

CoBaFi — Collaborative Bayesian Filtering

Alex Beutel
Carnegie Mellon University
Pittsburgh, PA 15213
abeutel@cs.cmu.edu

Christos Faloutsos
Carnegie Mellon University
Pittsburgh, PA 15213
christos@cs.cmu.edu

Kenton Murray
Carnegie Mellon University
Pittsburgh, PA 15213
kwmurray@cs.cmu.edu

Alexander J. Smola
CMU and Google
Pittsburgh, PA 15213
alex@smola.org

ABSTRACT

Given a large dataset of users' ratings of movies, what is the best model to accurately predict which movies a person will like? And how can we prevent spammers from tricking our algorithms into suggesting a bad movie? Is it possible to infer structure between movies simultaneously?

In this paper we describe a unified Bayesian approach to Collaborative Filtering that accomplishes all of these goals. It models the discrete structure of ratings and is flexible to the often non-Gaussian shape of the distribution. Additionally, our method finds a co-clustering of the users and items, which improves the model's accuracy and makes the model robust to fraud. We offer three main contributions: (1) We provide a novel model and Gibbs sampling algorithm that accurately models the quirks of real world ratings, such as convex ratings distributions. (2) We provide proof of our model's robustness to spam and anomalous behavior. (3) We use several real world datasets to demonstrate the model's effectiveness in accurately predicting user's ratings, avoiding prediction skew in the face of injected spam, and finding interesting patterns in real world ratings data.

Categories and Subject Descriptors

H.2.8 [Database Management]: Data mining; H.3.m [Information Storage and Retrieval]: Miscellaneous; J.4 [Social and Behavioral Sciences]: Miscellaneous

Keywords

Collaborative Filtering; Sampling; Clustering

1. INTRODUCTION

Collaborative filtering is one of the key tools for providing relevant content and for driving successful electronic commerce. Unsurprisingly, it has attracted a substantial amount of research. Of late, Bayesian approaches have demonstrated

good performance, when applied to collaborative filtering [33, 30]. This paper makes the following contributions to the recommender problem:

Rating Model: Many collaborative filtering models simply minimize the squared error between prediction and observed rating. Alternatively, others make the assumption of a Gaussian noise model, assuming that the observed rating is given by the latent rating plus a small perturbation. Unfortunately this assumption is not realistic since the observed ratings are *discrete*, typically ranging from 1 to 5. Moreover, they need not be unimodal — some products are quite polarizing, attracting a significant number of 1 and 5 ratings with very little in between. We address this by modeling such dependencies explicitly. This builds on [37], the key difference being that the present paper provides an efficient inference algorithm for such a rating distribution.

Profile Distribution: A common approach to modeling user and item profiles is to draw them iid from a distribution that is essentially normal around a common mean [33, 35]. However, often the objects we would like to recommend form clear clusters (e.g. romantic comedies, summer action movies). Hence, sharing of statistical strength even within the same object category is likely to improve accuracy. This extends work of [1, 3, 29] on dyadic factorization models.

Our motivation is that we have rather different amounts of information for different movies (and users). That is, some movies are very popular and using only a small number of latent factors amounts to considerable underfitting; as a result, using a larger number of parameters would be greatly beneficial. On the other hand, many movies have few ratings and it would be good if we could aggregate similar ones into a larger cluster for estimation purposes.

Inference: We propose an efficient sampling algorithm that can be parallelized with relative ease. It combines collapsed inference for discrete variables, uncollapsed inference for cluster parameters, and a Metropolis-Hastings iteration for nonconjugate terms arising from the recommender distribution.¹

Copyright is held by the International World Wide Web Conference Committee (IW3C2). IW3C2 reserves the right to provide a hyperlink to the author's site if the Material is used in electronic media.
WWW'14, April 7–11, 2014, Seoul, Korea.
ACM 978-1-4503-2744-2/14/04.
<http://dx.doi.org/10.1145/2566486.2568040>.

¹ It is over two orders of magnitude faster than the sampling algorithms carried out in Stan mc-stan.org using Hamiltonian Monte-Carlo [14].

Spam Detection: A useful side-effect of the clustering approach is that similar users and items are grouped into the same cluster. This means that users who exhibit consistently abnormal behavior are likely to aggregate into the same clusters, thus limiting their impact on other recommendations and making it easier to detect and remove them.

Outline: We begin discussing our model for addressing the discrete nature of the rating distribution in Section 3. This is followed in Section 4 by a description of a nonparametric mixture of Gaussians akin to [26]. We then give an overview of the statistical model used for collaborative filtering in Section 5, and we subsequently give an in-depth description of statistical inference techniques in Section 6. We then in Section 7 analyze the model’s ability to limit the impact of fraud. Last, we do an in depth analysis of our model on a variety of datasets in Section 8.

2. RELATED WORK

Collaborative filtering algorithms have been some of the best performing methods for recommendation systems. Koren [20] presents an overview of various methods for collaborative filtering. One of the simplest and most effective methods for collaborative filtering is low-rank matrix approximations. The intuition behind factor-based approaches is that preferences for users are determined by small numbers of unobserved factors [33]. Linear factor models make use of two factor vectors — a user factor vector and an item factor vector — with the inner product representing a user’s rating for an item. These vectors can be appended to form User Factor and Item Factor Matrices [33, 36]. Much work has been done on efficient algorithms such as stochastic gradient descent [20, 39] and SVD++ [19].

Several works consider alternative distributions over attribute vectors, e.g. sparse nonnegative factors [21], jointly sparse factorizations [15], cluster aggregation [1], or discrete factorizations [29]. Closest to our work are co-clustering models, such as [5]. Our work uses a Gaussian mixture model to capture information in users and attributes.

The least mean squares loss is a mainstay in collaborative filtering, not the least due to the Netflix contest. Nonetheless, there are many other and better objectives that one could strive to optimize for. For instance, [41, 39] study preference ranking instead of estimating scores directly. In fact, research on ranking teams with alternative formulations of preferences [7, 38]. There are variants using distance similarity [40]. However, there is only scarce research in attempting to model the skewed nature of ratings directly. [37] introduce the ranking distribution and show some first strategies for inference by optimization. A full Bayesian treatment of these cases is missing, though. We address this in the present paper by proposing a computationally efficient Metropolis-Hastings sampler.

Spam detection for online reviews is a significant problem, due to the financial incentives for malicious behavior. Many algorithms used for this purpose are custom-designed. For instance [17] recognized that fake reviews can be difficult and tedious for humans to detect, and creating a large labeled dataset is impractical. After first classifying exact duplicate reviews as spam, they obtained good performance using straightforward feature extraction with a logistic regression model. Later work focused on detecting more subtle

opinion spam in hotel reviews [22]. A labeled dataset was created by a panel of judges. This effort emphasized psycholinguistic ideas and heavy feature engineering from the review text. Their resulting SVM-based classifier had very high accuracy. [32] propose an algorithm with very good theoretical guarantees of robustness, albeit without the explicit ability to detect spammers. Most related to our work is [34] who use Bayesian mixture models for detecting outliers.

In a nutshell, we use a factorization model for recommendation where the factors are derived from a Gaussian mixture model. Moreover, we model the score distribution over ratings directly to deal with the often-observed bimodality of the reviews. This is all jointly addressed by a Metropolis-Hastings sampler for a unified Bayesian treatment. The advantage of our approach is that we only need to build *one* model to perform recommendation, spam detection, to model the distribution and confidence of a rating distribution explicitly, and to group users and movies.

3. RATING DISTRIBUTION

One of the key components of our approach is to provide a model which captures the bimodality in many of the reviews found on rating sites. For instance, consider the example of a WD15EZR hard disk (newegg.com, April 27, 2013). It clearly shows a very polarized view of the product, with significant numbers of users rating it very poorly whereas others rate it as excellent.

score	★	★★	★★★	★★★★	★★★★★
ratings	349	114	55	50	201

This type of rating distribution has been commonly ignored in collaborative filtering. In fact, when aiming for a least mean squares fit the optimal rating would be 2.5319 stars, the mean of the reviews. Such a score would be very misleading since only a small minority of reviews, namely only 7% of them actually assume such a value. Moreover, while an RMSE of 2.85 would indicate that the data is a poor fit, it would tell us precious little about the actual *distribution*. The distribution itself in many cases is just as important as the predicted rating: suggesting an item with a bimodal review distribution, be it an unreliable hard drive or a polarizing film, may be undesirable for a risk-avoiding user.

Note that this is not an isolated case. We have observed many such examples in both product reviews and movies. However, not all items are equally polarizing and, in fact, the majority of reviews follow a more conventional unimodal distribution. Hence it is desirable to design a likelihood model that takes such nonstandard settings into account.

For this purpose we employ an exponential families model of a discrete distribution over the set $\mathcal{X} = \{1, \dots, R\}$, where typically we have $R = 5$. In the following we will refer to this distribution as the *recommender distribution*. Define

$$\phi(x) = \left(\left(x - \frac{R}{2} \right), \left(x - \frac{R}{2} \right)^2 \right) \quad (1)$$

to be the sufficient statistic of the data on \mathcal{X} . In this case we have with $\theta \in \mathbb{R}^2$ the model

$$p(x|\theta) = e^{\langle \phi(x), \theta \rangle - g(\theta)} \text{ where } g(\theta) = \log \sum_{j=1}^R e^{\langle \phi(j), \theta \rangle}. \quad (2)$$

The above generalizes normal distributions to a discrete finite domain. Like the Gaussian, we can adjust mean and variance by modifying the first and second coordinate of

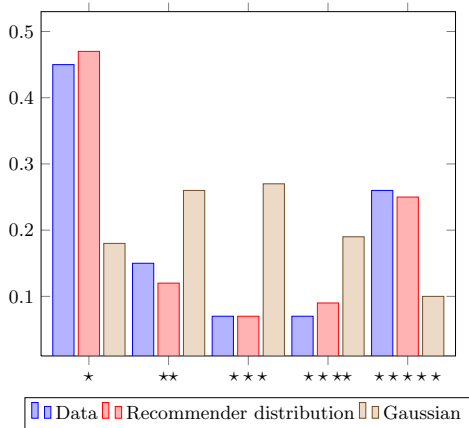
θ accordingly. However, quite unlike in a Gaussian, we can also choose coefficients $\theta_2 \geq 0$. Such a choice would amount to no variance or negative variance respectively, which is clearly impossible (with correspondingly diverging integrals). On a finite countable domain, however, normalization is easy. As common in exponential families, we have

$$-\partial_\theta \log p(x|\theta) = \sum_{x'} \phi(x')p(x'|\theta) - \phi(x) \quad (3)$$

$$-\partial_\theta^2 \log p(x|\theta) = \text{Cov}_{x' \sim p(x'|\theta)} [\phi(x')] \quad (4)$$

Note that the covariance is *independent* of x since it depends only on the log-partition function $g(\theta)$. We will exploit this property in the context of a Metropolis-Hastings sampler to infer user and movie dependent parameters.

To illustrate the model we fitted the recommender distribution to the data above. This yielded $\theta = (-0.157, 0.391)$. The associated distribution can be seen below. For comparison we also display the fit obtained by a Gaussian distribution when discretized over the domain $\{1, \dots, 5\}$ and using the mean and variances inherent in the data. It is clear that the Gaussian thoroughly misrepresents the data whereas the recommender distribution is an excellent fit.



The attraction of (2) is that it allows us infer the parameter θ efficiently, e.g. by means of simple convex optimization problem: as all exponential family models, the negative log-likelihood $-\log p(x|\theta)$ is convex in θ . The major difference is that we now need to fit two parameters: one for what effectively amounts to the mean, and one that amounts to the skew of the distribution. In summary the recommender distribution has the following properties:

- Negative values of θ_2 it will yield a unimodal distribution, as the log-likelihood mimics a discretized Gaussian. The magnitude of θ_2 determines how peaked the distribution is. Large values of θ_1 in this context imply that the popularity is fairly uncontested. This is nontrivial to achieve in a Gaussian fit of the data.
- Positive values of θ_2 can lead to a bimodal distribution. The extent of bimodality is determined by the value of θ_1 . If it is very large (or very small), it leads to what is essentially a single mode at either one of the extremes.
- Large (small) values of θ_1 correspond to an object that is rather more (less) popular. This can be seen in our example where we inferred $\theta_1 = -0.157$, i.e. the disk was rather unpopular.

In our model we will assume that the amount of polarization, i.e. the value of θ_2 , is determined in an additive fashion between users and items. That is, the distribution is bimodal if the item encourages such ratings (e.g. an unreliable hard disk) and if the user is prone to praise and condemnation. For the linear term we use an inner product model.

To assess this new rating model we need to evaluate its predictive log-likelihood, since a least mean squares fit clearly obscures the properties of the data. Hence, in our experiments we will compute $\sum_x -\log p(x|\theta)$. By basic facts from information theory [8], this is bounded from above by the entropy of the uniform distribution, i.e. $\log 5$ for 5 different outcomes. The lower bound is naturally the conditional entropy $H(R|U, M)$, where we conditioned on the user and item pairs. By definition, it is impossible to compute this lower bound explicitly, short of solving the recommendation problem optimally.

4. BI-CLUSTERING

Collaborative filtering is nontrivial whenever the number of ratings is nonuniform over items and users. Like much other natural data, the number of ratings often follows a power-law distribution on a per row and per column basis. For instance, blockbuster big-budget Hollywood movies will attract many more ratings than independent productions, regardless of their quality. Likewise, a small number of enthusiasts can generate an enormous amount of ratings when compared to more casual users.

Conventional collaborative filtering models assume that all users (and items respectively) are parametrized by a fixed dimensional model. In fact, many experiments show that the accuracy of the models plateaus (or even decreases) once a fairly modest number of parameters has been reached. Part of this effect is due to the fact that we have comparatively little data per row (or column) of the recommendation matrix. Furthermore, data scarcity is different, when viewed on a per-column or per-row basis. For instance, we might have millions of users but only thousands of movies.

This suggests that it would be very much desirable to have a mechanism for assigning capacity dynamically. That is, users who generate significant amounts of ratings are sufficiently well-specified to afford a ‘personal’ parameter vector for them. On the other hand, users who generate very little data are probably best modeled as unspecific members of a much larger pool of similar participants. Such a model can be obtained by clustering users, e.g. as mixture of Gaussians.

To accomplish this, we use bi-clustering of the rows and columns of the recommendation matrix to allow for dynamic allocation of statistical capacity between sets of users. This allows us to increase dimensionality beyond what is optimal when treating all users/items equally. Hence we need to introduce a statistical model for partitioning. We resort to the Dirichlet Process for this purpose. This yields a nonparametric mixture of Gaussians. To keep the presentation self-contained we give a brief overview of the model below.

One of the prototypical tools for nonparametric modeling is the Dirichlet Process $DP(H, \gamma)$. It allows for a discrete distribution of observations drawn from an arbitrary base measure H over the domain \mathcal{X} in such a way as that the marginals match draws from H , while simultaneously obtaining a countable set of distinct items. Denote by \mathcal{X}_i a partition of \mathcal{X} and let $c_i := \mu_H(\mathcal{X}_i)$ be the measure of \mathcal{X}_i under H . Then the discretization of draws from G_0 with

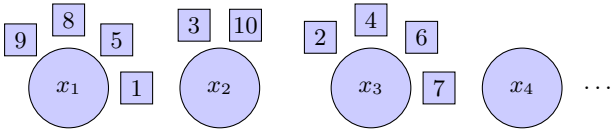


Figure 1: An example of the Chinese Restaurant Process metaphor. Here we observe 10 customers who have so far occupied tables 1 (customers 1, 5, 8, 9), 2 (customers 3, 10) and 3 (customers 2, 4, 6, 7). A new customer will pick the first three tables with probabilities $\frac{4}{10+\gamma}$, $\frac{2}{10+\gamma}$ and $\frac{4}{10+\gamma}$ respectively, or a new table with probability $\frac{\gamma}{10+\gamma}$.

$G_0 \sim DP(H, \gamma)$ onto the sets \mathcal{X}_i behaves as if it had been drawn from a multinomial distribution θ with $\theta \sim \text{Dir}(\gamma c)$.

A useful view of the Dirichlet Process is the Chinese Restaurant metaphor. In it, each data point is considered as a customer in a restaurant with an infinite number of tables. Initially all tables are empty. As the tables fill up, customers pick existing tables in proportion to their popularity, resulting in a “rich get richer” scenario. The probability for customer i to pick table j is given by

$$\Pr\{z_i = j\} = \begin{cases} \frac{N_j}{\sum_k N_k + \gamma} & \text{for an existing table} \\ \frac{\gamma}{\sum_k N_k + \gamma} & \text{for a new table.} \end{cases} \quad (5)$$

Here z_i encodes the choice of customer i . N_j denotes the *current* number of customers sitting at table j and $\sum_k N_k$ is the total number of customers so far. This makes the Chinese Restaurant Process a single parameter distribution over partitions of the integers. Figure 1 provides a pictographical representation of the process. One may show that the distribution over partitions is invariant in the order in which customers arrive, i.e. it is exchangeable.

A benefit of (5) is that it affords for collapsed Gibbs sampling inference by simply drawing from

$$z_i \sim \Pr(z_i | \text{rest}) \text{ where } \Pr(z_i | \text{rest}) \propto p(z_i | z \setminus z_i) p(\text{ratings} | z)$$

Inference proceeds by traversing users (or items) and resampling their cluster assignments in sequence.

5. STATISTICAL MODEL

We describe a model for obtaining collaborative filtering estimates in a pure rating setting rather than a recommendation setting [41]. That is, we assume that for a subset of (user, item) pairs (i, j) we are given ratings $r_{ij} \in \{1, \dots, R\}$. It is our goal to infer user and item specific parameters u_i, v_j such that we are able to estimate the distribution of $r_{ij} | i, j$ efficiently. For this purpose we employ a directed graphical model, as depicted in Figure 2.

This model extends existing recommendation algorithms such as [33, 35] by assuming a more complex structure for the latent user and item attributes. We assume that the ratings r_{ij} are drawn from the recommender distribution. The parameters $\langle u_i, v_j \rangle + u_i^{(1)} + v_j^{(1)}, u_i^{(2)} + v_j^{(2)}$, as associated with parts of the user and item specific vectors, account for preference and skew explicitly:

$$r_{ij} \sim \text{Recommender}(\langle u_i, v_j \rangle + u_i^{(1)} + v_j^{(1)}, u_i^{(2)} + v_j^{(2)})$$

From this we have for each user a $d - 2$ length vector u_i and scalars $u_i^{(1)}, u_i^{(2)}$ giving us d latent parameters per user

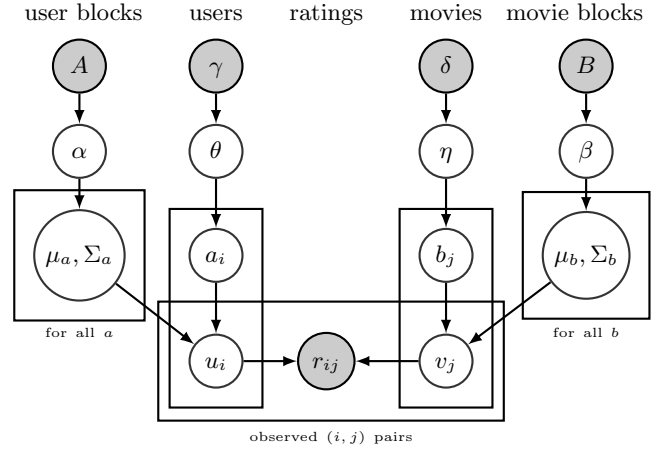


Figure 2: Factorized rating model. Note the symmetry between users and items. In both cases we use a mixture of Gaussians to represent the latent parameter distribution.

(and item). For simplicity, we will often refer to this set of parameters as a single vector u_i .

We assume that the user and item parameter distributions are described by a mixture of Gaussians, where

$$u_i \sim \mathcal{N}(\mu_{a_i}, \Sigma_{a_i}) \text{ and } v_j \sim \mathcal{N}(\mu_{b_j}, \Sigma_{b_j}).$$

Here a_i and b_j are the clusters that user i and item j belong to respectively. The cluster means and variances μ_a, Σ_a are drawn from a Gauss-Wishart distribution with ν_0 degrees of freedom and parameters $\{\mu_\alpha, W_\alpha, \lambda_\alpha\}$ and $\{\mu_\beta, W_\beta, \lambda_\beta\}$, respectively:

$$(\mu_a, \Sigma_a) \sim \text{GW}(\lambda_\alpha, W_\alpha, \mu_\alpha) \text{ and } (\mu_b, \Sigma_b) \sim \text{GW}(\lambda_\beta, W_\beta, \mu_\beta)$$

This is convenient since in this case the priors are conjugate and we are able to draw from $\mu_a, \Sigma_a | \alpha, u, a$ exactly without the need for approximation.

The common parameters W_α, μ_α (and W_β, μ_β) allow us to share information about variance and means between clusters via the conjugate Wishart distribution. Finally, we impose an inverse Gauss-Wishart hyperprior on $W_\alpha, \mu_\alpha, W_\beta, \mu_\beta$ respectively. For computational convenience we will maximize the likelihood with regard to those hyperparameters.

Finally, the cluster indicators, a_i and b_j , allow us to assign users (and items respectively) to particular groups. To sample cluster assignments, we draw

$$a_i \sim \text{Mult}(\theta) \text{ and } b_j \sim \text{Mult}(\eta) \quad (6)$$

In turn, we assume that the distribution over clusters follows a Dirichlet distribution conjugate to the multinomials governing cluster membership. Likewise in the nonparametric case (where the number of clusters itself is unbounded), we assume that it is given by a draw from a Dirichlet process.

$$\theta \sim \text{Dir}(\gamma) \text{ and } \eta \sim \text{Dir}(\delta) \quad (7)$$

Therefore, in a nutshell, we assume that the interaction between users u_i and items v_j can model ratings r_{ij} similarly to many other collaborative filtering methods. Unlike [33] who assume one common mean and variance for all users (and items respectively), clustering provides the ability to better model what is a similar user.

6. PARAMETER INFERENCE

Inference in the statistical model introduced previously requires sampling and (or) optimization. Matters are simple whenever the distributions are conjugate since the associated parameters can either be integrated out (collapsed) or, alternatively, it is easy to draw from them. The non-conjugate parts do not admit a closed-form treatment. In the following we discuss in detail how to sample the various parameters and variables for the sampler's Markov chain, subject to these constraints. Since user and item-related parameters are entirely identical, we only discuss users.

6.1 Sampling the user and item parameters u_i

We omit details for the case where r_{ij} is Gaussian, as it is almost identical to [33]. Since the recommender distribution does not have conjugacy, sampling the user and item parameters u_i and v_j is challenging and we resort to a Metropolis-Hastings procedure [6]. In its simplest form the idea is that rather than sampling from some difficult distribution $p(x)$ we draw \bar{x} from an approximation $q(x)$ and accept the move $x \rightarrow \bar{x}$ with probability $\min(1, p(\bar{x})q(x)/p(x)q(\bar{x}))$.

In our case the distribution p is given by the log-concave distribution $p(u_i|\text{rest})$ and we obtain q by performing a Laplace approximation of p . In other words, we perform a second order Taylor approximation of $\log p(u_i|\text{rest})$ at its mode, which yields a Gaussian. This is possible since log-concave distributions have a global mode and the second derivative of the negative log-posterior is positive semidefinite.

$$\begin{aligned} & -\log p(u_i|\text{rest}) - \text{const.} \\ &= \frac{1}{2}(u_i - \mu_a)^\top \Sigma_a^{-1}(u_i - \mu_a) + \\ & \quad \sum_{j \in \text{ratings}(i)} g(u_i^{(1)} + v_j^{(1)} + \langle u_i, v_i \rangle, u_i^{(2)} + v_j^{(2)}) - \\ & \quad \sum_{j \in \text{ratings}(i)} \left\langle \phi(r_{ij}), (u_i^{(1)} + v_j^{(1)} + \langle u_i, v_i \rangle, u_i^{(2)} + v_j^{(2)}) \right\rangle \end{aligned}$$

As before $u_i, u_i^{(1)}, u_i^{(2)}$ denote the latent parametrization, the linear biases and the quadratic biases for the user, with corresponding terms for item parameters v . The log-partition function g is as defined in (2) with appropriate derivatives given by (3) and (4).

Convex minimization of $-\log p(u_i|\text{rest})$ yields the mode $\bar{\mu}$. By first order conditions and since the objective is analytic, we know that at $\bar{\mu}$ the gradient vanishes and we may compute the 'variance' of $q(u_i)$ via

$$\bar{\Sigma}^{-1} = \sigma_a^{-1} + \sum_{j \in \text{ratings}(i)} \partial_{u_i}^2 g(u_i^{(1)} + v_j^{(1)} + \langle u_i, v_i \rangle, u_i^{(2)} + v_j^{(2)}).$$

Using $\bar{u} \sim \mathcal{N}(\bar{\mu}, \bar{\Sigma})$ we accept the MH proposal with

$$\min \left(1, \frac{p(\bar{u}|\text{rest})\mathcal{N}(u_i(t)|\bar{\mu}, \bar{\Sigma})}{p(u_i(t)|\text{rest})\mathcal{N}(\bar{u}|\bar{\mu}, \bar{\Sigma})} \right) \quad (8)$$

Otherwise we set $u_i(t+1) \leftarrow u_i(t)$. The advantage of this strategy is that we need to compute $\bar{\mu}$ and $\bar{\Sigma}$ only once per resample of u_i . Multiple MH steps ensure proper mixing.

The computational cost of this step is $O(d^3 + d^2 n_{i, \text{ratings}})$ per user, i.e. for the purpose of the Taylor approximation we need to aggregate all ratings ($n_{i, \text{ratings}}$) of a given user and subsequently draw from it. Hence the total cost is $O(d^3|V| + d^2|E|)$, when viewing the set of (user,item) pairs as a graph with V vertices and $|E|$ edges.

6.2 Sampling the cluster membership a_i

Our generative model represents cluster assignments of users and items as draws of indicator variables a_i and b_j from multinomial distributions, parametrized by θ and η . These multinomials are in turn drawn from their conjugate, the Dirichlet distribution. We take advantage of a collapsed Gibbs sampler to deal with the Dirichlet process in a (and b respectively) [26, 10]. More specifically, we may integrate out θ, η such that we are left with an *exchangeable* distribution over $\{a_i\}|\gamma$ (and corresponding terms in b, δ). This leaves us with (5), i.e.

$$p(a_i = \bar{a} | a \setminus \{a_i\}) = \begin{cases} \frac{\frac{n_{\bar{a}}}{n+\gamma-1}}{\frac{\gamma}{n+\gamma-1}} & \text{for existing cluster} \\ \frac{\gamma}{n+\gamma-1} & \text{for new cluster} \end{cases} \quad (9)$$

Analogous values can be obtained if we want to use the Pitman-Yor process [28]. Secondly, we need to evaluate the likelihood of $u_i|a_i$, as given by $\mathcal{N}(\mu_{a_i}, \Sigma_{a_i})$. This governs how well u_i fits to an *existing* cluster. Whenever we instantiate a new cluster, matters are slightly more involved — the acceptance probability of drawing a new random set of parameters μ, Σ from the associated Gaussian-Wishart distribution would be very low. Instead, we integrate out μ, Σ and evaluate directly $p(u_i|\mu_\alpha, W_\alpha, \lambda_\alpha)$. However, to do this we must integrate and collapse out the intermediate cluster parameters, see e.g. [11]. Thus the collapsed probability of u_i is

$$\begin{aligned} & p(u_i|\mu_\alpha, W_\alpha, \lambda_\alpha) \\ &= \mathcal{T} \left(\nu_0 - d + 2, \frac{\lambda_\alpha \mu_\alpha + u_i}{\lambda_\alpha + 1}, \frac{\lambda_\alpha + 2}{(\lambda_\alpha + 1)(\nu_0 - d + 2)} \Sigma_m \right) \end{aligned}$$

where $\Sigma_m = W_\alpha^{-1} + \frac{\lambda_\alpha}{\lambda_\alpha + 1}(u_i - \mu_\alpha)(u_i - \mu_\alpha)^\top$. Here \mathcal{T} is the student-t distribution, ν_0 is a user specified parameter and d is the rank of the decomposition. Ultimately, this gives us an estimate of the likelihood of a new cluster for u_i :

$$p(a_i = \bar{a} | \text{rest}) \propto \begin{cases} \frac{n_{\bar{a}}}{n+\gamma-1} \mathcal{N}(u_i|\mu_{\bar{a}}, \Sigma_{\bar{a}}) & \text{if cluster exists} \\ \frac{\gamma}{n+\gamma-1} p(u_i|\mu_\alpha, W_\alpha, \lambda_\alpha) & \text{for new cluster} \end{cases}$$

Lastly, if we draw a new cluster, we need to instantiate the values of μ_a, Σ_a . Since this is no different for clusters of size 1 or larger, we describe the general case below.

Resampling the cluster membership for a user requires us to evaluate the cluster likelihood for each existing (and one new) cluster. Each of these steps cost $O(d^2)$ computation. Given k clusters this amounts to $O(kd^2|V|)$ CPU cost for one sampling pass.

6.3 Sampling cluster parameters μ_a, Σ_a

Since the Gauss-Wishart distribution is conjugate to the Gaussian, it follows that the conditional posterior $p(\mu_a, \Sigma_a | \text{rest})$ is also a Gauss-Wishart. Moreover, the update equations for a cluster are particularly simple: draw μ_a, Σ_a from a Gauss-Wishart distribution with parameters

$$\bar{\lambda}_a = \lambda_\alpha + N_a \quad (10a)$$

$$\bar{\lambda}_a \bar{\mu}_a = \lambda_\alpha \mu_\alpha + \sum_{i: a_i = a} u_i \quad (10b)$$

$$\bar{\lambda}_a \bar{W}_a^{-1} = \lambda_\alpha \Sigma_\alpha^{-1} + \lambda_\alpha \mu_\alpha \mu_\alpha^\top + \sum_{i: a_i = a} u_i u_i^\top \quad (10c)$$

$$\bar{\nu} = \nu_0 + N_a \quad (10d)$$

Here $N_a = \sum_{i:a_i=a} 1$ is the size of the cluster a and $\bar{\mu}_a$ is the mean of the u_i vectors for the user in cluster a . See e.g. [25] for extended details and how to sample from it via a multivariate Student-t distribution.

The cost for sampling the cluster means and variances is $O(N_a d^2 + d^3)$ per cluster, hence the total cost is $O(|V|d^2 + kd^3)$ to sample all clusters in the algorithm.

6.4 Inference for hyperparameters $\mu_\alpha, W_\alpha, \lambda_\alpha$

The last remaining step for inference is to estimate a suitable value for the parameters of the Gauss-Wishart distribution. Choosing these terms appropriately is important since they provide the default (prior) for the cluster specific factors. We resort to maximization rather than sampling, since the number of clusters is large relative to the number of parameters involved in determining $\mu_\alpha, W_\alpha, \lambda_\alpha$, hence we expect the posterior to be rather peaked. For the sake of completeness, we give the full derivation below. In a nutshell we compute the parameters using a conjugate to the Gauss-Wishart distribution. While this is not available in closed form, it is easily constructed implicitly by adding an additional spurious normal distribution. We choose one with unit variance and zero mean. For notational convenience, we will include such a ‘prior’ in the set of $(\mu_a, \Sigma_a, \lambda_a)$ terms. The Gauss-Wishart distribution is given by

$$p(\mu, \Sigma | \mu_0, \lambda, W, \nu) = \mathcal{N}(\mu | \mu_0, \lambda^{-1} \Sigma) \mathcal{W}(\Sigma^{-1} | W, \nu) \quad (11)$$

$$\mathcal{W}(\Sigma^{-1} | W, \nu) = 2^{-\frac{\nu d}{2}} |\Sigma|^{\frac{\nu}{2}} \Gamma_d^{-1} \left(\frac{\nu}{2} \right) |W|^{\frac{\nu-d-1}{2}} e^{-\frac{1}{2} \text{tr} \Sigma W}$$

Here \mathcal{W} denotes the Wishart distribution with ν degrees of freedom and covariance scale W . Moreover, Γ_d denotes the multivariate Gamma function. The negative log-likelihood given k normal distributions is given (up to constants independent of μ_0, W, λ) by

$$\begin{aligned} & -\log p(\{\mu_a, \Sigma_a\} | \mu_0, \lambda, W, \nu) \\ &= \frac{kd}{2} \log \lambda + \frac{1}{2\lambda} \sum_{a=1}^k (\mu_a - \mu_0)^\top \Sigma_a^{-1} (\mu_a - \mu_0) + \frac{\nu}{2} \sum_{a=1}^k \log |\Sigma_a| \\ &+ k \log \Gamma_d \left(\frac{\nu}{2} \right) + \frac{k(\nu-d-1)}{2} \log |W| + \frac{1}{2} \text{tr} W \sum_{a=1}^k \Sigma_a + \text{const.} \end{aligned}$$

Taking derivatives with respect to μ_0, W , and λ respectively yields that the log-likelihood is maximized for

$$\mu_0 = \left[\sum_{a=1}^k \Sigma_a^{-1} \right]^{-1} \sum_{a=1}^k \Sigma_a^{-1} \mu_a \quad (12)$$

$$W = k(\nu - d - 1) \left[\sum_{a=1}^k \Sigma_a \right]^{-1} \quad (13)$$

$$\lambda = \frac{1}{kd} \sum_{a=1}^k (\mu_a - \mu_0)^\top \Sigma_a^{-1} (\mu_a - \mu_0) \quad (14)$$

That is, the mean μ_0 is given by what amounts to Gaussian posterior averaging; the prior on covariances W is given by the average covariance, suitably normalized by dimensions and degrees of freedom; the variance scale λ is given by an average over clusters and dimensions. The degrees of freedom $\nu > d - 1$ control the effective dimensionality of the space. Large values of ν encourage a larger volume of the associated covariance matrices, whereas a values emphasize low-dimensional per-cluster distributions. We set $\nu = \frac{3}{2}d$.

Computing W, μ_0 and λ costs $O(kd^3)$ work since we need to aggregate over all cluster centers. In summary, we have the following runtime properties:

Remark 1 *Per iteration the algorithm requires $O(kd^3 + |V|d^3 + |V|d^2k + |E|d^2)$ computation. That is, it is cubic in the number of latent parameters, linear in the number of clusters, and linear in the amount of data given.*

Furthermore, if needed, these steps could be parallelized efficiently, since most work is specific per user (or per item) with communication required only once for each round of sampling and optimization.

7. ROBUSTNESS TO SPAM

Fraud detection is a persistent challenge and requires a multi-tiered approach. By default, most recommender systems are not very resilient, allowing any spam that is not caught ahead of time to skew recommendations significantly. To illustrate this, consider Bayesian matrix factorization [33, 35]. In this case the influence of a malicious user cannot be bounded: the sufficient statistic governing the normal distribution over users is a linear combination of the statistics of all constituents. Hence it follows that each individual contribution can assume arbitrary amounts, provided that the associated statistic is also unbounded. Moreover, even for bounded sufficient statistics, the aggregate influence can be arbitrarily high, provided that a sufficiently large number of malicious users contributes. In other words, if a sufficiently high number of fraudulent likes for an objectionable movie is received, this will lead to the movie being rated very highly. The example below quantifies this effect.

Example 1 (Spam and sufficient statistics) *Denote by $\phi(x)$ the sufficient statistic of $x \in \mathcal{X}$. Moreover, let $X := \{x_1, \dots, x_m\}$ and $X' := \{x'_1, \dots, x'_{m'}\}$ be regular and malicious observations respectively. Let m_0, μ_0 be sufficient statistics arising from a conjugate prior. Then the posterior mean yields*

$$\hat{\mu} := \frac{1}{m_0 + m + m'} \left[m_0 \mu_0 + \underbrace{\sum_{i=1}^m \phi(x_i)}_{=: m \cdot \mu[X]} + \underbrace{\sum_{i=1}^{m'} \phi(x'_i)}_{=: m' \cdot \mu[X']} \right] \quad (15)$$

Hence parameter estimates will be influenced $O(\frac{m'}{m_0 + m + m'})$ by outliers whenever we have bounded sufficient statistics and by an unbounded amount whenever the statistics are unbounded, even for $m' = 1$.

Several researchers, such as [32], have attempted to address this problem. However, [32] deals primarily with robustness in the nearest neighbor case, and the extension to other estimators is highly nontrivial. Our approach borrows from [34] which investigates mixture models for outlier-robust density estimation. The key difference is that we perform bi-clustering of pairs of conditioning attributes. Moreover, we use clustering as latent information in an estimation problem.

Our analysis relies on the idea that when generating multiple clusters, malicious users can typically only affect a real users very little. *Whenever spammers are very different from regular users they will be aggregated into clusters of their*

own. Alternatively, whenever they are very similar to regular users, they cannot do much damage. It exploits the fact that whenever a user is rather different, (9) suggests that it should be assigned to a new cluster.

Denote by X and X' two sets of m and m' observations respectively. In a Dirichlet process, the probability of two vs. one cluster is given by $\frac{mm'}{(n+\alpha)^2}$ vs. $\frac{m+m'}{n+\alpha}$. Hence, if the data likelihood for two separate clusters exceeds that of a single joint cluster via

$$p_{\text{cluster}}(X)p_{\text{cluster}}(X')\frac{mm'}{(n+\alpha)^2} > p_{\text{cluster}}(X \cup X')\frac{m+m'}{(n+\alpha)}$$

then there is benefit in splitting observations. From this we see that a split is likely whenever m' is large, i.e. whenever we have a large number of spammers, or alternatively, whenever their behavior deviates substantially from regular users.

On the other hand, if the spammers try being subtle (to avoid detection), then their presence will improve overall parameter estimates. As a result, egregious spammers are clustered together, limiting their impact on others' recommendations and making them easier to directly spot, through analyzing the clusters or testing users within the clusters.

8. EXPERIMENTS

The model described above is particularly interesting because of the insight it gives into the many common datasets used throughout data mining. Here we demonstrate its effectiveness accurately modeling real world data, both matching some expectations of reviews and breaking others.

	Users	Items	Ratings
Netflix-48k	48,090	17,770	10,015,137
Netflix-24k	24,047	17,770	5,038,411
BeerAdvocate	12,652	5000	112,981
Amazon Electronics	29,294	2000	30,378
Amazon Clothing	16,017	5000	51,438

Table 1: Review datasets used in our experiments

8.1 Datasets and Set up

Various ratings systems and datasets have very different characteristics. We look at a few real-world datasets with varying distributional properties. As some domains may be more polarizing than others, the simple assumption that ratings will look the same across domains does not hold. We looked at four different sets of review data comprising movies, beer, clothing, and electronics. A summary of the datasets can be found in Table 1.

In analyzing movies data, we used the Netflix Prize dataset. We create two subsets of the data by randomly sampling users but keeping all movies. This creates one dataset of 48,090 users and 17,770 movies, which we will refer to as Netflix-48k, and one dataset of 24,047 users and 17,770 movies, which we will refer to as Netflix-24k. The Netflix-48k dataset has a total of 10,015,137 ratings and the Netflix-24k dataset has 5,038,411 ratings.

Second, we use user reviews of beers from BeerAdvocate.com, as was scraped in [24]. Here we again take a subset of the data, comprising 5000 beers and their associated reviews from 12,652 users. This makes for a total of 112,981 reviews.

We also use two datasets of Amazon reviews [23]. We first use reviews of a subset of products from Amazon's Electron-

	Uniform	BPMF	CoBaFi
Netflix-24k	1.6904	1.2525	1.1827
BeerAdvocate	2.1972	1.9855	1.6741

Table 2: Comparison of predictive log-likelihood on Netflix-24k and BeerAdvocate data.

ics category. This includes 2000 products and all of their associated from 29,294 users. The total number of reviews in the dataset is 30,378. We note that this is just over 1 review per user on average, making this dataset much sparser than the rest. Last, we use the reviews from a subset of the products in Amazon's Clothing & Accessories category. This is 51,348 reviews on 5000 products from 16,017 users.

Implementation: We implement our model and Gibbs sampling procedure in Matlab along with its parallel toolbox for speed. For simplicity and efficiency purposes we set a maximum number of clusters for users and items in each run, which we will specify for the experiments below. In all cases we use the PMF solution from [33] as our starting point. For all experiments below we set the rank (the length of u_i and v_j) $d = 30$ and $\nu_0 = \frac{3}{2}d$.

Comparing on predictive probability: To measure our model's fit, we use the predictive probability (PP):

$$PP = \frac{1}{N} \sum_{r_{ij}} -\log p(r_{ij}|u_i, v_j)$$

In order to make this a fair comparison with non-discrete distributions such as the typical Gaussian distribution, we discretize it first before any comparison. Because our recommender model can also act like a Gaussian, we can convert a Gaussian model $\mathcal{N}(\langle u_i, v_j \rangle, \sigma^2)$ to a recommender model $\text{Recommender}(\langle \hat{u}_i, \hat{v}_j \rangle, \hat{u}_i^{(2)} + \hat{v}_j^{(2)})$ as follows:

$$\begin{aligned} \hat{u}_i &= \sigma \cdot u_i & \hat{v}_j &= \sigma \cdot v_j \\ \hat{u}_i^{(1)} = \hat{v}_j^{(1)} &= 0 & \hat{u}_i^{(2)} = \hat{v}_j^{(2)} &= -\frac{1}{4\sigma^2} \end{aligned}$$

As a result, no weight of the distribution is wasted on regions for which there couldn't be a rating. We tested many different values of σ^2 that gave the best predictive probability for a Gaussian and ultimately chose to use $\sigma^2 = 1.0$ because it generally produced the best results. Additionally, to compare against BPMF we run our code with a maximum of 1 cluster and fitting a normal as was done in their model.

8.2 Model fit

We run our model on the Netflix-24k and BeerAdvocate datasets to verify that it better models the data. We compare the results of running BPMF [33] to that from our model with a maximum of 50 clusters. As explained before, we want to measure the *fit* of our model so we use average predictive probability. As seen in Table 2 (where smaller predictive probability is better), CoBaFi fits the data much better than BPMF for both Netflix-24k and BeerAdvocate.

Sampling from the recommender distribution: As was mentioned earlier, one of the challenges of fitting such a model correctly is approximating our recommender distribution to perform the Metropolis-Hastings sampling. To verify that our approximation was working well, we recorded the average acceptance rate for a couple of our datasets. In the Netflix-24k dataset we see an average acceptance rate of 77.77%. In the robustness tests below on the Amazon Electronics dataset, we see a 99.12% acceptance rate.

8.3 Robustness to spam

One of the benefits of incorporating clusters into a model is that similar users will get clustered together, whether it be users who have similar tastes, or anomalous users who may try to manipulate ratings for their own gain. Spammers are an issue in any large, community driven rating system as there is frequently a financial incentive for positive ratings and an incentive for negative ratings for competitors. By modeling the latent clusters of users, we are able to group similar users together. This means that anomalous users, such as spammers, may get grouped together, and thus have a smaller impact overall on ratings outside of their collective group or groups. To do this we ran two experiments. In each we take one of the datasets above and inject spammers who are trying to manipulate the ratings for a subset of items. We then measure how much these spammers effect the fit of our model on these items.

	PP Before	PP After
BPMF	1.7047	1.8146
CoBaFi	1.0549	1.7042

Table 3: Predictive probability on “attacked items” in the Amazon Electronics dataset before and after adding spammers.

	PP Before	PP After
BPMF	1.2375	1.3057
CoBaFi	0.9670	1.2935

Table 4: Predictive probability on “attacked movies” in the Netflix-24k dataset before and after add spam from hijacked accounts.

Dumb spammers: For the first experiment we model spammers who are quite flagrant in how they try to manipulate certain reviews and do not camouflage their behavior by also providing typical ratings. To do this we use the Amazon Electronics dataset and see if we can take generally well-liked products and bring their ratings down (as a competitor would want). We select 31 products with a high average rating and at least 100 reviews. These products include Logitech speakers, an HDMI cable, a TiVo USB Network Adapter, and an HP LaserJet Printer, among many others. For these products we randomly select 100 ratings for each project to be included in the training dataset and the rest to be held out for the test dataset. We subsequently create 100 new accounts, and for all of these products our spam accounts give a rating of 1.0. As such, each product being manipulated now has half real reviews and half fake. As can be seen in Table 3, the predictive probability of our model is better than that for BPMF before and after we add the spam. Additionally, while the model is clearly effected by the spammers as it shifts to cover both the 1 star and 5 star reviews, it does an even better job of clustering spammers and the products they attack. As can be seen in Figure 4(a), most of the spammers (83%) are placed in the same clustered. Additionally, not graphed due to its simplicity, the attacked products are *all* placed into the same cluster. In an industry setting, this sort of interpretability and ability to understand group behavior would be valuable for investigating suspicious behavior.

Spammers with hijacked accounts: Second we test the model’s robustness to more clever spammers - those

who hijack real accounts. In this case we take real users and assume their account has been hijacked and begin providing misleading dubious reviews. This is a common issue in online systems and is the worst-case form of spammers who add realistic-looking reviews to try to camouflage their fraudulent behavior. To test our model in this setting we use the Netflix-24k dataset. Here we choose 85 movies with an average rating over 4.3 and with at least 200 reviews. Additionally we select 99 users at random (their average number of ratings is approximately 209). As before, for each of the movies selected to be attacked we include 100 of their ratings at random in the training set and use the rest for the test set. For the hijacked accounts we merely add 1 star reviews to all of the movies attacked. Again, we see in Table 4 that our model has a better predictive probability than BPMF both before and after we add the spam. More impressively, CoBaFi is still successful in clustering together the hijacked users and the attacked movies, as seen in Figures 4(b-c). Note, this is particularly surprising for the hijacked users since they may have hundreds or thousands of legitimate reviews along with the only 85 fake reviews that contribute to their profile. From this we see that CoBaFi naturally handles spammers in a way that minimizes their impact on collaborative filtering and groups them together.

8.4 Natural clusters in real world

As mentioned earlier, being able to understand your data and interpret your latent parameters is useful in many applications and industry settings. Besides clustering anomalous behavior, our model provides interesting insight into the natural clusters of items based on their latent preferences in v_j .

We analyze the clusters produced from the Netflix-48k dataset. In fitting our model, we set the maximum number of clusters to 50, which given the number of movies/beers being rated leaves a still fairly coarse clustering. In order to isolate the effects of the clustering mechanism and not the flexible distribution, we use a Normal as our recommender distribution so that clusters are based only on v_j . In the Netflix-48k dataset we analyzed the clustering in two different ways. In Figure 3, we look at the distribution of different series across different clusters. We see that for most of the series nearly all of the seasons or films in that series are clustered together. Additionally, we see in Table 5 that clusters often contain movies within the same genre, with Cluster 28 containing generally comedies, Cluster 30 containing material for children, and Cluster 48 including science fiction. This sort of interpretable model is of course useful in practice to understand your data and user preferences.

Cluster 28	Cluster 30	Cluster 48
Simpsons	Scooby Doo	Star Trek
Family Guy	Spy Kids	Back to the Future
Monty Python	Stuart Little	Southpark
Curb your Enthusiasm	Dr. Dolittle	Lord of the Rings
The Twilight Zone	Lion King	Harry Potter
Arrested Development	Agent Cody Banks	The X-Files
Chappelle Show		
Monty Python		
Seinfeld		

Table 5: Clusters of movies and shows from Netflix. Series are only included if there are at least 2 items from that show or series in the cluster.

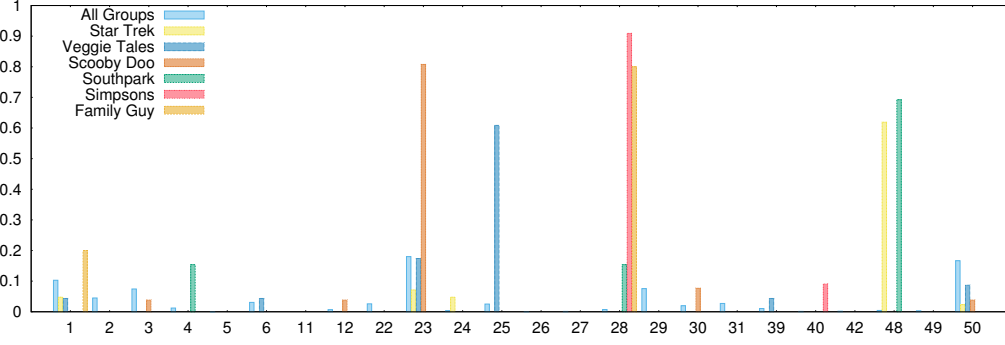


Figure 3: Distribution of movies from TV series over clusters. Movies from the same series are highly concentrated. Viewing series as ground truth, we see that the accuracy is fairly high. Note that movies within the same series are not necessarily uniform (different seasons, directors, script writers).

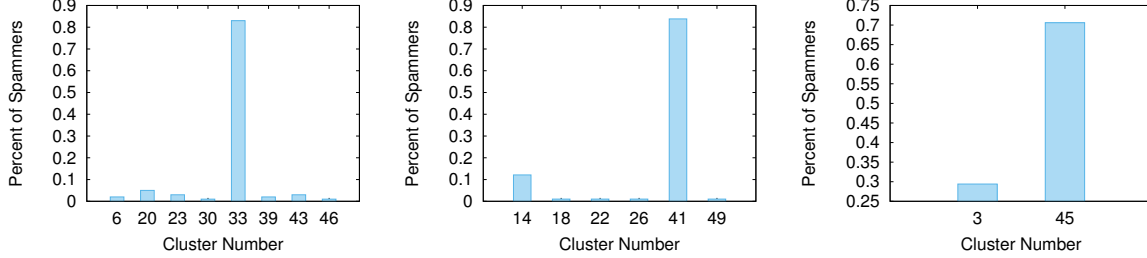


Figure 4: Distribution of spammers over clusters; Left: *Dumb spammers* on the electronics dataset; Middle: *Hijacked users* on the electronics dataset; Right: *Spammed movies* on the Netflix-24k spam dataset. In all three cases the spammers congregate tightly in a very small number of clusters.

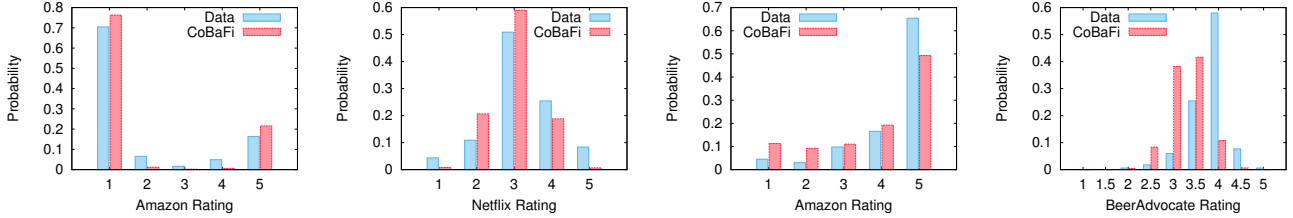


Figure 5: Rating distributions from the data and as estimated by CoBaFi. From left to right: *Sony 50CDQ80RS 80 Minute 700 MB 48x CD-R 50 Pack Spindle* on Amazon Electronics (the most skewed distribution); *The Rookie* on Netflix (the most Gaussian distribution); *Vanity Fair Women's My Favorite Illumination String Bikini Panty* on Amazon Clothing; *Ovila Abbey Saison - Sierra Nevada Brewing Co.* on BeerAdvocate.

8.5 Shape of real world data

The most interesting interpretable parameter of our model is understanding the wide range of distributions. In analyzing our results, we quickly see the world is not so normal. For each of our datasets we calculate the convexity of each item, $\hat{\theta}_j^{(2)}$:

$$\hat{\theta}_j^{(2)} = v_j^{(2)} + \frac{1}{|\text{ratings}(j)|} \sum_{i \in \text{ratings}(j)} u_i^{(2)}$$

With this term, we can generally see how convex (“U”-shaped) or concave (Gaussian) our item ratings actually are.

Beer Ratings: The simplest of the results, though surprising given the others, was with the BeerAdvocate ratings. For *all* beers we found $\hat{\theta}_j^{(2)} < 0$, meaning that the distribution was always Gaussian. Additionally, the distributions were not overly skewed as in other datasets; the items with the smallest values of $\hat{\theta}_j^{(2)}$ and thus the least variance

were typically centered at 4.0 or less as seen in Figure 5(d). Looking at the distribution of variances $\hat{\theta}_j^{(2)}$ we see a fairly wide range of variances across the beers. This explains why our model has such an improved predictive probability than BPF as shown above and demonstrates the importance of fitting the variance as well as the mean.

Netflix Ratings: Within the Netflix dataset, the results were not nearly as simple. Here we found that, possibly to the surprise of many data-miners, for most items on Netflix, $\hat{\theta}_j^{(2)} < 0$ and thus the distribution is Gaussian. Taking a closer look at the skews, we find some interesting patterns. In particular, as seen in Table 7, television shows were the most convex. For nearly all of these cases, we see that the shows produce highly polarized results that are usually heavily skewed in one direction. Alternatively, the items that were fitted to be the most Gaussian were mediocre movies such as “The Rookie.” A comparison of the distributions can be seen in Figure 5(b). Intuitively, this makes sense. Watch-

Amazon Clothing	Bra Disc Nipple covers Vanity Fair Women's String Bikini Panty Lee Men's Relaxed Fit Tapered Jean Carhartt Men's Dungaree Jean Wrangler Men's Cowboy Cut Slim Fit Jean
Amazon Electronics	Sony CD-R 50 Pack Spindle Olympus Stylus Epic Zoom Camera Sony AC Adapter Laptop Charger Apricorn Hard Drive Upgrade Kit Corsair 1GB Desktop Memory
BeerAdvocate	Weizenbock (Sierra Nevada) Ovila Abbey Saison (Sierra Nevada) Stoudt's Abbey Double Ale Stoudt's Fat Dog Stout Juniper Black Ale

Table 6: Items for Amazon Clothing and Amazon Electronics are those with the largest amount of skew, i.e. large positive θ_2 in the recommender distribution. This occurs for items that are universally liked, universally despised, or items that polarize. Alternatively, the beers from BeerAdvocate are those that are least skewed and have the largest negative θ_2 .

Most skewed	Most Gaussian
The O.C. Season 2	The Rookie
Samurai X: Trust and Betrayal	The Fan
Aqua Teen Hunger Force, Vol. 2	Cadet Kelly
Sealab 2001: Season 1	Money Train
Aqua Teen Hunger Force: Vol. 1	Alice Doesn't Live Here
Gilmore Girls: Season 3	Sea of Love
Felicity: Season 4	Boiling Point
The O.C.: Season 1	True Believer
The Shield: Season 3	Stakeout
Queer as Folk: Season 4	The Package

Table 7: A list of the most skewed and most Gaussian rating distributions on Netflix. Note that the skewed movies are all exclusively TV shows whereas the tightly peaked ‘Gaussian’ ratings all correspond to blockbuster movies.

ing a full television series is a lot more time consuming than viewing an individual movie, and users are probably more likely to self select, especially before rating a later season.

Amazon Ratings: For both the Amazon Electronics and Amazon Clothing datasets, we found that they exhibited very skewed distributions that were not Gaussian. Items were frequently viewed as very good, or very bad, with variances that were not well approximated by a normal distribution. Deviations were less likely to occur naturally. Furthermore, this makes sense as users often do not go online and rate an item they bought previously unless they have a strong reason for wanting to do so, e.g. the product is defective. In Figures 5(a,c) we see that our model handles the skew and bimodal distribution well. Additionally in Table 6 we list the most skewed distributions within both the Amazon Clothing and Amazon Electronics datasets.

Possible Explanations: We can hypothesize that such a difference in behavior can be attributed to a number of biases. For example, it is likely that only fans of a TV series will rate a later season of the show. Additionally, in rating the show is the rating a statement of (a) “This was a good/bad season of this show,” (b) “This was a good/bad

season of TV,” or (c) “This was a good/bad piece of video entertainment?” Similarly, the selection bias arising from the additional effort to watch a full season of a show may also arise in the additional effort to go online and rate a product on Amazon. Additionally, we may only be able to compare quality against highly similar items as has been found in pricing studies in behavioral economics. From this we see polarized views from items that are difficult to evaluate or we have fewer instances to compare against, e.g. printers, seasons of a TV show, jeans, and more Gaussian ratings from items that we can compare against a wide variety of past experiences, e.g. comedy movies, action movies, beers.

While there is no definitive explanation for why these patterns emerge, it is clear that ratings have complex meaning. While our model is effective at fitting the wide range of distributions generated from such varied motives, we believe these discovered discrepancies demonstrate the need to further understand online ratings and better understand the latent factors that contribute to rating decision.

9. SUMMARY

In this paper we demonstrated that a single model suffices to provide both good recommendation performance, the ability to capture nontrivial bimodal and skewed distributions, the ability to analyze and group users and movies, and a mechanism for detecting spammers. Often these four problems are addressed by separate tools, often even by separate teams in commercial contexts. Our research shows that this is not needed. Instead, it is likely better to jointly model all these effects in a moderately detailed statistical generative process. While this may be costlier than each individual specialized solution, the overall engineering cost of a joint model is considerably less than maintaining these specialized methods. Furthermore, additional data can help improve *all* aspects simultaneously. We anticipate that large integrated data modeling will become a pervasive trend for recommendation, ranking and content analysis.

Acknowledgments: We would like to thank A. Ahmed for valuable discussions. This research was supported by funds from Google, Amazon, an IBM Faculty Award, a Google Focused Research Award, the National Science Foundation under Grants No. CNS-1314632, IIS-0953330, IIS-0916345, IIS-0911032, a National Science Foundation Graduate Research Fellowship (Grant No. DGE-1252522), and the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory, the U.S. Government, the National Science Foundation, or other funding parties. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes not withstanding any copyright notation here on.

10. REFERENCES

- [1] D. Agarwal and S. Merugu. Predictive discrete latent factor models for large scale dyadic data. In *KDD*, pg. 26–35. ACM, 2007.
- [2] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola. Distributed large-scale natural graph factorization. In *WWW*, 2013.
- [3] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing. Mixed-membership stochastic blockmodels. *JMLR*, 9:1981–2014, 2008.
- [4] D.J. Aldous. Representations for partially exchangeable arrays of random variables. *J. Multivariate Analysis*, 11(4):581–598, 1981.
- [5] A. Anagnostopoulos, A. Dasgupta, and R. Kumar. Approximation algorithms for co-clustering. In *PODS*, pg. 201–210, 2008. ACM.
- [6] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan. An introduction to mcmc for machine learning. *Machine Learning*, 50:5–43, 2003.
- [7] C. J. Burges, Q. V. Le, and R. Ragno. Learning to rank with nonsmooth cost functions. In B. Schölkopf, J. Platt, and T. Hofmann, eds., *NIPS 19*, 2007.
- [8] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley and Sons, 1991.
- [9] P. Diaconis and S. Janson. Graph limits and exchangeable random graphs, 2007. arXiv:0712.2749.
- [10] T.L. Griffiths and M. Steyvers. Finding scientific topics. *PNAS*, 101:5228–5235, 2004.
- [11] T. Haines. Gaussian conjugate prior cheat sheet. http://thaines.com/content/misc/gaussian_conjugate_prior_cheat_sheet.pdf, 2011.
- [12] K. A. Heller and Z. Ghahramani. Bayesian hierarchical clustering. In *ICML*, pages 297–304, 2005.
- [13] R. Herbrich, T. Minka, and T. Graepel. TrueskillTM: A Bayesian skill ranking system. In *NIPS*, 2007.
- [14] M. D. Hoffman and A. Gelman. The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo. Technical report, arXiv, 2011. <http://arxiv.org/abs/1111.4246>.
- [15] L. Hong, A. Ahmed, S. Gurumurthy, A. Smola, and K. Tsioutsoulis. Discovering geographical topics in the twitter stream. In *World Wide Web*, 2012.
- [16] D.N. Hoover. Relations on probability spaces and arrays of random variables. *Institute of Advanced Studies, Princeton, NJ*, 1979.
- [17] N. Jindal and B. Liu. Opinion spam and analysis. In *WSDM*, pg. 219–230. ACM, 2008.
- [18] O. Kallenberg. *Probabilistic symmetries and invariance principles*. Springer, 2005.
- [19] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*, pages 426–434, 2008.
- [20] Y. Koren, R.M. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.
- [21] D. Lee and H. Seung. Algorithms for non-negative matrix factorization. In T. K. Leen, T. G. Dietterich, and V. Tresp, eds., *NIPS 13*. MIT Press, 2001.
- [22] E.-P. Lim, V.A. Nguyen, N. Jindal, B. Liu, and H. W. Lauw. Detecting product review spammers using rating behaviors. In *CIKM*, pg. 939–948. ACM, 2010.
- [23] J. McAuley and J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Recommender Systems*, 2013.
- [24] J. McAuley, J. Leskovec, and D. Jurafsky. Learning attitudes and attributes from multi-aspect reviews. In *ICDM*, pg. 1020–1025. IEEE, 2012.
- [25] Kevin P. Murphy. *Machine learning: a probabilistic perspective*. MIT Press, 2012.
- [26] R. Neal. Markov chain sampling methods for dirichlet process mixture models. TR-9815, U. Toronto, 1998.
- [27] K. Palla, D. A. Knowles, and Z. Ghahramani. An infinite latent attribute model for network data. In *ICML*, 2012.
- [28] J. Pitman and M. Yor. The two-parameter poisson-dirichlet distribution derived from a stable subordinator. *A. Probability*, 25(2):855–900, 1997.
- [29] I. Porteous, E. Bart, and M. Welling. Multi-HDP: A non parametric bayesian model for tensor factorization. In D. Fox and C.P. Gomes, eds., *AAAI*, pg. 1487–1490. AAAI Press, 2008.
- [30] S. Rendle and L. Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *WSDM*, pages 81–90. ACM, 2010.
- [31] J. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *ICML*, 2005.
- [32] B. Van Roy and X. Yan. Manipulation-resistant collaborative filtering systems. In *ACM RecSys*, 2009.
- [33] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In W.W. Cohen, A. McCallum, and S.T. Roweis, eds., *ICML*, volume 307, pages 880–887. ACM, 2008.
- [34] M. S. Shotwell and E. H. Slate. Bayesian outlier detection with dirichlet process mixtures. *Bayesian Analysis*, 6(4):665–690, 2011.
- [35] A.P. Singh and G.J. Gordon. A unified view of matrix factorization models. In W. Daelemans, B. Goethals, and K. Morik, eds., In *ECML*, pages 358–373. 2008.
- [36] N. Srebro, N. Alon, and T. Jaakkola. Generalization error bounds for collaborative prediction with low-rank matrices. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *NIPS 17*, 2005. MIT Press.
- [37] C. Tan, E. H. Chi, D. Huffman, G. Kossinets, and A. J. Smola. Instant foodie: Predicting expert ratings from grassroots. In *CIKM*, 2013.
- [38] M. N. Volkovs and R.S. Zemel. Boltzrank: Learning to maximize expected ranking gain. In *ICML*, 2009.
- [39] M. Weimer, A. Karatzoglou, Q. Le, and A. J. Smola. Cofi rank - maximum margin matrix factorization for collaborative ranking. In *NIPS 20*, 2008.
- [40] J. Weston, S. Bengio, and N. Usunier. Wsabee: Scaling up to large vocabulary image annotation. In *IJCAI 23*, pg. 2764–2770. AAAI Press, 2011.
- [41] S.-H. Yang, B. Long, A.J. Smola, H. Zha, and Z. Zheng. Collaborative competitive filtering: learning recommender using context of user choice. In *SIGIR*, pages 295–304, 2011.
- [42] Z. Zheng, H. Zha, T. Zhang, O. Chapelle, K. Chen, and G. Sun. A general boosting method and its application to learning ranking functions for web search. In *NIPS 20*, pg. 1697–1704. MIT Press, 2008.