Scalable Recommendation with Poisson Factorization

ABSTRACT

We develop hierarchical Poisson matrix factorization (HPF) for recommendation. HPF models sparse user behavior data, large user/item matrices where each user has provided feedback on only a small subset of items. HPF handles both explicit ratings, such as a number of stars, or implicit ratings, such as views, clicks, or purchases. We develop a variational algorithm for approximate posterior inference that scales up to massive data sets, and we demonstrate its performance on a wide variety of real-world recommendation problems–users rating movies, users listening to songs, users reading scientific papers, and users reading news articles. Our study reveals that hierarchical Poisson factorization definitively outperforms previous methods, including nonnegative matrix factorization, topic models, and state-of-the-art probabilistic matrix factorization techniques.

1. INTRODUCTION

Recommendation systems are a vital component of the modern Web. They help readers effectively navigate otherwise unwieldy archives of information and help websites direct users to items—movies, articles, songs, apps—that they will like. A recommendation system is built from user behavior data, historical data about which items each user has consumed. First, a statistical machine learning algorithm is used to uncover behavioral patterns that reveal the various types of users and the items they tend to like. Then, the system exploits these discovered patterns to recommend future items to its users.

As an example, Figure 1 illustrates movie recommendation for a user U from the MovieLens data set [13]. This data set is a large sparse matrix where rows are people and columns are movies. Each entry of the matrix (indexed by a user and a movie) contains the rating that the user gave to the movie, or a zero if she has not seen it. The list of movies at the top of the Figure 2 shows that user U enjoys various types of drama movies (such as the war drama "Breaker Morant" and the romantic drama "Leaving Las Vegas"). Of course, she has only seen a handful of the available movies, and the goal of a recommendation system is to suggest other movies. The list of movies at the bottom of the figure was suggested by our algorithm. It includes other war drama movies (such as "Apocalypse Now") and other romantic drama (such as "Breakfast at Tiffany's"). In this paper, we develop a new algorithm for building recommendation systems that is both more efficient and performs better than the existing state-of-the-art.

Currently, the workhorse method for recommendation systems is matrix factorization (MF). MF represents users and items with low dimensional vectors and computes the affinity between a user and item (that is, whether the user will like it) with the dot product of their respective representations. MF is typically fit with squared loss, where the algorithm finds representations that minimize the squared difference between the predicted value and the observed rating. (This corresponds to a Gaussian model of the data [27].) MF has been extended in many ways to implement modern recommendation systems [7, 20, 26, 29].

However, the assumptions behind traditional MF are fundamentally flawed when analyzing real-world user behavior data. In real-world data, each user has only rated a small subset of the large population of available items. An item a user did *not* rate can arise in two ways: either she considered it and chose not to rate it or she did not consider it at all. Each user has a limited budget (of money, attention, or time) and therefore most of the unrated items in the matrix arise from users not considering (as opposed to actively disliking) them.

The issue with traditional MF is that it treats all the missing cells as observations, as though every user has enough attention to consider every available item and decide whether to rate it. Thus, the missing cells are seen as evidence for users not liking the items, and this significantly biases the learned representations. To address the problem, researchers have patched MF in a variety of ways, for example by artificially down-weighting the contribution of the unrated items [16], by sub-sampling from the unrated items to give equal weight to the rated items [9, 8], or by explicitly modeling the unrated items as missing data [25].

This issue is particularly critical when analyzing binary data, such as product purchases or webpage clicks. Binary behavior data records whether each user consumed an item but does provide a rating. Building recommendation systems from such matrices is known as one-class collaborative filtering or recommendation with implicit feedback [16, 25].

In this paper, we develop a Bayesian Poisson factoriza-

Figure 1: The top 5 movies in each of the top 4 components of the user U illustrated in Fig 2.

"Drama, Romance"	"Drama"	"Children's Drama"	"Drama, War"
Breakfast at Tiffany's	Jean de Florette	The Secret Garden	The Bridge on the River Kwai
Casablanca	Manon of the Spring	The Secret of Roan Inish	The Right Stuff
The Graduate	Diva	A Little Princess	Patton
Shakespeare in Love	The Return of Martin Guerre	Fly Away Home	The Killing Fields
The African Queen	Blue Velvet	Black Beauty	Gandhi

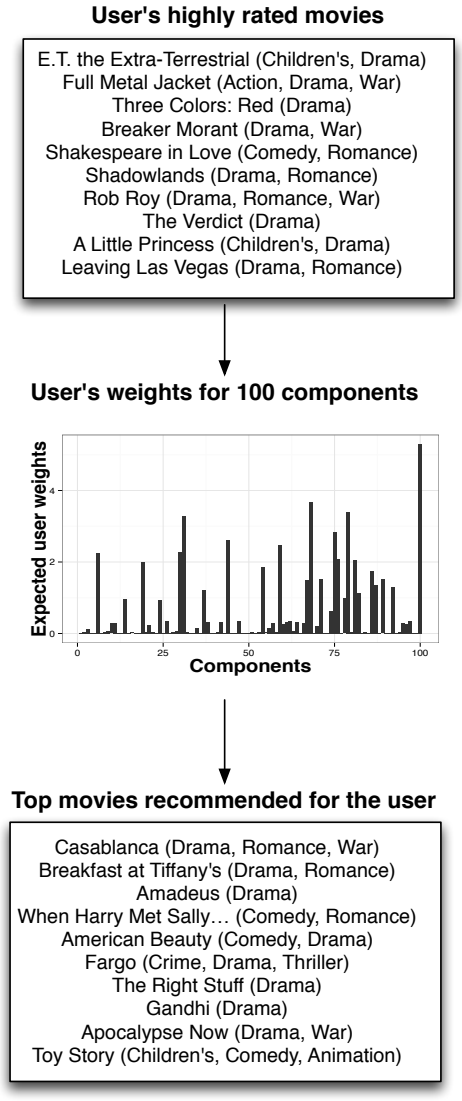


Figure 2: An illustration showing a subset of the highly rated movies of a selected user U in the MovieLens data set [13], and a subset of the movies in the top 15 recommended to the user by our algorithm. The expected user's K-vector of weights θ_u, inferred by our algorithm is shown. In our analysis, K was set to 100.

tion model as an alternative to traditional MF for building recommendation systems. Our model implicitly assumes that each user has a limited budget with which to consume items [12], and thus an item that a user has consumed provides a stronger signal about her preferences than an item that a user has not consumed. With several kinds of data sets—users rating movies [13, 21], users listening to songs [2], and users reading scientific papers [17]—we demonstrate that Poisson factorization leads to better recommendations than both traditional matrix factorization and its variants that adjust for sparse data.

Furthermore, Poisson factorization is computationally more efficient than traditional MF. Most algorithms for fitting MF must iterate over all user/item pairs, which is expensive for even modestly-sized user behavior matrices and cannot take advantage of the sparsity of the data [16]. (To address this issue, practical applications of matrix factorization rely on stochastic optimization [23].) In this paper, we derive efficient variational inference algorithms for Poisson factorization that take advantage of the sparsity of the data. Our algorithms need only iterate over the non-zero entries of the user behavior matrix. This lets us handle data at a scale that basic (non-stochastic) MF algorithms cannot handle.

2. POISSON RECOMMENDATION

We are given data about users and items. Each user has consumed and possibly rated a set of items. The observation y_{ui} is the rating that user i gave to item j, or zero if no rating was given. (In simple consumer data, y_{ui} equals one if user u consumed item i and zero otherwise.) User behavior data, such as purchases, ratings, clicks, "likes", or "check ins", are typically sparse. Most of the values of the matrix y are zero.

We model this data with factorized Poisson distributions [5]. Each user u and each item i is associated with a K-vector of positive weights, θ_u and β_i respectively. (The number K is fixed in advance.) The user/item observation y_{ui} is modeled with a Poisson parameterized by the inner product of the user's and item's weights $y_{ui} \sim \text{Poisson}(\theta_u^\top \beta_i)$. This is a variant of probabilistic matrix factorization [28] but where each user and item's weights are positive [22]—we place Gamma priors on them— and where the Poisson likelihood replaces the Gaussian likelihood.

This Poisson model has good properties when modeling user/item data. It implicitly assumes that each user consumes a negative-binomial number of items—a heavy-tailed distribution that is known to fit user activity well [12]–and this forces it to down-weight the contribution of the items that each user did not consume. The reason is that the model has two ways of explaining that a user does not consume an item: either she is not interested in it or she would be but has already consumed her allotted negative-binomial number. In contrast, a user that consumes an item must be interested in it. Thus, the model benefits more from mak-

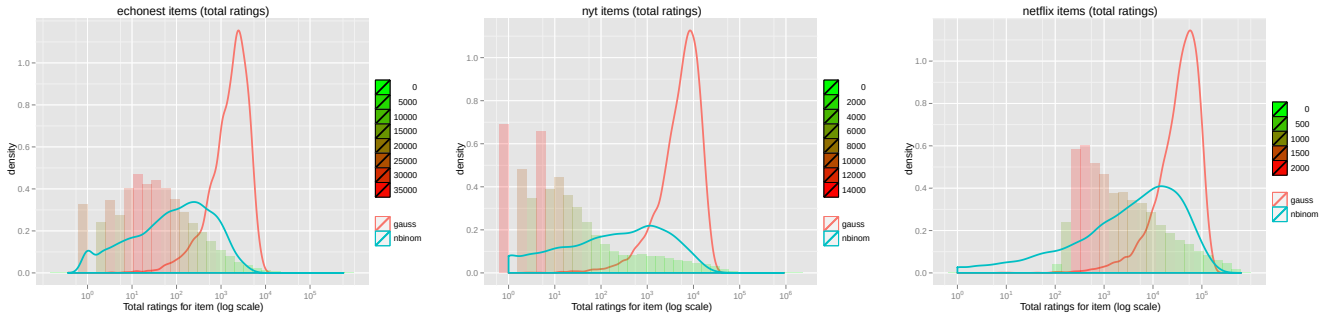


Figure 3: Empirical distribution of item popularity on real datasets, with fitted negative binomial and Gaussian distributions. The distributions were fit using maximum likelihood estimation. The negative binomial places significant probability mass on the left tail, i.e., items with few ratings. The colored bars show that such items are the most frequent. In contrast, the Gaussian distribution places negligible mass on the left tail and mainly captures popular items. The mode of the negative binomial distribution is also closer to the empirical mode than the Gaussian distribution.

ing a consumed user/item pair more similar than making an unconsumed user/item pair less similar.

Classical matrix factorization is based on Gaussian likelihoods (i.e., squared loss), which gives equal weight to consumed and unconsumed items. Consequently, when faced with a sparse matrix, matrix factorization places more total emphasis on the unconsumed user/item pairs. To address this, researchers have patched the model in complex ways, for example, by including per-observation confidences [21] or considering all zeroes to be hidden variables [25]. Poisson factorization more naturally solves this problem by capturing each user's rate of consumption.

As an example, consider two similar science fiction movies, "Star Wars" and "The Empire Strikes Back", and consider a user who has seen one of them. The Gaussian model pays an equal penalty for making the user similar to these items as it does for making the user different from them—with quadratic loss, seeing "Star Wars" is evidence for liking science fiction, but not seeing "The Empire Strikes Back" is evidence for disliking it. The Poisson model, however, will prefer to bring the user's latent weights closer to the movies' weights because it favors the information from the user watching "Star Wars". Further, because the movies are similar, this increases the Poisson model's predictive score that a user who watches "Star Wars" will also watch "The Empire Strikes Back".

2.1 The generative model

The generative process of the hierarchical Poisson factorization model (HPF) is as follows:

1. For each user u, choose activity
$$\xi_u \sim \text{Gamma}(a', b').$$
2. For each user u and each component k, choose weight
$$\theta_{uk} \sim \text{Gamma}(a, \xi_u).$$
3. For each item i, choose popularity
$$\eta_i \sim \text{Gamma}(c', d').$$
4. For each item i and each component k, choose weight
$$\beta_{ik} \sim \text{Gamma}(c, \eta_i).$$
5. For each user u and item i, choose rating
$$y_{ui} \sim \text{Poisson}(\theta_u^\top \beta_i).$$

The model places a Gamma prior on each of the Gamma rate parameters governing user and item weights. The prior on ξ_u captures the uncertainty around user u's consumption rate or activity; the prior on η_i captures the uncertainty around item i's popularity among users.

We also study a sub-class of the HPF where we fix the rate parameter ξ_u for each user, and the rate parameter η_i for each item. We call this model the Bayesian Poisson Factorization (BPF) model.

The HPF model is more powerful than the BPF because it allows for the sharing of statistical strength. In the HPF, for example, learning about the number of items consumed by one set of users provides weak, indirect evidence relevant to other users consumption. This can improve predictions, especially for users with little activity. A similar argument favors hierarchy over item popularities. In the HPF, we fix the shape parameters and share them across users and items. For a large shape, the weights will tend to have similar magnitude, and the representations will be dense. In contrast, a small shape will result in a sparse representation. Finally, the Gamma prior is conjugate to the Gamma distribution with fixed shape governing the weights. This results in a computationally efficient algorithm.

The posterior distribution of the latent variables $p(\theta, \beta, \xi, \eta \mid y)$ embeds users and items in a latent space of K-dimensional positive vectors.

We use the HPF to recommend items to users by predicting which of the unconsumed items each will like. We rank each user's unconsumed items by their posterior expected Poisson parameters,

$$\text{score}_{ui} = \text{E}[\theta_u^\top \beta_i \mid y]. \tag{1}$$

This amounts to asking the model to rank by probability which of the presently unconsumed items each user will likely consume in the future.

With these details in place, we highlight two statistical properties of the model. First, we mentioned above that the marginal distribution of each user's ratings is a negative binomial. Let $y_u = \sum_i y_{ui}$ be the sum of the ratings for user

u. Since each of the terms in the sum is a Poisson distribution with rate $\theta_u^\top \beta_i$, the sum is itself a Poisson random variable [18].

$$y_u \sim \text{Poisson}\left(\theta_u^\top (\textstyle\sum_i \beta_i)\right). \quad (2)$$

Holding the item weights fixed, note that the rate of this Poisson is a sum of scaled Gamma random variables $\beta_{ik}\theta_{uk}$, which is itself a Gamma variable [24]. Thus the marginal distribution of y_u is from an integrated Gamma-Poisson distribution, which is a negative binomial [10].

Second, the likelihood of the observed data depends only on the consumed items, that is, the non-zero elements of the user/item matrix y. Given the latent variables θ_u and β_i, the Poisson distribution of y_{ui} is

$$p(y_{ui} \mid \theta_u, \beta_i) = \left(\theta_u^\top \beta_i\right)^y \exp\left\{-\theta_u^\top \beta_i\right\} / y_{ui}! \quad (3)$$

Recall the elementary fact that $0! = 1$. The log probability of the complete matrix y is

$$\log p(y \mid \theta, \beta) = \left(\textstyle\sum_{\{y_{ui}>0\}} y_{ui} \log(\theta_u^\top \beta_i) - \log y_{ui}!\right) \quad (4)$$

$$- \left(\textstyle\sum_u \theta_u\right)^\top \left(\textstyle\sum_i \beta_i\right). \quad (5)$$

That this likelihood depends only on the non-zero elements facilitates inference with sparse matrices. In contrast, classical matrix factorization methods, especially when applied to massive data sets, must address the zeros either through sub-sampling [8] or approximation [16].

2.2 Related work

The roots of Poisson factorization come from nonnegative matrix factorization [22], where the objective function is equivalent to a factorized Poisson likelihood. Placing a Gamma prior on the user weights gives the GaP model [5], which was developed as an alternative text model to latent Dirichlet allocation (LDA) [3].

A difference between our treatment and GaP is that GaP fits the item weights with maximum likelihood via the expectation maximization algorithm. Our model places Gamma priors on these weights (step 2, above) and we approximate the full posterior with variational inference. Placing priors on both sets of weights further regularizes the model and lets us use the same inferential machinery in both user-space and item-space.

We note that GaP was developed as an alternative to LDA. In Appendix A, we show that LDA can be reinterpreted as an instance of Poisson factorization where we condition on the user counts and use an alternative prior on the item weights. (This connection was previously unknown.)

Independently of GaP, Bayesian Poisson factorization has been studied in the signal processing community for performing source separation from spectrogram data [6, 14]. This research includes variational approximations to the posterior, though the issues and details around spectrogram data differ significantly from user behavior data we consider and our derivation below (based on auxiliary variables) is more direct. As future work, the methods developed here could lead to improved methods for massive simultaneous analysis of audio spectrograms. In the context of network data, a Poisson model of overlapping communities was described by the authors of [1]. As with GaP, this model is unregularized and the authors fit the model with maximum likelihood via expectation maximization.

We discuss further differences between our method and previous work on matrix factorization in the empirical results below.

3. VARIATIONAL INFERENCE

The key computation for the HPF is the posterior distribution of the user weights θ_{uk}, item weights β_{ik}, user activity ξ_u and item popularity η_i given an observed matrix of user behavior y,

$$p(\theta, \beta, \xi, \eta \mid y) = \frac{p(\theta|\xi)p(\xi)p(\eta)p(\beta|\eta)p(y \mid \theta, \beta, \xi, \eta)}{\int_\theta \int_\xi \int_\eta \int_\beta p(\theta|\xi)p(\xi)p(\beta)p(\beta|\eta)p(y \mid \theta, \beta, \xi, \eta)}.$$

We need the posterior to form recommendations with the posterior expectations in Equation 1.

As for many Bayesian models of interest, however, the posterior is intractable to compute exactly. The problem is with the denominator, which is the marginal probability of the observed matrix and involves a complicated and high-dimensional integral. In this section, we show how to efficiently approximate the posterior with mean-field variational inference.

Variational inference is a general strategy for approximating posterior distributions in complex probabilistic models [19, 30]. Variational inference algorithms posit a family of distributions over the hidden variables, indexed by free "variational" parameters, and then find the member of that family that is closest in KL divergence to the true posterior. (The form of the family is chosen to make this optimization possible.) Thus, variational inference turns the inference problem into an optimization problem. Variational algorithms tend to scale better than alternative sampling-based approaches, like Monte Carlo Markov chain sampling, and have been deployed to solve many applied problems with complex models, including large-scale recommendation [25].

We develop mean-field variational inference algorithm for the HPF. We first describe the mean-field variational family and the corresponding variational objective. We then derive a batch algorithm that fits the variational distribution by repeatedly cycling through the non-zero data and updating its estimates of the latent representations. The simple structure of the algorithm lets us scale our approach to data sets like the full Netflix data (18,000 movies and 480,000 users) on a single CPU.

Before beginning these derivations, however, we give an alternative formulation of the model in which we add a layer of latent variables. These auxiliary variables allow us to take advantage of some general results for variational algorithms [11, 15]. For each user and item we add K latent variables $z_{uik} \sim \text{Poisson}(\theta_{uk}\beta_{ik})$, which are integers that sum to the user/item value y_{ui}. A sum of Poisson random variables is itself a Poisson with rate equal to the sum of the rates. Thus, these new latent variables preserve the marginal distribution of the observation, $y_{ui} \sim \text{Poisson}(\theta_u^\top \beta_i)$. These variables can be thought of as the contribution from component k to the total observation y_{ui}. Note that when $y_{ui} = 0$, these auxiliary variables are not random—the posterior distribution of z_{ui} will place all its mass on the zero vector. Consequently, our inference procedure need only consider z_{ui} for those user/item pairs where $y_{ui} > 0$.

3.1 Mean-field variational inference

The latent variables in the model are user weights θ_{uk}, item weights β_{ik}, and user-item contributions z_{uik}, which

we represent as a K-vector of counts z_{ui}. The mean-field family considers these variables to be independent and each governed by its own distribution,

$$q(\beta, \theta, \xi, \eta, z) = \prod_{i,k} q(\beta_{ik} \mid \lambda_{ik}) \prod_{u,k} q(\theta_{uk} \mid \gamma_{uk}) \prod_{u} q(\xi_u \mid \kappa_u) \prod_{i} q(\eta_i \mid \tau_i) \prod_{u,i} q(z_{ui} \mid \phi_{ui}). \tag{6}$$

Though the variables are independent, this is a flexible family of distributions because each variable is governed by its own free parameter. (We postpone specifying the forms of each of these factors to below.)

After specifying the family, we fit the variational parameters $\nu = \{\lambda, \gamma, \kappa, \tau, \phi\}$ to minimize the KL divergence to the posterior

$$\nu^* = \arg\min_{\nu} \mathrm{KL}(q(\beta, \theta, \kappa, \tau, z \mid \nu) || p(\beta, \theta, \kappa, \tau, z \mid y)).$$

We then use the corresponding variational distribution $q(\cdot \mid \nu^*)$ as a proxy for the posterior.[1] The mean-field factorization facilitates both optimizing the variational objective and downstream computations with the approximate posterior, such as the recommendation score of Equation 1.

3.2 Complete conditionals

Variational inference fits the variational parameters to minimize their KL divergence to the posterior. For a large class of models, we can easily perform this optimization with a coordinate-ascent algorithm, one in which we iteratively optimize each variational parameter while holding the others fixed. Specifically, we appeal to general results about the class of *conditionally conjugate* models [11, 15]. We define the class, show that the HPF is in the class, and then give the variational inference algorithm.

A *complete conditional* is the conditional distribution of a latent variable given the observations and the other latent variables in the model. A conditionally conjugate model is one where each complete conditional is in an exponential family (such as a Gaussian, Gamma, Poisson, multinomial, or others). This is a large class of models.

The HPF, with the z_{ui} variables described above, is a conditionally conjugate model. (Without the auxiliary variables, it is not conditionally conjugate.) For the user weights θ_{uk}, the complete conditional is a Gamma,

$$\theta_{uk} \mid \beta, \xi, z, y \sim \mathrm{Gamma}(a + \textstyle\sum_i z_{uik}, \xi_u + \sum_i \beta_{ik}). \tag{7}$$

The complete conditional for item weights β_{ik} is symmetric,

$$\beta_{ik} \mid \theta, \eta, z, y \sim \mathrm{Gamma}(a + \textstyle\sum_u z_{uik}, \eta_i + \sum_i \theta_{uk}). \tag{8}$$

These distributions stem from conjugacy properties between the Gamma and Poisson. In the user weight distribution, for example, the item weights β_{ik} act as "exposure" variables [10]. (The roles are reversed in the item weight distribution.) We can similarly write down the complete conditionals for the user activity ξ_u and the item popularity η_i.

$$\begin{aligned} \xi_u \mid \theta &\sim \mathrm{Gamma}(a' + Ka, b' + \textstyle\sum_k \theta_{uk}). \\ \eta_i \mid \beta &\sim \mathrm{Gamma}(c' + Kc, d' + \textstyle\sum_k \beta_{ik}). \end{aligned} \tag{9}$$

The final latent variables are the auxiliary variables. Recall that each z_{ui} is a K-vector of Poisson counts that sum to the observation y_{ui}. The complete conditional for this vector is

$$z_{ui} \mid \beta, \theta, y \sim \mathrm{Mult}\left(y_{ui}, \frac{\theta_u \beta_i}{\sum_k \theta_{uk} \beta_{ik}}\right). \tag{10}$$

Though these variables are Poisson in the model, their complete conditional is multinomial. The reason is that the conditional distribution of a set of Poisson variables, given their sum, is a multinomial for which the parameter is their normalized set of rates. See [18] (and Appendix A).

3.3 Variational algorithm

We now derive variational inference for the HPF. First, we set each factor in the mean-field family (Equation 6) to be the same type of distribution as its complete conditional. The complete conditionals for the item weights β_{ik} and user weights θ_{uk} are Gamma distributions (Equations 9 and 8); thus the variational parameters λ_{ik} and γ_{uk} are Gamma parameters, each containing a shape and a rate. Similarly, the variational user activity parameters κ_u and the variational item popularity parameter τ_i are Gamma parameters, each containing a shape and a rate. The complete conditional of the auxiliary variables z_{uik} is a multinomial (Equation 10); thus the variational parameter ϕ_{ui} is a multinomial parameter, a point on the K-simplex, and the variational distribution for z_{ui} is $\mathrm{Mult}(y_{ui}, \phi_{ui})$.

In coordinate ascent we iteratively optimize each variational parameter while holding the others fixed. In conditionally conjugate models, this amounts to setting each variational parameter equal to the expected parameter (under q) of the complete conditional.[2] The parameter to each complete conditional is a function of the other latent variables (by definition) and the mean-field family sets all the variables to be independent. These facts guarantee that the parameter we are optimizing will not appear in the expected parameter.

For the user and item weights, we update the variational shape and rate parameters. (We denote shape with the superscript "shp" and rate with the superscript "rte".) The updates are

$$\gamma_{uk} = \langle a + \textstyle\sum_i y_{ui} \phi_{uik}, b + \sum_i \lambda_{ik}^{\mathrm{shp}} / \lambda_{ik}^{\mathrm{rte}} \rangle \tag{11}$$

$$\lambda_{ik} = \langle c + \textstyle\sum_u y_{ui} \phi_{uik}, d + \sum_u \gamma_{ik}^{\mathrm{shp}} / \gamma_{ik}^{\mathrm{rte}} \rangle. \tag{12}$$

These are expectations of the complete conditionals in Equations 9 and 8. In the shape parameter, we use that the expected count of the kth item in the multinomial is $\mathrm{E}_q[z_{uik}] = y_{ui}\phi_{uik}$. In the rate parameter, we use that the expectation of a Gamma variable is the shape divided by the rate.

For the variational multinomial the update is

$$\phi_{ui} \propto \exp\{\Psi(\gamma_{uk}^{\mathrm{shp}}) - \log \gamma_{uk}^{\mathrm{rte}} + \Psi(\lambda_{ik}^{\mathrm{shp}}) - \log \lambda_{ik}^{\mathrm{rte}}\}, \tag{13}$$

[1] In fact, variational inference optimizes an equivalent objective that is the KL divergence up to an additive constant. But this detail is not needed here.

[2] It is a little more complex then this. We must be working with the natural parameterization of the corresponding exponential families. For details, see [15].

For all users and items, initialize the user parameters γ_u, $kappa_u^{\text{rte}}$ and item parameters λ_i, τ_i^{rte} to the prior with a small random offset. Set the user activity and item popularity shape parameters:

$$\kappa_u^{\text{shp}} = a + Ka'; \quad \tau_i^{\text{shp}} = c + Kc'$$

Repeat until convergence:

1. For each user/item such that $y_{ui} > 0$, update the multinomial:

$$\phi_{ui} \propto \exp\{\Psi(\gamma_{uk}^{\text{shp}}) - \log \gamma_{uk}^{\text{rte}} + \Psi(\lambda_{ik}^{\text{shp}}) - \log \lambda_{ik}^{\text{rte}}\}.$$

2. For each user, update the user weight and activity parameters:

$$\gamma_{uk}^{\text{shp}} = a + \textstyle\sum_i y_{ui}\phi_{uik}$$
$$\gamma_{uk}^{\text{rte}} = \frac{\kappa_u^{\text{shp}}}{\kappa_u^{\text{rte}}} + \textstyle\sum_i \lambda_{ik}^{\text{shp}}/\lambda_{ik}^{\text{rte}}$$
$$\kappa_u^{\text{rte}} = b' + \sum_k \Psi(\gamma_u^{\text{shp}}) - \log(\gamma_u^{\text{rte}})$$

3. For each item, update the item weight and popularity parameters:

$$\lambda_{ik}^{\text{shp}} = c + \textstyle\sum_u y_{ui}\phi_{uik}$$
$$\lambda_{ik}^{\text{rte}} = \frac{\tau_i^{\text{shp}}}{\tau_i^{\text{rte}}} + \textstyle\sum_u \gamma_{uk}^{\text{shp}}/\gamma_{uk}^{\text{rte}}$$
$$\tau_i^{\text{rte}} = d' + \sum_k \Psi(\lambda_i^{\text{shp}}) - \log(\lambda_i^{\text{rte}})$$

Figure 4: Batch variational inference for Poisson factorization. Each iteration only needs to consider the non-zero elements of the user/item matrix.

where $\Psi(\cdot)$ is the digamma function (the first derivative of the log Γ function). This update comes from the expectation of the log of a Gamma variable, for example $\mathrm{E}_q[\log \theta_{uk}] = \Psi(\gamma_{nk}^{\text{shp}}) - \log \gamma_{nk}^{\text{rte}}$.

The coordinate ascent algorithm iteratively executes these updates (see Figure 4). This algorithm is very efficient on sparse matrices. In step 1, the algorithm only needs to update variational multinomials for the non-zero user/item observations y_{ui}. In steps 2 and 3, the sums over users and items also only need to consider non-zero observations. In contrast, fitting a traditional matrix factorization with squared loss must iteratively consider every cell of the matrix, both zeros and non-zeros. This makes matrix factorization difficult to fit with large matrices, though there are innovative solutions based on stochastic optimization [23].

4. EMPIRICAL STUDY

We studied the Bayesian Poisson Factorization (BPF) algorithm of Figure 4 on a variety of large real data sets. We demonstrate in this section that BPF definitively outperforms matrix factorization (MF) in predictive performance on all data sets.

Data Sets. We study the Bayesian Poisson Factorization algorithm in Figure 4 on several large data sets:

- The **New York Times** data set with 1,615,675 users, 103390 articles movies and 80,071,435 ratings.
- The **Netflix** data set [21] with 480,000 users, 17,770 movies and 100 million ratings is similar to MovieLens but significantly larger. Unlike the MovieLens data, the Netflix data set is highly skewed and is more realistic: Some users rate more than 10,000 movies, while others rate less than 5 [28].
- The **Mendeley** data set [17] of scientific articles is a binary matrix of 80,000 users and 260,000 articles, where an article is considered consumed by a user if it's in the user's library.
- The **Echo Nest** music data set [2] consists of 1 million users, 385,000 distinct songs and 48 million (user, song, play count) triplets.

The scale and diversity of these data sets enables a robust evaluation of our algorithm. Both the Mendeley and the Echo Nest data sets are sparse in comparison to the movie data: only 0.001% of the matrix of ratings is non-zero in Mendeley, while 1% of the ratings are non-zero in Netflix and 4% in MovieLens. Further, the Mendeley data set has many more articles than users.

Pre-processing and training. We now describe how we the pre-process the input data, train the model, and assess the predictive accuracy on a held-out set.

Prior to training, we randomly select 20% of ratings in each data set to be used as a held-out test set comprised of items that the user has consumed. During training, these test set observations are treated as zeros. Additionally, we set aside 1% of the training ratings as a validation set and use it to determine the convergence of our algorithm.

During training, the input to our algorithm is the count data, for example, the movie ratings or the play count of a song. Notice that for the Mendeley data set, the input is binary. During training, we fix the shape and rate hyperparameters. This gives good performance on our validation set. We find that the algorithm is insensitive to small changes in the hyper-parameters.

We terminate the training process when the BPF algorithm converges. The convergence is measured by computing the prediction accuracy on our validation set. We approximate the probability that a user consumed an item using the variational approximations to posterior expectations of θ_u and β_i, and compute the average predictive log likelihood of the validation ratings. The BPF algorithm stops when the change in log likelihood is less than 0.0001%.

Exploratory analysis. The fitted model can be explored to discover latent structure among items and users and to confirm that the model is capturing the components in the data in a reasonable way. For example, in Figure 7 we illustrate the components discovered by our algorithm on the MovieLens and the Netflix movie ratings and the Mendeley data of users and scientific articles. For each data set, the illustration shows the top 10 items—items sorted in decreasing order of their expected weight β_i—from three of the 100 components discovered by our algorithm. These components naturally organize the movies and articles, and enable recommendation of new items to the user.

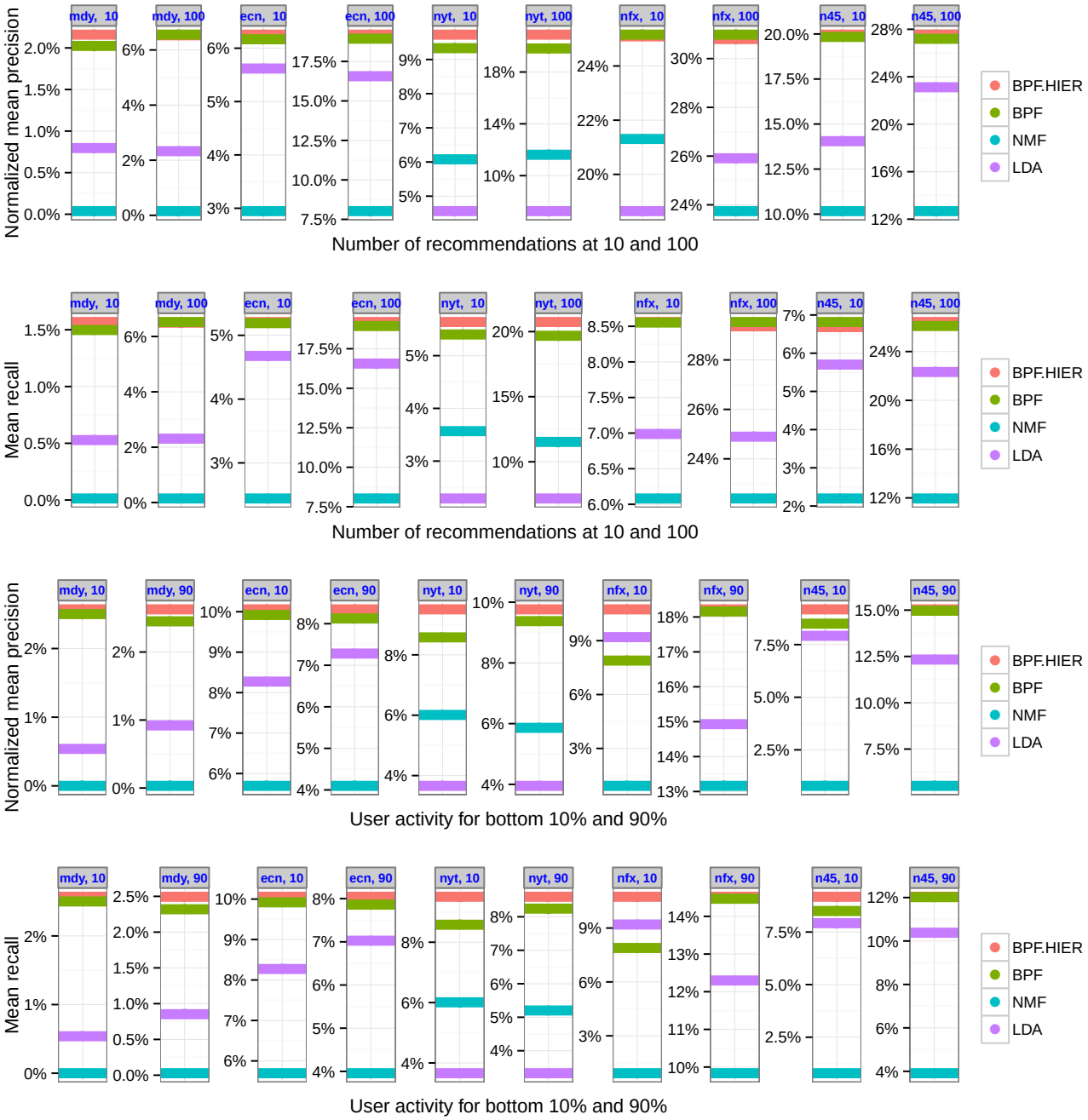
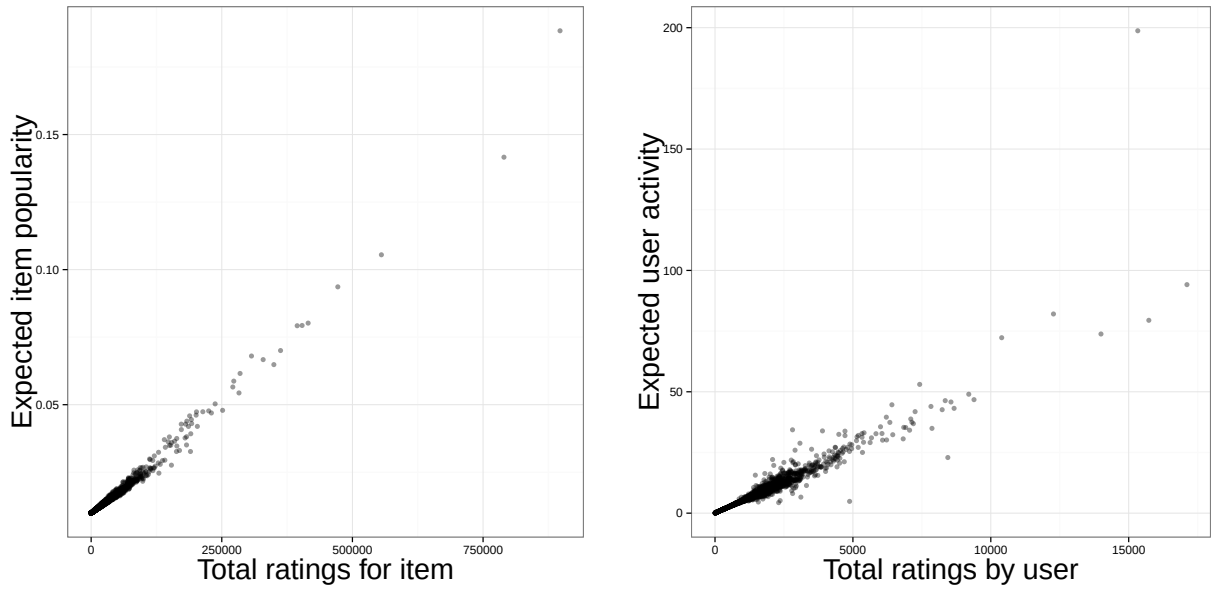


Figure 5: Predictive performance on datasets. The top two plots show normalized mean precision and mean recall at 10 and 100 recommendations. The bottom two plots show normalized mean precision and mean recall for the top 10% and 90% of users with lowest activity.

In Figure 2 we show a subset of the highly rated movies of a user from the MovieLens data set [13]. The top 15 movies recommended to this user using the trained BPF model, are also shown. The user's ratings are for primarily drama movies. We movies BPF recommends closely resemble the types of drama movies she is interested in, for example, "Children's drama" or "War drama". The expected user's K-vector of weights θ_u, inferred by our algorithm, is shown in Figure 2. In our analysis, K was set to 100. The θ_u are not sparse because the user's views span a range of movies in the small data set.

Testing. During testing, we generate the top M recommendations for each user as those items with the highest predictive score in Equation 1. The ranked list of items predicted for each user includes items in the test set, as well as items in the training set that were zeros. We compute precision-at-M, which measures the fraction of the top M recommendations present in the test set, varying M from



(a) (b)

Figure 6: Popularity of articles on the NYT dataset.

Table 1: Most popular articles in the NYT dataset

Title	Expected popularity	Number of factors
Blasts at Boston Marathon Kill 3 and Injure 100	0.043	1
FBI Posts Images of Pair Suspected in Boston Attack	0.042	1
Bombing Inquiry Turns to Motive and Russia Trip	0.037	1
War Zone at Mile 26: 'So Many People Without Legs'	0.025	1
Suspects Seemed Set for Attacks Beyond Boston	0.031	1
In Signal Image From Boston Bombing a Father Sees His Son	0.029	1
Skeleton in British Parking Lot Hailed as Richard III	0.027	4
Surviving Suspect is Charged by US in Boston Attack	0.027	1
The Boy With A Thorn In His Joints	0.026	3
Drowned in a Stream of Prescriptions	0.025	3

10 to 100 items. Likewise, we compute recall-at-M, which captures the fraction of items in the test set present in the top M recommendations.

Baselines. We compare our performance against traditional matrix factorization (MF). Ratings are modeled as

$$y_{ui} = c + a_u + b_i + \theta_u^\top \beta_i,$$

where c denotes a global intercept term, a_u captures the relative activity of the u-th user, and b_i accounts for the popularity of the i-th item. The final term quantifies interactions between a user and item via their K-dimensional latent factors, with θ_u specifying the user's interests and β_i describing the item's attributes. The model is fit by stochastic gradient descent using the open source Vowpal Wabbit package [31] to minimize squared error between predicted and actual ratings.

We note that while BPF takes only the non-zero observed ratings as input, traditional matrix factorization requires that we provide explicit zeros in the ratings matrix as negative examples. In practice, this amounts to either treating all missing ratings as zeros and down-weighting to balance the relative importance of observed and missing ratings [16], or generating negatives by randomly sampling from missing ratings in the training set [8]. We take the latter approach for computational convenience, employing two popular sampling schemes. In the first, denoted MF-UNI, we sample a fixed number of negative examples uniformly at random for each user such that there are approximately the same number of positive and negative examples. In the second, termed MF-POP, we sample users by activity—the number of items rated in the training set—and items by popularity—the number of training ratings an item received.

For MF, we cross-validate over the gradient update step size, the strength of an L_2-regularization term across all weights, and the number of passes over the training data. This results in a reasonably expensive grid search, from which we select the model that performs best on the validation set for each data set.

5. DISCUSSION

In settings where there are many more items than the typical user can consume, unobserved consumption is likely explained by finite attention, as opposed to an active dis-

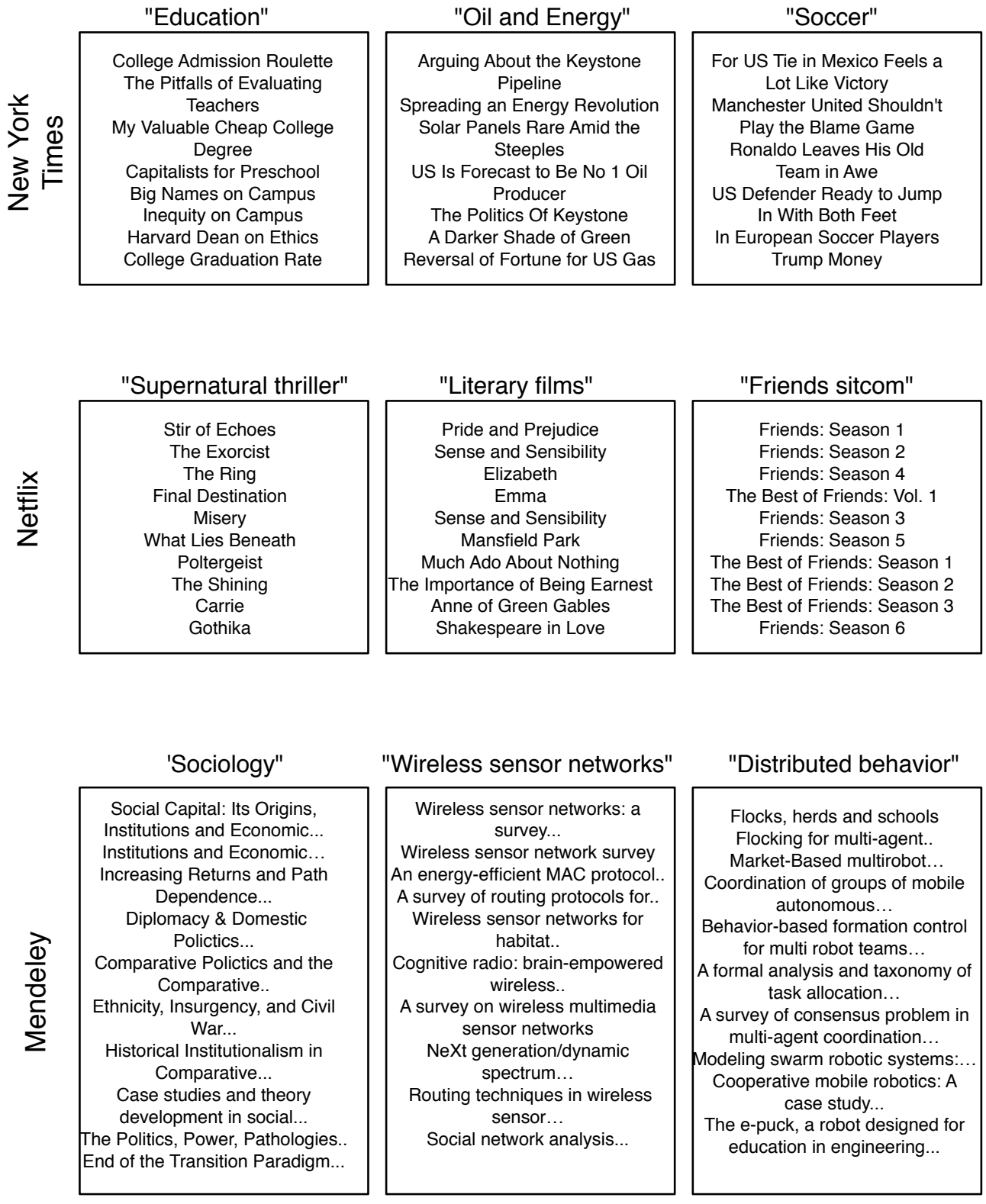


Figure 7: The top 10 items by the expected weight β_i from three of the 100 components discovered by our algorithm.

like for the associated content. Likewise, a user's choice in selecting a particular set of items amongst the many available options is a relatively strong indicator of her interests. BPF captures these features of sparse user data via the Poisson likelihood, which appropriately balances strong signals of consumption with weaker signals of unobserved activity.

Conveniently, the same Poisson likelihood also leads to computationally efficient inference on sparse data sets, as it requires evaluation of only the consumed user-item pairs, which comprise a small fraction of all possible observations. This avoids the issue faced by traditional matrix factorization in down-weighting or sampling negative examples during training. In addition to this computational advantage, BPF empirically outperforms classical MF across a wide array of data sets—from movies to music to scientific articles—in recommending relevant content to users.

6. REFERENCES

[1] B. Ball, B. Karrer, and M. Newman. Efficient and principled method for detecting communities in networks. *Physical Review E*, 84(3), Sept. 2011.

[2] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

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

[4] W. Buntine and A. Jakulin. Applying discrete PCA in data analysis. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pages 59–66. AUAI Press, 2004.

[5] J. Canny. GaP: A factor model for discrete data. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2004.

[6] A. Cemgil. Bayesian inference for nonnegative matrix factorization models. *Computational Intelligence and Neuroscience*, 2009.

[7] G. Dror, N. Koenigstein, and Y. Koren. Web-scale media recommendation systems. *Proceedings of the IEEE*, 100(9):2722–2736, 2012.

[8] G. Dror, N. Koenigstein, Y. Koren, and M. Weimer. The Yahoo! music dataset and KDD-cup '11. *Journal of Machine Learning Research*, 18:8–18, 2012.

[9] Z. Gantner, L. Drumond, C. Freudenthaler, and L. Schmidt-Thieme. Bayesian personalized ranking for non-uniformly sampled items. *JMLR W&CP*, Jan 2012.

[10] A. Gelman, J. Carlin, H. Stern, and D. Rubin. *Bayesian Data Analysis*. Chapman & Hall, London, 1995.

[11] Z. Ghahramani and M. Beal. Propagation algorithms for variational Bayesian learning. In *Neural Information Processing Systems*, pages 507–513, 2001.

[12] G. Goodhardt, S. Ehrenberg, and C. Chatfield. The Dirichlet: A comprehensive model of buying behavior. *Journal of the Royal Statistical Society, Series A*, 147(5):621–655, 1984.

[13] 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. ACM, 1999.

[14] M. Hoffman. Poisson-uniform nonnegative matrix factorization. In *ICASSP*, 2012.

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

[16] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. *Data Mining*, Jan 2008.

[17] J. Jack, Kris anwd Hammerton, D. Harvey, J. J. Hoyt, J. Reichelt, and V. Henning. MendeleyÊijs reply to the datatel challenge. *Procedia Computer Science*, 1(2):1–3, 2010.

[18] N. Johnson, A. Kemp, and S. Kotz. *Univariate Discrete Distributions*. John Wiley & Sons, 2005.

[19] M. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul. Introduction to variational methods for graphical models. *Machine Learning*, 37:183–233, 1999.

[20] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434, New York, NY, USA, 2008. ACM.

[21] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[22] D. Lee and H. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, October 1999.

[23] J. Mairal, J. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11:19–60, 2010.

[24] L. Norman, S. Kotz, and N. Balakrishnan. Continuous univariate distributions, 1994.

[25] U. Paquet and N. Koenigstein. One-class collaborative filtering with random graphs. *... of the 22nd international conference on ...*, Jan 2013.

[26] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Uncertainty in Artificial Intelligence*, pages 452–461, 2009.

[27] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the 25th international conference on Machine learning*, pages 880–887. ACM, 2008.

[28] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. *Advances in Neural Information Processing Systems*, 20:1257–1264, 2008.

[29] D. Stern, R. Herbrich, and T. Graepel. Matchbox: large scale online bayesian recommendations. *... of the 18th international conference on ...*, Jan 2009.

[30] M. Wainwright and M. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1–2):1–305, 2008.

[31] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual*

International Conference on Machine Learning, pages 1113–1120. ACM, 2009.

APPENDIX

A. POISSON FACTORIZATION AND LDA

We show that LDA is equivalent to Poisson factorization, conditioned on the per-user sums and where the item weights are constrained to sum to one (across items, for each component). To show this fact we appeal to the relationships between the Gamma and Dirichlet distributions and between the Poisson and multinomial distributions.

LDA is a mixed-membership model of word counts. There are a set of "topics" $\gamma_{1:K}$, distributions over a fixed vocabulary, and each document exhibits those topics with different proportions. Applied to the setting here, the vocabulary is the set of items; each user is a "document", represented as a sparse vector of item counts; a topic is a distribution over items, $\sum_j \gamma_{kj} = 1$; each user's topic proportions are denoted π_u, where $\sum_k \pi_{uk} = 1$. User/item data, treated as documents, was one of the original applications of LDA [3].

To see the connection to Poisson factorization, we take the "multinomial PCA" perspective of LDA [4]. Let Γ be the $K \times I$ matrix of topics, where each row γ_k is a distribution over I items. Recall that the Dirichlet distribution is a distribution over the simplex, non-negative vectors that sum to one. For a distribution on the K-simplex, the Dirichlet parameter is a positive K-vector. The generative process for LDA is

$$\begin{aligned} \gamma_k &\sim \quad \text{Dirichlet}(\eta) \qquad\quad k \in \{1, \ldots, K\} \\ \pi_u &\sim \quad \text{Dirichlet}(\alpha) \qquad\quad u \in \{1, \ldots, U\} \\ y_u &\sim \quad \text{Mult}(n_u, \pi_u^\top \Gamma). \qquad u \in \{1, \ldots, U\}. \end{aligned}$$

This process conditions on n_u, the sum of the counts for user u. Further, we assume exchangeable Dirichlets, that is, the hyperparameters α and η are scalars repeated K and I times, respectively, for their corresponding Dirichlet parameters. This generative process is different from but equivalent to the original process for LDA [3].

Before connecting LDA and Poisson factorization, we articulate the relationship between the Gamma and Dirichlet and between the Poisson and multinomial. The relationship between the Dirichlet and Gamma distributions is that we can write a Dirichlet random vector as a normalized vector of independent Gammas. Let π be a K-dimensional simplex vector and let α is a positive K-vector. If we generate π from the following two-stage process,

$$\begin{aligned} \theta_k &\sim \text{Gamma}(\alpha_k, 1) \\ \pi_k &= \frac{\theta_k}{\sum_j \theta_j}, \end{aligned}$$

then $\pi \sim \text{Dirichlet}(\alpha)$.

The relationship between the Poisson and multinomial is that a set of Poisson variables, conditioned on their sum, is a multinomial [18]. Let $z_{1:K}$ be a set of Poisson variables, each with different rates $\mu_{1:K}$. Conditioned on their sum $n = \sum_k z_k$, the joint distribution of $z_{1:K}$ is a multinomial (giving a vector of counts) whose proportions are the normalized rates,

$$z \sim \text{Mult}\,(n, \pi)$$

where $\pi_k = \mu_k / \sum_j \mu_j$.

With these two facts in hand, we can show that LDA is a type of Poisson factorization. First, we re-parameterize the Dirichlet topic proportions with Gamma distributions,

$$\begin{aligned} \theta_{uk} &\sim \text{Gamma}(\alpha, 1) \\ \pi_{uk} &= \theta_{uk} / \textstyle\sum_j \theta_{uj}. \end{aligned}$$

Second, we note that y_u coming from a multinomial is equivalent to a conditional bank of Poissons (denoted by Poisson_{n_u}),

$$y_u \,|\, n_u \sim \text{Poisson}_{n_u}(\pi_u^\top \Gamma).$$

This conditional Poisson is equivalent to any other with the rates scaled by a constant. We can thus use the original Gamma variables θ_{uk} because π is simply scaled by its sum,

$$y_u \,|\, n_u \sim \text{Poisson}_{n_u}(\theta_u^\top \Gamma).$$

Note that we cannot symmetrically scale by a Gamma representation of β_{ik} because the vector $\gamma_{\cdot i}$, which are the per-topic probabilities for a fixed item, cannot be represented by a normalized Gamma. (Rather, it is the topics themselves, across words, which are normalized Gammas.)

In summary, LDA is a form of Poisson factorization where (a) the scaling parameter on the gamma priors is fixed to be one (b) we condition on the marginal sums for each user (c) the per-item weights are scaled to sum to one for each component.