A Sparse Probabilistic Model of User Preference Data

Matthew Smith[1], Laurent Charlin[2], and Joelle Pineau[1]

[1] School of Computer Science, McGill University, Montréal, Quebec, Canada
{msmith108,jpineau}@cs.mcgill.ca
[2] HEC Montréal, Montréal, Quebec, Canada
laurent.charlin@hec.ca

Abstract. Modern recommender systems rely on user preference data to understand, analyze and provide items of interest to users. However, for some domains, collecting and sharing such data can be problematic: it may be expensive to gather data from several users, or it may be undesirable to share real user data for privacy reasons. We therefore propose a new model for generating realistic preference data. Our Sparse Probabilistic User Preference (SPUP) model produces synthetic data by sparsifying an initially dense user preference matrix generated by a standard matrix factorization model. The model incorporates aggregate statistics of the original data, such as user activity level and item popularity, as well as their interaction, to produce realistic data. We show empirically that our model can reproduce real-world datasets from different domains to a high degree of fidelity according to several measures. Our model can be used by both researchers and practitioners to generate new datasets or to extend existing ones, enabling the sound testing of new models and providing an improved form of bootstrapping in cases where limited data is available.

1 Introduction

User preference data has become one of the most valuable commodities, used by industry and governments, to inform several aspects of decision-making. Yet in many domains accurate user preference data can be difficult and expensive to obtain since collecting preference data requires access to a set of users, a set of items, and an interface for recording the users' preferences (clicks or ratings). There are also often limitations to sharing this data, which impedes progress of research, commercialization and policy development. There exist a few preference datasets commonly used in research on recommender systems (e.g., [2]). However confining research to a few datasets makes it difficult to ensure robust decision-making and explore diverse research directions.

In several fields of AI research, the use of synthetic data has provided an alternative for the rapid development and validation of new ideas. Examples abound, including in planning and reinforcement learning where there exist repositories of widely used synthetic benchmarks [3, 4, 5]. Synthetic datasets are also standard in social-network analysis research [6, 7], and in the field of statistics [8]. Realistic synthetic data can be used to explore the capabilities of models beyond what available real-world datasets can provide. Further, realistic synthetic data could be useful in application contexts where data has been collected, but cannot be shared, due to potential privacy or ethical

constraints. Yet a survey of the literature found very few examples of synthetic data generation for recommendation systems.

We present a new model for generating synthetic user preference data. Our model builds on the widely used probabilistic matrix factorization (PMF) model [9], which on its own was not designed to generate preference data. Our model generates a *mask matrix* which is used to censor the user preferences obtained by PMF. The mask matrix is parametrized by user budgets, item popularity as well as terms accounting for user-item interactions. Tuning these parameters allows the model to generate preference data with different attributes. The model could also take into account particular user and item interactions through *side information* (e.g., online friendship influenced by preferences).

While our model is capable of generating data from a wide variety of distributions, we show experimentally that, in particular, it can be used to generate realistic datasets that match important attributes of real data from movie, books, music and electronic-commerce domains. Note that while we use real datasets in order to evaluate our model, our procedure can be used to generate entirely novel datasets, from scratch. We also show that a popular recommendation model can be applied directly on our generated datasets in lieu of real data.

2 Related Work

A conceptual overview for a possible synthetic preference generation architecture is presented by Pasinato et al. [10]. The method outlined (never implemented as far as we know) involves the creation of user and item profiles, defining directly the distribution over the number of items sampled per user and the user's evaluation of each item. The authors suggest that one of the main benefits of this procedure is the ability to specify the probability density function of ratings. The model we propose is structured similarly, but the distributions are defined in terms of latent attributes of the users and items, providing more flexibility to structure the (preference) data.

Cluster Method. An alternative method based on clustering is presented by Tso and Schmidt-Thieme [11]. As far as we know, this is the only method in the literature that has been implemented and empirically validated for synthetic preference data generation. We use this approach as a baseline in our empirical evaluation below. This method involves the creation of user clusters, denoted C^U, and item clusters, C^I, which respectively represent groups of related users and related items. Each individual user is assigned to a user cluster according to a Dirichlet distribution over clusters, and likewise for items and item clusters. The method then generates a conditional distribution $P(C^U|C^I)$ of user clusters with respect to item clusters, using repeated draws from a modified χ^2 distribution that rejects values greater than 1, until the conditional normalized entropy, $H(C^U|C^I)$, reaches a preset value:

$$H(C^U|C^I) = -\sum_{i=0}^{|C^U|}\sum_{k=0}^{|C^I|} \frac{P(C^I = k, C^U = i)\log_2(P(C^U = i|C^I = k))}{\log_2(|C^I|)}.$$

Users then sample items from clusters determined by sampling from this conditional distribution, so the probability of user i, in user cluster C_i^U sampling from items in item cluster C_j^I, is $P(C_i^U|C_j^I)$. If a user has been determined to be sampling from a given item cluster, they then sample items according to a binomial distribution, with the parameter determining the probability that any given item within the cluster is sampled. This generates a final binary preference matrix, $\mathbf{R}$. While this model is general, it can be difficult to interpret in terms of how changing parameters will affect the resulting data, as probability density functions are not defined directly. In comparison, our model maintains the interpretability of a directly defined PDF, while remaining able to generate attribute information, since it models data as a product of an item attribute matrix.

Random Graph Models. Alternatively, we can view the user preference data generation problem as a random graph generation: we generate a (potentially weighted) bipartite graph, where edges exist between users and items. Many approaches to the generation of random graphs exist, with early advances focused around the Erdös-Rényi model. Though this model is too simple for user preferences, other models exist which enable specification of graph properties such as degree distribution. Caron and Fox [12] provide a method for the generation of sparse and exchangeable bipartite graphs, though the degree distribution is not directly specified. Newman et al. [13] present a method for sampling bipartite graphs with arbitrary degree distribution, though this method does not include latent factors (user-item interactions) and does not apply the budget-based approach constructed here. These models have not been used to construct user preference data, but may provide some theoretical foundation for our method.

3 A Sparse Probabilistic User Preference Model

The purpose of our model is to generate an N users by M items preference matrix, $\mathbf{R}$, to be used for recommender systems. The ith row represents how a user, i, has rated each item, either *implicitly* (by viewing or consuming the item, e.g., purchase decisions, movie views, article "shares"), or *explicitly* (giving feedback on some ratings scale) [14]. Here, as is common in recommender systems, a value of zero at entry ij of $\mathbf{R}$ indicates that user i has not rated or consumed item j. In typical recommendation system scenarios, $\mathbf{R}$ is a sparse matrix, although the methods proposed here can be used to generate arbitrarily dense datasets.

Our proposed **Sparse Probabilistic User Preferences (SPUP) model** produces ratings data in two steps. We first generate a dense preference matrix using a probabilistic matrix factorization model [9]; this matrix can be interpreted as how much users can be expected to like items. This matrix is then sparsified by generating budgets for each user, and subsampling from a user-specific distribution over items, as a function of both the user's expected preference, and the popularity of different items.

Probabilistic Matrix Factorization. Our model builds on the probabilistic matrix factorization (PMF) model [1, 9]. PMF is a generative probabilistic model that is standard in recommender systems. It involves the generation of a ratings matrix using the product of two matrices commonly interpreted as latent attributes of items and user latent

preferences over these attributes. Latent factors are usually inferred by optimizing for the reconstruction error of nonzero elements of the (observed) matrix.

The initial preference matrix, $\widetilde{\mathbf{R}}$, is modeled as a noisy product of latent user preference and item attribute matrices, which are sampled here, rather than inferred:

$$\mathbf{U}_i \sim \mathcal{N}(\mathbf{0}, \sigma_u^2 \mathbf{I}) \quad \text{and} \quad \mathbf{V}_j \sim \mathcal{N}(\mathbf{0}, \sigma_v^2 \mathbf{I}) \tag{1}$$

$$\widetilde{\mathbf{R}}_{ij} \sim \mathcal{N}(\mathbf{U}_i^\top \mathbf{V}_j, \sigma_p^2), \tag{2}$$

where $\mathbf{U}_i$ represents user i's latent preference vector, and $\mathbf{V}_j$ item j's latent attribute vector, $\mathbf{I}$ the identity matrix, and σ^2 parametrizes the Gaussian's variance.

Generating the Mask Matrix. The Aldous-Hoover representation theorem [15, 16] shows that the class of matrices generated by PMF (jointly exchangeable matrices) are dense (or empty) matrices. Thus our first step—recall that we begin by generating ratings with PMF—typically generates a matrix $\widetilde{\mathbf{R}}$ that is dense. However, our goal is not to infer the full user preference matrix, but rather to generate realistic preference data, which is typically very sparse.

Our second step ensures that we can generate such data by applying sparsification to the matrix generated by PMF ($\widetilde{\mathbf{R}}$). The steps for sparsification are as follows: a) sample a user budget to determine the activity-level of each user; b) sample an item popularity for each item; and c) for each user (and according to its budget) sample items according to their popularity and to their rating given by PMF.[3]

In order to achieve a realistic distribution over both items and products while sparsifying the matrix, we allocate each user with a budget, B_i, sampled from an exponential distribution, parameterized by rate β (rate is the inverse of the scale parameter):

$$B_i \sim \text{round}(\text{exponential}(\beta)) + c_b \tag{3}$$

where c_b is some positive hyperparameter set as the desired number of ratings for any user. Recall that the PDF of the exponential distribution is $\text{exponential}(x; \beta^{-1}) := \beta^{-1} \exp(-x\beta^{-1})$.

A user-specific distribution is then defined over the items. Users do not select items to rate at random. Empirically, across several datasets of consumer products, users rated more preferred items more frequently, with strong popularity effects, where many users are likely to have rated the same few items. We define the distribution over items for user i as a normalized element-wise product between an item-specific popularity vector (sampled from a power law distribution), and the underlying preference vector, $\widetilde{\mathbf{R}}_i$

$$p_j \sim \text{Power}(a) + c \tag{4}$$

$$\mathbf{D}_i = \frac{\mathbf{p} \circ (\widetilde{\mathbf{R}}_i + \min_j(\widetilde{\mathbf{R}}_{ij}))}{\|\mathbf{p} \circ (\widetilde{\mathbf{R}}_i + \min_j(\widetilde{\mathbf{R}}_{ij}))\|_1} \tag{5}$$

where p_j is the popularity factor of the jth item, a is the parameter for the power law, and c is some positive parameter, that gives a baseline probability of each item being

[3] The idea of using the combination of user budgets and item popularity has also been exploited for sampling preference matrices in the context of stochastic variational inference [17].

rated (so that no item has $p_j \equiv \mathbf{0}$), $\circ$ represents the Hadamard (elementwise) product between two vectors (or matrices) of equal size, and $\|\cdot\|_1$ is the 1-norm. Recall that the power law distribution function is $\text{Power}(x; a) := ax^{a-1}$.

B_i ratings for each user i are then sampled from the user-specific distribution $\mathbf{D}_i$, without replacement. This forms a masking matrix, $\mathbf{M}$, which is then applied to the latent ratings matrix $\widetilde{\mathbf{R}}$ by elementwise multiplication in order to form the masked raw ratings matrix: $\widetilde{\mathbf{R}}^{(\mathbf{M})} = \mathbf{M} \circ \widetilde{\mathbf{R}}$.

Most preference data is discrete (e.g., clicks or ratings). In order to form the final ratings matrix, $\mathbf{R}$, each entry in $\widetilde{\mathbf{R}}^{(\mathbf{M})}$ can be either binarized, in order to represent implicit feedback, or binned/scaled for ratings.[4]

The complete procedure for generating data is detailed in Algorithm 1. The time complexity of our method is dominated by the PMF product step (Eq. 2), which requires $\mathcal{O}(NM)$ operations to generate the complete ratings matrix $\widetilde{\mathbf{R}}$. In comparison, generating the mask ($\mathbf{M}$) has a time complexity that is linear in the order of the number of non-zero entries.

Algorithm 1: Generating preference data with SPUP.

Data: N, M, β, c_B, a, c
for *each user i and item j* **do**
  Sample a rating $\widetilde{r}_{ij}$ using Equation 2;
end
for *each user i* **do**
  Generate a budget B_i using Equation 3;
end
for *each item j* **do**
  Generate a popularity $\mathbf{p}_j$ using Equation 4;
end
Set $\mathbf{M}$ to be an all-zero N by M matrix;
for *each user i* **do**
  for $k \leftarrow 0$ **to** B_i **do**
    Generate $\mathbf{D}_i$ using Equation 5;
    Sample an item: $j \sim \mathbf{D}_i$;
    Set $\mathbf{M}_{ij} = 1$;
  end
end
return $\mathbf{M} \circ \widetilde{R}$

It should be noted that for large datasets, it is expensive to compute and store the full dense matrix $\widetilde{R}$ in memory, since this can contain tens of billions of real-valued entries. This is solved by performing the above masking procedure on individuals or groups of users, and concatenating the sparse matrices that this generates.

[4] Recent work has proposed the use of Poisson-observation matrix factorization models [18]. Using such models would alleviate the need for this discretization step but this is largely independent of our proposed approach.

4 Experimental Methods

To study our proposed SPUP model we evaluate how well it can simulate the attributes of real-world preference datasets. For several such datasets we adapt the hyperparameters of our model so as to generate data that is similar to that observed in the real-world, according to several attributes. We repeat this procedure for the cluster method [11] discussed in Related Work above and compare their output to the characteristics of the original datasets. We further validate our SPUP model by learning a standard collaborative filtering model on our synthetic data. We show that our synthetic data can indeed be learned: the performance of the model on our data is significantly higher than the performance of the same method on synthetic data generated using a random mask.

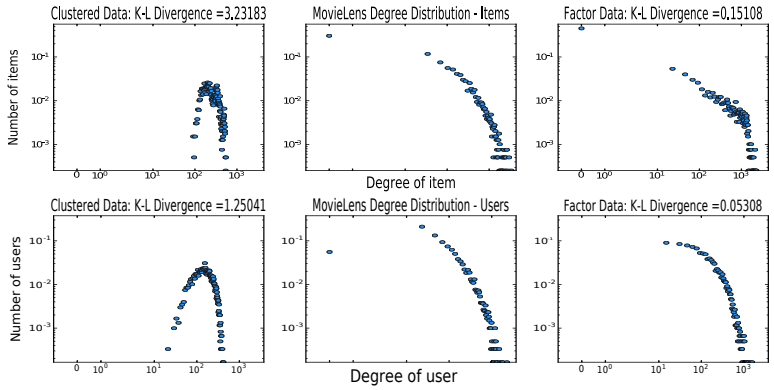


Fig. 1: Degree distribution of MovieLens data: cluster method (left), real data (middle), and SPUP (right). The x-axis represents the degree of items (top) or users (bottom) while the y-axis indicates the number of items or users (points on the figure indicate the number of items/user for a particular degree value). Error bars are omitted due to the very low variance of both methods. SPUP results (right) are more similar (visually and according to the KL divergence) to the real data (middle) than the cluster method (left).

We consider datasets from a range of domains with varying consumption patterns:

MovieLens Dataset. This data was collected from an online movie recommendation platform. Users score movies that they have seen on an integer scale from 1–5, and receive recommendations for movies to watch in the future. The MovieLens dataset used here contains 1 million ratings across 6K users and 4K movies [2].

Million Song Dataset (MSD). The million song dataset contains music listening data consisting of 1.4 million (user, song, playcount) triplets across 110K users and 160K songs. Playcounts in this dataset range from 1 to 923. This is a (random) subset of the original dataset [19].

Epinions Dataset. The Epinions dataset consists of data across 22K users and 300K items acquired from a product ratings website that allowed users to write reviews of and rate items that they had purchased on an integer scale from 1–5. Additionally, users

could form trust relationships with other users, which would influence the ratings which would appear on a user's feed [20, 21].

Book Crossings Dataset. This dataset was generated from a book sharing website that enables users to leave books in real-world locations, with a tag that allows the book to be tracked as it changes hands. Ratings data is either explicit values from 1–10, or implicit. This dataset contains 1.1 million ratings of 280K users by 270K books [22].

Comparison Measures. Synthetic and real datasets are compared across several key attributes. Our aim is to compare key attributes that define the structure of the preference datasets. We consider the density of the ratings matrix, the degree distribution for both users and items, and the normalized sorted sum of ratings distribution as first-order comparisons. We formally define these attributes below. Further as a second-order method, we consider the performance of a baseline PMF algorithm on a held-out subset of the synthetic data, relative to performance on a held-out subset of the real data.

When generating synthetic data for recommender systems, it is important to consider how well a recommender system performs under different amounts of data. Density provides a good measure as to the amount of information present in the ratings matrix (statistically uniformly denser matrices are easier to correctly estimate)—density is a widely reported statistic. As more ratings are added to the matrix, there is more information about the preferences of users and the attributes of items. If the purpose of the synthetic data is to extend or emulate a particular dataset, then it is important that the density of the synthetic data matches the density of the original data to avoid an information mismatch. Here, density of an $N \times M$ matrix $\mathbf{R}$ is expressed as: $\text{density}(R) = \sum_{i=0}^{N} \sum_{j=0}^{M} \frac{I(R_{ij} \neq 0)}{NM}$, where I is the indicator function. Both our SPUP model and the cluster method can control density directly, but since density is primarily determined by how many items users rate in each algorithm (B_i in our latent factor method, and the binomial parameter in the cluster method), parameterization to produce a particular density can impact the distributions of ratings for both users and items.

The second attribute is the *degree distribution* which describes how connected nodes in a graph are. If we interpret $\mathbf{R}$ as a bipartite graph, where users form one set of vertices and items form the other, this can be used to obtain the PDF of the number of connections (in our case, ratings) that users have formed and items have received. We can evaluate how much the ratings are governed by general patterns, such as popularity effects. For each dataset we report the degree distribution of users using a normalized histogram where the x-axis represents the number of connections per user (or item), and the y-axis is the number of users (or items) with a given number of connections.

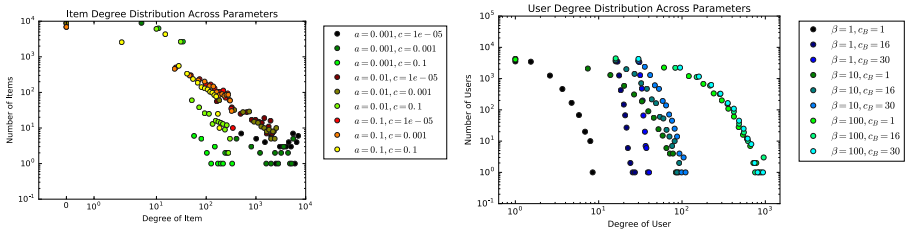
The third attribute is the *normalized sorted sum of ratings* (NSSR) which represents how ratings are distributed across users and items. For each user it is the ratio of the sum of that user's ratings over the sum of all ratings.

We report NSSR using a histogram over all users. NSSR is similar to degree distribution, in that it demonstrates how drastic effects such as item popularly or user activity are. We can read directly how much more certain items and users are engaged with by looking at the slope of the NSSR curve. If the slope is either very steep or very flat, then there may be strong factors (like item popularity) that influence the distribution of data. It should be noted that peaks in the NSSR graph correspond to points that have high degree. As such, NSSR demonstrates directly how items are rated by users. It can be

simpler to interpret than the degree distribution, but it does not show the PDF directly, so both methods are used.

In addition to reporting micro-level results using degree distribution and NSSR we can also compare methods more directly using their similarity according to these statistics. In particular since both statistics can be interpreted as distributions, we use the Kullback-Leibler divergence (KL), defined between two discrete probability distributions Y and Z as: $KL(Y||Z) = \sum_i Y(i) \log\left(\frac{Y(i)}{Z(i)}\right)$. Since KL-divergence is not well defined when Z is not continuous with respect to Y, Laplace smoothing is used wherever Y is nonzero and Z is.

Parameter Setting. To match characteristics of a specific real dataset with our synthetic data generation, we adjust the hyperparameters of the SPUP model using search. We first adjust hyperparameters in order to achieve similar density to the real dataset (using β and c_B), then the budget parameters (adjust β,c_B) while keeping the density constant, and finally, the item popularity parameters (using a,c). The first two objectives can be accomplished simultaneously by fitting β and c_B to the data: c_B is the minimum number of ratings across users in the dataset, and β is the mean number of ratings per user, minus c_B. Table 1 reports values of the hyper-parameters selected for each dataset.



(a) (b)

Fig. 2: An exploration of how the degree distribution of items (left) and users (right) vary as a function of hyperparameters. (This figure is best viewed in color.)

5 Results

We have described the methodology of our study above and will now present results. For the purpose of comparisons, we focus on the implicit-data case by binarizing all datasets, since the Cluster method cannot generate explicit preference data.

Comparison of Datasets. Figure 1 shows the results of the generation for the MovieLens dataset. We can see that degree distribution of the data (middle column) is shaped somewhere in between the shape of an exponential distribution and a power law across both items and users. The cluster method does not fit the degree distribution of either items or users (left). On the other hand the SPUP model achieves low KL-divergence for both users and items (right). The results for NSSR are qualitatively similar (Figure

4): SPUP generates more realistic data. Figure 2 illustrates how the budget parameters can be tuned to match the real data, even when looking at it on an item-by-item basis.

MSD is characterized by high sparsity—it has a density of only approx. 10^{-5}. It additionally has greater long-tail effects: while the mean number of listens per song is 10, some of its songs have several thousand listens. Despite the more complex nature of this data, SPUP is able to model it well (Fig. 5), while maintaining values across all parameters that seem intuitively plausible: the average user has rated 10 items, and item popularity falls off as a power of approximately $\frac{1}{10}$. We do not report the results for NSSR here as they are similar to the ones for the MovieLens datasets (likewise for the other datasets).

The results on the Epinions dataset (Fig. 6) show that SPUP provides a much better fit to the real data than the Cluster method in terms of user distributions, and fits similarly in item distribution.

For the Book Crossing dataset, both the cluster method and SPUP seem to provide good fits to the real data. Visually, the degree distributions match those of the Epinions data, and for lack of space are not included here. The KL-Divergence for the Cluster method was 0.1457 for items and 0.02235 for users, while for the SPUP method, KL-divergence was 0.00565 and 0.02235 correspondingly.

Parameter	Domain	Description	MovieLens	MSD	Epinions	Book Crossings
β	$\mathbb{R}^+$	Controls the distribution of user budgets	160	10.2	36.2	4
c_B	$\mathbb{N}^+$	Minimum number of ratings per user	15	3	2	0
a	$0 \leq a \leq 1$	Controls the distribution of item popularities	0.18	0.12	0.012	0.012
c	$\mathbb{R}^+$	Baseline probability of an item being rated	10^{-13}	0.02	0.02	0.02

Table 1: Values of hyperparameters used to generate the four dataset under study.

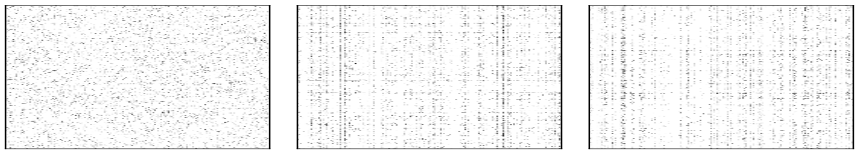


Fig. 3: Visualizing the observed entries of the MovieLens dataset (users in rows, items in columns) from the cluster method (left), real data (middle), SPUP (right). We shuffled the order of rows / columns to eliminate spurious spatial effects in the real data. (This figure is best viewed on screen.)

Model Attributes. One important feature of the SPUP model is the interpretable effect of the parameters on the generated data. Figure 2 demonstrates how the degree distribution across users and items varies as the relevant parameters change. Figure 2b shows how varying budget parameters control the density of the ratings matrix (shifting the distribution to the left or right). As β increases, the probability mass is moved to

higher values of x, widening the distribution. The desired number of ratings per user, c_B, can shift the distribution by adding a constant to all of the budgets.

The effects of parameter shifts on the degree distribution of items is somewhat more subtle. The position of the distribution is determined by the parameter settings for the budget, described above. Figure 2a shows the degree distribution as a and c are varied for budget parameters set to $\beta = 25, c_B = 5$. As a increases, the mass of the distribution is shifted up and to the left. This is because as a increases, the probability density flattens out as a power of x. The constant, c, is used to modify this effect by linearly scaling the vector before normalization. Thus, when c is increased, the distribution will shift horizontally to the left, due to a flattening out of the distribution. As a increases, however, the effects of changes in c are reduced significantly.

In addition to these quantitative evaluations we can also qualitatively compare the generated data from the different methods. Figure 3 shows visually the matrices corresponding to the generated data as the entries of a matrix. Data generated using the cluster method does not have visible structure whereas SPUP and the real data exhibit structure from users and items with varying levels of activity and popularity respectively.

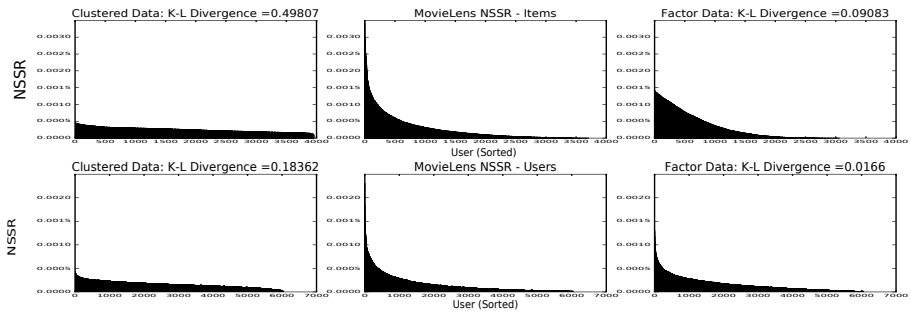


Fig. 4: Normalized sorted sum of ratings (NSSR) of MovieLens Data: cluster method (left), real data (middle), SPUP (right) for products (top) and users (bottom).

Learning with synthetic data. Above we compared different statistics of the synthetic datasets to the statistics of the real data. We now consider a more indirect measure of how realistic the synthetic data is by studying the performance of a collaborative filtering model fit to it.

The exact evaluation procedure is as follows: a) generate synthetic data (or use real data); b) split data into train/validation/test splits; c) evaluate the performance of a standard collaborative filtering model—we use PMF [9]—on this data. We ran this experiment on the MovieLens dataset which is the densest dataset. We did not attempt to compare to the Cluster method due to its overall lower performance at reproducing the real-world dataset.

When fitting PMF to synthetic data we obtain better performance compared to a random predictor of the actual dataset (Mean Normalized Discounted Cumulative Gain,

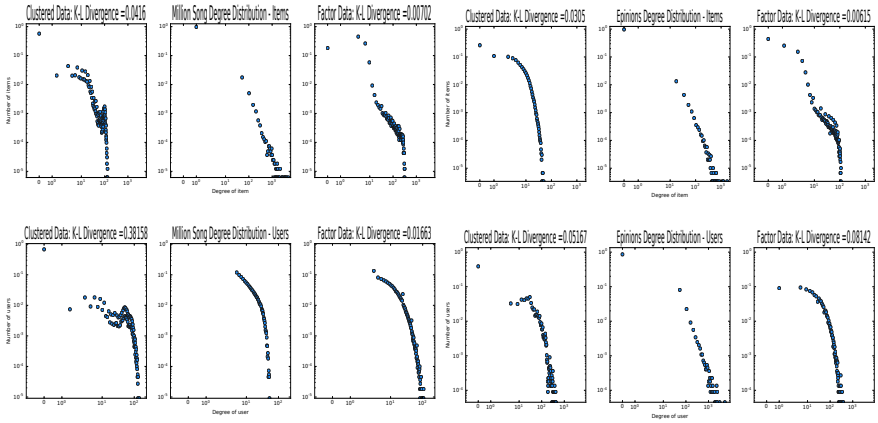


Fig. 5: Degree distribution for MSD. Fig. 6: Degree Distribution for Epinions. Axes represent same quantities as in Fig. 1

higher is better, of 0.28 vs 0.33 on a test set of MovieLens Data). This experiment suggests that there is structure in the data generated by SPUP that PMF can leverage.

6 Discussion

We presented a new model, SPUP, for generating user preference data and show that this model can produce synthetic data matching characteristics of four standard datasets from different preference domains. Further evidence from using it to learn a model suggests that data generated from our method contains meaningful structure.

Comparing the synthetic data generated by SPUP against the real-world datasets confirmed three key properties of our model: 1) SPUP is flexible: it can generate datasets characterized by many different attributes 2) SPUP is interpretable: it is easy to see how changing hyper-parameters lead to changes in the generated data 3) SPUP is stable: it tends to generate consistent-looking data for any given set of parameters, with low variability across measurements.

Given that SPUP is designed to be modular and extensible, it would be easy to modify the process to additionally produce side information (i.e., additional features of users, items or both). One such extension could involve the use of the user factors generated by probabilistic matrix factorization to generate a social network between users. This could then be used in the context of a recommender system to examine how social network data can improve the performance of the system. Another extension, in the same spirit as Pasinato et al. [10], involves the modification of the system to generate *contextual data*: preference data which is influenced in part by information about the context in which the items are consumed (time, location, social interactions). This enables the construction and validation of new models for rich datasets which may or may not exist, thus enabling more rapid testing and development in new domains.

Bibliography

[1] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems, 2009.

[2] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4), 2015.

[3] ICAPS. Ipc. http://www.icaps-conference.org/index.php/Main/Competitions.

[4] Tony Cassandra. Pomdp file repository. http://www.pomdp.org/examples/.

[5] RL-GLUE. Reinforcement learning glue. http://glue.rl-community.org/.

[6] J.P. Cointet and C. Roth. How realistic should knowledge diffusion models be. *Journal of Artificial Societies and Social Simulation*, 10(3), 2007.

[7] J. Leskovec. *Dynamics of large networks*. PhD thesis, Carnegie Mellon University, 2008.

[8] D.B. Rubin. Discussion statistical disclosure limitation. *JOS*, 9(2), 1993.

[9] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *NIPS*, pages 1257–1264, 2008.

[10] M. Pasinato, C. E. Mello, M. A. Aufaure, and G. Zimbro. Generating synthetic data for context-aware recommender systems. In *BRICS-CCI CBIC*, 2013.

[11] Karen H. L. Tso and Lars Schmidt-Thieme. Empirical analysis of attribute-aware recommender system algorithms using synthetic data. *J. of Computers*, 1(4), 2006.

[12] F. Caron and E. B. Fox. Sparse graphs using exchangeable random measures. *ArXiv e-prints*, January 2014.

[13] Mark EJ Newman, Steven H Strogatz, and Duncan J Watts. Random graphs with arbitrary degree distributions and their applications. *Physical review E*, 64(2): 026118, 2001.

[14] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08.*, pages 263–272. IEEE, 2008.

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

[16] D. N. Hoover. Relations on probability spaces and arrays of random variables. Technical report, Institute for Advanced Study, Princeton, NJ, 1979.

[17] Jose M. Hernandez-Lobato, Neil Houlsby, and Zoubin Ghahramani. Stochastic inference for scalable probabilistic modeling of binary matrices. In *ICML*, 2014.

[18] Prem Gopalan, Jake M. Hofman, and David M. Blei. Scalable recommendation with hierarchical poisson factorization. In *UAI*, 2015.

[19] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of 12th ISMIR*, 2011.

[20] J. Tang, H. Gao, and H. Liu. eTrust: Discerning multi-faceted trust in a connected world. In *ACM International Conference on web search and data mining*, 2012.

[21] J. Tang, H. Gao, H. Liu, and A. Das Sarma. eTrust: Understanding trust evolution in an online world. In *Proceedings of the 18th ACM SIGKDD*, pages 253–261. ACM, 2012.

[22] Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of WWW*, 2005.