delta(r_j^u, w)\delta(r_y^u, v)}{V + \sum_{w=1}^{V}\sum_{u=1}^{N} \delta(r_j^u, w)\delta(r_y^u, v)} \tag{4.15}$$

Input: $\mathbf{r}^a$, $\mathbf{r}$
Output: θ, β

for $y = 1$ to M, $v = 1$ to V **do**
  $\theta_{yv} \leftarrow \frac{1}{N+V}(1 + \sum_{u=1}^{N} \delta(r_y^u, v))$
  for $j = 1$ to M, $w = 1$ to V **do**
    $\beta_{yvjw} \leftarrow \frac{1+\sum_{u=1}^{N} \delta(r_j^u,w)\delta(r_y^u,v)}{V+\sum_{w=1}^{V}\sum_{u=1}^{N} \delta(r_j^u,w)\delta(r_y^u,v)}$
  end for
end for

Algorithm 4.2: NBClass-Learn

Input: $\mathbf{r}^a$, θ, β
Output: $\hat{\mathbf{r}}^a$

for $y = 1$ to M **do**
  $\hat{r}_y^a \leftarrow \arg\max_v \theta_{yv} \prod_{j \neq y} \prod_{w=1}^{V} \beta_{yvjw}{}^{\delta(r_y^a,w)}$
end for

Algorithm 4.3: NBClass-Predict

To predict the value of r_y^a given the profile $\mathbf{r}^a$ of a particular active user a we apply a slightly modified prediction rule to allow for missing values. This prediction rule is shown in equation 4.16. A complete prediction method is given in algorithm 4.3.

$$\hat{r}_y^a = \arg\max_v P(R_y = v) \prod_{j \neq y} \prod_{w=1}^{V} P(R_j = w | R_y = v)^{\delta(r_j^a, w)} \tag{4.16}$$

Applying a feature selection technique as described in section 4.2 may be useful for several reasons. First, it reduces the number of parameters that need to be stored from $O(M^2V^2)$ to $O(KMV^2)$. Second, the elimination of irrelevant attributes should decrease prediction error. Feature selection by empirical mutual information is an obvious candidate since the probabilities needed to compute the score are found when estimating the parameters of the classifier. However, the empirical mutual information scores computed for different feature items will be based on different numbers of observed ratings due to rating sparsity. Clearly we should have more confidence in a mutual information estimate computed using more observed rating values than one computed using fewer observed ratings. A simple heuristic score can be obtained by scaling the empirical mutual infor-

mation value for a feature item by the number of samples used to compute it. Zaffalon and Hutter present a principled, Bayesian approach to dealing with this problem based on estimating the distribution of mutual information [56].

4.2.2 Complexity

The computational cost of separately learning one Naive Bayes classifier for each item is $O(NM^2V^2)$. Storing the probabilities for a singe classifier takes $MV^2 + V$ space and thus $M^2V^2 + MV$ for all M classifiers. This space requirement begins to be prohibitive; however, it can be lowered by applying feature selection independently for each class item. For example, the empirical mutual information with the heuristic correction discussed previously can be computed between items at a computational cost of $O(NM^2V^2 + M^2 \log M)$. If the best K feature items are retained as input features, this lowers the storage requirement to $O(KMV^2)$. The computational complexity of predicting all unknown ratings for a singe user is $O(M^2V)$. If we restrict the input vectors to the best K features for each item, we obtain $O(KMV)$.

4.2.3 Results

Learning and prediction methods for the naive Bayes classifier were implemented according to algorithms 4.2, and 4.3. In addition, the heuristic mutual information score discussed in subsection 4.2.1 was applied to select the K best features after training the classifier for each item. Despite the issues we have outlined with the use of mutual information for feature selection, it was found to result in improved accuracy in preliminary testing.

Both weak generalization and strong generalization experiments were performed using the EachMovie and MovieLens data sets. The method was tested for $K = \{1, 20, 50, 100\}$. The mean error values are reported in terms of NMAE, along with the corresponding standard error values.

Table 4.3: NBClass-Predict: EachMovie Results

Data Set	$K = 1$	$K = 20$	$K = 50$	$K = 100$
Weak	0.5789 ± 0.0007	0.5258 ± 0.0022	0.5270 ± 0.0019	0.5271 ± 0.0011
Strong	0.5820 ± 0.0043	0.5319 ± 0.0057	0.5317 ± 0.0042	0.5295 ± 0.0047

Table 4.4: NBClass-Predict: MovieLens Results

Data Set	$K = 1$	$K = 10$	$K = 50$	$K = 100$
Weak	0.4803 ± 0.0027	0.4966 ± 0.0021	0.5042 ± 0.0026	0.5086 ± 0.0014
Strong	0.4844 ± 0.0016	0.4833 ± 0.0052	0.4942 ± 0.0065	0.4940 ± 0.0111

4.3 Other Classification and Regression Techniques

While the application of other standard types of classification and regression techniques including decision tree classifiers and, artificial neural networks is possible, the presence of missing values in the input is more problematic. In the case of decision tress, missing attribute values can be dealt with by propagating fractional instances during learning [42, p. 75]. This is the method used in the popular decision tree learning algorithm C4.5 [48]. In the case of neural networks missing values must be explicitly represented. Two main possibilities exist. First, 'missing' can simply be considered as an additional rating value. There are cases in statistical analysis of categorical data where this type of treatment of missing data is sensible; however, in the rating data case it is not justified. Second, the 1-of-V encoding scheme proposed by Billsus and Pazzani [5] can be applied. We have experimented briefly with some of these techniques, but the results were fairly poor. Neighborhood methods appear to achieve the best prediction accuracy in the presence of extremely sparse rating profiles of any of the classification or regression based prediction methods.

Chapter 5

Clustering

Given a set of M dimensional input vectors $\{\mathbf{x}_i\}$, the goal of clustering is to group similar input vectors together. A number of clustering algorithms are well known in machine learning, and they fall into two broad classes: hierarchical clustering, and standard clustering [3]. In hierarchical clustering a tree of clusters is constructed, and methods differ depending on whether the tree is constructed bottom-up or top-down. Standard clustering includes K-means, K-medians, and related algorithms. A key point in all clustering methods is deciding on a particular distance metric to apply. For ordinal data possibilities include, hamming distance, absolute distance, and squared distance.

Clustering has been applied to collaborative filtering in two basic ways. First, the items can be clustered to reduce the dimension of the item space and help alleviate rating sparsity. Second, users can be clustered to identify groups of users with similar or correlated ratings. Item clustering does not directly lead to rating prediction methods. It is a form of preprocessing step, which requires the subsequent application of a rating prediction method. O'Connor and Herlocker have studied item clustering as a preprocessing step for neighborhood based rating prediction [45]. They apply several clustering methods, but their empirical results show prediction accuracy actually *decreases* compared to the unclustered base case regardless of the clustering method used. A reduction

in computational complexity is achieved, however.

Unlike item clustering, user clustering methods can be used as the basis of simple rating prediction methods. Rating prediction based on user clustering is the focus of this chapter. We review clustering algorithms from both the standard, and hierarchical classes. We introduce a novel K-medians like rating prediction method with good prediction accuracy and low prediction complexity. We also discuss existing rating prediction methods for hierarchical clustering.

5.1 Standard Clustering

Standard clustering methods perform an iterative optimization procedure that shifts input vectors between K clusters in order to maximize an objective function. A representative vector for each cluster called a *cluster prototype* is maintained at each step. The objective function is usually the sum or mean of the distance from each input vector $\mathbf{x}_i$ to its cluster prototype [3, p.13]. The role of the underlying distance metric is crucial. It defines the exact form of the objective function, as well as the form of the cluster prototypes.

When squared distance is used, the objective function is the sum over input vectors of the squared distance between each input vector and the prototype vector of the cluster it is assigned to. For a particular assignment of input vectors to clusters, the optimal prototype for a given cluster is the mean of the input vectors assigned to that cluster. When absolute distance is used, the objective function is the sum of the absolute difference between the input vectors assigned to each cluster and the corresponding cluster prototype. For a particular assignment of input vectors to clusters, the optimal prototype for a given cluster is the median of the input vectors assigned to that cluster. In theory any distance function can be used, but some distance functions may not admit an analytical form for the optimal prototype.

To obtain a clustering of the input vectors which corresponds to a local minimum of the objective function, an iterative optimization procedure is required. We begin by initializing the K prototype vectors. On each step of the iteration we compute the distance from each input vector to each cluster prototype. We then assign each input vector to the cluster with the closest prototype. Lastly we update the cluster prototypes based on the input vectors assigned to each prototype. The general form of this algorithm is given below. When squared distance is used this algorithm is known as K-Means, and when absolute distance is used it is known as K-Medians.

1. $c_i^{t+1} = \arg\min_k d(\mathbf{x}_i, \mathbf{p}_k^t)$
2. $p_k^{t+1} = \arg\min_p \sum_{i=1}^N \delta(k, c_i^{t+1}) d(\mathbf{x}_i, \mathbf{p}_k^t)$

5.1.1 Rating Prediction

In user clustering the input vectors $\mathbf{x}_i$ correspond to the rows $\mathbf{r}^u$ of the user-item rating matrix. A simple rating prediction scheme based on user clustering can be obtained by defining a distance function that takes missing values into account. We choose to minimize total absolute distance since this corresponds to our choice of NMAE error measure. We modify the standard absolute distance function by taking the sum over components that are observed in both vectors as shown in equation 5.1. The objective function we minimize is thus given by equation 5.2 where c_u is the cluster user u is assigned to.

$$d(\mathbf{r}^u, \mathbf{p}^k) = \sum_{\{y | r_y^u, p_y^k \neq \perp\}} |r_y^u - p_y^k| \tag{5.1}$$

$$F[\mathbf{r}, c, \mathbf{p}] = \sum_{u=1}^{N} d(\mathbf{r}^u, \mathbf{p}^{c_u}) \tag{5.2}$$

The minimizer of this distance function occurs when all the values of p_y^k are set to $\perp$. However, if we also stipulate that the maximum number of components be defined, then the optimal prototype for cluster C_k is the median of the rating vectors assigned to C_k,

Input: $\{\mathbf{r}^u\}$, K
Output: $\{\mathbf{p}^l\}$

Initialize $\mathbf{p}^k$
while ($F[\mathbf{r}, c, \mathbf{p}]$ Not Converged) **do**
  for $u = 1$ to N **do**
    $c_u \leftarrow \arg\min_k \sum_{\{y|r_y^u, p_y^k \neq \perp\}} |r_y^u - p_y^k|$
  end for
  for $k = 1$ to K, $y = 1$ to M **do**
    $p_y^k \leftarrow \text{median}\{r_y^u | c_u = k, r_y^u \neq \perp\}$
  end for
end while

Algorithm 5.1: KMedians-Learn

Input: $\mathbf{r}^a$, $\{\mathbf{p}^l\}$, K
Output: $\hat{\mathbf{r}}^a$

$k \leftarrow \arg\min_l \sum_{\{y|r_y^a, p_y^l \neq \perp\}} |r_y^a - p_y^l|$
$\hat{r}_y^a \leftarrow \mathbf{p}^k$

Algorithm 5.2: KMedians-Predict

taking missing ratings into account. Specifically, the prototype value for item y is the median of the defined ratings for item y. It is only set to $\perp$ if no users assigned to cluster k have rated item y. In our experiments undefined components are not a problem due to the small number of clusters used compared to the large number of users.

Once the user clusters have been formed, we obtain a very simple algorithm for predicting all ratings for the active user a. We simply determine which cluster k user a belongs to and set $\hat{\mathbf{r}}^a$ to $\mathbf{p}^k$. We give learning and prediction procedures in algorithms 5.1 and 5.2.

5.1.2 Complexity

The complexity of learning the K-Medians cluster prototypes depends on the number of iterations needed to reach convergence. Assuming it takes I iterations to reach convergence, the total time complexity of the learning algorithm is $O(INMK)$. The space

Table 5.1: K-Medians Clustering: EachMovie Results

Data Set	$K = 5$	$K = 10$	$K = 20$	$K = 40$
Weak	0.4810 ± 0.0023	0.4676 ± 0.0016	0.4631 ± 0.0015	0.4668 ± 0.0013
Strong	0.4868 ± 0.0007	0.4725 ± 0.0021	0.4688 ± 0.0012	0.4694 ± 0.0035

Table 5.2: K-Medians Clustering: MovieLens Results

Data Set	$K = 5$	$K = 10$	$K = 20$	$K = 40$
Weak	0.4495 ± 0.0027	0.4596 ± 0.0052	0.4573 ± 0.0067	0.4677 ± 0.0058
Strong	0.4637 ± 0.0056	0.4585 ± 0.0056	0.4556 ± 0.0053	0.4612 ± 0.0091

complexity for the learned cluster prototype parameters is MK. Given a novel user profile, the time needed to compute predictions for all items is $O(MK)$.

5.1.3 Results

The KMedians-Learn method was run on the EachMovie and MovieLens weak data sets with $K = \{5, 10, 20, 40\}$. The cluster prototypes were initialized to K different randomly chosen user profile vectors. Preliminary testing indicated that good prediction accuracy was obtained after less than 25 iterations. This value was used as a hard limit on the number of iterations allowed in the learning implementation. After learning was complete the KMedians-Predict method was run on both weak and strong generalization data sets, and the mean prediction error rates were calculated. The NMAE values along with the standard error level are shown in tables 5.1, and 5.2.

5.2 Hierarchical Clustering

A hierarchical clustering method constructs a tree of clusters from the input vectors $\{\mathbf{x}_i\}$. The main property of a cluster tree or *dendrogram* is that the children of each cluster node C_l form a partition of the input vectors contained in C_l [3]. There are two ways of constructing a cluster tree: agglomeratively and divisively.

In agglomerative clustering each input vector is initially placed in its own cluster. On each subsequent step the two most similar clusters are identified and merged to obtain

their common parrent. The merging continues until only one node remains. This node forms the root of the tree and contains all the input vectors.

The central issue in agglomerative clustering is deciding which pair of clusters to merge next. A pair of clusters is selected by computing a linkage metric between all pairs of clusters, and choosing the pair of clusters that is closest with respect to the metric. Common linkage metrics include single linkage, complete linksage, and average linkage [18]. The linkage metric depends on a distance function between input vectors $d(\mathbf{x_a}, \mathbf{x_b})$.

$$\begin{aligned} \text{Single Linkage} \quad & l_s(C_k, C_l) = \min_{\mathbf{x_r} \in C_k, \mathbf{x_t} \in C_l} d(\mathbf{x_r}, \mathbf{x_t}) \\ \text{Complete Linkage} \quad & l_c(C_k, C_l) = \max_{\mathbf{x_r} \in C_k, \mathbf{x_t} \in C_l} d(\mathbf{x_r}, \mathbf{x_t}) \\ \text{Average Linkage} \quad & l_a(C_k, C_l) = \operatorname*{mean}_{\mathbf{x_r} \in C_k, \mathbf{x_t} \in C_l} d(\mathbf{x_r}, \mathbf{x_t}) \end{aligned}$$

A second method for cluster tree construction is to begin with all input vectors in the root node, and to recursively split the most appropriate node until a termination condition is reached. The construction method may be terminated when each leaf node contains less than a maximum number of input vectors, when a maximum number of clusters is reached, or when each cluster satisfies a condition on within-cluster similarity.

In the case of collaborative filtering we typically want a small set of clusters with respect to the number of items so divisive clustering a better choice in terms of total computational complexity. In divisive clustering the main issues are which cluster to select for splitting, and how to split the input vectors within a cluster. Both issues again require the definition of a distance measure $d(\mathbf{x}_a, \mathbf{x}_b)$ between input vectors.

Clusters are selected for splitting based on any number of heuristics including size, within-cluster similarity, and cluster cohesion [18]. A standard technique for splitting a cluster C_l is to randomly select an input vector $\mathbf{x}_r$ from the elements of C_l, and to determine the element $\mathbf{x}_t$ of C_l that is furthest from $\mathbf{x}_r$. These two input vectors are placed in their own clusters, and the remaining input vectors are assigned depending on which of $\mathbf{x}_r$ or $\mathbf{x}_t$ they are closer to.

5.2.1 Rating Prediction

Seng and Wang present a user clustering algorithm based on divisive hierarchical clustering called the Recommendation Tree algorithm (RecTree) [14]. In the RecTree algorithm a cluster node is expanded if it is at a depth less than a specified maximum, and its size is greater than a specified maximum. The exact sequence in which nodes are expanded is not critical, and a simple depth-first or breadth-first expansion of the nodes can be used until one of the termination conditions is met.

Interestingly, Seng and Wang do not use a common method for splitting the input vectors assigned to a cluster in the RecTree algorithm. Instead, they apply a K-means algorithm with $K = 2$ to each cluster. If the total number of leaf clusters is K', then the RecTree algorithm computes the clustering more efficiently than applying K-means to the data with K' clusters. However, the quality of the clustering produced by the RecTree algorithm will likely be lower than that obtained using K-means directly.

Ratings could be predicted using the computed prototypes similar to the method proposed above for K-medians. However, Seng and Wang choose to apply neighborhood-based rating prediction within each user cluster. This means the RecTree algorithm as given by Seng and Wang is more of a pre-processing method than a true rating prediction method. Experimental results presented by Seng and Wang show that the RecTree algorithm performs slightly better than neighborhood-based rating prediction methods on certain tasks [14].

Chapter 6

Dimensionality Reduction

Dimensionality reduction is a technique analogous to clustering. Instead of assuming that a single, discrete latent variable is the underlying cause for the observed data, we assume there are a small number of continuous latent variables. In general, the process of dimensionality reduction can be described as mapping a high dimensional input space into a lower dimensional latent space. A special case of dimensionality reduction is matrix factorization where a data matrix D is reduced to the product of several low rank matrices.

In this chapter we introduce three standard techniques used for dimensionality reduction: singular value decomposition (SVD), factor analysis (FA), and principal components analysis (PCA). We discuss how each technique can be applied or adapted to rating prediction. We provide rating prediction results for weighted singular value decomposition. We also introduce a new rating prediction technique which extends weighted singular value decomposition to novel user profiles.

6.1 Singular Value Decomposition

Singular value decomposition is a technique for matrix factorization. Given a data matrix D of size $N \times M$, the singular value decomposition of D is a factorization $D = U\Sigma V^T$

where U is of size $N \times M$, Σ is of size $M \times M$, and V is of size $M \times M$. In addition, U and V are orthonormal, and Σ is diagonal. The standard solution to the singular value decomposition is to let $\Sigma = \text{diag}(\sigma_1, \sigma_2, ..., \sigma_M)$ where σ_i is the i^{th} largest eigenvalue of DD^T, the columns of U are defined to be the eigenvectors of DD^T, and the columns of V are defined to be the eigenvectors of D^TD. The columns of U and V are ordered according to the sizes of their corresponding eigenvalues. Given this solution to the SVD, it is well known that $U_K\Sigma_K V_K^T$ is the best rank-K approximation to D under the Frobenius norm. Σ_K is obtained by preserving the first K diagonal elements of Σ and truncating the rest. U_K and V_K are obtained by preserving the first K columns of U and V and truncating the rest. The Frobenius norm is simply the sum of squares of all elements of a matrix, which in this case given by equation 6.1.

$$\mathcal{F}(D - U_K\Sigma_K V_K^T) = \sum_{n=1}^{N}\sum_{m=1}^{M}(D_{nm} - (U_K\Sigma_K V_K^T)_{nm})^2 \tag{6.1}$$

The components of the low rank approximation $U_K\Sigma_K V_K^T$ also have interpretations in terms of a latent space mapping. The columns of U_K can be interpreted as specifying a set of K basis vectors for an N dimensional space, while the columns of V_K specify a set of K basis vectors in an M dimensional space. Σ_K is responsible for scaling the dimensions of the latent space and can be multiplied into U_K and V_k.

6.1.1 Weighted Low Rank Approximations

As we have mentioned, $U_K\Sigma_K V_K^T$ is the best rank K approximation to D under the Frobenius norm $\mathcal{F}$. As Srebro and Jaakola argue, in certain situations it is natural to consider a weighted Frobenius norm [53]. For instance, if estimates of the noise variance associated with each measurement are available, and those estimates differ across measurements. Another case of special interest here is when some entries in the matrix are not observed. In this case unobserved elements of the matrix can be given a weight of 0, while observed elements are given a weight of 1. It is important to note that the

SVD of a matrix is undefined if the matrix contains missing entries.

In general the optimal rank K approximation to the data matrix D minimizes the difference between D and UV^T under the weighted Frobenius norm $\mathcal{F}_W$ as shown in equation 6.2. Here the only restriction on U and V are that U is of size $N \times K$, and that V is of size $M \times K$. W is an $N \times M$ matrix which specifies a non-negative weight value for each element of D.

$$\mathcal{F}_W(D - UV^T) = \sum_{n=1}^{N} \sum_{m=1}^{M} W_{nm}(D_{nm} - (UV^T)_{nm})^2 \tag{6.2}$$

Srebro and Jaakola show that the problem of computing an optimal rank K weighted approximation does not have an analytical solution, despite the fact that a minimizer can be found exactly under the unweighted Frobenius norm using standard SVD [53]. Srebro and Jaakola also show the existence of local minima in the weighted case. In the unweighted case every local minima is also a global minima. The obvious method for finding a locally optimal solution is thus to directly minimize $\mathcal{F}_W(D - UV^T)$ with respect to U and V using numerical optimization. Srebro and Jaakola present a numerical optimization method that performs coordinate descent in U and V. The minimum of $\mathcal{F}_W(D - UV^T)$ with respect to U can be found exactly for any given V. However, when the exact form of the minimizer is substituted into the gradient equation for V, the resulting system does not have an analytic solution. Thus, standard gradient descent must be used to find the minimum of $\mathcal{F}_{\mathcal{W}}(D - UV^T)$ with respect to V for a given U. We give the coordinate descent iteration equations below. The notation U_i, W_i, D_i indicates the i^{th} row of the corresponding matrix. The notation $\odot$ denotes elementwise multiplication. diag(v) is the diagonal matrix with the elements of the vector v along its diagonal.

$$U_i^* = (V^T \text{diag}(W_i) V)^{-1} V^T \text{diag}(W_i) D_i \tag{6.3}$$

$$\frac{\partial \mathcal{F}_W(D - U^* V^T)}{\partial V} = 2(W \odot (V U^{*T} - D^T)) U^* \tag{6.4}$$

In addition to the straightforward optimization approach for the general weighted low rank approximation problem, Srebro and Jaakola develop a simple EM algorithm for the missing data problem. This algorithm is based on iteratively applying standard SVD to obtain successively better low rank approximations $\hat{D} = U_K \Sigma_K V_K^T$. In this case if an element of D is unobserved it is assigned a weight of 0, and if it is observed it is assigned a weight of 1. In the expectation step of the EM algorithm, the missing values of D are filled in with values from the low rank reconstruction $\hat{D}$ forming a complete matrix X. In the maximization step a low rank approximation of X is computed and new values for U, Σ, and V are found. The low rank approximation needed in the M-step can be found using standard SVD since X is a completely specified matrix. The steps of this procedure are given in detail below.

$$\text{E-Step} \qquad X = W \odot D + (1 - W) \odot \hat{D} \tag{6.5}$$

$$\text{M-Step} \qquad [U, \Sigma, V] = \text{SVD}(X) \tag{6.6}$$

$$\hat{D} = U_K \Sigma_K V_K^T \tag{6.7}$$

In fact, this EM algorithm holds for any weight matrix W so long as W_{ij} lies in the interval $[0, 1]$ for all i and j [53]. Srebro and Jaakola note that both the number of iterations needed to achieve convergence, and the quality of the solution of the EM procedure depend strongly on the amount of missing data in the zero-one weight case [53]. They suggest an alternate procedure where the value of K is initialized significantly above the desired value, and on each iteration of the EM procedure the value of K is reduced until the desired value is reached. Once the desired value of K is reached, the EM procedure is run to convergence with K held constant.

6.1.2 Learning with Weighted SVD

While the presence of missing data prohibits the use of standard SVD for rating prediction, it does not pose a problem for either of the weighted low rank approximation

Input: R, W, L, K
Output: Σ, V

$\hat{R} \leftarrow 0$
while ($\mathcal{F}_W(R - \hat{R})$ Not Converged) **do**
  $X \leftarrow W \odot R + (1 - W) \odot \hat{R}$
  $[U, \Sigma, V] = \text{SVD}(X)$
  $U \leftarrow U_L$, $\Sigma \leftarrow \Sigma_L$, $V \leftarrow V_L$
  $\hat{R} \leftarrow U\Sigma V^T$
  if $(L > K)$ **then**
    Reduce L
  end if
end while

Algorithm 6.1: wSVD-Learn

schemes proposed by Srebro and Jaakola. The EM procedure is particularly attractive since many numerical computing packages include robust routines for computing the SVD of a complete matrix. Letting R represent the matrix of user ratings where each row of the matrix represents a different user profile, the EM procedure of Srebro and Jaakola can be applied without modification. In the presence of large amounts of missing data the quality of the solution can be greatly improved by slowly lowering the rank of the approximation from L to the desired K, as Srebro and Jaakola suggest. We give the complete learning procedure in algorithm 6.1.

6.1.3 Rating Prediction with Weighted SVD

Predicting ratings for profiles used to compute the low rank approximation is trivial given the final approximation matrix $\hat{R}$. The rating value predicted for user u and item y is simply $\hat{r}_y^u = \hat{R}_{uy}$. However, Srebro and Jaakola do not propose a method for prediction with novel user profiles [53]. In the complete data case the rank K latent space description $\mathbf{l}$ of a user profile $\mathbf{r}$ can easily be found by observing that $\mathbf{r} = \mathbf{l}\Sigma V^T$ and thus $\mathbf{l} = \mathbf{r}V\Sigma^{-1}$. We propose a simple iterative prediction method for novel user profiles based on this relationship. We give the prediction scheme in algorithm 6.2.

Input: $\mathbf{r}^a$, w^a, Σ, V, K
Output: $\hat{\mathbf{r}}^a$

$\hat{\mathbf{r}}^a \leftarrow 0$
while ($\mathcal{F}_{w^a}(r^a - \hat{\mathbf{r}}^a)$ Not Converged) **do**
  $x \leftarrow w^a \odot \mathbf{r}^a + (1 - w^a) \odot \hat{\mathbf{r}}^a$
  $\mathbf{l}^a \leftarrow xV\Sigma^{-1}$
  $\hat{\mathbf{r}}^a \leftarrow \mathbf{l}^a \Sigma V^T$
end while

Algorithm 6.2: wSVD-Predict

Recalling that Σ is a diagonal matrix with a trivial inverse, it is clear that each step of this iteration scheme can be quickly computed. In fact, a matrix of novel user profiles could be substituted for the single profile vector $\mathbf{r}$, and a set of predictions could be found simultaneously for all profiles in the matrix.

6.1.4 Complexity

The complexity of computing the singular value decomposition of a complete $N \times M$ matrix is $O(NM^2 + M^3)$ using Golub and van Loan's R-SVD algorithm [25, p. 254]. A slight gain can be made by employing the economy sized singular value decomposition, which only computes the first M singular values and vectors if $M < N$. The complexity of the weighted singular value decomposition EM algorithm is clearly dominated by the cost of performing the SVD at each step. The total computational complexity of the learning algorithm is given by $O(INM^2 + IM^3)$ where I is the number of iterations performed. In practice this means the weighted singular value decomposition learning method is extremely slow.

Predicting all ratings given a novel user profile using the proposed iterative scheme has computational complexity $O(IKM)$. Approximately one hundred iterations leads to acceptable convergence on the data sets we have studied. The only parameters we need to store are the K singular values and the $M \times K$ latent item space matrix V for a total of $MK + K$.

Table 6.1: wSVD-Predict: EachMovie Results

	$K = 5$	$K = 10$	$K = 20$	$K = 30$
Weak	0.5083 ± 0.0027	0.4725 ± 0.0023	0.4562 ± 0.0032	0.4618 ± 0.0034
Strong	0.5012 ± 0.0030	0.4752 ± 0.0005	0.4672 ± 0.0012	0.4673 ± 0.0021

Table 6.2: wSVD-Predict: MovieLens Results

	$K = 4$	$K = 6$	$K = 8$	$K = 10$
Weak	0.5450 ± 0.0011	0.5018 ± 0.0025	0.4914 ± 0.0021	0.4886 ± 0.0065
Strong	0.5260 ± 0.0024	0.4862 ± 0.0025	0.4710 ± 0.0042	0.4728 ± 0.0057

6.1.5 Results

The wSVD-Learn and wSVD-Predict methods described in algorithms 6.1 and 6.2 were implemented, and tested for both their strong and weak generalization performance. Due to the computational complexity of computing the SVD in each iteration of the wSVD-Learn algorithm, learning was performed using all 5000 training profiles for the MovieLens data set, but only the 5000 most dense training profiles for the EachMovie data set. The computation time needed to compute the SVD on all 30000 EachMovie training users was simply too prohibitive. EachMovie was tested using latent spaces of size 5, 10, 20,30. MovieLens was tested using latent spaces of size 4, 6, 8, 10. To reduce total computation time a limit of 100 iterations was imposed on all iterations.

6.2 Principal Components Analysis

Principal components analysis is a method for dimensionality reduction that seeks to identify orthogonal axes of variance given a data matrix D whose N rows consist of samples from an M dimensional space. Principal components analysis relies on a theorem of linear algebra which states that for any real symmetric matrix A there exists a unitary matrix Λ such that $\Sigma = \Lambda^T A \Lambda$, and Σ is diagonal. A solution to this problem can be found using the eigenvectors of A. In particular, let the columns of Λ be the eigenvectors of A ordered according to decreasing eigenvalues. Then $\Lambda^T A \Lambda$ must be diagonal with $\Sigma_{ii} = \lambda_i$, the i^{th} largest eigenvalue of A. In the case of principal components analysis, we

let $A = \frac{1}{N-1} D^T D$, the covariance matrix of D which is clearly real and symmetric. The eigenvalues of A then indicate the amount of variance along the direction given by the corresponding eigenvector. A reduction to dimension K is obtained by projecting the data matrix D on the subspace consisting of eigenvectors corresponding to the largest K eigenvalues of A. This projection preserves the maximum amount of variance of any projection to K dimensions.

Many current computational packages include routines for extracting eigenvalues and eigenvectors from a matrix, rendering the computation of principal components quite easy. An alternative method for performing PCA is to exploit the relationship between the diagonalization of $D^T D$ and the singular value decomposition of the centered data matrix D. Approximating D by $U\Sigma V^T$ we have $D^T D = (U\Sigma V^T)^T (U\Sigma V^T) = V\Sigma U^T U\Sigma V^T$ and since the columns of U are orthonormal we get $D^T D = V\Sigma^2 V^T$. Since the columns of V are also orthonormal we get $\Sigma^2 = V^T D^T DV$, which is a PCA solution since Σ^2 is a diagonal matrix.

6.2.1 Rating Prediction with PCA

Like with standard SVD, standard PCA can not be used in cases where the input matrix D contains missing data. Goldberg, Roeder, Gupta and Perkins propose a solution to this problem using a set of items they call the *gauge set* [24]. The gauge set consists of a small number of items that all users must rate completely. The same gauge set is used for all users. This means that PCA can be applied to the portion of the rating matrix consisting of ratings for the gauge set items. Goldberg et al. retain only the two first principal components, although in theory any number could be used.

A direct rating prediction scheme is not possible using this application of PCA. Instead, Goldberg et al. cluster users in the two dimensional latent space using a recursive rectangular clustering method, which is an instance of hierarchical divisive clustering (see sub-section 5.2). Next, they determine a mean rating vector for each cluster based on the

ratings of users in each cluster. When a prediction is needed for a new user, that user's rating profile for the gauge set is projected into the low dimensional latent space, and the user's cluster is determined. Next, predictions for unknown items are drawn from the pre-computed mean rating vector. This is precisely the simple mean rating prediction scheme discussed in sub-section 5.1.1. The combination of PCA applied to a small gauge set, clustering users in the latent space, and predicting cluster mean values results an the method Goldberg et al. refer to as the *EigenTaste* algorithm [24].

From a practical point of view there are several problems with this method. First, the gauge set must consist of the same items for all users, and all users must rate all the gauge set items. Goldberg et al. consider collaborative filtering of jokes in which case it is easy for a user to determine a rating for any item. However, this is not the case in collaborative filtering applications where items may not have concise and informative text based descriptions such as collaborative filtering of movies, music, or books. Another problem that is not addressed by Goldberg et al. is the selection of the items in the gauge set. Clearly this problem is key if we wish to extract the maximum amount of information about a set of users while asking a small number of rating queries.

6.3 Factor Analysis and Probabilistic PCA

Factor analysis is a dimensionality reduction method based on a simple, constrained linear Gaussian model. The standard factor analysis model is specified in equation 6.8 where $\mathbf{x}$ is an observed data vector, $\mathbf{z}$ is a vector in the latent space, Λ an $N \times K$ matrix which maps the K dimensional latent space vectors into the N dimensional data space, μ specifies a mean in the data space common to all data vectors, and ϵ is randomly sampled Gaussian noise unique to each data vector [54]. Λ is often referred to as the *factor loading* matrix. The corresponding graphical model is shown in figure 6.1.

$$\mathbf{x} = \Lambda\mathbf{z} + \mu + \epsilon \tag{6.8}$$

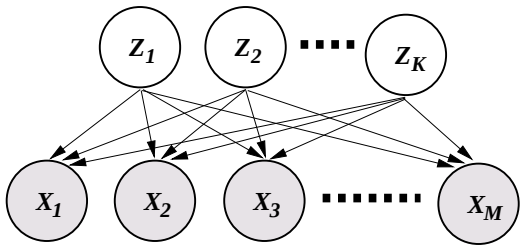

Figure 6.1: Factor Analysis and Probabilistic PCA Graphical Model.

In standard factor analysis $P(\mathbf{Z}) = \mathcal{N}(\mathbf{0}, I)$. The covariance of the Gaussian noise ϵ is assumed to be a diagonal matrix Ψ. Thus the conditional probability of a data vector given the latent space vector and parameters Λ, μ and Ψ is also Gaussian. It is given by $P(\mathbf{x}|\mathbf{z}, \mu, \mathbf{\Psi}) = \mathcal{N}(\mathbf{x}|\mu + \Lambda\mathbf{z}, \Psi)$. In fact, since we are dealing with products of Gaussians, the joint distribution $P(\mathbf{x}, \mathbf{z})$, the marginal distribution $P(\mathbf{x})$, and thus the conditional distribution $P(\mathbf{z}|\mathbf{x})$ are all Gaussian distributions. This factor analysis model is fit using an expectation maximization algorithm. The updates are given next.

$$\text{E-Step} \qquad L^n = (I + \Lambda^T \Psi^{-1} \Lambda)^{-1} \tag{6.9}$$

$$\mathbf{m}^n = L^n \Lambda^T \Psi^{-1} (\mathbf{x}^n - \mu)$$

$$\text{M-Step} \qquad \Lambda = \left(\sum_{n=1}^{N} \mathbf{x}^n \mathbf{m}^{nT} \right) \left(\sum_{n=1}^{N} L^n \right)^{-1} \tag{6.10}$$

$$\Psi = \frac{1}{N} \text{diag} \left(\sum_{n=1}^{N} \mathbf{x}^n \mathbf{x}^{nT} + \Lambda \sum_{n=1}^{N} \mathbf{m}^n \mathbf{x}^{nT} \right)$$

Probabilistic principal components analysis is a dimensionality reduction technique that can be seen both as a generalization of standard principal components analysis, and as a restricted version of factor analysis. Like factor analysis, probabilistic PCA is based on a simple, constrained linear Gaussian model as seen in equation 6.8. The only difference between the two is that in probabilistic PCA the covariance matrix Ψ is restricted to be spherical, not only diagonal. In other words, $\Psi = \sigma^2 I$ for some σ^2. Not surprisingly, the EM algorithm for probabilistic PCA is almost identical to that of factor analysis as seen next.

$$\text{E-Step} \quad L^n = (I + \sigma^{-2}\Lambda^T\Lambda)^{-1} \tag{6.11}$$

$$\mathbf{m}^n = \sigma^{-2}L^n\Lambda^T(\mathbf{x}^n - \mu)$$

$$\text{M-Step} \quad \Lambda = \left(\sum_{n=1}^{N}\mathbf{x}^n\mathbf{m}^{nT}\right)\left(\sum_{n=1}^{N}L^n\right)^{-1} \tag{6.12}$$

$$\sigma^2 = \frac{1}{NM}\text{trace}\left(\sum_{n=1}^{N}\mathbf{x}^n\mathbf{x}^{nT} + \Lambda\sum_{n=1}^{N}\mathbf{m}^n\mathbf{x}^{nT}\right)$$

The spherical covariance restriction decouples the parameters of the linear model leading to analytical solutions for Λ and σ in the complete data case. However, in the missing data case analytical solutions are not possible.

6.3.1 Rating Prediction with Probabilistic PCA

Canny has developed an EM algorithm for probabilistic principal components analysis specifically aimed at rating prediction for collaborative filtering [13]. Canny begins by defining an $M \times M$ trimming matrix T^u for each user u where T^u_{yy} is 1 if user u specified a rating for item y, and 0 other wise. The expectation step of the expectation maximization algorithm for probabilistic PCA remains simple, but the maximization step contains some subtleties related to the fact that the trimming matrix T^u is different for each user u. However, Canny is still able to find updates in closed form [13]. We give the sparse probabilistic PCA algorithm below where $\otimes$ represents the Kronecker tensor product, $\text{st}(X)$ is the vector obtained by vertically stacking the columns of X, and $|r^u|$ is the number of observed items for user u.

$$\text{E-Step} \quad L^u = (I + \sigma^{-2}\Lambda^T T^u\Lambda)^{-1} \tag{6.13}$$

$$\mathbf{m}^u = \sigma^{-2}L^u(T^u\Lambda)^T(\mathbf{r}^u - \mu)$$

$$\text{M-Step} \quad \text{st}(\Lambda) = \left(\sum_{n=1}^{U}\frac{1}{|r^u|}\text{st}(T^u\mathbf{r}^u\mathbf{m}^{nT})\right)\left(\sum_{n=1}^{U}\frac{1}{|r^u|}T^u \otimes (\mathbf{z}^u\mathbf{m}^{nT} + \sigma^2 L^u\right)^{-1} \tag{6.14}$$

$$\sigma^2 = \frac{1}{N}\sum_{n=1}^{U}\frac{1}{|r^u|}(\mathbf{r}^{uT}T^u\mathbf{r}^u - \text{trace}(\Lambda\mathbf{m}^u\mathbf{r}^{uT}T^u))$$

To use this model for rating prediction on novel user profiles $\mathbf{r}^u$ it suffices to compute Λ^u, L^u, and $\mathbf{m}^u$ according to the formulas in the E-Step above, and then to compute $\Lambda\mathbf{m}^n + \mu$. Note that estimating the mean vector μ can be done using either the sample mean of each users ratings, or the individual means for each item taken over all users.

Canny presents experimental results comparing pPCA-based rating prediction with the GroupLens neighborhood method, and several other methods. Several data sets are used including the EachMovie data set. The experimental procedure employed by Canny is similar to the strong generalization all-but-1 experimental procedures used here. Converted to our NMAE error measure, the pPCA method achieved an error rate of 0.5585 when trained with 5000 users. When trained with 50000 users, the error rate dropped to 0.5400.

Chapter 7

Probabilistic Rating Models

In this chapter we discuss a set of methods for collaborative filtering based on unsupervised learning of specialized probabilistic models. Several such models have been studied in the literature to date including a multinomial mixture model [10] [38], Hofmann's aspect model [29], and Marlin's user rating profile model [38]. We begin the discussion by describing a simple multinomial model, which forms the basis of the more complex models we investigate in this chapter. We move on to the mixture of multinomials model, and the aspect model which are both discrete mixture models. The User Rating Profile model is a continuous mixture model and can be seen as an extension of several existing models including the aspect model, and the latent Dirichlet allocation model of Blei, Ng, and Jordan [7]. We introduce the Attitude model family, a completely novel family of product models for categorical data. We will be concerned not only with deriving learning and prediction methods for all of these models, but also with describing the underlying modeling assumptions. We will be particularly interested the generative process underlying each model, and whether the generative process makes intuitive sense in the context of collaborative filtering.

The models we describe differ significantly from factor analysis (FA) and probabilistic principal component analysis (pPCA), which both assume ratings are generated by a

linear process with additive Gaussian noise. In both pPCA and FA the observed rating variables are modeled as real valued random variables. Recall that we have defined ratings to be ordinal valued, so the density under a linear Gaussian model is incorrect. As a result, pPCA and FA models can predict values that do not correspond to ordinal ratings. In fact, they may even predict some ratings that are off the predefined rating scale. However, such models are still useful for prediction if we do not mind truncating values that are off the rating scale, and making predictions that are not actual rating values. The models we describe in this chapter all assume that the distribution over rating values is a multinomial, or conditional multinomial. This means we treat ratings as categorical random variables, which is a better approximation for ordinal ratings than continuous random variables, but ignores the inherent ordering of the rating values.

7.1 The Multinomial Model

The multinomial model is a simple probabilistic model for categorical data. The main modeling assumption at the profile level is that the values of ratings for each item are statistically independent of each other. This corresponds to the assumption that $P(\mathbf{R} = \mathbf{r}^u) = \prod_{y=1}^{M} P(R_y = r_y^u)$. At the data set level the multinomial model asserts that there is only one type of user. In the corresponding generative process, a rating profile is generated by independently sampling one rating for each item according to the unique multinomial distribution over ratings for that item. It is important to note that the generative process outputs a complete user profile with no missing ratings. From an inference standpoint, the complete independence assumption means that knowing any subset of the rating values in a user profile tells you nothing more about the ratings of the remaining items. The multinomial model is thus too weak to provide personalized recommendations. However, it serves as a good introduction to models based on multinomial rating distributions.

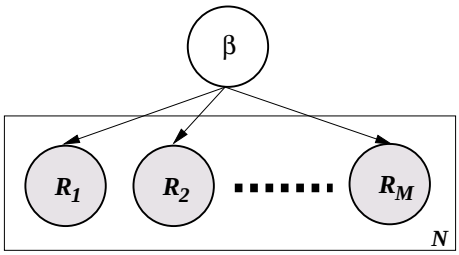

Figure 7.1: Multinomial Model

We show the graphical model in figure 7.1. Each node labeled R_y represents a multinomial random variable corresponding to the rating for item y. The value of random variable R_y is said to be observed if the active user has specified a rating for item y as given by r_y^a, and is unobserved otherwise. The node labeled β is a model parameter that encodes the multinomial distributions for each item. It can be thought of as a $V \times M$ matrix where β_{vy} represents $P(R_y = v)$.

7.1.1 Learning

Since all variables are independent, learning the parameters of the multinomial model simplifies to estimating the $P(R_y = v)$ for all y and v. This is quickly and easily done using frequency counts. It is generally a good idea to smooth the probability estimates in the case where little data is available for items to avoid zero probabilities . In the equation below we compute the standard Laplace estimate of $P(R_y = v)$. We give a learning method for the multinomial model in algorithm 7.1.

$$P(R_y = v) = \frac{1 + \sum_{u=1}^{N} \delta(r_y^u, v)}{V + \sum_{v'=1}^{V} \sum_{u=1}^{N} \delta(r_y^u, v')} \tag{7.1}$$

7.1.2 Rating Prediction

Given a learned multinomial model, we have already mentioned that the predictions for unknown items are independent of the ratings specified in the profile of the active user, $\mathbf{r}^a$. However, there are still several methods for computing a prediction based on the

Inputs: $\{\mathbf{r}^u\}$
Outputs: $\hat{r}$

Initialize θ, β
for $y = 1$ to M **do**
  for $v = 1$ to V **do**
    $p_v \leftarrow \frac{1+\sum_{u=1}^{N} \delta(r_y^u, v)}{V+\sum_{v'=1}^{V}\sum_{u=1}^{N} \delta(r_y^u, v')}$
  end for
  $\hat{r}_y \leftarrow \text{median}\, p$
end for

Algorithm 7.1: Multi-Learn

Input: $\mathbf{r}^a$, $\hat{r}$
Output: $\hat{\mathbf{r}}^a$

$\hat{r}^a \leftarrow \hat{r}$

Algorithm 7.2: Multi-Predict

distribution over ratings, and each method minimizes a different error measure. For a particular item y, if we predict the expected rating $\hat{r}_y^a = \sum_{v=1}^{V} vP(R_y = v)$ we will minimize the mean squared error defined in equation 3.1. If we predict the median rating defined by $\hat{r}_y^a = \{v|P(R_y < v) \leq 1/2, P(R_y > v) \geq 1/2\}$ we minimize the mean absolute error defined in equation 3.2. Lastly, if we predict the most probable rating value defined as $\hat{r}_y^a = \arg\max_v P(R_y = v)$ we minimize the mean prediction error defined in equation 3.3. This result holds not only for the multinomial model, but for all models based on multinomials over ratings. Existing work on these models often combines mean prediction with the mean absolute error measure. If the goal is to minimize prediction error, then this is clearly the incorrect choice. We use median prediction with all models since it minimizes the mean absolute error measure we have chosen.

7.1.3 Complexity

Learning the multinomial model has time complexity $O(MNV)$. As a non-personalized prediction method, predictions for each item need only be computed once, a step that can

be incorporated into the learning phase at an addition cost of $O(MV)$. At prediction time all unknown ratings for the active user can be looked up in the pre-computed prediction vector at no additional computational cost as we show in algorithms 7.1 and 7.2.

7.1.4 Results

Learning and prediction methods for the multinomial model were implemented, and tested for both their strong and weak generalization performance. On the EachMovie data set a weak generalization NMAE rate of 0.5383 ± 0.0022 was obtained, along with a strong generalization NAME rate of 0.5446 ± 0.0029. On the MovieLens data set a weak generalization NMAE rate of 0.4694 ± 0.0020 was obtained, along with a strong generalization NAME rate of 0.4746 ± 0.0035. The multinomial model is too simplistic to be of serious interest; however, these results serve as a useful baseline for comparing the performance of the more complex models we will study next.

7.2 Mixture of Multinomials Model

The mixture of multinomials model posits that there are K types of users underlying all profiles, and that the values of rating variables are independent of each other and the user's identity given the user's type. A user's type is modeled as a latent variable Z that takes K settings. The assertion that the ratings of items are conditionally independent given the value of the latent variable Z is sometimes called the naive Bayes assumption.

From a generative point of view, a profile is generated by sampling a user type z according to a prior distribution over user types, and then sampling a rating for each item according to the distribution over ratings for that item, given the chosen user type. The model parameters are the components of the multinomial distribution over settings of the latent variable, and the components of the distribution over rating values for each item and user type.

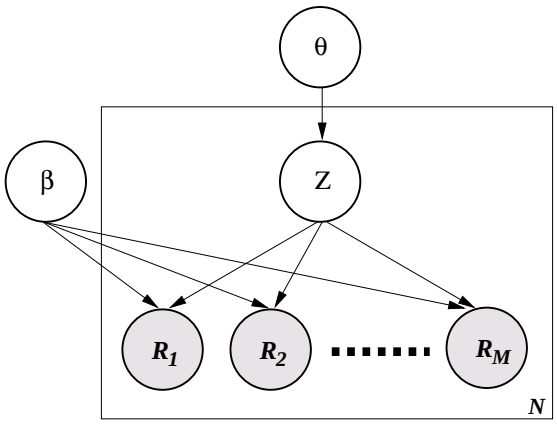

Figure 7.2: Mixture of Multinomials Model.

We show the graphical model in figure 7.2. Again, each node labeled R_y represents a multinomial rating random variable for item y. The node labeled Z is the latent variable corresponding to the user type. The node labeled β is a parameter that encodes the multinomial distributions for each item given a value of the latent variable. It can be thought of as a $V \times M \times K$ matrix where β_{vyz} represents $P(R_y = v|Z = z)$. The node labeled θ is a parameter that encodes a prior distribution over the latent variable Z. It can be thought of as a length K vector where θ_z gives $P(Z = z)$.

The joint probability of observing a user of type z with profile $\mathbf{r}^u$ is given by 7.2. Of course, we never actually know the value of the latent variable because we are never told what type a user is. To compute the marginal probability of a particular user profile with the user type unobserved, we simply sum out the latent variable obtaining 7.3. The resulting density is a mixture of K multinomial distributions.

$$P(\mathbf{R} = \mathbf{r}^u, Z = z) = P(Z = z) \prod_{y=1}^{M} P(R_y = r_y^u | Z = z) \tag{7.2}$$

$$P(\mathbf{R} = \mathbf{r}^u) = \sum_{z=1}^{K} P(Z = z) \prod_{y=1}^{M} P(R_y = r_y^u | Z = z) \tag{7.3}$$

7.2.1 Learning

Learning in the mixture of multinomials model is more difficult than in the simple multinomial case because the value of the latent variable is never observed. This necessitates the use of an Expectation Maximization (EM) procedure [16]. We have already described several EM algorithms in connection with weighted singular value decomposition, factor analysis, and probabilistic principal components analysis in Chapter 6. In this sub-section we derive an EM procedure for the mixture of multinomials model by applying the free energy approach of Neal and Hinton described in chapter 3 [44]. This approach provides a unified framework for deriving many variants of EM.

To begin we define the total free energy of the mixture of multinomials model, which depends on the definition of a distribution $q(Z|\mathbf{R} = \mathbf{r}^u)$. In general the distribution $q(Z|\mathbf{R} = \mathbf{r}^u)$ can be an approximation to the true posterior $P(Z|\mathbf{R} = \mathbf{r}^u)$; however, in the case of the mixture of multinomials model the true posterior can be found exactly. We parameterize the distribution $q(Z|\mathbf{R} = \mathbf{r}^u)$ as a multinomial with components ϕ_z^u. The total free energy function $F[\phi, \theta, \beta]$ is shown in equation 7.4.

$$
\begin{aligned}
F[\phi, \theta, \beta] &= \sum_{u=1}^{N} E_q[\log P(\mathbf{R} = \mathbf{r}^u, Z = z|\theta, \beta)] + \sum_{u=1}^{N} H[q(Z = z|\mathbf{R} = \mathbf{r}^u)] \quad (7.4) \\
&= \sum_{u=1}^{N} \sum_{z=1}^{K} q(Z = z|\mathbf{R} = \mathbf{r}^u) \log P(\mathbf{R} = \mathbf{r}^u, Z = z|\theta, \beta) \\
&\quad - \sum_{u=1}^{N} \sum_{z=1}^{K} q(Z = z|\mathbf{R} = \mathbf{r}^u) \log q(Z = z|\mathbf{R} = \mathbf{r}^u) \\
&= \sum_{u=1}^{N} \sum_{z=1}^{K} \phi_z^u \log \left(\theta_z \prod_{y=1}^{M} \prod_{v=1}^{V} \beta_{vyz}^{\delta(r_y^u, v)} \right) - \sum_{u=1}^{N} \sum_{z=1}^{K} \phi_z^u \log \phi_z^u \quad (7.5)
\end{aligned}
$$

Learning is now cast as the task of optimizing the function $F[\phi, \theta, \beta]$ with respect to the q distribution parameters ϕ_z^u, and the model parameters θ_z and β_{vyz}. We derive an iterative gradient descent procedure for optimizing the parameters as is done in standard EM. For the mixture of multinomials model this is easily accomplished as we show below. We first derive the update for the ϕ_z^u parameters. Recall that these parameters encode multinomial distributions so we must enforce the normalization constraint $\sum_{z=1}^{K} \phi_z^u = 1$.

$$\begin{aligned}
\frac{\partial F[\phi,\theta,\beta]}{\partial \phi_z^u} &= \log\left(\theta_z \prod_{y=1}^{M}\prod_{v=1}^{V} \beta_{vyz}^{\delta(r_y^u,v)}\right) - \frac{\phi_z^u}{\phi_z^u} - \log\phi_z^u - \lambda = 0 \qquad (7.6)\\
\log\phi_z^u &= \log\left(\theta_z \prod_{y=1}^{M}\prod_{v=1}^{V} \beta_{vyz}^{\delta(r_y^u,v)}\right) - 1 - \lambda\\
\phi_z^u &= e^{-1-\lambda}\theta_z \prod_{y=1}^{M}\prod_{v=1}^{V} \beta_{vyz}^{\delta(r_y^u,v)}\\
\sum_{z=1}^{K}\phi_z^u &= \sum_{z=1}^{K} e^{-1-\lambda}\theta_z \prod_{y=1}^{M}\prod_{v=1}^{V} \beta_{vyz}^{\delta(r_y^u,v)}\\
e^{1+\lambda} &= \sum_{z=1}^{K}\theta_z \prod_{y=1}^{M}\prod_{v=1}^{V} \beta_{vyz}^{\delta(r_y^u,v)}\\
\lambda &= \log\sum_{z=1}^{K}\theta_z \prod_{y=1}^{M}\prod_{v=1}^{V} \beta_{vyz}^{\delta(r_y^u,v)}\\
\phi_z^u &= \frac{\theta_z \prod_{y=1}^{M}\prod_{v=1}^{V} \beta_{vyz}^{\delta(r_y^u,v)}}{\sum_{z'=1}^{K}\theta_{z'}\prod_{y=1}^{M}\prod_{v=1}^{V} \beta_{vyz'}^{\delta(r_y^u,v)}} \qquad (7.7)
\end{aligned}$$

Next we move on to the updates for the model parameters. We begin with θ, which has the same constraint as ϕ^u. Namely, we must ensure that $\sum_{z=1}^{K}\theta_z = 1$.

$$\begin{aligned}
\frac{\partial F[\phi,\theta,\beta]}{\partial \theta_z} &= \sum_{u=1}^{N}\phi_z^u \frac{\prod_{y=1}^{M}\prod_{v=1}^{V}\beta_{vyz}^{\delta(r_y^u,v)}}{\theta_z\prod_{y=1}^{M}\prod_{v=1}^{V}\beta_{vyz}^{\delta(r_y^u,v)}} - \lambda = 0 \qquad (7.8)\\
\theta_z &= \frac{1}{\lambda}\sum_{u=1}^{N}\phi_z^u\\
\sum_{z=1}^{K}\theta_z &= \frac{1}{\lambda}\sum_{z=1}^{K}\sum_{u=1}^{N}\phi_z^u\\
\lambda &= \sum_{z=1}^{K}\sum_{u=1}^{N}\phi_z^u\\
\theta_z &= \frac{\sum_{u=1}^{N}\phi_z^u}{\sum_{z=1}^{K}\sum_{u=1}^{N}\phi_z^u} \qquad (7.9)
\end{aligned}$$

Lastly, we derive the update for the β model parameter. Recall that β_{vyz} gives the multinomial parameters of the distribution $P(R_y = v|Z = z)$. In optimizing with respect to β_{vyz} we must enforce the constraint that $\sum_{v=1}^{V}\beta_{vyz} = 1$.

Inputs: $\{\mathbf{r}^u\}$,K
Outputs: θ, β

Initialize θ, β
while ($F[\phi, \theta, \beta]$ Not Converged) **do**
  for $u = 1$ to N **do**
    $\phi_z^u \leftarrow \frac{\theta_z \prod_{y=1}^{M} \prod_{v=1}^{V} \beta_{vyz}{}^{\delta(r_y^u, v)}}{\sum_{z'=1}^{K} \theta_{z'} \prod_{y=1}^{M} \prod_{v=1}^{V} \beta_{vyz'}{}^{\delta(r_y^u, v)}}$
  end for
  for $z = 1$ to K **do**
    $\theta_z \leftarrow \frac{\sum_{u=1}^{N} \phi_z^u}{\sum_{z=1}^{K} \sum_{u=1}^{N} \phi_z^u}$
    for $y = 1$ to M, $v = 1$ to V **do**
      $\beta_{vyz} \leftarrow \frac{\sum_{u=1}^{N} \phi_z^u \delta(r_y^u, v)}{\sum_{v'=1}^{V} \sum_{u=1}^{N} \phi_z^u \delta(r_y^u, v')}$
    end for
  end for
end while

Algorithm 7.3: MixMulti-Learn

$$\frac{\partial F[\phi, \theta, \beta]}{\partial \beta_{vyz}} = \sum_{u=1}^{N} \phi_z^u \frac{\delta(r_y^u, v)}{\beta_{vyz}} - \lambda = 0 \tag{7.10}$$

$$\begin{aligned} \beta_{vyz} &= \frac{1}{\lambda} \sum_{u=1}^{N} \phi_z^u \delta(r_y^u, v) \\ \sum_{v=1}^{V} \beta_{vyz} &= \frac{1}{\lambda} \sum_{v=1}^{V} \sum_{u=1}^{N} \phi_z^u \delta(r_y^u, v) \\ \lambda &= \sum_{v=1}^{V} \sum_{u=1}^{N} \phi_z^u \delta(r_y^u, v) \\ \beta_{vyz} &= \frac{\sum_{u=1}^{N} \phi_z^u \delta(r_y^u, v)}{\sum_{v'=1}^{V} \sum_{u=1}^{N} \phi_z^u \delta(r_y^u, v')} \end{aligned} \tag{7.11}$$

Given these update rules, the optimization procedure for learning the mixture of multinomials model is straightforward. We randomly initialize the parameters θ and β ensuring that the constraints $\sum_{z=1}^{K} \theta_z = 1$, and $\sum_{v=1}^{V} \beta_{vyk} = 1$ hold. Iterating the derrived updates until the value of the objective function $F[\phi, \theta, \beta]$ converges defines a standard EM algorithm as given in algorithm 7.3.

Input: $\mathbf{r}^a$, θ, β
Output: $\hat{\mathbf{r}}^a$

for ($z = 1$ to K) **do**
  $\phi_z \leftarrow \frac{\theta_z \prod_{y=1}^{M} \prod_{v=1}^{V} \beta_{vyz}^{\delta(r_y^a, v)}}{\sum_{z'=1}^{K} \theta_{z'} \prod_{y=1}^{M} \prod_{v=1}^{V} \beta_{vyz'}^{\delta(r_y^a, v)}}$
end for
for $y = 1$ to M **do**
  for $v = 1$ to V **do**
    $p_v \leftarrow \sum_{z=1}^{K} \beta_{vyz} \phi_z$
  end for
  $\hat{r}_y \leftarrow \text{median}\, p_v$
end for

Algorithm 7.4: MixMulti-Predict

7.2.2 Rating Prediction

Once learning has converged, the resulting model (θ, β) can be used to predict ratings given any novel user profile $\mathbf{r}^a$. Rating prediction with the mixture of multinomials model consists of first estimating $P(R_y = v|\mathbf{r}^a, \theta, \beta)$ as shown in equation 7.12, and then applying a prediction rule such as median prediction to the estimated distribution. Estimating the distribution consists of an inference step that determines the mixing coefficients $P(Z = z|\mathbf{r}^a)$, and the actual mixture step. Intuitively $P(Z = z|\mathbf{r}^a)$ is the degree to which the active user exhibits traits of a type z user through the profile $\mathbf{r}^a$. In the mixture step, the distribution over ratings for an item is computed as a mixture of the distributions over ratings for the item given by each user type. Note that while the generative semantics of the mixture of multinomials model assert that every user belongs to exactly one of the K user types, the predicted distributions are actually computed as if the user was a mixture of different user types. The complete prediction method is given in algorithm 7.4.

$$\begin{aligned} P(R_y = v|\mathbf{r}^a, \theta, \beta) &= \sum_{z=1}^{K} P(R_y = v|Z = z) P(Z = z|\mathbf{r}^a) \\ &= \sum_{z=1}^{K} \beta_{vyz} \frac{\theta_z \prod_{y=1}^{M} \prod_{v=1}^{V} \beta_{vyz}^{\delta(r_y^a, v)}}{\sum_{z'=1}^{K} \theta_{z'} \prod_{y=1}^{M} \prod_{v=1}^{V} \beta_{vyz'}^{\delta(r_y^a, v)}} \end{aligned} \tag{7.12}$$

7.2.3 Complexity

In the E-step of the MixMulti-Learn algorithm we compute an estimate ϕ_z^u for all z and u. The total computational complexity of the E-Step is $O(NMVK)$. The M-step consists of computing updates for θ_z and β_{vyz}. The computational complexity of the update for θ is $O(MK)$. The computational complexity of the update for β is $O(NMVK)$. Letting I be an upper bound on the number of iterations needed to reach convergence we obtain a total time complexity of $O(INMVK)$. The number of parameters that need to be stored for the mixture of multinomials model is $K + MVK$.

The MixMulti-Predict algorithm first computes ϕ_z for the active user at a computational cost of $O(MVK)$. For each item a distribution over ratings is calculated at a cost of $O(VK)$ per item. The total computational cost of the MixMulti-Predict algorithm is thus $O(MVK)$.

7.2.4 Results

The MixMulti-Learn and MixMulti-Predict methods described in algorithms 7.3 and 7.4 were implemented, and tested for both their strong and weak generalization performance. EachMovie was tested using 5, 10, 20, and 30 user types. MovieLens was tested using 4, 6, 8, and 10 user types. The learning method was found to converge reliably in 30 to 40 iterations with θ and β initialized randomly. The results of the prediction performance experiments with the mixture of multinomials model are shown in tables 7.1, and 7.2.

Table 7.1: MixMulti-Predict: EachMovie Results

	$K = 5$	$K = 10$	$K = 20$	$K = 30$
Weak	0.4755 ± 0.0013	0.4579 ± 0.0006	0.4559 ± 0.0010	0.4557 ± 0.0012
Strong	0.4744 ± 0.0038	0.4631 ± 0.0027	0.4602 ± 0.0011	0.4573 ± 0.0007

Table 7.2: MixMulti-Predict: MovieLens Results

	$K = 4$	$K = 6$	$K = 8$	$K = 10$
Weak	0.4444 ± 0.0032	0.4480 ± 0.0020	0.4535 ± 0.0020	0.4528 ± 0.0049
Strong	0.4573 ± 0.0124	0.4383 ± 0.0048	0.4405 ± 0.0045	0.4339 ± 0.0023

7.3 The Aspect Model

In this section we describe the aspect model for rating prediction [29]. The rating prediction version of the aspect model is closely related to the aspect model for probabilistic latent semantic analysis of text documents, also referred to as pLSA or pLSI [28]. To avoid confusion we will refer to the rating prediction version as the triadic aspect model, and the text analysis version as the dyadic aspect model. It is important to understand the relationship between the dyadic and triadic models, and more generally the relationship between vectorspace or bag-of-words text analysis, collaborative filtering, and rating prediction. These relationships play a crucial role in the development of the next section.

In the dyadic aspect model applied to text analysis, a corpus of documents is modeled as a set of pairs (d, w), where d is a document index and w is a word index. A fixed vocabulary is assumed. This type of data is often called co-occurrence or dyadic data, thus our choice of name for the model. The graphical representation of the dyadic aspect model applied to text analysis appears in figure 7.3. Each document is represented as a unique distribution over the K settings of the latent variable Z. Each setting of the latent variables Z corresponds to an underlying "topic". Associated with each topic is a distribution over words in the vocabulary. Thus, a document is seen as a distribution over topics where each topic is described by a different distribution over words. A word is generated for a document by choosing a topic and then selecting a word according to the distribution over words for the chosen topic. This is an interesting model for

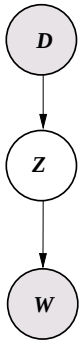

Figure 7.3: Dyadic aspect model.

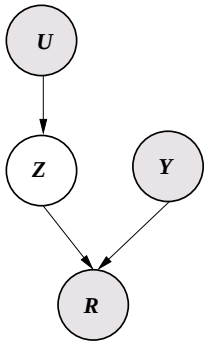

Figure 7.4: Triadic aspect model.

text because it allows different documents to be generated by different distributions over topics (settings of the latent variable), which is not possible with a multinomial mixture model where all documents would share a common distribution over topics.

The dyadic aspect model can also be applied to collaborative filtering, but within a different formulation. Recall from our discussion of preference indicators in sub-section 2.1.1 that co-occurrence data is semantically distinct from explicit ratings. To apply the dyadic aspect model to a pure, non-sequential, co-occurrence formulation of collaborative filtering, we simply exchange documents for users and words for items. The data consists of (u, y) pairs representing the fact that user u accessed, or viewed item y. A data set may contain the same pair multiple times, just as a corpus of text documents may contain the same word-document pair multiple times.

The correct analogy between collaborative filtering and bag-of-words text modeling should now be clear: only the pure, non-sequential, co-occurrence formulation of collaborative filtering is equivalent to the bag-of-words representation of text documents. Any model that can be applied in one domain can be applied in the other. Formulations based on other types of preference indicators including ratings are not equivalent to the this representation of text documents, and models of each are not exchangeable.

However, a slight extension of the dyadic aspect model yields the triadic aspect model, which is capable of representing preferences based on explicit ratings. The particular instance of the triadic aspect model we are interested in assumes that the basic data element is a triple (u, y, v) where u is a user, y is an item, and v is a rating value. Each triple represents the fact that user u assigned rating v to item y. Hofmann's depiction of the model is shown in figure 7.4 [29]. Each setting of the latent variable Z can be interpreted as a "user type", or "user attitude". A particular user u is modeled as a unique distribution over user types $P(Z|U = u)$. These distributions are seen as parameters and are encoded by θ^u. A user type is represented as a multinomial distribution $P(R|Z = z, Y = y)$ over rating values for each item.

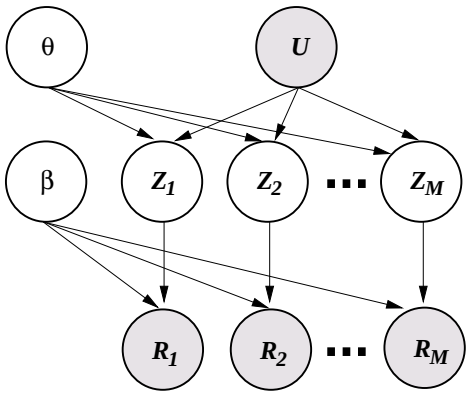

Figure 7.5: Vector aspect model.

The triadic aspect model is flawed in several ways, and it shares some of these flaws with the dyadic version. First, the distribution over types for each user u must be specified *outside* of the generative process as a parameter θ^u. If we wish to generate ratings for N users then we must be given N distributions $\theta^1, ..., \theta^N$. The triadic aspect model itself provides no probabilistic mechanism for sampling θ^u. As a result the model has improper generative semantics at the user level. The number of model parameters also grows with the number of user profiles we are trying to model. This is a highly undesirable feature for a model based approach. In addition, the model lacks a principled inference procedure for novel user profiles since the θ^u parameters are unknown for an arbitrary profile. This means it is impossible to perform rating prediction with novel user profiles.

The lower levels of the generative process are also problematic. Once the θ^u parameters are given, the generative process iterates through all the users. For each user and each item a value of Z is sampled from $P(Z|\theta^u, U = u)$, and a rating is sampled from $P(R|Z = z, Y = y)$. While it is possible to force the aspect model into generating a complete profile for each user by clamping the U and Y nodes to the desired values, there is nothing in the model itself which restricts it to one rating per user-item combination.

The problems with the generative process at the higher levels of the model have a nontrivial solution as we will see in the next section. However, we can fix the repeated sampling problems at the lower levels of the aspect model quite easily. We simply expand

the lower levels of the model to ensure that exactly one rating is generated for each item. The necessary modification has no impact on model fitting or inference if we assume that the data contains at most one rating for each user-item pair. It also has no effect on the definition or interpretation of the model parameters. Our vector version of the aspect model appears in figure 7.5. The joint probability of a user rating profile and choice of user attitudes is shown in equation 7.13. The probability of of a user rating profile is shown in equation 7.14. The likelihood of the complete set of user profiles under the aspect model is given in equation 7.15.

$$P(\mathbf{r}^u, \mathbf{z}|\theta^u, \beta) \prod_{y=1}^{M} \prod_{v=1}^{V} (P(Z_y = z_y|\theta^u) P(R_y = r_y^u|Z_y = z_y, \beta))^{\delta(r_y^u, v)} \tag{7.13}$$

$$P(\mathbf{r}^u|\theta, \beta) = \prod_{y=1}^{M} \prod_{v=1}^{V} (\sum_{z=1}^{K} P(Z_y = z|\theta^u) P(R_y = r_y^u|Z = z, \beta))^{\delta(r_y^u, v)} \tag{7.14}$$

$$P(\mathbf{r}|\theta, \beta) = \prod_{u=1}^{N} \prod_{y=1}^{M} \prod_{v=1}^{V} (\sum_{z=1}^{K} P(Z_y = z|\theta^u) P(R_y = r_y^u|Z = z, \beta))^{\delta(r_y^u, v)} \tag{7.15}$$

7.3.1 Learning

In this subsection we derive an EM procedure for model fitting based on the vector aspect model. We again apply the free energy approach of Neal and Hinton. Hofmann gives a set of update rules for the triadic version of the aspect model, but the rules we derive for the vector aspect model make the handling of missing rating values explicit.

In the generative process recall that for each item we independently choose one setting of the latent variable Z. We assume a corresponding factorization of the q-distribution $q(\mathbf{Z} = \mathbf{z}|\mathbf{R} = \mathbf{r}^u) = \prod_{y=1}^{M} q(Z_y = z_y|\mathbf{R} = \mathbf{r}^u)$. Note that Z_y depends only on the value of r_y^u so we can further factorize the q-distribution obtaining $\prod_{y=1}^{M} q(Z_y = z_y|r_y^u)$ We parameterize $q(Z_y = z|r_y^u)$ as a multinomial distribution with parameters given by ϕ_{vyz}^u. Note that this distribution is normalized with respect to Z so that we must have $\sum_z \phi_{vyz}^u = 1$. The total free energy function $F[\phi, \theta, \beta]$ is shown next.

$$
\begin{aligned}
F[\phi,\theta,\beta] &= \sum_{u=1}^{N} E_q[\log P(\mathbf{R}=\mathbf{r}^u,\mathbf{Z}=\mathbf{z}|\theta,\beta)] + \sum_{u=1}^{N} H[q(\mathbf{Z}=\mathbf{z}|\mathbf{R}=\mathbf{r}^u)] \qquad (7.16)\\
&= \sum_{u=1}^{N}\sum_{y=1}^{M}\sum_{v=1}^{V}\delta(r_y^u=v)\sum_{z=1}^{K} q(Z_y=z|R_y=v)\log P(R_y=v,Z=z|\theta^u,\beta)\\
&- \sum_{u=1}^{N}\sum_{y=1}^{M}\sum_{v=1}^{V}\delta(r_y^u,v)\sum_{z=1}^{K} q(Z_y=z|R_y=v)\log q(Z_y=z|R_y=v)\\
&= \sum_{u=1}^{N}\sum_{y=1}^{M}\sum_{v=1}^{V}\delta(r_y^u,v)\sum_{z=1}^{K}\phi_{vyz}^u\log(\beta_{vyz}\theta_z^u) - \sum_{u=1}^{N}\sum_{y=1}^{M}\sum_{v=1}^{V}\delta(r_y^u,v)\sum_{z=1}^{K}\phi_{vyz}^u\log\phi_{vyz}^u
\end{aligned}
$$

We first work out the update for the ϕ_{vyz}^u parameters. Recall that these parameters are multinomial distributions so we must enforce the normalization constraint $\sum_{z=1}^{K}\phi_{vyz}^u = 1$ for all u.

$$
\begin{aligned}
\frac{\partial F[\phi,\theta,\beta]}{\partial\phi_{vyz}^u} &= \delta(r_y^u,v)\log(\beta_{vyz}\theta_z^u) - \delta(r_y^u=v)(\log\phi_{vyz}^u+1) - \lambda = 0 \qquad (7.17)\\
\log\phi_{vyz}^u &= \log(\beta_{vyz}\theta_z^u) - 1 - \frac{\lambda}{\delta(r_y^u,v)}\\
\phi_{vyz}^u &= \beta_{vyz}\theta_z^u e^{-1-\frac{\lambda}{\delta(r_y^u,v)}}
\end{aligned}
$$

$$
\begin{aligned}
\sum_{z=1}^{K}\phi_{vyz}^u &= \sum_{z=1}^{K}\beta_{vyz}\theta_z^u e^{-1-\frac{\lambda}{\delta(r_y^u,v)}}\\
e^{1+\frac{\lambda}{\delta(r_y^u,v)}} &= \sum_{z'=1}^{K}\beta_{vyz'}\theta_{z'}^u\\
1+\frac{\lambda}{\delta(r_y^u,v)} &= \log(\sum_{z'=1}^{K}\beta_{vyz'}\theta_{z'}^u)\\
\lambda &= \delta(r_y^u,v)(\log(\sum_{z'=1}^{K}\beta_{vyz'}\theta_{z'}^u)-1)
\end{aligned}
$$

$$
\begin{aligned}
\phi_{vyz}^u &= \beta_{vyz}\theta_z^u e^{-1-\frac{\delta(r_y^u,v)(\log(\sum_{z'=1}^{K}\beta_{vyz'}\theta_{z'}^u)-1)}{\delta(r_y^u,v)}}\\
&= \frac{\beta_{vyz}\theta_z^u}{\sum_{z'=1}^{K}\beta_{vyz'}\theta_{z'}^u} \qquad (7.18)
\end{aligned}
$$

Next we move on to the problem of the updates for the model parameters. Unlike the mixture of multinomials case, each user has a unique distribution over user types parameterized by θ^u. Again we must ensure that $\sum_{z=1}^{K}\theta_z^u = 1$.

$$\begin{aligned}
\frac{\partial F[\phi,\theta,\beta]}{\partial \theta^u_z} &= \sum_{y=1}^{M}\sum_{v=1}^{V}\delta(r^u_y,v)\frac{\phi^u_{vyz}}{\theta^u_z} - \lambda = 0 \qquad (7.19)\\
\lambda &= \sum_{y=1}^{M}\sum_{v=1}^{V}\delta(r^u_y,v)\frac{\phi^u_{vyz}}{\theta^u_z}\\
\theta^u_z &= \frac{1}{\lambda}\sum_{y=1}^{M}\sum_{v=1}^{V}\delta(r^u_y,v)\phi^u_{vyz}\\
\sum_{z'=1}^{K}\theta^u_{z'} &= \frac{1}{\lambda}\sum_{z'=1}^{K}\sum_{y=1}^{M}\sum_{v=1}^{V}\delta(r^u_y,v)\phi^u_{vyz'}\\
\lambda &= \frac{1}{\sum_{z'=1}^{K}\sum_{y=1}^{M}\sum_{v=1}^{V}\delta(r^u_y,v)\phi^u_{vyz'}}\\
\theta^u_z &= \frac{\sum_{y=1}^{M}\sum_{v=1}^{V}\delta(r^u_y,v)\phi^u_{vyz}}{\sum_{z'=1}^{K}\sum_{y=1}^{M}\sum_{v=1}^{V}\delta(r^u_y,v)\phi^u_{vyz'}} \qquad (7.20)
\end{aligned}$$

Lastly, we derive the update for the β model parameter. β encodes the multinomial parameters of the distribution $P(R_y = v|Z = z)$ as in the mixture of multinomials model. In optimizing with respect to β_{vyz} we enforce the constraint that $\sum_{v=1}^{V}\beta_{vyz} = 1$.

$$\begin{aligned}
\frac{\partial F[\phi,\theta,\beta]}{\partial \beta_{vyz}} &= \sum_{u=1}^{N}\delta(r^u_y,v)\frac{\phi^u_{vyz}}{\beta_{vyz}} - \lambda = 0 \qquad (7.21)\\
\beta_{vyz} &= \frac{1}{\lambda}\sum_{u=1}^{N}\delta(r^u_y,v)\phi^u_{vyz}\\
\sum_{v'=1}^{V}\beta_{v'yz} &= \frac{1}{\lambda}\sum_{v'=1}^{V}\sum_{u=1}^{N}\delta(r^u_y,v')\phi^u_{v'yz}\\
\lambda &= \sum_{v'=1}^{V}\sum_{u=1}^{N}\delta(r^u_y,v')\phi^u_{v'yz}\\
\beta_{vyz} &= \frac{\sum_{u=1}^{N}\delta(r^u_y,v)\phi^u_{vyz}}{\sum_{v'=1}^{V}\sum_{u=1}^{N}\delta(r^u_y,v')\phi^u_{v'yz}} \qquad (7.22)
\end{aligned}$$

Given these update rules, the optimization procedure for learning the aspect model is straightforward. We randomly initialize the parameters θ and β ensuring that the constraints $\sum_{z=1}^{K}\theta^u_z = 1$, and $\sum_{v=1}^{V}\beta_{vyz} = 1$ hold for all u,y and z. We then iterate the updates until the value of the objective function $F[\phi,\theta,\beta]$ converges. The vector aspect model learning procedure is specified in algorithm 7.5.

Inputs: $\{\mathbf{r}^u\}$, K
Outputs: θ, β

Initialize θ, β
while ($F[\phi, \theta, \beta]$ Not Converged) **do**
  for $u = 1$ to N, $v = 1$ to V, $z = 1$ to Z **do**
    $\phi^u_{vyz} \leftarrow \frac{\beta_{vyz}\theta^u_z}{\sum_{z'=1}^{K}\beta_{vyz'}\theta^u_{z'}}$
  end for
  for $u = 1$ to N, $z = 1$ to K **do**
    $\theta^u_z \leftarrow \frac{\sum_{y=1}^{M}\sum_{v=1}^{V}\delta(r^u_y,v)\phi^u_{vyz}}{\sum_{z'=1}^{K}\sum_{y=1}^{M}\sum_{v=1}^{V}\delta(r^u_y,v)\phi^u_{vyz'}}$
  end for
  for $y = 1$ to M, $v = 1$ to V, $z = 1$ to K **do**
    $\beta_{vyz} \leftarrow \frac{\sum_{u=1}^{N}\delta(r^u_y,v)\phi^u_{vyz}}{\sum_{v'=1}^{V}\sum_{u=1}^{N}\delta(r^u_y,v')\phi^u_{v'yz}}$
  end for
end while

Algorithm 7.5: Aspect-Learn

7.3.2 Rating Prediction

Rating prediction with the aspect model is somewhat different than with the mixture of multinomials model. Once learning has converged we have a model (θ,β), but this model can only be used to make predictions for the N training users in the data set since we can not properly perform inference. Assuming the active user a is one of the N training users we find the distribution over ratings for an unknown item y according to equation 7.23.

$$\begin{aligned} P(R_y = v|\mathbf{r}^a, U = a, \theta, \beta) &= P(R_y = v|U = a, \theta^a, \beta) \\ &= \sum_{z=1}^{K} P(R_y = v|Z = z)P(Z = z|U = u) \\ &= \sum_{z=1}^{K} \beta_{vyz}\theta^a_z \end{aligned} \tag{7.23}$$

Input: $\mathbf{r}^a$, θ^a, β
Output: $\hat{\mathbf{r}}^a$

for $y = 1$ to M **do**
  for $v = 1$ to V **do**
    $p_v \leftarrow \sum_{z=1}^{K} \beta_{vyz} \theta_z^a$
  end for
  $\hat{r}_y \leftarrow \text{median}\, p_v$
end for

Algorithm 7.6: Aspect-Predict

7.3.3 Complexity

In the E-step of the Aspect-Learn algorithm we compute an estimate ϕ_{vyz}^u. The total computational complexity of the E-Step is thus $O(NMVK)$. The M-step consists of computing updates for θ_z^u and β_{vyz}. The computational complexity of the update for θ is $O(NMVK)$. The computational complexity of the update for β is also $O(NMVK)$. Letting I be an upper bound on the number of iterations needed to reach convergence we obtain a total time complexity of $O(INMVK)$. The number of parameters that need to be stored for the aspect model is $KN + MVK$.

7.3.4 Results

The Aspect-Learn and Aspect-Predict methods described in algorithms 7.5 and 7.6 were implemented, and tested for their weak generalization performance. As mentioned previously, a learned aspect model can not be applied to a set of novel users in a principled manner. Thus the strong generalization error of the model can not be assessed.

As with the mixture of multinomials model, the learning method was found to converge reliably in 30 to 40 iterations with θ and β initialized randomly. However, with larger numbers user types, the vector aspect model tended to over fit the training data with respect to the prediction error measure in preliminary testing. Hofmann suggests using early stopping of the EM iteration to help overcome this problem. In the implemen-

tation an additional validation set was extracted from the training set. During EM the validation error was evaluated after every iteration. When the validation error increased on three successive iterations, the method was stopped. The best set of parameters according to the validation error estimate were selected as the final model. The results of the prediction performance experiments with the vector aspect model are shown in tables 7.1, and 7.2.

Table 7.3: Aspect-Predict: EachMovie Results

	$K = 5$	$K = 10$	$K = 20$	$K = 30$
Weak	0.4744 ± 0.0038	0.4631 ± 0.0027	0.4602 ± 0.0011	0.4573 ± 0.0007

Table 7.4: Aspect-Predict: MovieLens Results

	$K = 4$	$K = 6$	$K = 8$	$K = 10$
Weak	0.4573 ± 0.0124	0.4383 ± 0.0048	0.4405 ± 0.0045	0.4339 ± 0.0023

7.4 The User Rating Profile Model

In this section we present the User Rating Profile (URP) model recently introduced by Marlin [38]. As was mentioned in the previous section, the triadic aspect model is flawed in several ways. Changing the lower level of the model from a triadic to vector representation correct one set of problems, but issues remain with the user level generative semantics. The main problem is that the $P(Z|U = u)$ distributions are viewed as parameters outside the generative process of the model. As a result the aspect model lacks a maximum likelihood procedure for performing inference on novel user profiles. This means it is impossible to make rating predictions for users not in the training set. There are also a variety of undesirable secondary effects including the number of parameters in the model increasing with the number of users in the training set. The URP model has been proposed as a generative version of the vector aspect model, which solves all of these problems.

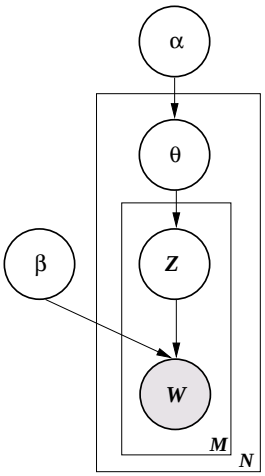

Figure 7.6: LDA model.

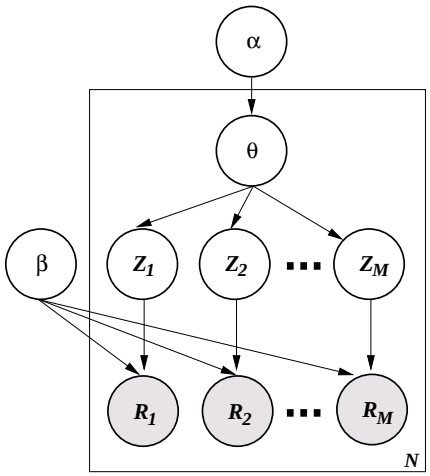

Figure 7.7: URP model.

Blei, Ng, and Jordan point out similar problems with the high level generative semantics of the dyadic aspect model. They propose the Latent Dirichlet Allocation (LDA) model as a correct generative version of the dyadic aspect model [7]. The obvious correction to the dyadic aspect model is to remove the index variable D and view the distribution over Z as a hidden variable θ. The complete generative process is to sample θ for each document, and Z and W for each of the M positions in the document. The simplest generative model with proper document level semantics would place a uniform distribution on θ. However, other priors on θ can also be used such as the Dirichlet prior, which is conjugate to the multinomial distribution. If a Dirichlet prior is used, the model we have just described is precisely the Latent Dirichlet Allocation (LDA) model proposed by Blei et al. [7].

Recall from our discussion of the dyadic, triadic, and vector aspect models that dyadic models can only be applied to a formulation of collaborative filtering based on co-occurrence data. Therefore LDA is clearly not applicable to rating-based collaborative filtering. Blei et al. slightly confuse this issue by applying LDA to a rating-based collaborative filtering data set using a preprocessing step [7, p. 1014]. All rating values above a threshold were set to 1 and those below the threshold were set to 0. This converts the rating data into a special form of co-occurrence data where the pair (u, y) indicates that

user u *likes* item y.

The user rating profile (URP) model is a true generative model for rating profiles. It can be seen as a generative version of the vector aspect model in the same sense that LDA can be seen as a generative version of the dyadic aspect model. To obtain URP from the vector aspect model we remove the user index node U and view the distribution over the latent variable Z as a random variable θ instead of a parameter. As in LDA we place a Dirichlet prior on θ since it is more general than a uniform prior, and is convenient to work with. The complete generative process for the URP model is to sample θ for each user from a Dirichlet prior with parameter α, then for each item y sample Z_y according to $P(Z|\theta)$, and a rating value R_y according to $P(R_y|Z=z,\beta)$.

Based on this description we can derive the joint probability of observing a complete user profile $\mathbf{r}^u$ along with corresponding values of θ and $\mathbf{Z}$ as shown in equation 7.24. To obtain the probability of observing a particular user profile $\mathbf{r}^u$ we sum and integrate out the values of the hidden variables from the joint probability as shown in equation 7.25. As in LDA, the Dirichlet prior renders the computation of the exact posterior distribution shown in equation 7.26 intractable, and variational techniques must be used to fit the URP model.

$$P(\mathbf{r}^u,\mathbf{z},\theta|\alpha,\beta) = P(\theta|\alpha)\prod_{y=1}^{M}\prod_{v=1}^{V}(P(Z_y=z_y|\theta)P(R_y=r_y^u|Z_y=z_y,\beta))^{\delta(r_y^u,v)} \tag{7.24}$$

$$P(\mathbf{r}^u|\alpha,\beta) = \int_\theta P(\theta|\alpha)\prod_{y=1}^{M}\prod_{v=1}^{V}(\sum_{z=1}^{K}P(Z_y=z|\theta)P(R_y=r_y^u|Z=z,\beta))^{\delta(r_y^u,v)}d\theta \tag{7.25}$$

$$P(\theta,\mathbf{z}|\mathbf{r}^u,\alpha,\beta) = P(\mathbf{r}^u,\theta,\mathbf{z}|\alpha,\beta)/P(\mathbf{r}^u|\alpha,\beta) \tag{7.26}$$

7.4.1 Variational Approximation and Free Energy

The procedure used for fitting the URP model is a variational Expectation Maximization algorithm. We choose to apply a fully factored variational q-distribution as shown in equation 7.27. We define $q(\theta|\gamma^u)$ to be a Dirichlet distribution with Dirichlet parameters

γ_i^u, and $q(Z_y|\phi_y^u)$ to be a multinomial distribution with parameters ϕ_{iy}^u.

$$q(\theta, \mathbf{z}|\gamma^u, \phi^u) = q(\theta|\gamma^u) \prod_{y=1}^{M} q(Z_y = z_y|\phi_y^u) \tag{7.27}$$

We give the probability of θ under a Dirichlet distribution with parameter α in equation 7.28. We also need the expectation $E[\log \theta_i]$ to fully compute the free energy. We give the formula for this expectation under a Dirichlet with parameter α in equation 7.29. $\Psi()$ denotes the digamma function, the first derivative of the log-gamma function.

$$P(\theta|\alpha) = \frac{\Gamma(\sum_{i=1}^K \alpha_i)}{\prod_{i=1}^K \Gamma(\alpha_i)} \prod_{i=1}^{K} \theta_i^{\alpha_i - 1} \tag{7.28}$$

$$E[\log \theta_i|\alpha] = \Psi(\alpha_i) - \Psi(\sum_{j=1}^{K} \alpha_j) \tag{7.29}$$

The total free energy of the URP model with respect to this fully factorized q-distribution is given by $F[\phi, \gamma, \alpha, \beta] = \sum_{u=1}^N (E_q[\log P(\theta, \mathbf{z}, \mathbf{r}^u|\alpha, \beta)] + H[q(\theta, \mathbf{z}|\gamma^u, \phi^u)])$. We expand $F[\phi, \gamma, \alpha, \beta]$ according to the factorizations of the joint and q distributions, and then compute each expectation. We use the notation $E_{q\theta}[\cdot]$ and $E_{q\mathbf{z}}[\cdot]$ to denote expectation with respect to each part of the factorized q-distribution. Note that $E_q[\cdot] = E_{q\theta}[E_{q\mathbf{z}}[\cdot]] = E_{q\mathbf{z}}[E_{q\theta}[\cdot]]$.

$$F[\phi, \gamma, \alpha, \beta] = \sum_{u=1}^{N} E_q[\log p(\theta|\alpha) + \log p(\mathbf{z}|\theta) + \log p(\mathbf{r}^u|\mathbf{z}, \beta) - \log q(\theta|\gamma^u) - \log q(\mathbf{z}|\phi^u)]$$

$$\begin{aligned} E_q[\log p(\theta|\alpha)] &= E_q[\log \Gamma(\sum_{i=1}^K \alpha_i) - \sum_{i=1}^K \log \Gamma(\alpha_i) + \sum_{i=1}^K (\alpha_i - 1) \log \theta_i] \\ &= \log \Gamma(\sum_{i=1}^K \alpha_i) - \sum_{i=1}^K \log \Gamma(\alpha_i) + \sum_{i=1}^K (\alpha_i - 1) E_{q\theta}[\log \theta_i] \\ &= \log \Gamma(\sum_{i=1}^K \alpha_i) - \sum_{i=1}^K \log \Gamma(\alpha_i) + \sum_{i=1}^K (\alpha_i - 1)(\Psi(\gamma_i^u) - \Psi(\sum_{j=1}^K \gamma_j^u)) \end{aligned}$$

$$\begin{aligned} E_q[\log p(\mathbf{z}|\theta)] &= E_q[\sum_{y=1}^M \log P(Z_y = z|\theta)] \\ &= \sum_{y=1}^M E_{q\mathbf{z}}[E_{q\theta}[\log \theta_z]] \end{aligned}$$

$$
\begin{aligned}
&= \sum_{y=1}^{M} E_{q\mathbf{z}}[\Psi(\gamma_z) - \Psi(\sum_{j=1}^{K}\gamma_j)] \\
&= \sum_{y=1}^{M}\sum_{z=1}^{K} q(Z_y = z|\phi^u)(\Psi(\gamma_z^u) - \Psi(\sum_{j=1}^{K}\gamma_j^u)) \\
&= \sum_{y=1}^{M}\sum_{z=1}^{K} \phi_{zy}^u(\Psi(\gamma_z^u) - \Psi(\sum_{j=1}^{K}\gamma_j^u))
\end{aligned}
$$

$$
\begin{aligned}
E_q[\log p(\mathbf{r}^u|\mathbf{z},\beta)] &= E_q[\sum_{y=1}^{M}\sum_{v=1}^{V}\delta(r_y^u, v)\log P(R_y = v|Z = z_y,\beta)] \\
&= \sum_{y=1}^{M}\sum_{v=1}^{V}\delta(r_y^u, v)E_{q\mathbf{z}}[\log P(R_y = v|Z = z_y,\beta)] \\
&= \sum_{y=1}^{M}\sum_{v=1}^{V}\delta(r_y^u, v)\sum_{z=1}^{K} q(Z_y = z|\phi_{zy}^u \log P(R_y = v|Z = z_y,\beta) \\
&= \sum_{y=1}^{M}\sum_{v=1}^{V}\delta(r_y^u, v)\sum_{z=1}^{K}\phi_{zy}^u \log\beta_{vyz}
\end{aligned}
$$

$$
\begin{aligned}
E_q[\log q(\theta|\gamma^u)] &= E_q[\log\Gamma(\sum_{i=1}^{K}\gamma_i^u) - \sum_{i=1}^{K}\log\Gamma(\gamma_i^u) + \sum_{i=1}^{K}(\gamma_i^u - 1)\log\theta_i] \\
&= \log\Gamma(\sum_{i=1}^{K}\gamma_i^u) - \sum_{i=1}^{K}\log\Gamma(\gamma_i^u) + \sum_{i=1}^{K}(\gamma_i^u - 1)E_{q\theta}[\log\theta_i] \\
&= \log\Gamma(\sum_{i=1}^{K}\gamma_i^u) - \sum_{i=1}^{K}\log\Gamma(\gamma_i^u) + \sum_{i=1}^{K}(\gamma_i^u - 1)(\Psi(\gamma_i^u) - \Psi(\sum_{j=1}^{K}\gamma_j^u))
\end{aligned}
$$

$$
\begin{aligned}
E_q[\log q(\mathbf{z}|\phi^u)] &= E_q[\sum_{y=1}^{M}\log q(Z_y = z|\phi^u)] \\
&= \sum_{y=1}^{M} E_{q\mathbf{z}}[\log q(Z_y = z|\phi^u)] \\
&= \sum_{y=1}^{M}\sum_{z=1}^{K} q(Z_y = z|\phi^u)\log q(Z_y = z|\phi^u) \\
&= \sum_{y=1}^{M}\sum_{z=1}^{K}\phi_{zy}^u\log\phi_{zy}^u
\end{aligned}
$$

Now we compile the expanded expectations yielding the total free energy function $F[\phi,\gamma,\alpha,\beta]$ shown in equation 7.30.

$$
F[\phi,\gamma,\alpha,\beta] = \sum_{u=1}^{N}\left(\log\Gamma(\sum_{i=1}^{K}\alpha_i) - \sum_{i=1}^{K}\log\Gamma(\alpha_i) + \sum_{i=1}^{K}(\alpha_i - 1)(\Psi(\gamma_i^u) - \Psi(\sum_{j=1}^{K}\gamma_j^u))\right)
$$

$$
\begin{aligned}
&+ \sum_{u=1}^{N}\sum_{y=1}^{M}\sum_{i=1}^{K}\phi_{iy}^{u}(\Psi(\gamma_i^u) - \Psi(\sum_{j=1}^{K}\gamma_j^u)) \\
&+ \sum_{u=1}^{N}\sum_{y=1}^{M}\sum_{v=1}^{V}\delta(r_y^u, v)\sum_{i=1}^{K}\phi_{iy}^{u}\log\beta_{vyi} \\
&- \sum_{u=1}^{N}\left(\log\Gamma(\sum_{i=1}^{K}\gamma_i^u) - \sum_{i=1}^{K}\log\Gamma(\gamma_i^u) + \sum_{i=1}^{K}(\gamma_i^u - 1)(\Psi(\gamma_i^u) - \Psi(\sum_{j=1}^{K}\gamma_j^u))\right) \\
&- \sum_{u=1}^{N}\sum_{y=1}^{M}\sum_{i=1}^{K}\phi_{iy}^{u}\log\phi_{iy}^{u} \qquad (7.30)
\end{aligned}
$$

7.4.2 Learning Variational Parameters

To find the update for the variational multinomial parameter ϕ_{yi}^u we differentiate $F[\phi, \gamma, \alpha, \beta]$ with respect to ϕ_{yi}^u, and solve the resulting equation. Note that ϕ_y^u is a distribution and we must enforce the normalization constraint $\sum_{i=1}^{K}\phi_{yi}^u = 1$. Intuitively, ϕ_{yi}^u is the probability that user u's rating for item y was generated by user attitude i given the value r_y^u. Note that if r_y^u is unobserved the distribution becomes uniform.

$$
\begin{aligned}
\frac{\partial F[\phi,\gamma,\alpha,\beta]}{\partial\phi_{yi}^u} &= \Psi(\gamma_i^u) - \Psi(\sum_{j=1}^{K}\gamma_j^u) + \sum_{v=1}^{V}\delta(r_y^u, v)\log\beta_{vyi} - 1 - \log\phi_{iy}^u - \lambda \\
\log\phi_{iy}^u &= \Psi(\gamma_i^u) - \Psi(\sum_{j=1}^{K}\gamma_j^u) + \sum_{v=1}^{V}\delta(r_y^u, v)\log\beta_{vyi} - 1 - \lambda \\
\phi_{iy}^u &= \exp(-\lambda)(\exp(\Psi(\gamma_i^u) - \Psi(\sum_{j=1}^{K}\gamma_j^u) + \sum_{v=1}^{V}\delta(r_y^u, v)\log\beta_{vyi} - 1)
\end{aligned}
$$

$$
\begin{aligned}
\exp(\lambda) &= \sum_{i=1}^{K}\exp(\Psi(\gamma_i^u) - \Psi(\sum_{j=1}^{K}\gamma_j^u) + \sum_{v=1}^{V}\delta(r_y^u, v)\log\beta_{vyi} - 1) \\
\lambda &= \log(\sum_{i=1}^{K}\exp(\Psi(\gamma_i^u) - \Psi(\sum_{j=1}^{K}\gamma_j^u) + \sum_{v=1}^{V}\delta(r_y^u, v)\log\beta_{vyi} - 1)
\end{aligned}
$$

$$
\begin{aligned}
\phi_{iy}^u &= \frac{\exp(\Psi(\gamma_i^u) - \Psi(\sum_{j=1}^{K}\gamma_j^u) + \sum_{v=1}^{V}\delta(r_y^u, v)\log\beta_{vyi})}{\sum_{i=1}^{K}\exp(\Psi(\gamma_i^u) - \Psi(\sum_{j=1}^{K}\gamma_j^u) + \sum_{v=1}^{V}\delta(r_y^u, v)\log\beta_{vyi})} \\
&= \frac{\prod_{v=1}^{V}\beta_{vyi}^{\delta(r_y^u, v)}\exp(\Psi(\gamma_i^u) - \Psi(\sum_{j=1}^{K}\gamma_j^u))}{\sum_{i=1}^{K}\prod_{v=1}^{V}\beta_{vyi}^{\delta(r_y^u, v)}\exp(\Psi(\gamma_i^u) - \Psi(\sum_{j=1}^{K}\gamma_j^u))} \qquad (7.31)
\end{aligned}
$$

Input: α, β, $\mathbf{r}$, I
Output: ϕ, γ

$\phi_{ky} \leftarrow 1/K$ for all k, y
$\gamma_k \leftarrow \alpha_k + V/K$ for all k
for $t = 1$ to I **do**
  for $y = 1$ to M, $k = 1$ to K **do**
    $\phi_{yk} \leftarrow \frac{\prod_{v=1}^{V} \beta_{vyk}^{\delta(r_y,v)} \exp(\Psi(\gamma_k) - \Psi(\sum_{j=1}^{K} \gamma_j))}{\sum_{i=1}^{K} \prod_{v=1}^{V} \beta_{vyi}^{\delta(r_y,v)} \exp(\Psi(\gamma_i) - \Psi(\sum_{j=1}^{K} \gamma_j))}$
    $\gamma_k \leftarrow \alpha_k + \sum_{y=1}^{M} \phi_{ky}$
  end for
end for

Algorithm 7.7: URP-VarInf

Next we derive the update rule for the variational Dirichlet parameters γ_i^u. Intuitively γ_i^u are the parameters of a Dirichlet distribution over θ given the observed ratings.

$$\begin{aligned}
\frac{\partial F[\phi, \gamma, \alpha, \beta]}{\partial \gamma_i^u} &= \alpha_i(\Psi'(\gamma_i^u) - \Psi'(\sum_{j=1}^{K} \gamma_j^u)) + \sum_{y=1}^{M} \phi_{iy}^u(\Psi'(\gamma_i^u) - \Psi'(\sum_{j=1}^{K} \gamma_j^u)) \\
&\quad -\gamma_i^u(\Psi'(\gamma_i^u) - \Psi'(\sum_{j=1}^{K} \gamma_j^u)) = 0 \\
\gamma_i^u &= \alpha_i + \sum_{y=1}^{M} \phi_{iy}^u \qquad (7.32)
\end{aligned}$$

By iterating the updates derived in equations 7.31 and 7.32 we are guaranteed to reach a local maximum of the free energy function $F[\phi, \gamma, \alpha, \beta]$ for fixed α, and β. This iterative procedure defines a variational inference algorithm for the URP model as shown in algorithm 7.7.

7.4.3 Learning Model Parameters

In this sub-section we derive estimates for the URP model parameters α and β. β_{yz} represents the parameters of a multinomial distribution so we must ensure that $\sum_{v=1}^{V} \beta_{vyz} = 1$. While β, and both variational parameters admit closed form solutions, α does not. We begin by deriving the updates for the β parameters, as the α parameters will require

special treatment.

$$\begin{aligned}
\frac{\partial F[\phi,\theta,\beta]}{\partial \beta_{vyz}} &= \sum_{u=1}^{N} \delta(r_y^u, v)\frac{\phi_{yz}^u}{\beta_{yz}} - \lambda = 0 \qquad (7.33)\\
\beta_{vyz} &= \frac{1}{\lambda}\sum_{u=1}^{N} \delta(r_y^u, v)\phi_{yz}^u\\
\sum_{v'=1}^{V} \beta_{v'yz} &= \frac{1}{\lambda}\sum_{v'=1}^{V}\sum_{u=1}^{N} \delta(r_y^u, v')\phi_{yz}^u\\
\lambda &= \sum_{v'=1}^{V}\sum_{u=1}^{N} \delta(r_y^u, v')\phi_{yz}^u\\
\beta_{vyz} &= \frac{\sum_{u=1}^{N} \delta(r_y^u, v)\phi_{yz}^u}{\sum_{v'=1}^{V}\sum_{u=1}^{N} \delta(r_y^u, v')\phi_{yz}^u} \qquad (7.34)
\end{aligned}$$

As we will see, the α_i are unfortunately coupled together and do not yield an analytic form for the maximizer. However, Minka has proposed two iterative methods for estimating a Dirichlet distribution from probability vectors that can be used here. We give Minka's fixed-point iteration, which yields very similar results compared to the alternative Newton iteration. Details for both procedures including the derivation for the inversion of the psi function may be found in [40].

$$\begin{aligned}
\frac{\partial F[\phi,\theta,\beta]}{\partial \alpha_i} &= N(\Psi(\sum_{i=1}^{k}\alpha_j) - \Psi(\alpha_i)) + \sum_{u=1}^{N}\Psi(\gamma_i^u) - \Psi(\sum_{j=1}^{K}\gamma_j^u)\\
\Psi(\alpha_i) &= \Psi(\sum_{i=1}^{k}\alpha_j) + 1/N(\sum_{u=1}^{N}\Psi(\gamma_i^u) - \Psi(\sum_{j=1}^{K}\gamma_j^u))\\
\alpha_i &= \Psi^{-1}\left(\Psi(\sum_{i=1}^{k}\alpha_j) + 1/N(\sum_{u=1}^{N}\Psi(\gamma_i^u) - \Psi(\sum_{j=1}^{K}\gamma_j^u))\right) \qquad (7.35)
\end{aligned}$$

The fixed point iteration for computing α_i in itself is quite simple, but it requires a method for inverting the psi function. This can also be done iteratively using a fixed point iteration also due to Minka [40]. Minka claims the inner loop of the iteration requires just five iterations to reach fourteen decimal places of accuracy if the proposed initialization scheme is used. We give the complete procedure for updating α in algorithm 7.8.

We give a variational EM procedure for model fitting based on the updates derived for model parameters α, β, and the variational parameters γ, ϕ. We also discuss an initialization method that has proved to be very effective for the URP model.

Input: α,γ
Output: α

while (α Not Converged) **do**
  for $i = 1$ to K **do**
    $x_i = \Psi(\sum_{i=1}^{k} \alpha_j) + 1/N(\sum_{u=1}^{N} \Psi(\gamma_i^u) - \Psi(\sum_{j=1}^{K} \gamma_j^u))$
    $\alpha_i \leftarrow \begin{cases} \exp(x_i) + 1/2 & \text{if } y \geq -2.22 \\ \frac{-1}{x_i - \Psi(1)} & \text{if } y < -2.22 \end{cases}$
    while α_i Not Converged **do**
      $\alpha_i \leftarrow \alpha_i - \frac{\Psi(\alpha_i) - x_i}{\Psi'(\alpha_i)}$
    end while
  end for
end while

Algorithm 7.8: URP-AlphaUpdate

Since each iteration of the variational inference procedure increases the total free energy $F[\phi, \gamma, \alpha, \beta]$, any number of variational inference steps I can be used and overall convergence is guaranteed. Iterating to convergence corresponds to performing a full variational E-Step. This is the approach used by Blei et all to fit the LDA model [7]. On the other hand, a single step of variational inference can be used. This is the approach adopted by Buntine to fit the Multinomial PCA (mPCA) model , a slight generalization of LDA [11]. A further refinement is to allow the number of steps of variational inference to vary for each user by defining a heuristic function that may depend on the user, $H(u)$. Empirically, we have found that a simple function of the number of observed ratings in the user profile such as $H(u) = \lfloor(|\mathbf{r}^u| + 4)/4\rfloor$ provides a good heuristic. Note that while in theory any choice of $H(u)$ is valid since we are guaranteed to increase the free energy, different choices lead to model parameters with different characteristic. The details of the fitting procedure are given in algorithm 7.9.

7.4.4 An Equivalence Between The Aspect Model and URP

Recently Girolami and Kabán have shown an interesting equivalence between the dyadic aspect model and LDA [20]. They show that fitting an LDA model with a uniform

Input: $\{\mathbf{r}^u\}$, K
Output: α, β

Initialize α, β
while ($F[\phi, \gamma, \alpha, \beta]$ Not Converged) **do**
  for $u = 1$ to N **do**
    $[\phi^u, \gamma^u] \leftarrow$ URP-VarInf$(\alpha, \beta, \mathbf{r}^u, H(u))$
  end for
  for $v = 0$ to V, $y = 1$ to M, $z = 1$ to K **do**
    $\beta_{vyz} \leftarrow \frac{\sum_{u=1}^{N} \delta(r_y^u, v)\phi_{yz}^u}{\sum_{v'=1}^{V}\sum_{u=1}^{N} \delta(r_y^u, v')\phi_{yz}^u}$
  end for
  $\alpha \leftarrow$ URP-AlphaUpdate(α, γ)
end while

Algorithm 7.9: URP-Learn

Dirichlet prior using a mixed maximum a posteriori and maximum likelihood fitting procedure is equivalent to fitting the dyadic aspect model using maximum likelihood. Essentially the same relationship holds between the vector aspect model and the URP model.

In the URP model exact inference is not possible so approximate inference methods must be used for model fitting. As with LDA, an alternative to the variational inference method we have adopted is to compute a MAP estimate of θ for each user. The MAP estimate θ_{MAP}^a for a particular user a maximizes the posterior probability $P(\theta^a|\mathbf{r}^a, \beta, \alpha)$. If we assume a uniform Dirichlet prior $\alpha_k = 1$ for all k, Then $\theta_{MAP}^a = \theta_{ML}^a = \arg\max_{\theta^a} P(\mathbf{r}^a|\theta^u, \beta) = \arg\max_{\theta^a} \prod_{u=1}^{N} P(\mathbf{r}^u|\theta^u, \beta)$. Given the value of θ_{ML}^u for all users u we may then obtain a maximum likelihood estimate of β by computing $\beta_{ML} = \arg\max_{\beta} \prod_{u=1}^{N} P(\mathbf{r}^u|\theta_{ML}^u, \beta)$. Now we expand the quantity $\prod_{u=1}^{N} P(\mathbf{r}^u|\theta^u, \beta)$ under the URP model as seen in equation 7.36.

$$P(\mathbf{r}^u|\theta, \beta) = \prod_{u=1}^{N}\prod_{y=1}^{M}\prod_{v=1}^{V}(\sum_{z=1}^{K} P(Z_y = z|\theta^u)P(R_y = r_y^u|Z = z, \beta))^{\delta(r_y^u, v)} \tag{7.36}$$

This quantity is exactly the likelihood of a complete set of user profiles under the vector aspect model as given in equation 7.15. Thus, iterating the maximization for θ_{ML}

and β_{ML} results in a set of maximum likelihood parameters for the vector aspect model. This demonstraits that the maximum likelihood procedure used to fit the vector aspect model can be interpreted as a mixed maximum a posteriori and maximum likelihood fitting procedure for the URP model which enforces a uniform dirichlet prior.

Since the fitting procedure for the vector aspect model can be seen as a lower quality approximation than the variational methods we have adopted to fit the URP model, we expect the variational methods to result in more accurate rating prediction results. Similarly, we might expect a model fitting procedure based on expectation propagation to result in more accurate rating prediction results than our variational methods [41].

7.4.5 URP Rating Prediction

Computing the distribution over rating values for a particular unrated item given a user profile $\mathbf{r}^u$ requires applying variational inference. For rating prediction we generally run variational inference for a fixed number of steps since even a crude approximation of the distribution over rating values can lead to correct predictions. The rating distribution equation for the URP model is shown in equation 7.37.

$$p(R_y = v|\mathbf{r}^u) = \int_\theta \sum_{z=1}^{K} p(R_y = v|Z_y = z)p(z|\theta)p(\theta|\mathbf{r}^u)d\theta \tag{7.37}$$

This quantity may look quite difficult to compute, but by interchanging the sum and integral, and appealing to our variational approximation $q(\theta|\gamma^u) \approx p(\theta|u)$ we obtain an expression in terms of the model and variational parameters.

$$\begin{aligned} p(R_y = v|\mathbf{r}^u) &= \sum_{z=1}^{K} p(R_y = v|Z_y = z) \int_\theta p(Z_y = z|\theta)p(\theta|\mathbf{r}^u)d\theta \\ &\approx \sum_{z=1}^{K} p(R_y = v|Z_y = z) \int_\theta p(Z_y = z|\theta)q(\theta|\gamma^u)d\theta \\ &= \sum_{z=1}^{K} p(R_y = v|Z_y = z)E_{q\theta}[\theta_z] \\ &= \sum_{z=1}^{K} \beta_{ryz} \frac{\gamma_z^u}{\sum_{i=1}^{k} \gamma_i^u} \end{aligned} \tag{7.38}$$

Input: $\mathbf{r}^a$, α, β
Output: $\hat{\mathbf{r}}^a$

$[\phi, \gamma] \leftarrow \text{URP-VarInf}(\alpha, \beta, \mathbf{r}^a, H(a))$
for $y = 1$ to M **do**
  for $v = 1$ to V **do**
    $p_v \leftarrow \sum_{z=1}^{K} \beta_{ryz} \frac{\gamma_z^u}{\sum_{i=1}^{k} \gamma_i^u}$
  end for
  $\hat{r}_y^a \leftarrow \text{median}\, p_v$
end for

Algorithm 7.10: URP-Predict

To compute $p(R_y = v|\mathbf{r}^u)$ according to equation 7.38 given the model parameters α and β, it is necessary to apply the variational inference procedure to compute γ^u. However, this only needs to be done once for any profile $\mathbf{r}^u$ regardless of the number of predictions that need to be calculated.

7.4.6 Complexity

In the E-step of the URP-Learn algorithm consists of running the iterative variational inference algorithm for each user to update the variational parameters. The complexity of the variational inference algorithm is $O(I_1MVK)$ where I_1 is a bound on the number of steps of variational inference taken for each user. The total time complexity of the E-step is thus $O(I_1NMVK)$. The complexity of the E-step dominates the M-step where the model parameters are updated. Letting I_2 be a bound on the number of EM iterations needed to reach convergence, we obtain a total time complexity of $O(I_1I_2NMVK)$. The total space complexity of the learned representation is $O(MVK + K)$.

The URP-Predict algorithm consists of first performing variational inference on the user profile at a cost of $O(I_1MVK)$, and then computing the distribution over ratings for each item at a cost of $O(MVK)$. The total time complexity is thus $O(I_1MVK)$.

7.4.7 Results

Fitting the URP model can be quite difficult starting from randomly initialized α and β parameters. The method we have adopted for initialization is to partially fit a mixture of multinomials model with K user types. A small, fixed number of EM iterations is used. The mixture of multinomials model yields a multinomial distribution β' over ratings conditioned on the item and user type, as well as a single multinomial distribution over user types θ'. To initialize the URP model we set $\beta \leftarrow \beta'$, $\alpha \leftarrow \kappa\theta'$ where κ is a positive constant. Setting $\kappa = 1$ appears to give good results in practice.

The URP-Learn and URP-Predict methods described in algorithms 7.9 and 7.10 were implemented, and tested for their strong and weak generalization performance. We report the results in tables 7.5 and 7.6.

Table 7.5: URP: EachMovie Results

	$K = 5$	$K = 10$	$K = 20$	$K = 30$
Weak	0.4621 ± 0.0016	0.4442 ± 0.0013	0.4422 ± 0.0018	0.4422 ± 0.0008
Strong	0.4755 ± 0.0013	0.4579 ± 0.0006	0.4559 ± 0.0010	0.4557 ± 0.0012

Table 7.6: URP: MovieLens Results

	$K = 4$	$K = 6$	$K = 8$	$K = 10$
Weak	0.4386 ± 0.0044	0.4341 ± 0.0023	0.4402 ± 0.0035	0.4403 ± 0.0019
Strong	0.4476 ± 0.0016	0.4444 ± 0.0032	0.4480 ± 0.0020	0.4535 ± 0.0020

7.5 The Attitude Model

The URP model has consistent generative semantics at both the user level and the profile level. However, the question of whether the generative semantics of URP correspond well to intuition has not been addressed. Recall that in the URP model the latent space representation of a user is a distribution $P(Z|\theta^u)$ over user types. A rating for item y is generated by sampling a particular attitude z, and then sampling a rating from $P(R_y|Z_y = z)$. Thus the user is represented as a distribution over user attitudes, but

a different stochastically selected attitude is used to generate the rating value for each item.

The URP model is descended from the dyadic aspect model, and the stochastic selection of user attitudes corresponds directly to the stochastic selection of topics in the document case. In the dyadic model a document is represented as a distribution over topics $P(Z|\theta)$. A word is generated by stochastically selecting a topic z, and then selecting a word according to the distribution $P(W|Z = z)$. In this case the empirical distribution of words generated for a particular document will reflect the distribution over topics for that document. In the URP model, the distribution over user attitudes will be reflected in the rating profile as a whole, but not in the choice of rating value for any particular item. This is disappointing because intuitively we expect that different user attitudes interact to determine a rating for each item.

The network structure of the attitude model can be seen in figure 7.8. Instead of a distribution over attitude values or user types, the attitude model has a set of marginally independent attitude nodes labeled A_k with a factorial prior distribution parameterized by η as seen in equation 7.39. The latent space description of a user is a vector of attitude expression levels $\mathbf{a}^u = (a_1, ..., a_k)$. Note that this vector does not represent a distribution so the attitudes A_k can take on different expression levels a_k independently of each other.

$$P(\mathbf{A} = \mathbf{a}^u|\eta) = \prod_{k=1}^{K} P(A_k = a_k^u|\eta) \tag{7.39}$$

Each attitude A_k has associated with it a set of real valued preference parameters ρ_{vyk}. These can be thought of as parameters of a distribution over rating values for each item; however, they are not the parameters of an actual multinomial distribution over ratings. We can obtain the parameters of corresponding multinomial distribution over ratings using the softmax function obtaining $P(R_y = v|k) = exp(\rho_{vyk})/\sum_{v'} \rho_{v'yk}$. This choice of parameterization was made because it avoids the use of constrained optimization that is needed to fit multinomial parameters directly. Given a particular expression level a_k for attitude A_k, the distribution over ratings for a particular item y is defined according to

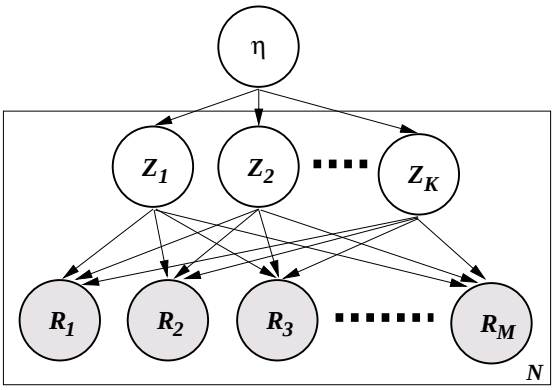

Figure 7.8: The attitude model.

equation 7.40. Given an attitude vector $\mathbf{a}^u$, the probability of rating value v for item y is a product of attitude expression levels and preference parameters as seen in equation 7.41. Given an attitude vector $\mathbf{a}^u$, the probability of a rating profile $\mathbf{r}^u$ with unobserved ratings is shown in equation 7.42. We define $s_y^u = \sum_{v=1}^{V} \delta(r_y^u, v)$ to simplify the notation. Lastly, the joint probability of observing a user profile $\mathbf{r}^u$ and an attitude vector $\mathbf{a}^u$ is given in equation 7.43

$$P(R_y = v | A_k = a_k^u, \rho) = \frac{\exp(a_k^u \rho_{vyk})}{\sum_{v'=1}^{V} \exp(a_k^u \rho_{v'ik})} \tag{7.40}$$

$$\begin{aligned} P(R_y = v | \mathbf{A} = \mathbf{a}^u, \rho) &= \frac{\prod_{k=1}^{K} P(R_y = v | A_k = a_k^u)}{\sum_{v'=1}^{V} \prod_{k=1}^{K} P(R_y = v' | A_k = a_k^u)} \\ &= \frac{\exp(\sum_{k=1}^{K} a_k^u \rho_{vyk})}{\sum_{v'=1}^{V} \exp(\sum_{k=1}^{K} a_k^u \rho_{v'ik})} \end{aligned} \tag{7.41}$$

$$P(\mathbf{R} = \mathbf{r}^u | A^u, \rho) = \prod_{y=1}^{M} \frac{\prod_{v=1}^{V} \exp(\sum_{k=1}^{K} a_k^u \rho_{vyk})^{\delta(r_y^u, v)}}{\sum_{v'=1}^{V} \exp(\sum_{k=1}^{K} a_k^u \rho_{v'ik})^{s_y^u}} \tag{7.42}$$

$$P(\mathbf{R} = \mathbf{r}^u, A^u | \eta, \rho) = \prod_{k=1}^{K} P(A_k = a_k^u | \eta) \prod_{y=1}^{M} \frac{\prod_{v=1}^{V} \exp(\sum_{k=1}^{K} a_k^u \rho_{vyk})^{\delta(r_y^u, v)}}{\sum_{v'=1}^{V} \exp(\sum_{k=1}^{K} a_k^u \rho_{v'ik})^{s_y^u}} \tag{7.43}$$

This is a novel approach for collaborative filtering, and is quite distinct from the mixture models we have described to this point. As we have mentioned, the URP and aspect models employ stochastic selection of user attitudes to generate ratings, while the attitude model allows every user attitude to influence every rating. In the stochastic selection case we would expect preference patterns to be learned competitively, and

that each user attitude would attempt to explain the ratings for all items. On the other hand, we would expect the preference patterns in the attitude model to be learned co-operatively. A particular user attitude k can have no opinion about a certain item y by learning a set of ρ_{vyk} that are approximately uniform over the range of v. This may allow different attitudes to specialize to a subset of the items, leading to a more efficient use of parameters.

7.5.1 Variational Approximation and Free Energy

As with the URP model, exact inference is intractable due to a sum over all attitude expression vectors of length k. We assume a factorial q-distribution for the hidden attitude nodes as shown in equation 7.44. In the case of attitudes with binary expression levels, $q(A_k = 1|\mu_k^u) = \mu_k^u$ and $q(A_k = 0|\mu_k^u) = 1 - \mu_k^u$. In the case of more general integer-valued attitudes, a Poisson distribution with mean μ_k^u would be appropriate for $q(A_k = a|\mu_k^u)$.

$$q(\mathbf{A} = \mathbf{a}^u|\mathbf{R} = \mathbf{r}^u, \mu) = \prod_{k=1}^{K} q(A_k = a_k^u|\mu_k) \tag{7.44}$$

We derive the free energy of the attitude model $F[\mu, \eta, \rho] = E_q[\log P(\mathbf{r}, \mathbf{a}|\eta, \rho)] + H[q(\mathbf{a}|\mathbf{r}, \mu)]$ without assuming particular forms for the distributions $P(A_k|\eta_k)$ and $q(A_k|\mu_k)$. Later this will allow us to easily derive learning algorithms for an attitude model with binary expression levels, as well as an attitude model with integer valued expression levels.

Expanding the first term of the free energy and then taking the expectation with respect to the q-distribution yields equation 7.45.

$$\begin{aligned}
\log P(\mathbf{r}, \mathbf{a}|\eta, \rho) &= \sum_{u=1}^{N} (\log P(\mathbf{r}^u|\mathbf{a}^u, \rho) + \log P(\mathbf{a}^u|\theta)) \\
&= \sum_{u=1}^{N}\sum_{y=1}^{M}\sum_{v=1}^{V} \delta(r_y^u, v) \sum_{k=1}^{K} a_k^u \rho_{vyk} - \sum_{u=1}^{N}\sum_{y=1}^{M} s_y^u \log\left(\sum_{v=1}^{V} \exp(\sum_{k=1}^{K} a_k^u \rho_{vyk})\right) \\
&\quad + \sum_{u=1}^{K}\sum_{k=1}^{K} \log P(A_k = a_k^u|\eta_k)
\end{aligned}$$

$$
\begin{aligned}
E_q[\log P(\mathbf{r}, \mathbf{a}|\eta, \rho)] &= \sum_{u=1}^{N}\sum_{y=1}^{M}\sum_{v=1}^{V}\delta(r_y^u, v)\sum_{k=1}^{K}E_q[a_k^u \rho_{vyk}] \\
&\quad - \sum_{u=1}^{N}\sum_{y=1}^{M} s_y^u E_q\left[\log\left(\sum_{v=1}^{V}\exp(\sum_{k=1}^{K}a_k^u\rho_{vyk})\right)\right] \\
&\quad + \sum_{u=1}^{N}\sum_{k=1}^{K}E_q[\log P(A_k^u = a_k^u|\eta_k)] \qquad (7.45)
\end{aligned}
$$

To complete the derivation of the free energy we expand the entropy term obtaining equation 7.46.

$$
\begin{aligned}
\log q(\mathbf{a}|\eta) &= \sum_{k=1}^{K}\log q(A_k = a_k^u|\mu_k^u) \\
E_q[\log q(A^u|R^u, \theta)] &= \sum_{k=1}^{K}E_q[\log q(A_k^u = a_k|\mu_k^u)] \qquad (7.46)
\end{aligned}
$$

Combining these quantities we obtain the general form of the attitude model free energy function as seen in equation 7.47.

$$
\begin{aligned}
F[\mu, \eta, \rho] &= \sum_{u=1}^{N}\sum_{y=1}^{M}\sum_{v=1}^{V}\delta(r_y^u, v)\sum_{k=1}^{K}E_q[a_k^u\rho_{vyk}] - \sum_{u=1}^{N}\sum_{y=1}^{M}s_y^u E_q\left[\log\left(\sum_{v=1}^{V}\exp(\sum_{k=1}^{K}a_k^u\rho_{vyk})\right)\right] \\
&\quad + \sum_{u=1}^{N}\sum_{k=1}^{K}E_q[\log P(A_k^u = a_k^u|\eta_k)] - \sum_{u=1}^{N}\sum_{k=1}^{K}E_q[\log q(A_k^u = a_k^u|\mu_k^u)] \qquad (7.47)
\end{aligned}
$$

We obtain lower bound on the free energy $\tilde{F}[\mu, \eta, \rho] \leq F[\mu, \eta, \rho]$ which is easier to work with by applying Jensen's inequality to the second term in 7.47. A further simplification is obtained by noting that the A_k are marginally independent and thus the expected value of their product is equal to the product of their expected values.

$$
\begin{aligned}
\tilde{F}[\mu, \eta, \rho] &= \sum_{u=1}^{N}\sum_{y=1}^{M}\sum_{v=1}^{V}\delta(r_y^u, v)\sum_{k=1}^{K}\mu_k^u\rho_{vyk} - \sum_{u=1}^{N}\sum_{y=1}^{M}s_y^u\log\sum_{v=1}^{V}\prod_{k=1}^{K}E_q[\exp(A_k^u\rho_{vyk})] \\
&\quad + \sum_{u=1}^{N}\sum_{k=1}^{K}E_q[\log P(A_k^u = a_k^u|\eta_k)] - \sum_{u=1}^{N}\sum_{k=1}^{K}E_q[\log q(A_k^u = a_k^u|\mu_k^u)] \qquad (7.48)
\end{aligned}
$$

7.5.2 Learning

We begin the development of a learning algorithm for the attitude model by finding the derivatives of the bound on the free energy $\tilde{F}[\mu, \eta, \rho]$ with respect to the model

parameters ρ_{vyk} and η_k, as well as the variational parameters μ_k^u. We will later use the general derivatives found here to develop model fitting procedures for both binary and integer attitude models. We define intermediate variables γ_{vyk}^u and α_{vy}^u to simplify the derivative equations:

$$\gamma_{vyk}^u = E_q[\exp(a_k^u \rho_{vyk})] \qquad \alpha_{vy}^u = \prod_{k=1}^{K} \gamma_{vyk}^u$$

$$\begin{aligned}
\frac{\partial \tilde{F}[\mu,\eta,\rho]}{\partial \mu_k^u} &= \sum_{y=1}^{M}\sum_{v=1}^{V} \delta(r_y^u, v)\rho_{vyk} - \frac{\partial}{\partial \mu_k^u}(E_q[\log q(a_k^u|\mu_k^u)] + E_q[\log P(a_k^u|\eta_k)]) \\
&\quad - \sum_{y=1}^{M} s_y^u \frac{\sum_{v=1}^{V} \frac{\partial}{\partial \mu_k^u}(E_q[\exp(a_k^u \rho_{vyk})])\prod_{i\neq k}^{K} E_q[\exp(a_y^u \rho_{vyi})]}{\sum_{v=1}^{V}\prod_i^K E_q[\exp(a_k^u \rho_{vyi})]} \\
&= \sum_{y=1}^{M}\sum_{v=1}^{V} \delta(r_y^u, v)\rho_{vyk} - \frac{\partial}{\partial \mu_k^u}(E_q[\log q(a_k^u|\mu_k^u)] - E_q[\log P(a_k^u|\theta)]) \\
&\quad - \sum_{y=1}^{M} s_y^u (\sum_{v'=1}^{V} \frac{\partial \gamma_{v'yk}^u}{\partial \mu_k^u}\frac{\alpha_{vy}}{\gamma_{vyk}^u})/(\sum_{v'=1}^{V}\alpha_{v'y})
\end{aligned} \tag{7.49}$$

$$\begin{aligned}
\frac{\partial \tilde{F}[\mu,\eta,\rho]}{\partial \rho_{vyk}} &= \sum_{u=1}^{N} \delta(r_y^u, v)\mu_k^u - \sum_{u=1}^{N} s_y^u \frac{\frac{\partial}{\partial \rho_{vyk}}(E_q[\exp(a_k^u \rho_{vyk})])\prod_{i\neq k}^{K} E_q[\exp(a_i^u \rho_{vyi})]}{\sum_{v'=1}^{V}\prod_i^K E_q[\exp(a_i^u \rho_{v'yi})]} \\
&= \sum_{u=1}^{N} \delta(r_y^u, v)\mu_k^u - \sum_{u=1}^{N} s_k^u \frac{\frac{\partial \gamma_{vyk}^u}{\partial \rho_{vyk}}}{\gamma_{vyk}} \frac{\alpha_{vy}^u}{\sum_{v'=1}^{V}\alpha_{vy}^u}
\end{aligned} \tag{7.50}$$

$$\frac{\partial \tilde{F}[\mu,\eta,\rho]}{\partial \eta_k} = \sum_{u=1}^{N} \frac{\partial}{\partial \eta_l} E_q[\log P(a_k^u|\eta_k)] \tag{7.51}$$

Learning now reduces to the problem of performing gradient ascent to maximize the free energy of the model $\tilde{F}[\mu,\eta,\rho]$. As we will see in subsequent sections, some care must be taken in maximizing the free energy because some of the variational and model parameters are constrained under different distributions. Learning will thus require the use of an iterative, constrained optimization procedure.

7.5.3 Rating Prediction

Prediction with the attitude model is much more computationally intensive than in the other models studies to this point, including the URP model. In this subsection we

define the general prediction equation for the attitude model, and explain how it can be approximated. In later subsections we show how the prediction equation specializes to the binary and integer attitude models. The probability that a given item y will have rating v given a user profile $\mathbf{r}^u$ is shown in equation 7.52

$$\begin{aligned} P(R_y = v|\mathbf{R} = \mathbf{r}^u) &= \sum_{\mathbf{a}} P(R_y = v|\mathbf{A} = \mathbf{a}) P(\mathbf{A} = \mathbf{a}|\mathbf{R} = \mathbf{r}^u) \\ &\approx \sum_{\mathbf{a}} \frac{\exp(\sum_{k=1}^{K} a_k \rho_{vyk})}{\sum_{v'=1}^{V} \exp(\sum_{k=1}^{K} a_k \rho_{vyk})} q(\mathbf{A} = \mathbf{a}|\mathbf{R} = \mathbf{r}^u) \\ &= \sum_{\mathbf{a}} \frac{\exp(\sum_{k=1}^{K} a_k \rho_{vyk})}{\sum_{v'=1}^{V} exp(\sum_{k=1}^{K} a_k \rho_{v'yk})} \prod_{k=1}^{K} P(A_k = a_k|\mu_k^u) \end{aligned} \quad (7.52)$$

It is important to note two facts about the expression given above. First, we have appealed to the variational approximation $P(\mathbf{A} = \mathbf{a}|\mathbf{R} = \mathbf{r}^u) \approx q(\mathbf{A} = \mathbf{a}|\mu)$. This means that to compute $P(R_y = v|\mathbf{R} = \mathbf{r}^u)$, we must first perform variational inference to obtain estimates of the variational parameters μ_k^u. However, this only needs to be done once in order to make predictions for all unknown items. Second, the sum over $\mathbf{a}$ is a sum over all possible attitude vectors of length K. In the binary case the number of attitude vectors is exponential in the number of attitudes, K. In the integer case the number of possible attitude vectors is infinite. To preserve computational tractability in either case requires that the true value of $P(R_y = v|\mathbf{R} = \mathbf{r}^u)$ be approximated by sampling a relatively small number of attitudes vectors $\mathbf{a}$.

7.5.4 Binary Attitude Model

In this subsection we derive a model fitting procedure assuming binary attitude expression. In this case the prior distribution on the A_k is Bernoulli. We have $P(A_k = 1) = \eta_k$ and $P(A_k = 0) = 1 - \eta_k$. We defines the q-distribution to be Bernoulli as well. The q-distribution has the form $q(A_k = 1|\mu_k^u) = \mu_k^u$ and $q(A_k^u = 0|\mu_k^u) = 1 - \mu_k^u$. The intermediate variables γ_{vyk}^u and α_{vy}^u as well as their derivatives are shown below.

$$\gamma_{vyk}^u = E_q[\exp(a_k^u \rho_{vyk})] = \mu_k^u \exp(\rho_{vyk}) + 1 - \mu_k^u$$

$$\begin{aligned}
\alpha^u_{vy} &= \prod_{k=1}^{K} \gamma^u_{vik} = \prod_{k=1}^{K} \mu^u_k \exp(\rho_{vyk}) + 1 - \mu^u_k \\
\frac{\partial \gamma^u_{vyk}}{\partial \mu_k} &= \exp(\rho_{vyk}) - 1 \\
\frac{\partial \gamma^u_{vyk}}{\partial \rho_{vyk}} &= \mu^u_k \exp(\rho_{vyk})
\end{aligned}$$

To derive a variational learning algorithm for the binary attitude model we must find the gradient of $\tilde{F}[\mu, \eta, \rho]$ with respect to the model parameters ρ_{vyk} and η_k, as well as the variational parameters μ^u_k. This is easily accomplished by substituting the values of the intermediate variables and their derivatives into equations 7.49 to 7.51.

$$\begin{aligned}
\frac{\partial \tilde{F}^B[\mu, \eta, \rho]}{\partial \mu^u_k} &= \sum_{y=1}^{M} \sum_{v=1}^{V} \delta(r^u_y, v) \rho_{vyk} + \log(\eta_k) - \log(1 - \eta_k) - \log(\mu^u_k) + \log(1 - \mu^u_k) \\
&\quad - \sum_{y=1}^{M} s^u_y \Big(\sum_{v'=1}^{V} \alpha_{vy} \frac{(\exp(\rho_{vyk}) - 1)}{\mu^u_k (\exp(\rho_{vyk}) - 1) + 1} \Big/ \Big(\sum_{v=1}^{V} \alpha_{vy} \Big) \qquad (7.53)
\end{aligned}$$

$$\frac{\partial \tilde{F}^B[\mu, \eta, \rho]}{\partial \rho_{vyk}} = \sum_{u=1}^{N} \delta(r^u_y, v) \mu^u_k - \sum_{u=1}^{N} s^u_k \frac{\mu^u_k \exp(\rho_{vyk})}{\mu^u_k (\exp(\rho_{vyk}) - 1) + 1} \frac{\alpha^u_{vy}}{\sum_{v'=1}^{V} \alpha^u_{vy}} \qquad (7.54)$$

$$\frac{\partial \tilde{F}[\mu, \eta, \rho]}{\partial \eta_k} = \eta_k - \frac{1}{N} \sum_{u=1}^{N} \mu^u_k \qquad (7.55)$$

As we noted previously analytical updates for ρ_{vyk}, and μ^u_k can not be found due to coupling of parameters in their respective gradient equations. We must thus resort to iterative optimization techniques for learning. In the binary attitude model the ρ_{vyk} parameters are unconstrained, but μ^u_k parameters represent Bernoulli probabilities and are constrained to lie within the interval $[0, 1]$.

A number of optimization methods exist for iteratively solving box constrained optimization problems. However, since the number of users in a collaborative filtering data set ranges from tens of thousands to hundreds of thousands and the number of attitude variables may be on the order of hundreds, clearly any method relying on second derivatives will be computationally intractable. Two methods that rely only on first order derivatives are the log-barrier method, and the projected gradient method [4, p. 76].

Input: η, ρ, $\mathbf{r}$, I
Output: μ

Initialize μ. $\xi \leftarrow 1$. Compute s.
for $t = 1$ to I **do**
  for $k = 1$ to K **do**
    $d_k \leftarrow \sum_{y=1}^{M} \sum_{v=1}^{V} \delta(r_y, v)\rho_{vyk} + \log(\eta_k) - \log(1 - \eta_k) - \log(\mu_k)$
      $+ \log(1 - \mu_k) - \sum_{y=1}^{M} s_y(\sum_{v'=1}^{V} \alpha_{vy} \frac{(\exp(\rho_{vyk})-1)}{\mu_k(\exp(\rho_{vyk})-1)+1} / (\sum_{v=1}^{V} \alpha_{vy})$
    while $(\tilde{F}_u^B[\mathcal{P}(\mu - \xi d), \eta, \rho] > \tilde{F}_u^B[\mu, \eta, \rho])$ **do**
      $\xi \leftarrow \kappa\xi$
    end while
    $\mu \leftarrow \mathcal{P}(\mu - \xi d)$
  end for
end for

Algorithm 7.11: AttBin-VarInf

The log-barrier method is well known to exhibit extremely slow convergence in most cases. The projected gradient method is a modification of regular gradient descent. The method has an extremely simply form for problems where each variable x_i is constrained to lie in the interval $[lb_i, ub_i]$. In this case The projected gradient method replaces the standard gradient descent step $x^{t+1} = x^t - \xi^t \nabla f(x^t)$ with the projected gradient step $x^{t+1} = \mathcal{P}(x^t - \xi \nabla f(x^t))$ where $\mathcal{P}(x)$ is the projection function. For box constrained problems $\mathcal{P}(x)_i = \text{median}(lb_i, x_i, ub_i)$ [4, p. 92]. To ensure convergence the step size α^t must be chosen by an inexact line search procedure which satisfies sufficient decrease and curvature conditions. A backtracking line search is particularly easy to implement.

We obtain a variational inference procedure by iteratively maximizing $\tilde{F}^B[\mu, \eta, \rho]$ with respect to μ using the projected gradient method. It is important to note that while the μ^u parameters are coupled for each user, they are not coupled *across* users. It is thus natural to define a per user objective function $\tilde{F}_u^B[\mu^u, \eta, \rho]$ such that $\tilde{F}^B[\mu, \eta, \rho] = \sum_{u=1}^{N} \tilde{F}_u^B[\mu^u, \eta, \rho]$. We summarize the resulting variational inference procedure in algorithm 7.11. κ is a parameter that controls the speed of backtracking in the line search. Its value must satisfy $0 < \kappa < 1$

Input: $\{\mathbf{r}^u\}$, K
Output: η, ρ

Initialize η, ρ. $\xi \leftarrow 1$. Compute s.
while ($\tilde{F}^B[\mu, \eta, \rho]$ Not Converged) **do**
  for $u = 1$ to N **do**
    $\mu^u \leftarrow$ AttBin-VarInf$(\eta, \rho, \mathbf{r}, H(u))$
  end for
  for $v = 1$ to V, $y = 1$ to M, $k = 1$ to K **do**
    $d_{vyk} \leftarrow \sum_{u=1}^{N} \delta(r_y^u, v)\mu_k^u - \sum_{u=1}^{N} s_k^u \frac{\mu_k^u \exp(\rho_{vyk})}{\mu_k^u(\exp(\rho_{vyk})-1)+1} \frac{\alpha_{vy}^u}{\sum_{v'=1}^{V} \alpha_{vy}^u}$
  end for
  while ($\tilde{F}^B[\mu, \eta, \rho - \xi d] > \tilde{F}^B[\mu, \eta, \rho]$) **do**
    $\xi \leftarrow \kappa\xi$
  end while
  $\rho \leftarrow \rho - \xi d$
  for $k = 1$ to K **do**
    $\eta_k \leftarrow \frac{1}{N} \sum_{u=1}^{N} \mu_k^u$
  end for
end while

Algorithm 7.12: AttBin-Learn

An iterative procedure for learning the parameters ρ_{vyk} and η_k of the binary attitude model can now be defined. The ρ_{vyk} parameters are unconstrained, so standard gradient descent with line search can be used. The η_k parameters have an analytic update. We give the model fitting procedure in algorithm 7.12. Lastly, we give the prediction equations for the binary attitude model. Recall that in the binary case the q-distribution is given by $q(A_k = a_k|\mu_k^u) = a_k\mu_k^u + (1 - a_k)(1 - \mu_k^u)$. To make predictions for any user we must first apply the variational inference algorithm to compute the values of the variational parameters. As mentioned previously, computing the distribution $P(R_y|\mathbf{R} = \mathbf{r}^u)$ is intractable even when the variational approximation is used. This is because the computation involves a sum over all binary vectors of length K. To overcome this problem, we compute an approximation to the true prediction distribution by sampling attitude vectors according to their probability under the q-distribution. Luckily, the attitudes are marginally independent so sampling an attitude vector reduces to independently sampling

Input: $\mathbf{r}^a$, η, ρ
Output: $\hat{\mathbf{r}}^a$

$\mu \leftarrow$ AttBin-VarInf$(\eta, \rho, \mathbf{r}^a, H(a))$
for $s = 1$ to S **do**
  Sample $\mathbf{a}^s$ from Bernoulli(μ)
end for
for $y = 1$ to M **do**
  for $v = 1$ to V **do**
    $p_v \leftarrow \sum_{s=1}^{S} \frac{\exp(\sum_{k=1}^{K} a_k^s \rho_{vyk})}{\sum_{v'=1}^{V} \exp(\sum_{k=1}^{K} a_k^s \rho_{vyk})} \prod_{k=1}^{K} (a_k^s \mu_k + (1 - a_k^s)(1 - \mu_k))$
  end for
  $p \leftarrow \frac{p}{\sum_{v=1}^{V} p_v}$
  $\hat{r}_y^a \leftarrow \text{median}(p)$
end for

Algorithm 7.13: AttBin-Predict

each component. We give a complete prediction method in algorithm 7.13.

$$\begin{aligned} P(R_y = v|\mathbf{R} = \mathbf{r}^u) &\approx \sum_{\mathbf{a}} \frac{\exp(\sum_{k=1}^{K} a_k \rho_{vyk})}{\sum_{v'=1}^{V} \exp(\sum_{k=1}^{K} a_k \rho_{vyk})} \prod_{k=1}^{K} (a_k \mu_k^u + (1 - a_k)(1 - \mu_k^u)) \\ &\approx \frac{P^s(R_y = v|\mathbf{R} = \mathbf{r}^u)}{\sum_{v'=1}^{V} P^s(R_y = v'|\mathbf{R} = \mathbf{r}^u)} \\ P^s(R_y = v|\mathbf{R} = \mathbf{r}^u) &= \sum_{s=1}^{S} \frac{\exp(\sum_{k=1}^{K} a_k^s \rho_{vyk})}{\sum_{v'=1}^{V} \exp(\sum_{k=1}^{K} a_k^s \rho_{vyk})} \prod_{k=1}^{K} (a_k^s \mu_k^u + (1 - a_k^s)(1 - \mu_k^u)) \end{aligned}$$

7.5.5 Integer Attitude Model

In this section we derive a model fitting procedure assuming integer attitude expression. In this case we assume a Poisson prior distribution on the A_k such that $P(A_k^u = a|\eta) = \exp(-\eta_k)\eta_k^a/a!$, as well as a Poisson q-distribution $q(A_k^u = a|\mu) = \exp(-\mu_k)\mu_k^a/a!$. The intermediate variables γ_{vyk}^u and α_{vy}^u as well as their derivatives are shown below.

$$\begin{aligned} \gamma_{vyk}^u &= \exp[\mu_k^u(\exp(\rho_{vyk}) - 1)] \\ \alpha_{vy}^u &= \prod_{k=1}^{K} \exp[\mu_k^u(\exp(\rho_{vyk}) - 1)] \\ \frac{\partial \gamma_{vyk}^u}{\partial \mu_k} &= \gamma_{vyk}^u(\exp(\rho_{vyk}) - 1) \\ \frac{\partial \gamma_{vyk}^u}{\partial \rho_{vyk}} &= \gamma_{vyk}^u \mu_k^u \exp(\rho_{vyk}) \end{aligned}$$

To derive a variational model fitting algorithm for the integer attitude model we need the gradient of the total free energy with respect to the model parameters ρ_{vyk} and η_k, as well as the variational parameters μ_k^u. Again we simply substitute the values of the intermediate variables and their derivatives into equations 7.49 to 7.51.

$$\frac{\partial \tilde{F}[\mu,\eta,\rho]}{\partial \mu_k^u} = \sum_{y=1}^{M}\sum_{v=1}^{V}\delta(r_y^u,v)\rho_{vyk} + \log(\eta_k) - \log(\mu_k^u) - \sum_{y=1}^{M} s_y^u \frac{\sum_{v=1}^{V}[\exp(\rho_{vyk})-1]\alpha_{vk}^u}{\sum_{v=1}^{V}\alpha_{vk}^u} \tag{7.56}$$

$$\frac{\partial \tilde{F}[\mu,\eta,\rho]}{\partial \rho_{vyk}} = \sum_{u=1}^{N}\delta(r_y^u,v)\mu_k^u - \sum_{u=1}^{N} s_y^u \frac{\mu_k^u \exp(\rho_{vyk})\alpha_{vy}^u}{\sum_v' \alpha_{v'y}^u} \tag{7.57}$$

$$\frac{\partial \tilde{F}[\mu,\eta,\rho]}{\partial \eta_k} = \eta_k - \frac{1}{N}\sum_{u=1}^{N}\mu_k^u \tag{7.58}$$

As in the binary case, analytical updates for ρ_{vyk}, and μ_k^u can not be found due to coupling of parameters in their respective gradient equations. We again resort to iterative optimization techniques for model fitting. In the integer attitude model the μ_l^u parameters represent the mean of a poisson distribution over integer attitude expression levels, and are thus constrained to lie in the interval $[0,\infty)$. This type of optimization problem is also considered to be box constrained, and the projected gradient method described in previous sub-section can be applied without modification. In the Poisson case lb_k^u is set to 0 for all k and u, while ub_k^u is set to ∞. The ρ_{vyk} parameters remain unconstrained and η_k parameters have the same analytical update as in the binary case. We give the variational inference method for the integer attitude model in algorithm 7.14, followed by a model fitting procedure in algorithm 7.15.

Lastly, we give the prediction equations for the integer attitude model. Recall that in the integer case the q-distribution is given by $q(A_k = a_k|\mu_k^u) = \exp(-\mu_k^u)(\mu_k^u)^{a_k}/a_k!$. To make predictions for any user we must first apply the variational inference algorithm to compute the values of the variational parameters. In the integer case the calculation of the prediction distribution $P(R_y|\mathbf{R} = \mathbf{r}^u)$ is clearly intractable because it involves a sum over all integer valued vectors of length K. However, we can still approximate the

Input: η, ρ, $\mathbf{r}$, I
Output: μ

Initialize μ. $\xi \leftarrow 1$. Compute s.
for $t = 1$ to I **do**
  for $k = 1$ to K **do**
    $d_k \leftarrow \sum_{y=1}^{M} \sum_{v=1}^{V} \delta(r_y, v)\rho_{vyk} + \log(\eta_k) - \log(\mu_k)$
      $- \sum_{y=1}^{M} s_y \frac{\sum_{v=1}^{V} [\exp(\rho_{vyk}) - 1]\alpha_{vk}}{\sum_{v=1}^{V} \alpha_{vk}}$
    while $(\tilde{F}_u^I[\mathcal{P}(\mu - \xi d), \eta, \rho] > \tilde{F}_u^I[\mu, \eta, \rho])$ **do**
      $\xi \leftarrow \kappa\xi$
    end while
    $\mu \leftarrow \mathcal{P}(\mu - \xi d)$
  end for
end for

Algorithm 7.14: AttInt-VarInf

prediction distribution. In the integer case the attitudes are marginally independent so sampling an attitude vector reduces to independently sampling each component as in the binary case. Sampling the components is slightly more complicated than in the binary case because the q-distribution is Poisson and not Bernoulli.

$$\begin{aligned} P(R_y = v|\mathbf{R} = \mathbf{r}^u) &\approx \sum_A \frac{\exp(\sum_{k=1}^{K} a_k \rho_{vyk})}{\sum_{v'=1}^{V} \exp(\sum_{k=1}^{K} a_k \rho_{vyk})} \prod_{k=1}^{K} \exp(-\mu_k^u)(\mu_k^u)^{a_k}/a_k! \\ &\approx \frac{P^s(R_y = v|\mathbf{R} = \mathbf{r}^u)}{\sum_{v'=1}^{V} P^s(R_y = v'|\mathbf{R} = \mathbf{r}^u)} \\ P^s(R_y = v|\mathbf{R} = \mathbf{r}^u) &= \sum_{s=1}^{S} \frac{\exp(\sum_{k=1}^{K} a_k^s \rho_{vyk})}{\sum_{v'=1}^{V} \exp(\sum_{k=1}^{K} a_k^s \rho_{vyk})} \prod_{k=1}^{K} \exp(-\mu_k^u)(\mu_k^u)^{a_k^s}/a_k^s! \end{aligned}$$

7.5.6 Complexity

The computational and space complexity of the binary and integer attitude models are asymptotically equivalent. The complexity of computing the per-user free energy function is $O(MKV)$. The backtracking phase of the variational inference method thus has computational complexity $O(I_1 MKV)$ where I_1 is a bound on the number of backtracking iterations needed to ensure a decrease in the free energy function. By preserving the

Input: $\{\mathbf{r}^u\}$, K
Output: η, ρ

Initialize η, ρ. $\xi \leftarrow 1$. Compute s.
while ($\tilde{F}^I[\mu, \eta, \rho]$ Not Converged) **do**
  for $u = 1$ to N **do**
    $\mu^u \leftarrow$ AttInt-VarInf$(\eta, \rho, \mathbf{r}, H(u))$
  end for
  for $v = 1$ to V, $y = 1$ to M, $k = 1$ to K **do**
    $d_{vyk} \leftarrow \sum_{u=1}^{N} \delta(r_y^u, v)\mu_k^u - \sum_{u=1}^{N} s_y^u \frac{\mu_k^u \exp(\rho_{vyk})\alpha_{vy}^u}{\sum_v' \alpha_{v'y}^u}$
  end for
  while ($\tilde{F}^I[\mu, \eta, \rho - \xi d] > \tilde{F}^I[\mu, \eta, \rho]$) **do**
    $\xi \leftarrow \kappa\xi$
  end while
  $\rho \leftarrow \rho - \xi d$
  for $k = 1$ to K **do**
    $\eta_k \leftarrow \frac{1}{N} \sum_{u=1}^{N} \mu_k^u$
  end for
end while

Algorithm 7.15: AttInt-Learn

value of ξ for each user between calls to variational inference method, I_1 is observed to drop to 1 on average as the learning method progresses. The cost of computing the gradient with respect to μ^u is $O(MVK)$. The total complexity of the variational inference method is thus $O(I_2 I_1 MVK)$ where I_2 is a bound on the number of iterations needed for the objective function to converge. In the implementation a hard limit of five steps was imposed to control computation time.

The learning method is also an iterative optimization algorithm. In each step the variational inference algorithm is run for each user to update the μ values at a total cost of $O(I_2 I_1 NMVK)$. The gradient with respect to ρ is also computed at a cost of $O(NMVK)$. If a full line search is used to update ρ as we suggest, the backtracking procedure requires $O(I_3 NMVK)$, where I_3 is a bound on the number of backtracking iterations. The value of I_3 was also observed to go to 1 on average as the number of optimization steps increased. The cost of updating η is negligible at $O(NK)$. The total computational complexity of

Input: $\mathbf{r}^a$, ρ, η
Output: $\hat{\mathbf{r}}^a$

$\mu \leftarrow \text{AttInt-VarInf}(\eta, \rho, \mathbf{r}^a, H(u))$
for $s = 1$ to S **do**
  Sample $\mathbf{a}^s$ from Poisson(μ)
end for
for $y = 1$ to M **do**
  for $v = 1$ to V **do**
    $p_v \leftarrow \sum_{s=1}^{S} \frac{exp(\sum_{k=1}^{K} a_k^s \rho_{vyk})}{\sum_{v'=1}^{V} \exp(\sum_{k=1}^{K} a_k^s \rho_{vyk})} \prod_{k=1}^{K} \exp(-\mu_k)(\mu_k)^{a_k^s}/a_k^s!$
  end for
  $p \leftarrow \frac{p}{\sum_{v=1}^{V} p_v}$
  $\hat{r}_y^a \leftarrow \text{median}\, p$
end for

Algorithm 7.16: AttInt-Predict

the learning algorithm is thus $O(I_4 I_2 I_1 NMVK + I_4 I_3 NMVK)$, where I_4 is a bound on the number of iterations of the learning algorithm needed to obtain convergence of the total free energy function. Again, a hard limit of 50 iterations was imposed to control computation time. The space complexity of the learned representation is similar to the other probabilistic models we have investigated requiring $MVK + K$ parameters.

The prediction method first call the variational inference procedure to compute μ^a for the active user at a cost of $O(I_2 I_1 MVK)$. Next, the rating distributions for each item are computed using S samples at a cost of $SMVK$. This gives a total prediction complexity of $O(I_2 I_1 MVK + SMVK)$.

7.5.7 Results

The AttBin-Learn and AttBin-Predict methods described in algorithms 7.12 and 7.13 were implemented, and tested for both their strong and weak generalization performance. EachMovie was tested using 5, 10, and 20 attitudes. MovieLens was also tested using 5, 10, and 20 attitudes. The results are presented in tables 7.7, and 7.8.

Like with the URP model, the attitude model is very sensitive to the initial values

for the variational parameters μ and the model parameters ρ. Random initialization of these parameters lead to fairly poor results in preliminary testing. An initialization scheme was adopted where the attitude model was fit to a random subset of the training users for several iterations. At the end of the initialization phase the learned ρ values were retained, the μ values were reinitialized randomly for all training users, and the full training phase was started.

Prediction performance is also highly sensitive to the step size parameters used in the line search procedures for optimizing μ. The line search procedures ensure that the free energy decreases monotonically to convergence regardless of the settings of the learning parameters, but some solutions are clearly of higher quality than others. Setting the initial step size too high in the variational inference procedure will cause the μ vectors for many users to immediately jump very close to the extreme values 0 and 1. This tends to lead to parameters with poor predictive ability. A fair amount of experimentation is needed to obtain reasonable initial step sizes for each data set.

Due to the complexity of the attitude model learning procedure, the complete EachMovie training data sets EMWeak1, EMWeak2, and EMWeak3 were not used for experimentation. Instead, a random sample of 5000 users was drawn from each of these data sets, and the model was trained on each random sample. Of course, the final prediction error on the weak data sets was calculated using all training users.

Table 7.7: AttBin-Predict: EachMovie Results

	$K = 5$	$K = 10$	$K = 20$
Weak	0.4803 ± 0.0023	0.4664 ± 0.0030	0.4520 ± 0.0016
Strong	0.4785 ± 0.0024	0.4601 ± 0.0043	0.4550 ± 0.0023

Table 7.8: AttBin-Predict: MovieLens Results

	$K = 5$	$K = 10$	$K = 20$
Weak	0.4384 ± 0.0060	0.4320 ± 0.0055	0.4338 ± 0.0034
Strong	0.4400 ± 0.0086	0.4375 ± 0.0028	0.4400 ± 0.0082

Chapter 8

Comparison of Methods

In this chapter we present a comparison of ratings prediction methods. We have implemented a total of nine methods including a classical Pearson correlation K nearest neighbor regression method (PKNN), a naive Bayes classification method (NBClass), a K-Medians clustering method (K-Medians), the wighted singular value decomposition method (wSVD), a simple multinomial model (Multi), a mixture of multinomials model (MixMulti), the vector aspect model (Aspect), the user rating profile model (URP), and the binary attitude model (Attitude).

For each model we give the computational complexity of the learning and prediction algorithms, as well as the space complexity of the representation learned by each method. We also report the mean weak and strong generalization error rates achieved by each method on each data set. For methods with a complexity parameter K, we report the lowest mean error rate achieved by the settings of K we have tested.

8.1 Complexity

From a strict complexity-based point of view, the most attractive feature of rating prediction method are a non-iterative prediction algorithm, prediction complexity that is independent of the number of users, reasonable learning complexity, and a compact

Table 8.1: Computational Complexity Of Learning and Prediction Methods

Method	Learning	Prediction
PKNN	$O(1)$	$O(NM + N \log N + KM)$
NBClass	$O(NM^2V^2 + M^2 \log M)$	$O(MVK)$
K-Medians	$O(INMK)$	$O(MK)$
wSVD	$O(INM^2 + IM^3)$	$O(IKM)$
Multi	$O(NMV)$	$O(1)$
MixMulti	$O(INMVK)$	$O(MVK)$
Aspect	$O(INMVK)$	$O(MVK)$
URP	$O(I_1I_2NMVK)$	$O(I_1MVK)$
Attitude	$O(I_4I_2I_1NMVK + I_4I_3NMVK)$	$O(I_2I_1MVK + SMVK)$
N: #users M: #items V: #rating values I: #iterations S: #samples K: complexity		

learned representation.

The computational complexity of each learning and rating prediction method is given in table 8.1. The learning method with the lowest computational complexity is the neighborhood method PKNN-Learn. As an instance based method, PKNN-Learn simply stores all training profiles in the learning step. However, the complexity of the PKNN-Predict method is $O(NM+N \log N+KM)$, making it the only method with prediction complexity that scales with the number of users N. On the other hand, the multinomial model learning method Multi-Learn scales linearly with the number of users, but the prediction method Multi-Predict has a computation time of $O(1)$. These two methods illustrate a fundamental tradeoff in computational complexity between the learning algorithm and the prediction algorithm. From a systems standpoint it is much more desirable to tradeoff higher learning time for lower prediction time. Learning can be done offline, while prediction must be done online, and often in real time for Internet-based recommendation services. The remainder of the methods we have studied fall somewhere in between these two. All learning methods other than PKNN-Learn scale linearly with respect to the number of users N, while all prediction methods other than Multi-Predict scale linearly with the number of items M.

The methods which perform clustering, dimensionality reduction, and learn proba-

bilistic latent variable model all require iterative learning procedures. The computational complexity of these methods all depend on the number of iterations needed for an objective function to converge. The K-medians clustering method Kmedians-Learn reliably converged from random initializations in approximately 20 iterations at a cost of $O(NMK)$ per iteration. The MixMulti-Learn and Aspect-Learn algorithms also converged very reliably from random initializations in approximately 30 iterations both at a cost of $O(NMVK)$ per iteration.

Running the URP-Learn algorithm with a randomly initialized set of parameters resulted in extremely unreliable convergence regardless of the number of steps used. However, when initialized with learned parameters from a multinomial mixture model, URP-Learn would reliably converge in approximately 10 iterations. URP-Learn is a doubly iterative algorithm since the variational inference method for the model is itself iterative. While theory dictates that the variational inference algorithm should also be run to convergence, limits proportional to the number of observed ratings in each user profile were imposed.

The two learning methods that exhibited the greatest number of iterations to yield reasonable results were the wSVD-Learn method, and the Attitude-Learn method. In the case of the wSVD-Learn method, convergence is known to be slow in the zero/one weight case. This condition is further aggravated by the extreme sparsity of the rating data. A limit of 100 iterations was imposed to control the total computation time of the experiments; however, the method was still making slow but steady progress at the end of these iterations in all cases. Better results may be obtained using a greater number of iterations, but this simply is not practical due to the $O(NM^2 + M^3)$ computational complexity of each iteration.

The attitude model is triply iterative, which makes it one of the slowest learning methods. A limit of 50 iterations was imposed on the learning method. A limit of 5 iterations was imposed on the variational inference method for each user. The line

Table 8.2: Space Complexity of Learned Representation

Method	Space Complexity of Learned Representation
PKNN	NM
NBClass	$MV^2K + MV$
K-Medians	MK
wSVD	$MK + K$
Multi	MV
MixMulti	$MVK + K$
Aspect	$MVK + NK$
URP	$MVK + K$
Attitude	$MVK + K$
N: #users M: #items V: #rating values K: complexity	

search procedures were permitted to backtrack until the step size fell below 10^{-5}. The method yields excellent performance within these computational limits, but allowing more iterations would likely result in improved performance. We note that a major factor affecting the quality of the solution is the choice of initial step length and backtracking parameters for the line search procedures employed by the method. Small values for both quantities will yield good solutions, but increase time to convergence and the number of backtracking steps. A large initial step length decreases time to convergence, but results in a poor model.

The majority of the prediction methods have fixed computational complexity, except for wSVD-Predict, URP-Predict, and AttBin-Predict. The per-iteration cost of the wSVD method is $O(KM)$, which is fairly low. However, a relatively large number of iterations is needed, and a limit of 100 iterations was imposed. The per-iteration cost of the URP-Predict method is also quite low at $O(MVK)$. A number of iterations proportional to the number of observed ratings was used in practice. The AttBin-Predict algorithm contains both a doubly iterative variational inference step and a sampling step, which make it the slowest prediction method. Even with the largest number of possible binary attitude configurations tested, 2^20, the prediction method yields excellent results based on only 200 samples. This is due to the fact that for most users the mean μ_k^u of the Bernoulli distribution for most attitudes k is close to either 0 or 1.

In terms of space complexity of the learned representation, the multinomial model and the K-Medians cluster prototypes are approximately tied for the most compact learned representation. The two methods with the largest learned representation are the PKNN method and the vector aspect model. The PKNN method takes space proportional to NM, or more correctly, the total number of observed ratings in the training data. This is not surprising since PKNN is an instance based learning method: learning simply consists of storing the train data. On the other hand, the vector aspect model actually learns a representation of the data that grows linearly with the number of user in the training set. This is an artifact of a probabilistic model with a generative process containing incomplete semantics. This particular problem, along with several others, is solved by the URP model which learns a representation of size $MVK + K$.

Only two of the nine methods we have studied satisfy all the complexity-based criteria we have outlined. K-Medians clustering method has a simple, non-iterative prediction method of complexity $O(MK)$, learning complexity of $O(NMK)$, and MK model parameters. The mixture of multinomials method involves calculations with distribution over ratings, so naturally the number of rating values enters into the complexity of the method. The mixture of multinomials prediction method is noniterative due to the fact that inference in the model is simple and exact. The prediction complexity is $O(MVK)$. The complexity of learning is $O(NMVK)$, while the number of parameters in the model is $O(MVK)$.

8.2 Prediction Accuracy

In terms of prediction performance, the goal of any rating prediction method is to obtain the lowest possible prediction error. We have introduced two separate experimental protocols to evaluate prediction performance. The weak generalization protocol tests a method's ability to generalize to new items for the users it was trained on. The strong

Table 8.3: EachMovie: Prediction Results

Method	Weak Comp	Weak NMAE	Strong Comp	Strong NMAE
PKNN	1	0.4886 ± 0.0014	1	0.4933 ± 0.0006
NBClass	20	0.5258 ± 0.0022	100	0.5295 ± 0.0047
K-Medians	20	0.4631 ± 0.0015	20	0.4688 ± 0.0012
wSVD	20	0.4562 ± 0.0032	20	0.4672 ± 0.0012
Multi	1	0.5383 ± 0.0022	1	0.5446 ± 0.0029
MixMulti	30	0.4557 ± 0.0012	30	0.4573 ± 0.0007
Aspect	30	0.4573 ± 0.0007	N/A	N/A
URP	30	0.4422 ± 0.0008	30	0.4557 ± 0.0008
Attitude	20	0.4520 ± 0.0016	20	0.4550 ± 0.0023

Table 8.4: MovieLens: Prediction Results

Method	Weak Comp	Weak NMAE	Strong Comp	Strong NMAE
PKNN	1	0.4539 ± 0.0010	1	0.4621 ± 0.0022
NBClass	1	0.4803 ± 0.0027	10	0.4831 ± 0.0052
K-Medians	5	0.4495 ± 0.0027	20	0.4556 ± 0.0013
wSVD	10	0.4886 ± 0.0065	8	0.4710 ± 0.0042
Multi	1	0.4694 ± 0.0020	1	0.4746 ± 0.0035
MixMulti	4	0.4444 ± 0.0007	6	0.4383 ± 0.0048
Aspect	10	0.4339 ± 0.0023	N/A	N/A
URP	6	0.4341 ± 0.0023	6	0.4444 ± 0.0032
Attitude	10	0.4320 ± 0.0055	10	0.4375 ± 0.0028

generalization protocol tests the ability of a method to generalize to completely novel user profiles.

While the weak generalization protocol has typically been used to evaluate rating prediction methods, the strong generalization results give a better indication of a method's online performance. In an online context it is not practical to re-learn a model whenever a new rating is received, or a new rating profile is created. Instead a model is applied that, having been trained at some time in the past, was trained without complete knowledge of the current set of user profiles. This situation arises often in Internet recommendation services, making strong generalization performance most relevant for that application area.

For methods with a model size parameter K, we have conducted experiments testing a range of values of K. For each method and each setting of the model size parameter, each

experimental procedure was carried out using three partitions of the available data for each data set. We report the lowest normalized mean absolute error rate attained by each method along with the corresponding standard error value (NMAE), and the number of components at which the lowest error rate was obtained (Comp). We present results for the weak and strong generalization performance of each method on the EachMovie data set in table 8.2. We present results for the weak and strong generalization performance of each method on the MovieLens data set in table 8.3. In addition, we present bar charts of weak and strong generalization error in figures 8.1 to 8.4. We have ranked the methods along the horizontal axis from highest to lowest mean NMAE.

The performance of a range Pearson correlation neighborhood methods has long been established. Our version, the PKNN method, consistently obtains better error rates than the multinomial model, but is consistently worse than the specialized probabilistic models and the K-medians method. All of the model-based methods learn representations of the data that are more compact than the instance-based PKNN method, and have prediction complexities not directly dependant on the number of users N.

The naive Bayes classification method is quite clearly a failure. In two cases it performs only slightly better than the baseline multinomial model, and in two case it performs slightly worse. This is an interesting result because if a given user has not rated any of the feature items for a given class item, the naive Bayes model reverts to a multinomial model for that item. Thus in the worst case we should expect the error of the multinomial model to upper bound the error of the naive Bayes classification method. The fact that this doesn't hold indicates that some selected feature items or combinations of feature items actually result in *decreased* performance. The performance of this method could likely be improved by exploring an approach to feature selection that would incrementally select feature items only if they resulted in an increase in prediction performance. However, the added computational cost of such a learning method would probably not be worth the resulting gain in performance.

For a method based on hard clustering, the K-Medians method performs much better than expected. It achieves error rates comparable with some of the specialized probabilistic models using a very reasonable number of components. It also outperforms the PKNN method by a significant amount, while having far lower prediction complexity. While not a top performer, taking the reliability and speed of convergence into account along with its excellent prediction complexity, the K-Medians method is among the top "light-weight" prediction methods we have studied. It is actually quite astonishing that early work on collaborative filtering failed to popularize the use of standard clustering methods for rating prediction given the performance of these methods.

The performance of the wSVD algorithm is quite interesting. On the EachMovie data set, which has a smaller number of items than the MovieLens data set by roughly half, the wSVD method obtains error rates very close to the specialized probabilistic models. It is among the top four methods with respect to both weak generalization error and strong generalization error. On the smaller MovieLens data set wSVD ranks among the bottom three methods. It is the only method which exhibits such a decrease in performance across the two data sets. This would seem to indicate a drop in prediction performance of the wSVD-Learn algorithm as the number of items increases. However, it may also be the case that wSVD actually over fits on the training data due to the lower number of user profiles. The training error rates were observed to be much lower than the test error rates at the end of the wSVD learning procedure, which would tend to support the second view. Further testing could not be carried out due again to the complexity of the method.

The mixture of multinomials method performs better than expected. It ties the Attitude model for best strong generalization performance on the MovieLens data set, and is top three on both EachMovie tasks. The mixture of multinomials model also serves as a very useful method for initializing the URP model, as we have noted previously. The availability of simple, exact inference coupled with good performance makes this model

very well suited for use in an online context.

The vector aspect model achieves very good accuracy on the EachMovie data set and ties with URP and the Attitude model for the best weak generalization performance on the MovieLens data set. The structure and parameterization of the vector aspect model imply that inference can not be performed in a principled manner for new user profiles. However, the interpretation of the vector aspect model fitting procedure in terms of an approximate, restricted fitting procedure for the URP model in fact justifies the use of the heuristic "folding in" procedure for inference with new users [29]. However, this new perspective also makes it clear that the variational approximation method we have proposed for fitting the URP model is more accurate than the mixed MAP/ML approximation. Thus, in terms of prediction accuracy, URP completely superseeds the vector aspect model.

The URP model obtains the lowest weak generalization error on the EachMovie data set, and ties the attitude and multinomial mixture models for lowest strong generalization error on the EachMovie data set. URP also places in the top three on both MovieLens tasks. Overall, the URP model is a close second to the attitude model in terms of prediction performance. However, it attains this performance level at a lower cost in terms of both learning and prediction complexity.

Lastly, the binary attitude model achieves mean error rates lower than all the other models in this study in three out of four experiments. This is an appealing result because the attitude model was designed to have generative semantics that are more realistic than the stochastic selection semantics of the multinomial mixture, aspect, and URP models. In the attitude model, all latent attitudes interact to determine the rating for each user. The more sophisticated generative semantics result in more complex model fitting and prediction procedures, but the result is a model that obtains the best overall performance of any of the models we have studied.

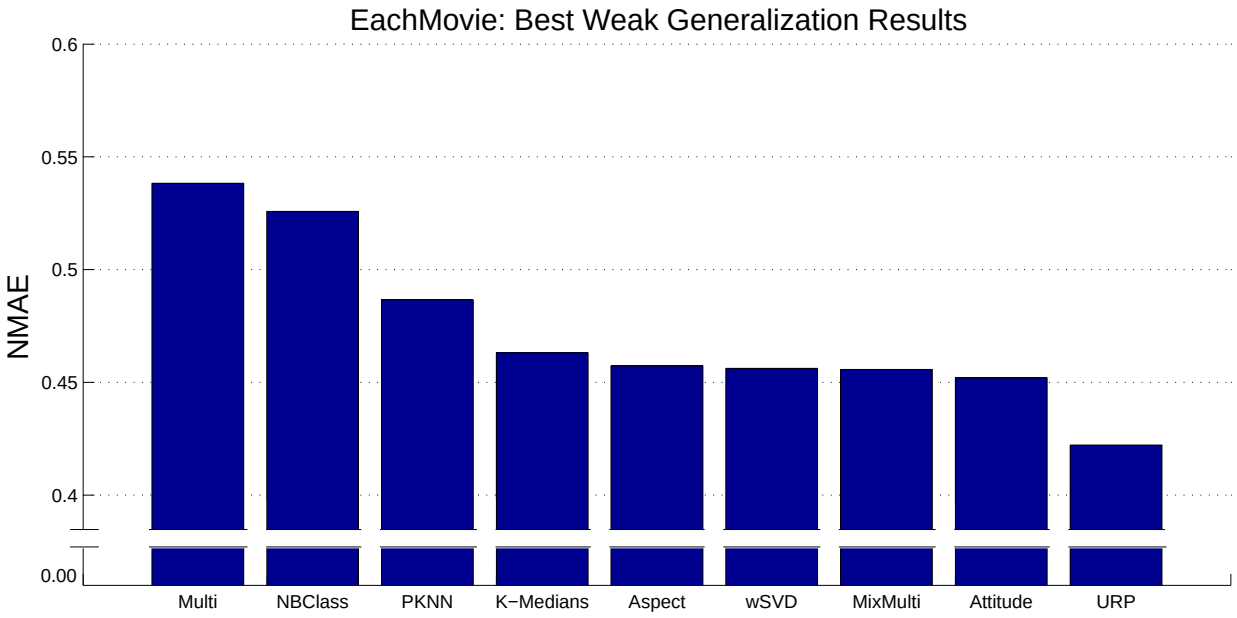

Figure 8.1: Comparison of EachMovie weak generalization performance.

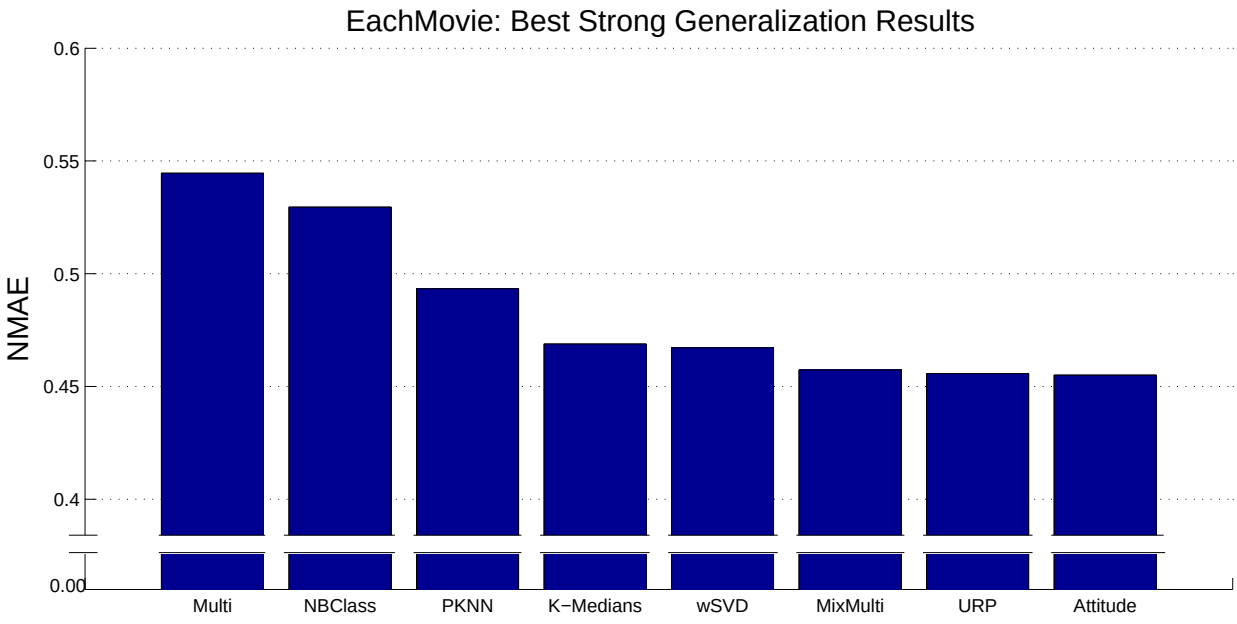

Figure 8.2: Comparison of EachMovie strong generalization performance.

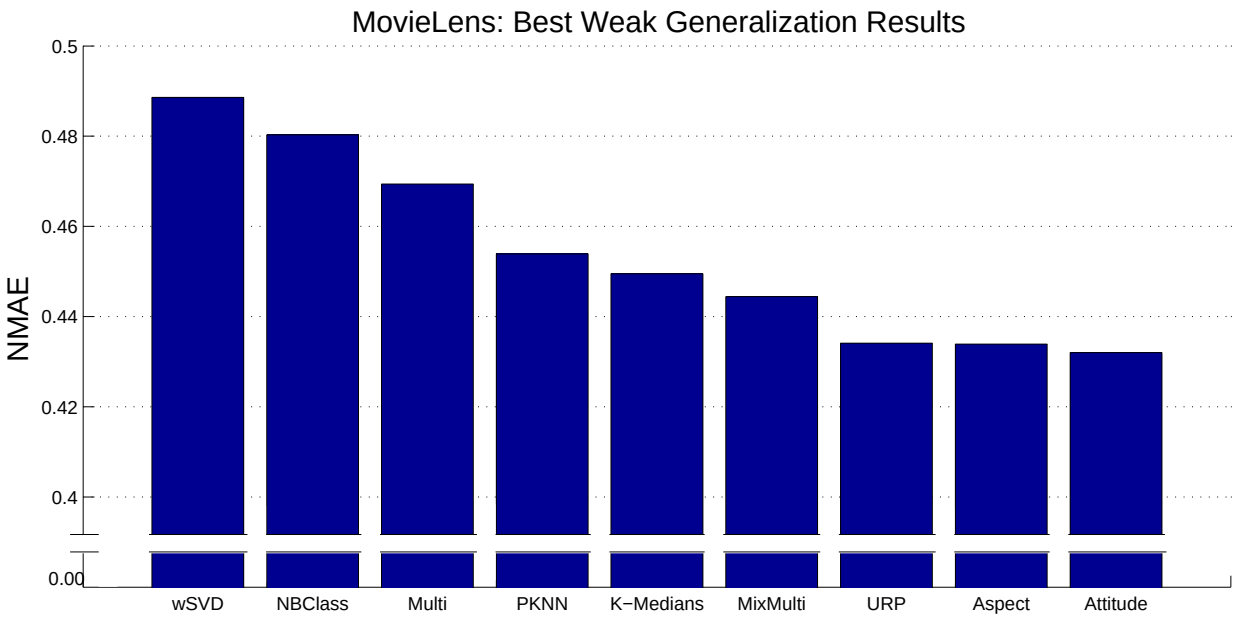

Figure 8.3: Comparison of MovieLens weak generalization performance.

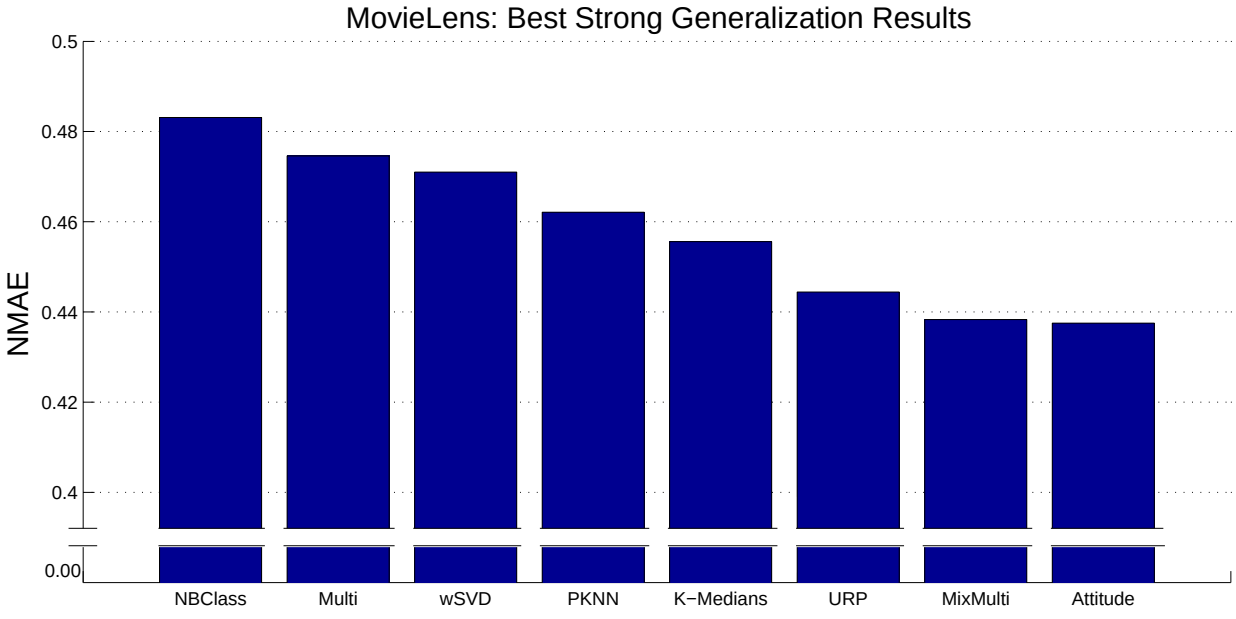

Figure 8.4: Comparison of MovieLens strong generalization performance.

Chapter 9

Conclusions

We begin by summaryzing the work contained in this thesis, outlining the main developments, and indicating the primary results. Next we adopt a broader perspective, analyze what we have accomplished, and indicate interesting directions for future research. Finally, we consider the feasibility of scaling some of the methods we have proposed for use with large electronic document collections such as the Web.

9.1 Summary

In the preceding chapters we have presented a comprehensive study of rating-based, pure, non-sequential collaborative filtering. We have given detailed descriptions and derivations for a variety of methods, shown their relationship to standard machine learning algorithms, analyzed their computational and space requirements.

We show that the original GroupLens method is a modification of the well known K nearest neighbor classifier. We introduce a new application of the standard naive Bayes classifier to the task of rating prediction. We discuss several rating prediction methods based on user clustering and introduce a new method based on the standard K-medians clustering algorithm. We review dimensionality reduction techniques including singular value decomposition, weighted singular value decomposition, principal components anal-

ysis, probabilistic principal components analysis, and factor analysis. We introduce a new rating prediction technique for weighted singular value decomposition. In the area of probabilistic models we review the multinomial model, the mixture of multinomials mode, the aspect model, and the URP model. We also introduce the attitude model family, a new family of product models for collaborative filtering

We implement and analyze learning and prediction methods for a neighborhood method, the naive Bayes classification method, the K-medians clustering method, weighted singular value decomposition, the multinomial model, the mixture of multinomials model, the aspect model, the URP model, and the binary attitude model. We introduce the notion of strong generalization as the ability of a learning method to correctly predict ratings for novel user profiles, profiles other than those used for training. This is an important form of generalization for rating prediction, which is especially relevant when considering the online performance of a prediction method. We introduce an experimental protocol for assessing strong generalization to complement the existing protocols for assessing weak generalization.

The primary result of this study has been the identification of several new and promising rating prediction methods. The K medians clustering method was found to exhibit excellent rating prediction ability given its relative simplicity. It is an excellent candidate for use in an interactive recommendation service. The attitude model family was designed to have intuitive and appealing generative semantics for rating based collaborative filtering. While the binary attitude model exhibits the most complex learning and prediction methods, we have found that it achieves the best overall prediction accuracy of any of the methods we have studied.

9.2 Future Work

In this section we consider possible directions for future work. We discuss the work left to be done with existing models, and consider removing some of the assumptions underlying the development of these models. We discuss extending some of the models we have studied to new formulations of collaborative filtering. Lastly, we indicate other application areas where our current techniques and their extensions may be useful.

9.2.1 Existing Models

We have performed a fairly comprehensive evaluation of a wide range of models; however, some work with existing models remains to be done. One of the main omissions in our set of methods was probabilistic principal components analysis. Weighted singular value decomposition was selected as a representative dimensionality reduction method due to the fact that it had not been extensively tested for collaborative filtering. pPCA is an intersting method in its own right. It has been tested by Canny on protocols and data sets comparable to the EachMovie strong generalization experiment we have used, but the error rate reported is equivalent to the error rate achieved by the multinomial model in our tests. This is curiously poor given that simple clustering methods achieve superior performance in our tests. Experimentation with the pPCA model could be performed to determine if its performance improves on the particular data sets we use, or if it remains the same as reported by Canny.

As we have noted previously, the weighted singular value decomposition method turned out to be fairly enigmatic. It was the only method to exhibit a large change in prediction performance across the EachMovie and MovieLens data sets. This result calls for a separate study to more closely examine the effect of the size and sparsity of the rating matrix on the performance of the method, and the number of iterations needed for the method to converge. The discrepancy in performance could be a result of an insuffi-

cient number of iterations in the MovieLens case, but none of the other methods studied exhibited this problem. In fact, the general trend over all other models is that performance on the MovieLens data set is slightly better than performance on the EachMovie data set. An alternative explanation is that the wSVD method overfits on the training data during model fitting. This is extremely hard to believe when the model consists of as few as 10 item space basis vectors. On the other hand, the feasibility of such an in depth study is somewhat questionable given the high computational complexity if the wSVD learning method. In addition, any improvements that result from further insight into this problem are highly unlikely to result in prediction performance that surpasses the specialized probabilistic models.

The equivalence between the vector aspect model and the URP model is another area we have not fully explored. The MAP/ML fitting procedure for URP represents a significant savings in terms of learning and prediction complexity. If computing time is not an issue, then the variational fitting procedure should always be used in place of the MAP/ML procedure. However, an intersting situation arises when we consider using the URP model in an online context such as an interactive recommender system. The variational model fitting algorithm could be used with a fixed, uniform Dirichlet prior for offline model fitting. For online prediction the MAP inference procedure could be used in place of full variational inference. Such a prediction method would not be iterative, and could be fast enough for interactive use. Experiments similar to the ones we have already carried out could be applied to determine the degree to which the strong generalization performance of this hybrid method degrades compared to the full variational prediction methods.

Our strongly positive results for the binary attitude model call for further testing with it and other members of the attitude model family. In the binary case we were forced to impose hard limits on the number of iterations allowed for model fitting, variational inference, and line search procedures. The only justification offered for the choice of

these limits was that the resulting models exhibited excellent prediction performance. However, a more thorough study of the effect of data set size, data set sparsity, step size, and backtracking on convergence is clearly needed. A better understanding of how all these variables interact will likely lead to model fitting procedures that attain better prediction accuracy than we have reported.

Based on the success of the binary attitude model, an investigation of the integer attitude model is also required. The integer attitude model has more flexible generative semantics and latent user representation than the binary version. This means it has the capacity to outperform the binary version. However, we expect it to suffer from similar learning issues as the binary version including a sensitivity to initial step sizes and backtracking parameters.

9.2.2 The Missing at Random Assumption

One of the most important simplifying assumptions made in this study was that missing ratings were missing at random. This is an assumption which has been made either implicitly or explicitly in all existing research on rating prediction methods. However, we also noted that in the case of rating-based collaborative filtering there is reason to believe that this assumption does not hold. In the standard interaction model with a recommender system, a user may initially be required to rate a subset of a randomly selected set of items. After these initial ratings are collected, the user is free to rate any items they choose. In the case of information items such as movies or books, it is reasonable to hypothesizes that a user will only see a movie or read a book if they anticipate liking it or finding it useful, and that a user can only rate items they have seen or read. This means that the lower a users estimate of their rating for a particular item, the less likely they are to supply a rating for that item. This is exactly the type of scenario where the missing at random assumption fails to hold.

If our hypothesis about this missing data mechanism is correct, one consequence

would be a skewing of the observable rating distribution toward high rating values. This skewing is observed in both the EachMovie and MovieLens data sets. Theory dictates that if the missing at random assumption does not hold, then any maximum likelihood estimates derived from the observed data without taking the missing data mechanism into account will be biased. Since learning in all the models we have described is based on finding maximum likelihood parameters, not taking the missing data mechanism into account may be resultsing in biased parameter estimates.

An interesting test of whether the proposed missing data mechanism does affect learning and prediction would be to generate a dense data set from a learned model such as URP or the attitude model, and then create two training sets from it. One in which ratings are removed uniformly randomly, and a second where ratings are removed inversely proportional to their values. The same learning method could be trained on each of these training data sets, and the resulting models could be tested on the same test set to see if prediction performance is affected by the pattern of missing data.

If a significant effect is observed, this opens the possibility of developing a novel set of models for rating prediction based on incorporating prior knowledge of the missing data mechanism directly into the generative model. An obvious first step in this direction would be to extend the multinomial mixture model to include a missing data mechanism, and evaluate its empirical performance on real data sets compared to the standard multinomial mixture model. It is our belief that such models will result in increased prediction accuracy.

9.2.3 Extensions to Additional Formulations

As we discussed in chapter 2, rating-based, pure, non-sequential collaborative filtering is one out of a large number of possible formulations. Even within this small class of problems we have focused on the case where ratings are ordinal valued and neglected the continuous case. Some of the methods we have studied including the PKNN method, the

K-medians method, and the wSVD method can be applied to either continuous or ordinal rating data, but all the specialized probabilistic models represent ratings using categorical random variables. This means they can not be applied to the case where rating values are continuous. One simple extension of the present work would be to change the lower levels of all probabilistic models to Gaussian distributed continuous random variables and then re-derrive all the model fitting and prediction equations. Hofmann has recently carried out this exercise with the triadic aspect model [30].

A close second to our chosen formulation in terms of research activity is the co-occurrence, pure, non-sequential formulation. We briefly discussed this formulation in conjunction with LDA and the dyadic aspect model, but it is also quit significant. URP already has a co-occurrence analog in LDA, but the attitude model family could easily be extended (or, more accurately, retracted) to this type of preference information as well.

Several other more complex and more interesting formulations are beginning to receive attention from the research community. Of particular interest are sequential formulations, which remove the assumption that preferences are static. Recently Girolami and Kabán have introduced a method for learning dynamic user profiles based on simplicial mixtures of first order Markov chains in a pure, sequential, co-occurrence formulation of collaborative filtering. In theory their model could be extended to dynamic user rating profiles as well, although data for testing such a model is not currently available.

The principled integration of preference information with additional content based features in a single generative model is also an interesting direction for future research. The addition of content-based item features may help alleviate the cold start and new user problems, which are common to all recommender systems. The cold start problem arises when a system is initialized with little preference information. The new user problem occurs in an operational system when a new user has little preference information.

9.2.4 Generalizations to Additional Applications

We have just considered additional formulations of collaborative filtering and how our existing methods could be extended to yield algorithms in some of these formulations. An alternative is to consider generalizations of the rating prediction problem itself. In a pure, non-sequential, rating-based formulation of collaborative filtering, predicting missing ratings for all users can be thought of more generally as an imputation problem. Such problems occur fairly frequently. One particularly interesting example comes from the analysis of gene expression data. Some simple machine learning techniques have been applied for imputing missing data values in microarrays. Continuous valued versions of any of the probabilistic models we have studied could be applied to this problem. Once candidate is a Gaussian version of the binary or integer attitude model, both of which have fairly sophisticated generative semantics. In an application area like computational biology, the complexity of the prediction methods is not as critical as in the collaborative filtering case. In addition, the data is many times less sparse. There is reason to believe that our methods will yield good results in this area.

9.3 The Last Word

We began this thesis by introducing the problem of information overload. We indicated that the major sources of information overload were web pages and Usenet news articles available through the Internet, as well as books, movies and music. We motivated content based information filtering, and collaborative filtering as two independent methods for dealing with information overload.

As a result of the limited data sets currently available, our work has been constrained to the area of movie rating prediction. A fair critique of the present work is that few people tend to feel overwhelmed by the volume of movies being produced. Indeed, the items and users in the data sets we have used only number is in the thousands. While

these movie rating data sets are considered fairly substantial by normal machine learning standards, they pale in comparison with the billions of web pages routinely processed by state-of-the-art content based filtering systems.

Clearly scaling up the size of a collaborative filtering system to rival content based filtering systems such as Google requires some thought. Google indexes on the order of 10^9 web pages. At peak times it serves thousands of queries per second. An equivalent collaborative filtering system might be expected to serve 10^6 users each with an average of 10^3 ratings. For deployment on the Internet we would also want to maintain a document collection of 10^9 web pages and serve 10^3 recommendations per second.

Suppose that we were given computational resources equivalent to those currently used by Google: 10^4 CPU's running at 500MHz each capable of performing 10^8 additions and subtractions per second [1]. The method with the most reasonable computational complexity that we have studied is the K-medians method, which has a learning complexity of $O(INMK)$, and a prediction complexity of $O(MK)$. If implemented using data structures that take sparsity into account, the learning time can be decreased to $O(IKT)$, and the prediction time can be reduced to $O(KT^u)$ where T is the total number of observed ratings and T^u is the total number of observed ratings for user u. A reasonable number of clusters for this problem might be $K = 10^4$. Note that for recommendations, the prototype vectors could be sorted off line after they are learned at a savings of $O(M \log M)$ per recommendation. The complexity of generating each set of recommendations is thus $O(KT^u)$.

Under these assumptions we find that processing each recommendation request would take only $O(10^7)$ additions and subtractions. With a cluster capable of 10^{12} additions and subtractions per second we could serve over 10^4 queries per second. This is quite remarkable. The number of operations needed for each iteration of the K-medians learning algorithm under our assumptions is $O(10^{13})$. Interestingly, the K-medians learning algorithm can be completely parallelized meaning each iteration of the learning method

would take as little as 10 seconds! Even if 10^3 iterations were needed to learn the cluster prototypes, learning would only take a few hours.

These simple order of magnitude computations neglect the time needed to move data around in the system, which is likely to dominate the time needed for computation. However, in a specialized architecture where each of the 10^4 cluster nodes keeps one prototype vector in active memory at a space complexity of about 1GB, we would only need to move the relatively small user profiles in and out of memory. Even if our naive calculations are off by two orders of magnitude it would still be possible to serve 10^3 queries per second and perform over three learning iterations every hour. Learning could be performed in approximately ten days.

The point of this exposition has been to show that some of the collaborative filtering methods we have described, such as K-medians, could scale up to the level of state-of-the-art content-based filtering systems like Google. The only factor obstructing the application of collaborative filtering techniques to large electronic document collections like the Web is the commitment of computational resources. We leave this as an exercise for the interested reader.

Bibliography

[1] Luiz André Barroso, Jeffrey Dean, and Urs Hölzle. Web search for a planet: The Google Cluster Architecture. *IEEE Micro*, 23(2):22–28, March/April 2003.

[2] Chumki Basu, Haym Hirsh, and William W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *AAAI/IAAI*, pages 714–720, 1998.

[3] Pavel Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA, 2002.

[4] D P Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press, New York, 1982.

[5] Daniel Billsus and Michael J. Pazzani. Learning collaborative information filters. In *Proc. 15th International Conf. on Machine Learning*, pages 46–54. Morgan Kaufmann, San Francisco, CA, 1998.

[6] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet Allocation. In *Proceedings of the 14th Conference on Advances in Neural Information Processing Systems (NIPS 14)*, 2002.

[7] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, January 2003.

[8] Hendrik Blockeel, Luc De Raedt, and Jan Ramon. Top-down induction of clustering trees. In J. Shavlik, editor, *Proceedings of the 15th International Conference on Machine Learning*, pages 55–63. Morgan Kaufmann, 1998.

[9] M. E. Brand. Incremental singular value decomposition of uncertain data with missing values. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 2350, pages 707–720, May 2002.

[10] John S. Breese, David Heckerman, and Carl Kadie. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence*, pages 43–52, July 1998.

[11] W. Buntine. Variational extensions to EM and multinomial PCA. In *Proceedings of the European Conference on Machine Learning ECML*, 2002.

[12] Wray Buntine and Sami Perttu. Is Multinomial PCA Multi-faceted Clustering or Dimensionality Reduction? In *9th International Workshop on Artificial Intelligence and Statistics (AISTAT-2003)*, 2003.

[13] John Canny. Collaborative filtering with privacy via factor analysis. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 238–245. ACM Press, 2002.

[14] Sonny Han Seng Chee, Jiawei Han, and Ke Wang. RecTree: An efficient collaborative filtering method. In *Data Warehousing and Knowledge Discovery (DaWaK)*, Munich, Germany, 2001.

[15] Mark Claypool, Phong Le, Makoto Wased, and David Brown. Implicit interest indicators. In *Intelligent User Interfaces*, pages 33–40, 2001.

[16] A.P. Dempster, N.M. Laird, and D.B.Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[17] P. J. Denning. Electronic junk. *Communications of the ACM*, 25(3):163–165, 1982.

[18] Chris Ding and Xiaofeng He. Cluster merging and splitting in hierarchical clustering algorithms. In *IEEE International Conference on Data Mining (ICDM'02)*, Japan, 2002.

[19] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. Wiley-Interscience, New York, New York, 1973.

[20] M. Girolami and A. Kabán. On an equivalence between PLSI and LDA. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval*, pages 433–434, 2003.

[21] Mark Girolami and Ata Kabán. Simplicial mixtures of markov chains: Distributed modelling of dynamic user profiles. In *Proceedings of the Seventeenth Annual Conference on Neural Information Processing Systems (NIPS-2003)*, 2003.

[22] A. Gokhale and M. Claypool. Thresholds for more accurate collaborative filtering. In *In Proceedings of the IASTED International Conference on Artificial Intelligence and Soft Computing*, Honolulu, Hawaii, USA, 1999.

[23] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[24] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval Journal*, 4(2):133–151, July 2001.

[25] G.H. Golub and C.F. van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, second edition, 1989.

[26] Isabell Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research. Special Issue on Variable and Feature Selection.*, 3:1157–1182, 2003.

[27] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237, 1999.

[28] Thomas Hofmann. Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI'99)*, 1999.

[29] Thomas Hofmann. Learning What People (Don't) Want. In *Proceedings of the European Conference on Machine Learning (ECML)*, 2001.

[30] Thomas Hofmann. Collaborative filtering via gaussian probabilistic latent semantic analysis. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 259–266. ACM Press, 2003.

[31] Thomas Hofmann and Jan Puzicha. Unsupervised learning from dyadic data. Technical Report TR-98-042, University of Californian at Berkeley, Berkeley, CA, 1998.

[32] Thomas Hofmann and Jan Puzicha. Latent Class Models for Collaborative Filtering. In *Proceedings of the International Joint Conference in Artificial Intelligence*, 1999.

[33] Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[34] A. Kohrs and B. Merialdo. Clustering for collaborative filtering applications, 1999.

[35] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *Proceedings of the 12th Conference on Advances in Neural Information Processing Systems (NIPS 12)*, pages 556–562, 2000.

[36] Roderick J. A. Little and Donald B. Rubin. *Statistical analysis with missing data*. John Wiley & Sons, Inc., 1987.

[37] Shoshana Loeb and Douglas Terry. Information filtering. *Communications of the ACM*, 35(12):26–28, 1992.

[38] Benjamin Marlin. Modeling user rating profiles for collaborative filtering. In *Proceedings of the Seventeenth Annual Conference on Neural Information Processing Systems (NIPS-2003)*, 2003.

[39] P. Melville, R. Mooney, and R. Nagarajan. Content-boosted collaborative filtering. In *Proceedings of the ACM SIGIR Workshop on Recommender Systems*, 2001.

[40] Thomas Minka. Estimating a Dirichlet Distribution. *Technical report*, 2000.

[41] Thomas Minka and John Lafferty. Expectation-Propagation for the Generative Aspect Model. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, 2002.

[42] Tom M. Mitchell. *Machine Learning*. WCB McGraw-Hill, Boston, 1997.

[43] Koji Miyahara and Michael J. Pazzani. Collaborative filtering with the simple bayesian classifier. In *Pacific Rim International Conference on Artificial Intelligence*, pages 679–689, 2000.

[44] R. M. Neal and G. E. Hinton. A new view of the EM algorithm that justifies incremental, sparse and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. Kluwer Academic Publishers, 1998.

[45] Mark O'Connor and Jon Herlocker. Clustering items for collaborative filtering. In *ACM SIGIR '99 Workshop on Recommender Systems: Algorithms and Evaluation*, 1999.

[46] Dmitry Y. Pavlov and David M. Pennock. A maximum entropy approach to collaborative filtering in dynamic, sparse, high dimensional domains. In *Proceedings of the Sixteenth Annual Conference on Neural Information Processing Systems (NIPS-2002)*, 2002.

[47] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, California, 1988.

[48] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1992.

[49] P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186, Chapel Hill, North Carolina, 1994. ACM.

[50] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of dimensionality reduction in recommender systems–a case study. In *ACM WebKDD Workshop*, 2000.

[51] Andrew I. Schein, Alexandrin Popescul, and Lyle H. Ungar. Methods and metrics for cold-start recommendations. In *Proceedings of the 25'th annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2002.

[52] Upendra Shardanand and Patti Maes. Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, volume 1, pages 210–217, 1995.

[53] Nathan Srebro and Tommi Jaakola. Weighted low-rank approximations. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, 2003.

[54] M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society*, 61(3):611–622, 1999.

[55] L. Ungar and D. Foster. Clustering methods for collaborative filtering. In *Proceedings of the Workshop on Recommendation Systems*, Menlo Park California., 1998. AAAI Press.

[56] Marco Zaffalon and Marcus Hutter. Robust feature selection using distributions of mutual information. In A. Darwiche and N. Friedman, editors, *Proceedings of the 18th International Conference on Uncertainty in Artificial Intelligence (UAI-2002)*, pages 577–584, San Francisco, CA., 2002. Morgan Kaufmann.