Scalable Recommendation with Hierarchical Poisson Factorization

Prem Gopalan
Department of Computer Science
Princeton University
Princeton, NJ

Jake M. Hofman
Microsoft Research
641 Sixth Avenue, Floor 7
New York, NY

David M. Blei
Departments of Statistics & Computer Science
Columbia University
New York, NY

Abstract

We develop hierarchical Poisson matrix factorization (HPF), a novel method for providing users with high quality recommendations based on implicit feedback, such as views, clicks, or purchases. In contrast to existing recommendation models, HPF has a number of desirable properties. First, we show that HPF more accurately captures the long-tailed user activity found in most consumption data by explicitly considering the fact that users have finite attention budgets. This leads to better estimates of users' latent preferences, and therefore superior recommendations, compared to competing methods. Second, HPF learns these latent factors by only explicitly considering positive examples, eliminating the often costly step of generating artificial negative examples when fitting to implicit data. Third, HPF is more than just one method—it is the simplest in a class of probabilistic models with these properties, and can easily be extended to include more complex structure and assumptions. We develop a variational algorithm for approximate posterior inference for HPF that scales up to large data sets, and we demonstrate its performance on a wide variety of real-world recommendation problems—users rating movies, listening to songs, reading scientific papers, and reading news articles.

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, products—that they will like. A recommendation system is built from historical data about which items each user has consumed, be it clicked, viewed, rated, or purchased. First, it uncovers the behavioral patterns that characterize various types of users and the kinds of items they tend to like. Then, it exploits these discovered patterns to recommend future items to its users.

In this paper, we develop Hierarchical Poisson factorization (HPF) for generating high-quality recommendations. Our algorithms easily scale to massive data and outperform several existing methods. We show HPF is tailored to real-world properties of user behavior data: the heterogeneous interests of users, the varied types of items, and a realistic distribution of the finite resources that users have to consume these items.

In more detail, HPF is a probabilistic model of users and items. It associates each user with a latent vector of preferences, each item with a latent vector of attributes, and constrains both sets of vectors to be sparse and non-negative. The model assumes that each cell of the observed behavior matrix is drawn from a Poisson distribution—an exponential family distribution over non-negative integers—whose parameter is a linear combination of the corresponding user preferences and item attributes. The main computational problem is posterior inference: given an observed matrix of user behavior, we would like to discover the latent attributes that describe the items and the latent preferences of the users, which we can then use to make predictions and recommendations.

This inferential computation is common to many variants of matrix factorization. We find, however, that HPF enjoys significant quantitative advantages over classical methods for a variety of *implicit feedback* data sets. Figure 4 shows that HPF performs better than competing methods—including the industry standard of matrix factorization with user and item biases (MF) fit using stochastic gradient descent—for large data sets of Netflix users watching movies, Last.FM users listening to music, scientists reading papers, and *New York Times* readers clicking on articles.

We review related work in detail in Section 4. We now discuss details of the Poisson factorization model, including

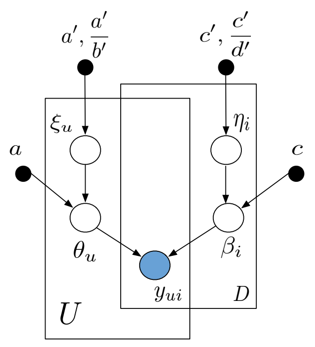

Figure 1: The hierarchical Poisson factorization model.

its statistical properties and methods for scalable inference.

2 POISSON RECOMMENDATION

In this section we describe the Poisson factorization model for recommendation, and discuss its statistical properties.

We are given data about users and items, where each user has consumed and possibly rated a set of items. The observation y_{ui} is the rating that user u gave to item i, or zero if no rating was given. In the "implicit" consumer data that we consider here, y_{ui} equals one if user u consumed item i and zero otherwise. User behavior data, such as purchases, clicks, or views, are typically sparse. Most of the values of the matrix y are zero.

We model these data with factorized Poisson distributions [4], where each item i is represented by a vector of K latent attributes β_i and each user u by a vector of K latent preferences θ_u. The observations y_{ui} are modeled with a Poisson distribution, parameterized by the inner product of the user preferences and item attributes, $y_{ui} \sim \text{Poisson}(\theta_u^\top \beta_i)$. This is a variant of probabilistic matrix factorization [32] but where each user and item's weights are positive [24] and where the Poisson replaces the Gaussian. While a Bernoulli distribution may seem more appropriate for modeling binary data, we demonstrate in Section 2.1 that the additivity of independent Poissons result in models that capture the marginal user, item distributions well. [1]

Beyond the basic data generating distribution, we place Gamma priors on the latent attributes and latent preferences, which encourage the model towards sparse representations of the users and items. Furthermore, we place additional priors on the user and item-specific rate parameter of those Gammas, which controls the average size of the representation. This hierarchical structure allows us to capture the diversity of users, some tending to consume more than others, and the diversity of items, some being more popular than others. The literature on recommendation systems suggests that a good model must capture such heterogeneity across users and items [22].

[1]Our ongoing work considers censored Poisson distributions. Our initial results indicate that it is computationally expensive but does not give better performance.

Putting this together, the generative process of the hierarchical Poisson factorization model (HPF), illustrated in the graphical model in Figure 1, is as follows:

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

This process describes the statistical assumptions behind the model. We note that this contains, as a sub-class, a factorization model where with fixed rate parameters for all users. We call this model Bayesian Poisson Factorization (BPF).

The central computational problem is posterior inference, which is akin to "reversing" the generative process. Given a user behavior matrix, we want to estimate the conditional distribution of the latent per-user and per-item structure, $p(\theta_{1:N}, \beta_{1:M} \mid y)$, termed the posterior, which is the key to recommendation. We estimate the posterior expectation of each user's preferences, each items attributes and, subsequently, form predictions about which unconsumed items each user will like. We discuss posterior inference in Section 2.2.

Once the posterior is fit, we use 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.

2.1 Properties of HPF

With the modeling details in place, we highlight several statistical properties of hierarchical Poisson factorization.

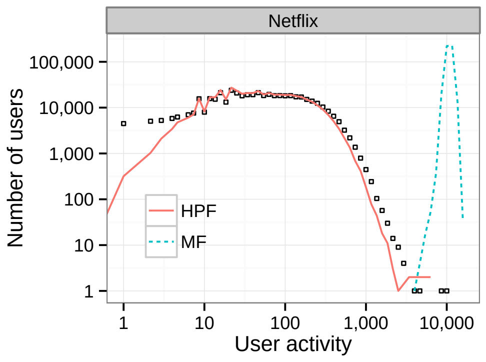

Figure 2: A posterior predictive check of the distribution of total ratings for the Netflix data set. The black squares show the empirical count of the number of users who have rated a given number of items, while the red and blue curves show the simulated totals from fitted Poisson and traditional matrix factorization models, respectively. The Poisson marginal closely matches the empirical, whereas classical matrix factorization fits a large mean to account for skew in the distribution and the missing ratings.

These properties provide advantages over classical Gaussian matrix factorization.[2]

HPF captures sparse factors. As mentioned above, the Gamma priors on preferences and attributes encourages sparse representations of users and items. Specifically, by setting the shape parameter to be small, most of the weights will be close to zero and only a few will be large. This leads to a simpler, more interpretable model.

HPF models the long-tail of users and items. One statistical characteristic of real-world user behavior data is the distribution of user activity (i.e., how many items a user consumed) and item popularity (i.e., how many users consumed an item). These distributions tend to be long-tailed: while most users consume a handful few items, a few "tail users" consume thousands of items. A question we can ask of a statistical model of user behavior data is how well it captures these distributions. We found that HPF captures them well, while classical matrix factorization does not.

To check this, we implemented a *posterior predictive check* (PPC) [30, 9], a technique for model assessment from the Bayesian statistics literature. The idea behind a PPC is to simulate a complete data set from the posterior predictive distribution—the distribution over data that the posterior induces—and then compare the generated data set to the true observations. A good model will produce data that captures the important characteristics of the observed data.

[2]Specifically, by classical matrix factorization we mean L2 regularized matrix factorization with bias terms for users and items, fit using stochastic gradient descent [22]. Without the bias terms, this corresponds to maximum a-posteriori inference under Probabilistic Matrix Factorization [32]. We generate negatives by randomly sampling from missing ratings in the training set [8, 7, 28].

We developed a PPC for matrix factorization algorithms on user behavior data. First, we formed posterior estimates of user preferences and item attributes for both classical MF and HPF. Then, from these estimates, we simulated user behavior by drawing values for each user and item. (For classical matrix factorization, we truncated these values at zero and rounded to one in order to generate a plausible matrix.) Finally, we compared the matrix generated by the posterior predictive distribution to the true observations.

Figure 2 illustrates our PPC for the Netflix data. In this figure, we illustrate three distributions over user activity: the observed distribution (squares), the distribution from a data set replicated by HPF (red line), and a distribution from a data set replicated by Gaussian MF with generated negatives using popularity-based sampling (blue line). HPF captures the truth much more closely than Gaussian MF, which overestimates the distribution of user activity. (We note that this is true for the item popularities as well, and for the other data sets.) This indicates that HPF better represents real data when measured by its ability to capture distributions of user activity and item popularity. In fact, this is encoded in its assumptions. We can rewrite the Poisson observation model as a two stage process where a user u first decides on a budget b_u she has to spend on items, and then spends this budget rating items that she is interested in:

$$b_u \sim \text{Poisson}(\theta_u^T \sum_i \beta_i)$$

$$[y_{u1}, \cdots, y_{uM}] \sim \text{Mult}(b_u, \frac{\theta_u^T \beta_i}{\theta_u^T \sum_i \beta_i}).$$

This shows that learning a PF model for user-item ratings is effectively the same as learning a budget for each user while also learning that budget is distributed across items.

HPF downweights the effect of zeros. Another advantage of HPF is that it implicitly down-weights the contribution of the items that each user did not consume. With an appropriate fit to user activity, the model has two ways of explaining an unconsumed item: either the user is not interested in it or she would be interested in if she the opportunity to consider it. In contrast, a user that consumes an item must be interested in it. Thus, the model benefits more from making latent factors for a consumed user/item pair more similar compared to making them less similar for an unconsumed user/item pair.

Classical MF 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 and implicit feedback, i.e., binary consumption data, matrix factorization places more total emphasis on the unconsumed user/item pairs. (This too can be seen to stem from classical MF's overestimation of the distribution of user activity.) To address this, researchers have patched MF in complex ways, for example, by including per-observation confidences [22] or considering all zeroes to be hidden variables [28]. Poisson factorization more naturally solves this problem with a more realistic model of user activity.

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". This explains why HPF outperforms other methods.

Fast inference with sparse matrices. Finally, the likelihood of the observed data under HPF depends only on the consumed items, that is, the non-zero elements of the user/item matrix y. This facilates computation for the kind of sparse matrices we observe in real-world data.

We can see this property from the form of the Poisson distribution. Given the latent preferences θ_u and latent attributes β_i, the Poisson distribution of the rating y_{ui} is

$$p(y_{ui} \mid \theta_u, \beta_i) = \left(\theta_u^\top \beta_i\right)^y \exp\left\{-\theta_u^\top \beta_i\right\} / y_{ui}! \qquad (2)$$

Recall the elementary fact that $0! = 1$. With this, the log probability of the complete matrix y can be written as

$$\begin{aligned}\log p(y \mid \theta, \beta) = &\textstyle\sum_{\{y_{ui}>0\}} y_{ui} \log(\theta_u^\top \beta_i) - \log y_{ui}! \\ &- \left(\textstyle\sum_u \theta_u\right)^\top \left(\textstyle\sum_i \beta_i\right).\end{aligned}$$

This avoids the need for sub-sampling [7], approximation [16], or stochastic optimization [26] that complicate other approaches.

2.2 INFERENCE WITH VARIATIONAL METHODS

Using HPF for recommendation hinges on solving the posterior inference problem. Given a set of observed ratings, we would like to infer the user preferences and item attributes that explain these ratings, and then use these inferences to recommend new content to the users. In this section we discuss the details and practical challenges of posterior inference for HPF, and present a mean-field variational inference algorithm as a practical and scalable approach. Our algorithm easily accommodates data sets with millions of users and hundreds of thousands of items on a single CPU.

Given a matrix of user behavior, we would like to compute the posterior distribution of user preferences θ_{uk}, item attributes β_{ik}, user activity ξ_u and item popularity η_i. As for many Bayesian models, however, the exact posterior is computationally intractable. We show how to efficiently approximate the posterior with mean-field variational inference.

Variational inference is an optimization-based strategy for approximating posterior distributions in complex probabilistic models [20, 34]. Variational 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 Kullback-Liebler (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 inference has previously been used for large-scale recommendation [28].

We will describe a simple variational inference algorithm for HPF. To do so, however, we first give an alternative formulation of the model in which we add an additional layer of latent variables. These auxiliary variables facilitate derivation and description of the algorithm [10, 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 $y_{ui} > 0$.

With these latent variables in place, we now describe the algorithm. First, we posit the variational family over the hidden variables. Then we show how to optimize its parameters to find an approximation to the posterior.

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,

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

Though the variables are independent, this is a flexible fam-

For all users and items, initialize the user parameters γ_u, κ_u^{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:

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

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

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

Figure 3: Variational inference for Poisson factorization. Each iteration only needs to consider the non-zero elements of the user/item matrix.

ily of distributions because each variable is governed by its own free parameter. The variational factors for preferences θ_{uk}, attributes β_{ik}, activity ξ_u, and popularity η_i are all Gamma distributions, with freely set scale and rate variational parameters. The variational factor for z_{ui} is a free multinomial, i.e., ϕ_{ui} is a K-vector that sums to one. This form stems from z_{ui} being a bank of Poisson variables conditional on a fixed sum y_{ui}, and the property that such conditional Poissons are distributed as a multinomial [19, 5].

After specifying the family, we fit the variational parameters $\nu = \{\lambda, \gamma, \kappa, \tau, \phi\}$ to minimize the KL divergence to the posterior, and then use the corresponding variational distribution $q(\cdot \mid \nu^*)$ as its proxy. 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.

We optimize the variational parameters with a coordinate ascent algorithm, iteratively optimizing each parameter while holding the others fixed. The algorithm is illustrated in Figure 3. We denote shape with the superscript "shp" and rate with the superscript "rte". We provide a detailed derivation in the Appendix.

Note that our algorithm is very efficient on sparse matrices. In step 1, we need only update variational multinomials for the non-zero user/item observations y_{ui}. In steps 2 and 3, the sums over users and items need only to consider non-zero observations. This efficiency is thanks the likelihood of the full matrix only depending on the non-zero observations, as we discussed in the previous section. Both HPF and BPF enjoy this property and have the same computational overhead, but HPF allows for more flexibility in modeling the variation in activity and popularity across users and items, respectively.

We terminate the algorithm when the variational distribution converges. Convergence is measured by computing the prediction accuracy on a validation set. Specifically, 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 HPF algorithm stops when the change in log likelihood is less than 0.0001%. We find that the algorithm is largely insensitive to small changes in the hyper-parameters. To enforce sparsity, we set the shape hyperparameters a', a, c and c' to provide exponentially shaped prior Gamma distributions. We fixed each hyperparameter at 0.3. We set the hyperparameters b' and d' to 1, fixing the prior mean at 1.

3 EMPIRICAL STUDY

We evaluate the performance of the Hierarchical Poisson factorization (HPF) algorithm on a variety of large-scale user behavior data sets: users listening to music, users watching movies, users reading scientific articles, and users reading the newspaper. We find that HPF provides significantly better recommendations than competing methods. We provide an exploratory analysis of preferences and attributes on the New York Times data set in the appendix.

Data Sets. We study the HPF algorithm in Figure 3 on several data sets of user behavior:

- The **Mendeley** data set [18] of scientific articles is a binary matrix of 80,000 users and 260,000 articles, with 5 million observations. Each cell indicates the presence or absence of an article in a user's library.
- The **Echo Nest** music data set [2] is a matrix of 1 million users and 385,000 songs, with 48 million observations. Each observation is the number of times a user played a song.

- The **New York Times** data set is a matrix of 1,615,675 users and 103,390 articles, with 80 million observations. Each observation is the number of times a user viewed an article.

- The **Netflix** data set [22] contains 480,000 users and 17,770 movies, with 100 million observations. Each observation is the rating (from 1 to 5 stars) that a user provided for a movie.

The scale and diversity of these data sets enables a robust evaluation of our algorithm. The Mendeley, Echo Nest, and New York Times data are sparse compared to Netflix. For example, we observe only 0.001% of all possible user-item ratings in Mendeley, while 1% of the ratings are non-zero in the Netflix data. This is partially a reflection of large number of items relative to the number of users in these data sets.

Furthermore, the intent signaled by an observed rating varies significantly across these data sets. For instance, the Netflix data set gives the most direct measure of stated preferences for items, as users provide an star rating for movies they have watched. In contrast, article click counts in the New York Times data are a less clear measure of how much a user likes a given article—most articles are read only once, and a click through is only a weak indicator of whether the article was fully read, let alone liked. Ratings in the Echo Nest data presumably fall somewhere in between, as the number of times a user listens to a song likely reveals some indirect information about their preferences.

As such, we treat each data set as a source of implicit feedback, where an observed positive rating indicates that a user likes a particular item, but the rating value itself is ignored. The Mendeley data are already of this simple binary form. For the Echo Nest and New York Times data, we consider any song play or article click as a positive rating, regardless of the play or click count. As in previous work, we consider an implicit version of the Netflix data where only 4 and 5 star ratings are retained as observations [28].

Competing methods. We compare Poisson factorization against an array of competing methods:

- **NMF**: Non-negative Matrix Factorization [24]. In NMF, user preferences and item attributes are modeled as non-negative vectors in a low-dimensional space. These latent vectors are randomly initialized and modified via an alternating multiplicative update rule to minimize the Kullback-Leibler divergence between the actual and modeled rating matrices. We use the GraphLab implementation of NMF [23] to scale to our large data sets.

- **LDA**: Latent Dirichlet Allocation [3]. LDA is a Bayesian probabilistic generative model where user preferences are represented by a distribution over different topics, and each topic is represented by a distribution over items. Interest and topic distributions are randomly initialized and updated using stochastic variational inference [15] to approximate these intractable posteriors.

- **MF**: Probabilistic Matrix Factorization with user and item biases. We use a variant of matrix factorization popularized through the Netflix Prize [22], where a linear predictor—comprised of a constant term, user activity and item popularity biases, and a low-rank interaction term—is fit to minimize the mean squared error between the predicted and observed rating values, subject to L2 regularization to avoid overfitting. Weights are randomly initialized and updated via stochastic gradient descent using the Vowpal Wabbit package [36]. This corresponds to maximum a-posteriori inference under Probabilistic Matrix Factorization [32].

- **CliMF**: Collaborative Less-is-More filtering [33] maximizes mean reciprocal rank to improve the top-n predictive performance on binary relevance data sets. We use the GraphLab implementation of CliMF [23] to scale to our large data sets.

We note that while HPF and LDA take only the non-zero observed ratings as input, traditional matrix factorization requires that we provide explicit zeros in the ratings matrix as negative examples for the implicit feedback setting. In practice, this amounts to either treating all missing ratings as zeros (as in NMF) 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, 7, 28]. We take the latter approach for computational convenience, employing a popularity-based sampling scheme: 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 to generate negative examples.[3]

Finally, we note that a few candidate algorithms failed to scale to our data sets. The fully Bayesian treatment of the Probabilistic Matrix Factorization [31], uses a MCMC algorithm for inference. The authors [31] report that a single Gibbs iteration on the Netflix data set with 60 latent factors, requires 30 minutes, and that they throw away the first 800 samples. This implies at least 16 days of training, while the HPF variational inference algorithm converges within 13 hours on the Netflix data. Another alternative, Bayesian Personalized Ranking (BPR) [29, 8], optimizes a ranking-based criteria using stochastic gradient descent.

[3]We also compared this to a uniform random sampling of negative examples, but found that the popularity-based sampling performed better.

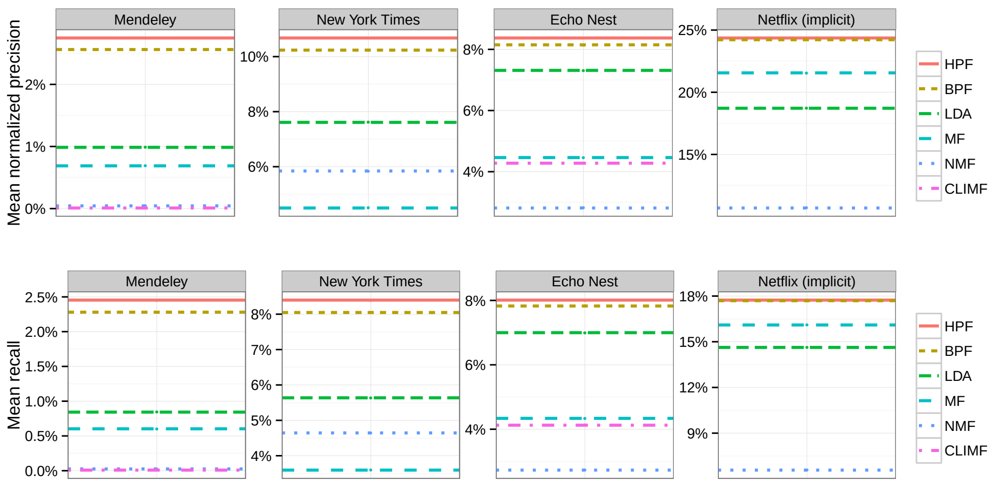

Figure 4: Predictive performance on data sets. The top and bottom plots show normalized mean precision and mean recall at 20 recommendations, respectively. While the relative performance of the competing methods varies across data sets, HPF consistently outperforms each of them.

The algorithm performs an expensive bootstrap sampling step at each iteration to generate negative examples from the vast set of unobserved. We found time and space constraints to be prohibitive when attempting to use BPR with the data sets considered here. Finally, the GraphChi implementation of CLiMF [23] failed with an error on the Netflix and New York Times data sets.

Evaluation. Prior to training any models, 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. Additionally, we set aside 1% of the training ratings as a validation set and use it to determine algorithm convergence and to tune free parameters. We used the HPF settings described in Section 2.2 across all data sets, and set the number of latent components K to 100.

During testing, we generate the top M recommendations for each user as those items with the highest predictive score under each method. For each user, we compute a variant of precision-at-M that measures the fraction of relevant items in the user's top-M recommendations. So as not to artificially deflate this measurement for lightly active users who have consumed fewer than M items, we compute *normalized* precision-at-M, which adjusts the denominator to be at most the number of items the user has in the test set. Likewise, we compute recall-at-M, which captures the fraction of items in the test set present in the top M recommendations.

Figure 4 shows the normalized mean precision at 20 recommendations for each method and data sets. We see that HPF outperforms other methods on all data sets by a sizeable margin—as much as 8 percentage points. Poisson factorization provides high-quality recommendations—a relatively high fraction of items recommended by HPF are found to be relevant, and many relevant items are recommended. While not shown in these plots, the relative performance of methods within a data set is consistent as we vary the number of recommendations shown to users. We also note that while Poisson factorization dominates across all of these data sets, the relative quality of recommendations from competing methods varies substantially from one data set to the next. For instance, LDA performs quite well on the Echo Nest data, but fails to beat classical matrix factorization for the implicit Netflix data set.

We also study precision and recall as a function of user activity to investigate how performance varies across users of different types. In particular, Figure 5 shows the mean difference in precision and recall to HPF, at 20 recommendations, as we look at performance for users of varying activity, measured by percentile. For example, the 10% mark on the x-axis shows mean performance across the bottom 10% of users, who are least active; the 90% mark shows the mean performance for all but the top 10% of most active users. Here we see that Poisson factorization outperforms other methods for users of all activity levels.

4 RELATED WORK

The roots of Poisson factorization come from nonnegative matrix factorization [24], where the objective function is equivalent to a factorized Poisson likelihood. The original NMF update equations have been shown to be an expectation-maximization (EM) algorithm for maximum likelihood estimation of a Poisson model [5].

Placing a Gamma prior on the user weights results in the

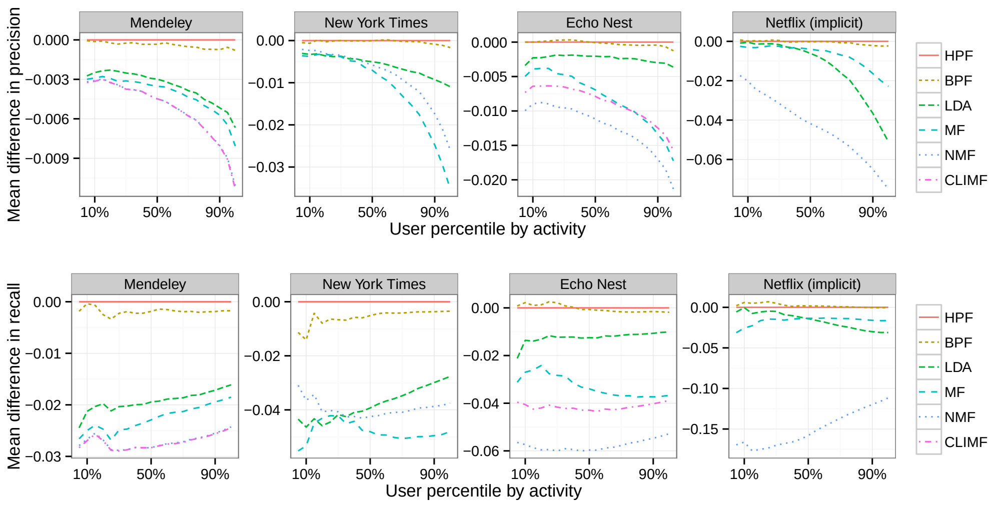

Figure 5: Predictive performance across users. The top and bottom plots show the mean difference in precision and recall to HPF at 20 recommendations, respectively, by user activity.

GaP model [4], which was developed as an alternative text model to latent Dirichlet allocation (LDA) [3, 17]. The GaP model is fit using the expectation-maximization algorithm to obtain point estimates for user preferences and item attributes. The Probabilistic Factor Model (PFM) [25] improves upon GaP by placing a Gamma prior on the item weights as well, and using multiplicative update rules to infer an approximate maximum a posteriori estimate of the latent factors. Our model uses a hierarchical prior structure of Gamma priors on both user and item weights, and Gamma priors over the rate parameters from which these weights are drawn. Furthermore, we approximate the full posterior over all latent factors using a scalable variational inference algorithm.

Independently of GaP and user behavior models, Poisson factorization has been studied in the context of signal processing for source separation [5, 14] and for detecting community structure in network data [1, 12]. This research includes variational approximations to the posterior, though the issues and details around these data differ significantly from user data we consider and our derivation in the supplement (based on auxiliary variables) is more direct.

When modeling implicit feedback data sets, researchers have proposed merging factorization techniques with neighborhood models [21], weighting techniques to adjust the relative importance of positive examples [16], and sampling-based approaches to create informative negative examples [8, 7, 28]. In addition to the difficulty in appropriately weighting or sampling negative examples, there is a known selection bias in provided ratings that causes further complications [27]. HPF does not require such special adjustments and scales linearly in the observed ratings.

Comparison to Gaussian MF. Many of the leading MF methods are based on Gaussian likelihoods (i.e., squared loss). When applied to explicit data, Gaussian models are fit only to the observed ratings [22] and infer distributions over user preferences. For each user, the items she did not consume, i.e., the zero observations, are treated as missing. Gaussian models make up the state of the art in this setting [31, 32, 22].

In implicit data sets of user consumption, there is a fundamental asymmetry that allows one to infer which items a user consumed, and therefore liked, but not which items a user did not like [16]. In this setting, Gaussian MF applied to all observations gives equal weight to consumed and unconsumed items. Consequently, when faced with a sparse matrix and implicit feedback, matrix factorization places more total emphasis on the unconsumed user/item pairs.

To address this limitation of Gaussian MF, researchers have patched it in complex ways. There are two main approaches. The first approach, proposed by [16], is to treat the unconsumed items with greater uncertainty and increase confidence as the rating for an item increases. This converts the raw observations into two separate quantities with distinct interpretations: user preferences and confidence levels. Hu et al. [16] present an alternating least squares algorithm that considers all observations but whose per-iteration complexity is still linear in the number of nonzero observations. A disadvantage of this method is that the assignment of per-observation (class) confidence weights may require exhaustive search via cross-validation, or other model selection methods, hindering scalability.

The second approach is to randomly synthesize negative examples [7, 8, 28]. In this approach, unconsumed items are subsampled for each user to balance out the consumed items. As Dror et al. [7] note, it is unclear how to balance these two sets of items. Do we use an equal number of consumed and consumed items, or do we use the full set of unconsumed items [6, 16]? Further, the subsampling of negative or unconsumed items is often expensive, and can account for a substantial fraction of resources devoted to model fitting.

Another issue with most Gaussian models is they do not capture the fact that users have limited resources to consume, view, or rate items from a vast number of options. Gaussian MF with subsampled zeros, fit using SGD, systematically overestimates the users' budgets, as confirmed in Section 3 using a posterior predictive check [9]. This misfit leads to an overweighting of the zeros, which explains why practitioners require complex methods for downweighting them [7, 8, 16, 28]. Poisson factorization does not need to be modified in this way.

Further, the HPF algorithm retains the linear-scaling of Gaussian MF with downweighted zeros [16]. HPF algorithms only need to iterate over the consumed items in the observed matrix of user behavior. This follows from the mathematical form of the Poisson distribution. In contrast, the subsampling-based Gaussian MF methods [7, 8, 28] must iterate over both positive and negative examples in the implicit setting. This makes it difficult to take advantage of data sparsity to scale to massive data sets.

Finally, unlike Gaussian MF which typically provides dense latent representations of users and items, PF models provide sparse latent representations. This property arises from the PF log-likelihood which can be shown to minimize the information (Kullback-Leibler) divergence under NMF [5], and from the Gamma priors in the HPF model.

Recent extensions. Building on the HPF model and algorithm we presented in a preprint, recent extensions have been proposed. One extension is a combined model of article text and reader preferences [13]. This model takes advantage of the sparse, non-negative representations in PF, which are useful in capturing different types of discrete data, such as word counts and user ratings. Further, they exploit the additive properties of independent Poisson random variables to capture dependencies between discrete data, for example, the dependence of user ratings of an article on its content. Another recent work proposes a Bayesian nonparametric model [11] that adapts the dimensionality of the latent representations, learning the preference patterns (and their number) that best describe the users. Both models exploit the scalability of PF algorithms to study massive data sets. These extensions testify to the modeling flexibility of PF models.

5 DISCUSSION

We have demonstrated that Poisson factorization is an efficient and effective means of generating high quality recommendations across a variety of data sets ranging from movie views to scientific article libraries. It significantly outperforms a number of leading methods in modeling implicit behavior data without the need for ad hoc modifications. Variational inference for HPF scales to massive data and differs from traditional methods in its ability to capture the heterogeneity amongst users and items, accounting for the wide range of activity and popularity amongst them, respectively. The HPF algorithm is a robust, off-the-shelf tool, providing high accuracy even with fixed hyperparameter settings.

Finally, we emphasize that HPF is more than just one method—it is the simplest in a class of probabilistic models with these properties, and has already been extended to a combined model of article content and reader ratings [13], and a Bayesian nonparametric model that adapts the dimensionality of the latent representations [11].

A notable innovation in Gaussian MF is the algorithm of [16] that explicitly downweights zeros using confidence parameters. We presented the empirical study in this paper comparing to the Gaussian MF with subsampled zeros [22]. We attempted to compare to a GraphChi implementation [23] of [16], but it gave unexpectedly poor results. We found another implementation [35], and these comparisons are ongoing work. Another piece of ongoing work includes bringing the confidence-weighting of [16] into HPF. This will allow downweighting of the zeros beyond that provided implicitly by Poisson factorization.

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 *ISMIR*, 2011.

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

[4] J. Canny. GaP: A factor model for discrete data. In *ACM SIGIR*, 2004.

[5] A. T. Cemgil. Bayesian inference for nonnegative matrix factorisation models. *Computational Intelligence and Neuroscience*, 2009, May 2009.

[6] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 39–46. ACM, 2010.

[7] 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.

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

[9] A. Gelman, X. Meng, and H. Stern. Posterior predictive assessment of model fitness via realized discrepancies. *Statistica Sinica*, 6:733–807, 1996.

[10] Z. Ghahramani and M. Beal. Propagation algorithms for variational Bayesian learning. In *NIPS*, pages 507–513, 2001.

[11] P. Gopalan, F. J. Ruiz, R. Ranganath, and D. M. Blei. Bayesian nonparametric Poisson factorization for recommendation systems. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, pages 275–283, 2014.

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

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

[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. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 263–272. IEEE, 2008.

[17] D. Inouye, P. Ravikumar, and I. Dhillon. Admixture of Poisson MRFs: A topic model with word dependencies. In *ICML*, pages 683–691, 2014.

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

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

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

[21] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *ACM SIGKDD*, pages 426–434. ACM, 2008.

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

[23] A. Kyrola, G. E. Blelloch, and C. Guestrin. Graphchi: Large-scale graph computation on just a PC. In *OSDI*, volume 12, pages 31–46, 2012.

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

[25] H. Ma, C. Liu, I. King, and M. R. Lyu. Probabilistic factor models for web site recommendation. In *ACM SIGIR*, pages 265–274. ACM Press, 2011.

[26] 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.

[27] B. Marlin, R. S. Zemel, S. Roweis, and M. Slaney. Collaborative filtering and the missing at random assumption. *arXiv preprint arXiv:1206.5267*, 2012.

[28] U. Paquet and N. Koenigstein. One-class collaborative filtering with random graphs. In *WWW*, 2013.

[29] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461, 2009.

[30] D. Rubin. Bayesianly justifiable and relevant frequency calculations for the applied statistician. *The Annals of Statistics*, 12(4):1151–1172, 1984.

[31] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *ICML*, pages 880–887. ACM, 2008.

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

[33] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, N. Oliver, and A. Hanjalic. Climf: learning to maximize reciprocal rank with collaborative less-is-more filtering. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 139–146. ACM, 2012.

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

[35] C. Wang and D. M. Blei. Collaborative topic modeling for recommending scientific articles. In *ACM SIGKDD*, KDD '11, pages 448–456, 2011.

[36] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In *ICML*, pages 1113–1120, 2009.