Factorization Meets the Item Embedding: Regularizing Matrix Factorization with Item Co-occurrence

Dawen Liang*
Netflix Inc.
Los Gatos, CA
dliang@netflix.com

Jaan Altosaar
Princeton University
Princeton, NJ
altosaar@princeton.edu

Laurent Charlin
HEC Montréal
Montreal, Canada
laurent.charlin@hec.ca

David M. Blei
Columbia University
New York, NY
david.blei@columbia.edu

ABSTRACT

Matrix factorization (MF) models and their extensions are standard in modern recommender systems. MF models decompose the observed user-item interaction matrix into user and item latent factors. In this paper, we propose a co-factorization model, CoFactor, which jointly decomposes the user-item interaction matrix and the item-item co-occurrence matrix with shared item latent factors. For each pair of items, the co-occurrence matrix encodes the number of users that have consumed both items. CoFactor is inspired by the recent success of word embedding models (e.g., word2vec) which can be interpreted as factorizing the word co-occurrence matrix. We show that this model significantly improves the performance over MF models on several datasets with little additional computational overhead. We provide qualitative results that explain how CoFactor improves the quality of the inferred factors and characterize the circumstances where it provides the most significant improvements.

Keywords

Collaborative filtering; matrix factorization; item embedding; implicit feedback.

1. INTRODUCTION

Recommender systems model users through their preferences for items. User preferences are often encoded as sets of user-item-preference triplets. For instance "user A gave item B a 4-star rating" or in the case of *implicit data*, which we focus on, "user A clicked on item B". The task of interest is to predict missing user-item preferences given the observed triplets. Predicted preferences can then be used downstream to fuel recommendations.

*This work was mostly done while DL was at Columbia University and LC was at McGill University.

RecSys '16, September 15-19, 2016, Boston, MA, USA
 ISBN 978-1-4503-4035-9/16/09…$15.00
DOI: http://dx.doi.org/10.1145/2959100.2959182

The preference triplets can be seen as the sparse representation of a user-item *preference matrix* (or *click matrix*). Predicting preferences can be seen as filling in the missing entries of this matrix. Models such as matrix factorization—which decompose the preference matrix into user and item factors [10, 20]—are standard for preference prediction: their performance is high [10], maximum *a posteriori* inference can be done efficiently with closed-form updates [9], and they can be composed to incorporate additional side information (e.g., [1, 4, 13, 23, 26]).

Encoding user preferences in a matrix and modeling it with matrix factorization is a particular modeling assumption. In this paper we explore an alternative which models item co-occurrence across users. We posit that pairs of items which are often consumed in tandem by different users are similar. This is similar to modeling a set of documents (users) as a bag of co-occurring words (items). In that context frequently co-occurring words are likely to be about the same topic. For example, in a corpus of scientific papers "planet" and "Pluto" are likely to frequently co-occur. A similar idea has been explored in recommendations for next-item prediction [24]. Item co-occurrence information is, in principle, available to matrix factorization, but it may not be easy to infer from the click matrix: matrix factorization models are bi-linear with limited modeling capacity.

We propose a co-factorization model, CoFactor, which simultaneously factorizes both the click matrix and the item co-occurrence matrix. The factorization of item co-occurrence is inspired by the recent models for learning word embedding from sequences of words [14, 11]. We learn item embedding using the sets of items each user has consumed (or rated), and the co-occurrence counts of these items across users in the data.

We show that learning CoFactor from data can be done efficiently with coordinate updates. We use a sequence of closed-form updates which scale in the number of observed preference triplets. CoFactor outperforms matrix factorization [9] across datasets of user clicking on scientific articles, rating movies, and listening to music. We also provide exploratory results to better understand the effectiveness of our method. This shows that we outperform standard matrix factorization due to the ability of our model to exploit co-occurrence patterns for rare items (items not consumed by many users).

2. THE COFACTOR MODEL

We first introduce the two building blocks of the CoFactor model: matrix factorization (MF) and item embedding. Then we describe CoFactor and how to compute with it.

Matrix factorization. MF is standard in collaborative filtering [10]. Given the sparse user-item interaction matrix $\boldsymbol{Y} \in \mathbb{R}^{U \times I}$ from U users and I items, MF decomposes it into the product of user and item latent factors denoted $\theta_u \in \mathbb{R}^K$ $(u = 1, \ldots, U)$ and $\beta_i \in \mathbb{R}^K$ $(i = 1, \ldots, I)$, respectively. Here we focus on implicit feedback data [9, 16]. However, this is not a limiting aspect of CoFactor—it can be readily extended to the explicit feedback setting. Matrix factorization for implicit feedback model [9] optimizes the following objective:

$$\mathcal{L}_{\text{mf}} = \sum_{u,i} c_{ui}(y_{ui} - \theta_u^\top \beta_i)^2 + \lambda_\theta \sum_u \|\theta_u\|_2^2 + \lambda_\beta \sum_i \|\beta_i\|_2^2. \tag{1}$$

The scaling parameter c_{ui} is a hyperparameter that is normally set to be $c_{y=1} > c_{y=0}$. It can be tuned to balance the unobserved ratings $(y = 0)$ which far outnumber the observed ratings $(y = 1)$ in most click data. The regularization parameters λ_θ and λ_β are hyperparameters which are selected from validation data. The (global) optimum of $\mathcal{L}_{\text{mf}}$ can be interpreted as the maximum *a posteriori* estimate of the probabilistic Gaussian matrix factorization model [20].

Word embedding. Word embedding models (e.g., word2vec [14]) have gained success in many natural language processing tasks. Given a sequence of words, these models embed each word into a low-dimensional continuous space (relative to the vocabulary size). In the skip-gram model, the objective is to predict context words—surrounding words within a fixed window—given the current word. Stochastic gradient descent (SGD) with negative sampling is normally used to train a word embedding model (see Mikolov et al. [14]).

Levy and Goldberg [11] show an equivalence between skip-gram word2vec trained with negative sampling value of k and implicit factorizing the pointwise mutual information (PMI) matrix shifted by $\log k$. PMI between a word i and its context word j is defined as:

$$\text{PMI}(i, j) = \log \frac{P(i, j)}{P(i)P(j)}.$$

Empirically, it is estimated as:

$$\text{PMI}(i, j) = \log \frac{\#(i, j) \cdot D}{\#(i) \cdot \#(j)}. \tag{2}$$

Here $\#(i, j)$ is the number of times word j appears in the context of word i. $\#(i) = \sum_j \#(i, j)$ and $\#(j) = \sum_i \#(i, j)$. D is the total number of word-context pairs.

After making the connection between word2vec and implicit matrix factorization, Levy and Goldberg [11] further propose to perform word embedding by spectral dimensionality reduction (e.g., singular value decomposition) on the (sparse) shifted positive PMI (SPPMI) matrix:

$$\text{SPPMI}(i, j) = \max\{\text{PMI}(i, j) - \log k, 0\}$$

This is attractive because it does not require tuning an optimization procedure.[1] We will follow the similar approach in CoFactor.

Item embedding. Users consume items sequentially. Sequences of items are analogous to sequences of words, so we can use word embedding models to learn item embeddings (e.g., Guàrdia-Sebaoun et al. [5]).

Define $\boldsymbol{M} \in \mathbb{R}_+^{I \times J}$ as the co-occurrence SPPMI matrix for item consumptions.[2] We can obtain item embedding by factorizing $\boldsymbol{M}$. In this paper, we do not assume that the order (or timestamp) of each user's item consumption is available. For a consumed item i from a particular user, we define its context j as all other items in this user's click history. We use Equation (2) to compute m_{ij} with the empirical estimates of $\text{PMI}(i, j)$. In this estimate, $\#(i, j)$ corresponds to the number of users that consumed both item i and j. By defining the context in this way, CoFactor does not require any additional information other than what is already available in the standard MF model. This departs, at least in spirit, from the traditional word embedding models and Guàrdia-Sebaoun et al. [5], where context is defined as neighbors within a fixed window. Those methods require the order within a sequence.

Our definition of context is a design choice and CoFactor can work with any type of co-occurrence SPPMI matrix. Context could be data- and problem-specific. For example, if we did know the order of item consumption, we could break it down into shorter segments of sequences.

The CoFactor model. Both MF and item embedding models infer latent item representations. The difference is that the item representations inferred from MF encode users' preferences for items, while the item embeddings must explain item co-occurrence patterns. We propose learning these representations jointly:

$$\mathcal{L}_{\text{co}} = \overbrace{\sum_{u,i} c_{ui}(y_{ui} - \theta_u^\top \beta_i)^2}^{\text{MF}} + \overbrace{\sum_{m_{ij} \neq 0} \left(m_{ij} - \beta_i^\top \gamma_j - w_i - c_j\right)^2}^{\text{item embedding}} + \lambda_\theta \sum_u \|\theta_u\|_2^2 + \lambda_\beta \sum_i \|\beta_i\|_2^2 + \lambda_\gamma \sum_j \|\gamma_j\|_2^2 \tag{3}$$

This is the CoFactor model objective. The item factor (or item embedding) β_i is shared by both the MF and item embedding parts of the objective. We include context embedding γ_j as additional model parameter. We do not regularize item and context biases w_i and c_j. Notice that the item embeddings β_i must account for both user-item interactions and item-item co-occurrence. We can also interpret this objective as regularizing the traditional MF objective $\mathcal{L}_{\text{mf}}$ (Equation (1)) with item embeddings learned by factorizing the item co-occurrence matrix.[3]

The scaling parameter c_{ui} in the CoFactor objective enforces a balance between the unobserved zeros and observed ones in the click matrix. Additionally, its relative scale will balance the MF and item embedding parts of the model. Setting its relative scale smaller will impose more regularization

[1] Under this setting, k becomes a hyperparameter that controls the sparsity of the SPPMI matrix, as opposed to the original skip-gram word2vec model [14] where k explicitly controls the number of negative examples sampled in SGD.

[2] For generality, we use $i = 1, \ldots, I$ and $j = 1, \ldots, J$ to index an item and its context. In experiments, the set of items and the set of context items are the same.

[3] This model does not have a clear generative interpretation as the regular MF model, since the user-item interaction matrix $\boldsymbol{Y}$ deterministically defines the co-occurrence SPPMI matrix $\boldsymbol{M}$.

from the co-occurrence matrix and vice versa. This scale is an important hyperparameter. In our empirical study, we will select it based on recommendation performance on a validation set.

2.1 Inference

We take the gradient of CoFactor objective $\mathcal{L}_{\text{co}}$ (Equation (3)) with respect to the model parameters $\{\theta_{1:U}, \beta_{1:I}, \gamma_{1:J}, w_{1:I}, c_{1:J}\}$ and set it to zero. This gives the following closed-form coordinate updates:

$$\theta_u \leftarrow \left(\sum_i c_{ui}\beta_i\beta_i^\top + \lambda_\theta \mathrm{I}_K\right)^{-1}\left(\sum_i c_{ui}y_{ui}\beta_i\right)$$

$$\beta_i \leftarrow \left(\sum_u c_{ui}\theta_u\theta_u^\top + \sum_{j:m_{ij}\neq 0}\gamma_j\gamma_j^\top + \lambda_\beta \mathrm{I}_K\right)^{-1} * \left(\sum_u c_{ui}y_{ui}\theta_u + \sum_{j:m_{ij}\neq 0}(m_{ij}-w_i-c_j)\gamma_j\right)$$

$$\gamma_j \leftarrow \left(\sum_{i:m_{ij}\neq 0}\beta_i\beta_i^\top + \lambda_\gamma \mathrm{I}_K\right)^{-1}\left(\sum_{i:m_{ij}\neq 0}(m_{ij}-w_i-c_j)\beta_i\right)$$

$$w_i \leftarrow \frac{1}{|\{j : m_{ij}\neq 0\}|}\sum_{j:m_{ij}\neq 0}(m_{ij}-\beta_i^\top\gamma_j-c_j)$$

$$c_j \leftarrow \frac{1}{|\{i : m_{ij}\neq 0\}|}\sum_{i:m_{ij}\neq 0}(m_{ij}-\beta_i^\top\gamma_j-w_i)$$

These updates are very similar to the alternating least squares (ALS) of Hu et al. [9]. The main difference is in how we update β_i in CoFactor. We can view each update of ALS as a weighted ridge regression. Therefore, the update for β_i is collectively performing ridge regression with two sources of data: the click data y_{ui} with covariates θ_u, and the co-occurrence data m_{ij} with covariates γ_j. Iteratively performing these updates reaches a stationary point of the model objective $\mathcal{L}_{\text{co}}$.

Complexity We set the scaling parameter c_{ui} as $c_{ui} = \ell(1+\alpha\cdot y_{ui})$ where ℓ is the relative scale. This is similar to Hu et al. [9] and we use the same pre-computing trick to speed up the computation for updating θ_u and β_i. For the item embeddings, the complexity scales linearly with the number of non-zero entires in the SPPMI matrix, which is usually very sparse. Furthermore, all the coordinate updates are embarrassingly parallelizable across users and items. Constructing the SPPMI matrix can be time-consuming as it scales quadratically with the number of items consumed by each user. Fortunately, the SPPMI matrix only needs to be constructed once and it is parallelizable across users. Our Python implementation of the CoFactor model scales easily to large datasets.[4]

3. RELATED WORK

The user and item factors learned through matrix factorization are usually regularized with their ℓ_2-norm. It is also standard to further regularize the factors with *side information* [1, 2, 7, 12], which comprises user, item, and/or user-item covariates (e.g., item metadata or user demographics which can be indicative of user preferences). We can interpret CoFactor as regularizing the learned item factors beyond simple ℓ_2-norm. The distinguishing feature of CoFactor is that the regularization comes from a deterministic non-linear transformation of the original user-item preference data instead of additional data.

This kind of regularization (either through incorporation of side information or re-encoding the input as in CoFactor) can alternatively be viewed as enforcing more complex structure in the prior of the latent factors. For example, Ranganath et al. [17] find that imposing a deep exponential family prior on the matrix factorization model, which implicitly conditions on consumption counts in a non-linear fashion, can be helpful in reducing the effect of extremely popular (or extremely unpopular) items on held-out recommendation tasks. This is analogous to our findings with the CoFactor model.

Collective matrix factorization [25] explores co-factorizing multiple matrices in the context of relational learning. For example, a database with users, movies, and genres contains two relational matrices: one representing users' ratings of movies, and the other representing the genres each movie belongs to. Collective matrix factorization jointly factorizes these two matrices with shared movie factors to leverage the additional genre information for better preference prediction. We can consider CoFactor as a special case of collective matrix factorization with the user-item preference relation deterministically defines the item-item co-occurrence relation. Our contribution lies in the construction of the item-item relation, which is motivated by the recent development of word embedding models [14, 11].

Guàrdia-Sebaoun et al. [5] observe that user trajectories over items can be modeled much like sequences of words. In detail, they use a word embedding model [14] to learn item embeddings. User embeddings are then inferred as to predict the next item in the trajectory (given the previous item). User and item embeddings are finally used as covariates in a subsequent regression model of user-item preferences. Learning item embeddings from user trajectories is similar to our work. The difference is that we treat the items consumed by each user as exchangeable (i.e., we do not assume that the trajectories are given, rather we assume that we are given an unordered set of items for each user). Additionally we show that jointly learning all parameters yields higher performance.

In item-based collaborative filtering [22], an item-item similarity matrix is built from user consumption data, much like the SPPMI matrix in CoFactor. The difference is that in CoFactor we only use the SPPMI matrix to regularize the item factors, while in item-based collaborative filtering we directly use the similarity matrix to predict missing preferences. Previous work shows that the performance of item-based collaborative filtering is highly sensitive to the choice of similarity metric and data normalization [8]. It would be interesting to consider SPPMI matrix as similarity matrix in item-based collaborative filtering.

4. EMPIRICAL STUDY

We study the performance of CoFactor both quantitatively and qualitatively. We provide insights into its performance by exploring the resulting fits. We highlight the following results:

- On three datasets, the CoFactor model performs significantly better than the state-of-the-art weighted matrix factorization (WMF) model [9].
- We investigate where CoFactor improves performance

[4] https://github.com/dawenl/cofactor

	ArXiv	ML-20M	TasteProfile
# of users	25,057	111,148	221,830
# of items	63,003	11,711	22,781
# interactions	1.8M	8.2M	14.0M
% interactions	0.12%	0.63%	0.29%
with timestamps	yes	yes	no

Table 1: Attributes of datasets after preprocessing. Interactions are non-zero entries (listening counts, watches, and clicks). % interactions refers to the density of the user-item interaction matrix ($\boldsymbol{Y}$). For datasets with timestamps, we ensure there is no overlap in time between the training and test sets.

to understand why jointly factoring both the user click matrix and item co-occurrence matrix boosts certain recommendations.

- We demonstrate the importance of joint learning by comparing to a two-stage model that first does word2vec for item embeddings and then fits a MF.

4.1 Datasets

We study three medium- to large-scale user-item consumption datasets from various domains: 1) scientific articles data from the arXiv[5]; 2) users' movie viewing data from MovieLens [6]; 3) the taste profile subset of the million song dataset [3]. In more details:

ArXiv: User-paper clicks derived from log data collected in the first half of 2012 from the arXiv pre-print server. The data is binarized (multiple clicks by the same user on the same paper are considered to be a single click). We preprocess the data to ensure that all users and items have a minimum of ten clicks.

MovieLens-20M (ML-20M): User-movie ratings collected from a movie recommendation service. We binarize explicit data by keeping the ratings of four or higher and interpret them as implicit feedback. We only keep users who have watched at least five movies.

TasteProfile: User-song play counts collected by the music intelligence company Echo Nest[6]. We binarize play counts and interpret them as implicit preference data. We keep users with at least 20 songs in their listening history and songs that are listened to by at least 50 users.

ArXiv and Movielens are time stamped. To create the training/validation/test splits for these data, we order all the user-item interactions by time and take the first 80% as training data, from which we randomly selected 10% as the validation set. For the remaining 20% of the data, we only keep the users and items that appear in the training and validation sets to obtain the test set. For TasteProfile, which is not timestamped, we randomly split the observed user-item interactions into training/validation/test sets with 70/10/20 proportions. Table 1 summarizes the dimensions of all the datasets after preprocessing.

4.2 Metrics

We use ranking-based metrics: Recall@M, truncated normalized discounted cumulative gain (NDCG@M), and mean average precision (MAP@M). For each user, all the metrics compare the predicted rank of (unobserved) items with their true rank. While Recall@M considers all items ranked within the first M to be equivalent, NDCG@M and MAP@M use a monotonically increasing discount to emphasize the importance of higher ranks versus lower ones. Formally, define π as a permutation over all the items, $\mathbb{1}\{\cdot\}$ is the indicator function, and $u(\pi(i))$ returns 1 if user u has consumed item $\pi(i)$. CoFactor predicts ranking π for each user by sorting the predicted preference $\theta_u^\top \beta_i$ for $i = 1, \ldots, I$.

Recall@M for user u is

$$\text{Recall@}M(u, \pi) := \sum_{i=1}^{M} \frac{\mathbb{1}\{u(\pi(i)) = 1\}}{\min(M, \sum_{i'}^{I} \mathbb{1}\{u(\pi(i')) = 1\})}.$$

The expression in the denominator evaluates to the minimum between M and the number of items consumed by user u. This normalizes Recall@M to have a maximum of 1, which corresponds to ranking all relevant items in the top M positions.

DCG@M for user u is

$$\text{DCG@}M(u, \pi) := \sum_{i=1}^{M} \frac{2^{\mathbb{1}\{u(\pi(i))=1\}} - 1}{\log(i + 1)}.$$

NDCG@M is the DCG@M normalized to $[0, 1]$, where one signifies a perfect ranking.

Mean average precision (MAP@M) calculates the mean of users' average precision (AP). The average precision AP@M for a user u is

$$\text{AP@}M(u, \pi) := \sum_{i=1}^{M} \frac{\text{Precision@}i(u, \pi)}{\min(i, \sum_{i'}^{I} \mathbb{1}\{u(\pi(i')) = 1\})}.$$

4.3 Experiment protocols

We compare CoFactor to weighted matrix factorization (WMF) [9]. Given the similarity between the CoFactor objective $\mathcal{L}_{\text{co}}$ (Equation (3)) and WMF objective $\mathcal{L}_{\text{mf}}$ (Equation (1)), we can attribute the performance differences to the regularization imposed by the co-occurrence SPPMI matrix.

In all fits, the dimension of the latent space K is set to 100. The CoFactor model is trained following the inference algorithm described in Section 2.1. We monitor the convergence of the algorithm using NDCG@100 on the validation set for both CoFactor and WMF.

Both CoFactor and WMF require hyperparameter tuning. We first select the best hyperparameters for WMF (the weight $c_{y=0}$ and $c_{y=1}$, the regularization parameters $\lambda = \lambda_\theta = \lambda_\beta$) based on their performance on the validation set. For CoFactor, we then keep the same $c_{y=1}/c_{y=0}$ ratio and regularization parameters, and grid search for the best relative scale $\ell \in \{0.01, 0.05, 0.1, \ldots, 1, 5, 10\}$ (Section 2.1). Note when scaling c_{ui}, we also scale λ_θ and λ_β. The value of the relative scale indicates the importance of regularization with co-occurrence information. If a large scaling parameter c_{ui} is selected, the MF part of the model can be effective at recommendation on its own without significant help from the information provided by the co-occurrence matrix. (As the scaling parameter goes to infinity, CoFactor effectively reduces to WMF, as the MF part of the objective will completely dominate.) On the other hand, a smaller scaling parameter indicates that the model benefits from the co-occurrence patterns in the observed user behavior

[5] http://arxiv.org
[6] http://the.echonest.com

	ArXiv		ML-20M		TasteProfile	
	WMF	CoFactor	WMF	CoFactor	WMF	CoFactor
Recall@20	0.063	**0.067**	0.133	**0.145**	0.198	**0.208**
Recall@50	0.108	**0.110**	0.165	**0.177**	0.286	**0.300**
NDCG@100	0.076	**0.079**	0.160	**0.172**	0.257	**0.268**
MAP@100	0.019	**0.021**	0.047	**0.055**	0.103	**0.111**

Table 2: Comparison between the widely-used weighted matrix factorization (WMF) model [9] and our CoFactor model. CoFactor significantly outperforms WMF on all the datasets across all metrics. The improvement is most pronounced on the movie watching (ML-20M) and music listening (TasteProfile) datasets.

data. We also grid search for the negative sampling value $k \in \{1, 2, 5, 10, 50\}$, which effectively modulates how much to shift the empirically estimated PMI matrix.

4.4 Analyzing the CoFactor model fits

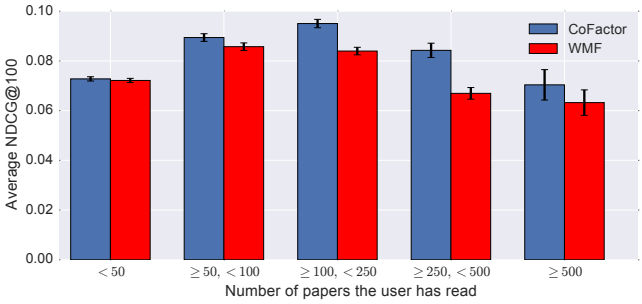
Table 2 summarizes the quantitative results. Each metric is averaged across all users in the test set. CoFactor outperforms WMF [9] on all datasets and across all metrics. The improvement is very clear for the MovieLens (ML-20M) and music (TasteProfile) data. We emphasize that both models make use of the same data and optimize similar objectives, with CoFactor benefiting from an extra co-occurrence regularization term.

Exploratory analysis

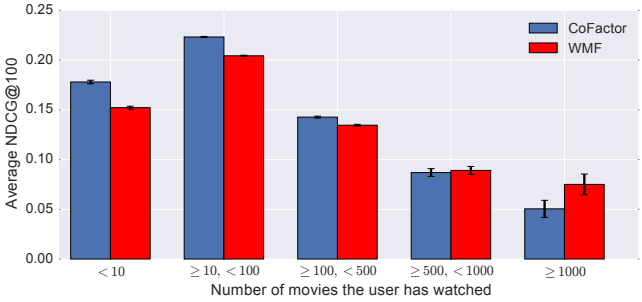
When does CoFactor do better/worse? Figure 1 shows the breakdown of NDCG@100 by user activity in the training set for all three datasets. Although details vary across datasets, the CoFactor model consistently improves recommendation performance for users who have only consumed a small number of items. This makes sense because MF cannot accurately infer inactive users' preferences, while CoFactor explicitly makes use of the additional signal from item co-occurrence patterns to learn better latent representations, even when user-item interaction data is scarce. For active users (the rightmost bar pairs in each plot), WMF does worse than CoFactor on ArXiv, better on ML-20M, and roughly the same on TasteProfile. However, since active users are the minority (most recommendation datasets have a long tail), the standard error is also bigger.

To understand when CoFactor makes better recommendation, we compare it to WMF in how it ranks rare items. Figure 2 shows the histogram of ranks (derived from predicted scores) for movies with less than 10 viewings in the training set (which consists of 3,239 movies out of the 11,711 total movies). We report results using four randomly-selected users on the MovieLens ML-20M data. (The general trend is consistent across the entire dataset, and we observe a similar patten for the TasteProfile dataset.) WMF mostly ranks rare movies around the middle of its recommendation list, which is expected because the collaborative filtering model is driven by item popularity.[7] On the other hand, CoFactor can push these rare movies both to the top and bottom of its recommendations. We did not observe as clear of a pattern on the ArXiv data. We conjecture that this is because the ArXiv data is less popularity-biased than ML-20M and TasteProfile: the mean (median) of users who consumed an

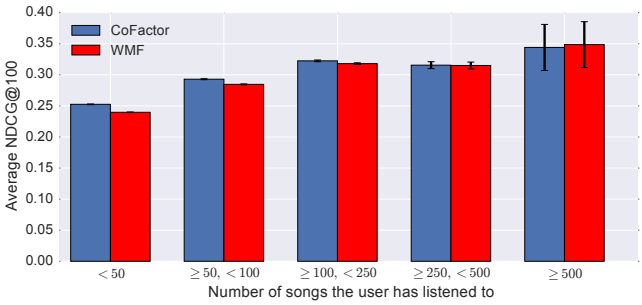
[7]It is also reasonable given the relatively high model uncertainty around rare items.



(a) ArXiv



(b) ML-20M



(c) TasteProfile

Figure 1: Average normalized discounted cumulative gain (NDCG@100) breakdown for both CoFactor and weighted matrix factorization (WMF) [9] based on user activity on different datasets (higher is better). Error bars correspond to one standard error. There is some variation across datasets, but CoFactor is able to consistently improve recommendation performance for users who have only consumed a small amount of items.

item is 597 (48) for ML-20M and 441 (211) for TasteProfile, while only 21 (12) for ArXiv.

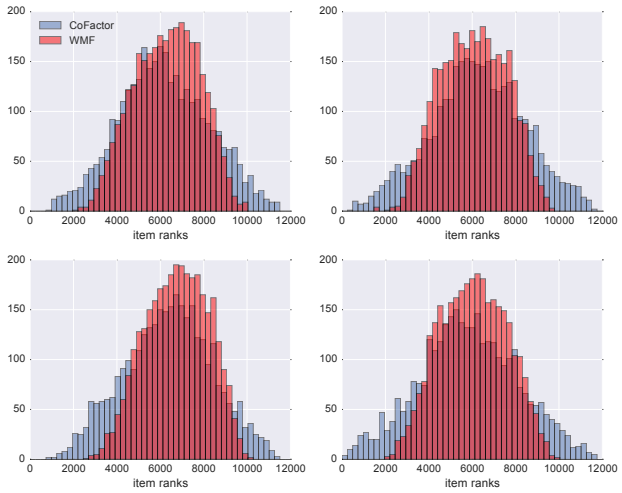


Figure 2: The histogram of ranks from four randomly selected users for movies with less than 10 watches in the MovieLens ML-20M training set for both CoFactor and weighted matrix factorization (WMF) [9]. The extra regularization in CoFactor allows it to push these rare items further up and down its recommendation list for each user.

In Figure 3, we compare CoFactor and WMF for a particular user who has watched many popular movies (e.g., "Toy Story" and "Fight Club"), as well as some rare French movies (e.g. "Mouchette" and "Army of Shadows"). We find that WMF highly ranks popular movies and places them on top of its recommendations. Even when it recommends a French movie ("That Obscure Object of Desire") to this user, it is a relatively popular one (as indicated by the number of users who have watched it). In contrast, CoFactor will better balance between the popular movies and relevant (but relatively rare) French movies in its recommendation. This makes sense because rare French movies co-occur in the training set, and this is captured by the SPPMI matrix. We find similar exploratory examples in TasteProfile.

This explains the performance gap between CoFactor and WMF among active users on ML-20M, as CoFactor could potentially rank popular movies lower than WMF. For active users, they are more likely to have watched popular movies in both the training and test sets. When we look at the top 100 users where WMF does better than CoFactor in terms of NDCG@100, we find that the performance difference is never caused by WMF recommending a rare movie in the test set that CoFactor fails to recommend.

How important is joint learning? One of the main contributions of the proposed CoFactor model is incorporating both MF and item embedding in a joint learning objective. This "end-to-end" approach has proven effective in many machine learning models. To demonstrate the advantage of joint learning, we conduct an alternative experiment on ArXiv where we perform the learning in a two-stage process. First, we train a skip-gram word2vec model on the click data, treating users' entire click histories as the context—this is equivalent to factorizing the SPPMI matrix in CoFactor model. We then fix these learned item embeddings from word2vec as the latent factors β_i in the MF model, and learn user latent factors θ_u. Learning θ_u in this way is the same as one iteration of CoFactor, which is effectively doing a weighted ridge regression.

	WMF	CoFactor	word2vec + reg
Recall@20	0.063	0.067	0.052
Recall@50	0.108	0.110	0.095
NDCG@100	0.076	0.079	0.065
MAP@100	0.019	0.021	0.016

Table 3: Comparison between joint learning (CoFactor) and learning from a separate two-stage (word2vec + reg) process on ArXiv. Even though they make similar modeling assumptions, CoFactor provides superior performance.

We evaluate the model learned from this two-stage process (word2vec + reg) and report the quantitative results in Table 3. (The results for WMF and CoFactor are copied from Table 2 for comparison.) The performance of the two-stage model is much worse. This is understandable because the item embeddings learned from word2vec are certainly not as well-suited for recommendation as the item latent factors learned from MF. Using this item embedding in the second stage to infer user preferences will likely lead to inferior performance. On the other hand, CoFactor is capable of finding the proper balance between MF and item embeddings to get the best of both worlds.

5. DISCUSSION

It is interesting that CoFactor improves over WMF by re-using the preference data in a different encoding. The item co-occurrence information is, in principle, available from the user-item interaction data to MF. However, as a bilinear model with limited capacity, MF cannot uncover such information. On the other hand, highly non-linear models, such as deep neural networks, have shown limited success at the task of preference prediction [21, 17]. Our approach is somewhat in between, where we have used a deterministic non-linear transformation to re-encode the input.

A natural question is whether it is always better to incorporate item co-occurrence information. This depends on the problem and specific data. However, as mentioned in Section 4.3, as the relative scale on the weight c_{ui} goes to infinity, CoFactor is effectively reduced to WMF. Therefore, as long as there is additional information that is useful in the co-occurrence matrix, in principle, the performance of CoFactor is lower bounded by that of WMF.

5.1 Possible extensions to the model

It is straightforward to extend our CoFactor model to explicitly handle sessions of user-item consumption data by constructing a session-based co-occurrence matrix, as discussed in Section 2. This is likely to yield improved performance in some settings, such as in purchase behavior or music listening data, where the notion of a 'shopping cart' or 'playlist' induce session-based patterns.

Adding user-user co-occurrence to our model to regularize user latent factors is another natural extension, which we leave to future work. We anticipate this extension to yield further improvements, especially for users in the long tail. The type of regularization we have added to MF models can

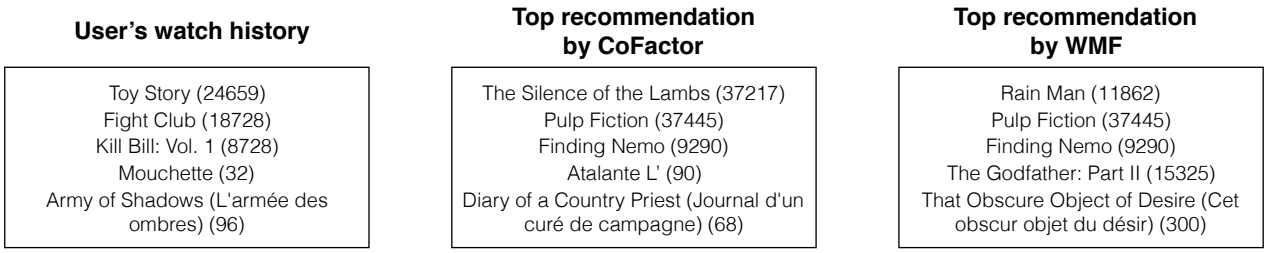


Figure 3: Comparison between CoFactor and WMF [9] for a particular user in MovieLens ML-20M data. On the right, we show a subset of the user's watch history from the training set. The number in the parenthesis after the movie title is the number of users who have watched this movie in the training set. The top recommendations by CoFactor and WMF are shown in the middle and left tables, respectively. Here only the movies that are watched by the same user in the test set are shown. The NDCG@100 of this user is high for both CoFactor and WMF (0.893 vs 0.783).

also be extended to higher-order co-occurrence patterns. It is unclear which datasets may benefit from this, but this type of prior structure may be interesting to explore. It may help for complex recommendation data where long-term higher-order interactions are assumed to be important. For example, in the ArXiv dataset, researchers reading the same sets of papers over many years could exhibit such patterns. It is unclear how best to capture this type of co-occurrence in a procedure. Our regularization approach provides an initial solution.

Another line of future work is to explore CoFactor's co-occurrence regularization in other collaborative filtering methods. For example, we can regularize Bayesian Personalized Ranking (BPR) [19], factorization machine [18], or sparse linear method (SLIM) [15] with item and/or user co-occurrence patterns.

6. CONCLUSION

We present CoFactor, a regularization scheme for MF models inspired by word embeddings. We force the standard MF objective to share item representations with a factorized shifted positive pointwise mutual information matrix of item co-occurrence counts. This is analogous to word embedding models [11, 14] which factorize a similar matrix of word co-occurrence patterns in text corpora. We show that this type of joint factorization yields performance boosts in recommendation metrics in a variety of settings: recommending papers to researchers on ArXiv, movies on MovieLens, and music on TasteProfile.

We identify the scenarios where CoFactor outperforms standard MF, which gives further insights into the benefits of 'reusing' the data. We show that regularizing using item co-occurrence counts enables CoFactor to recommend rare items by capturing their co-occurrence patterns, whereas this feature is absent in standard MF. We point to the potential of alternative methods such as regularizing with user-user co-occurrence or in the context of other widely-used collaborative filtering models.

Acknowledgement

We thank the anonymous reviewers for helpful feedback. This work is supported by NSF IIS-1247664, ONR N00014-11-1-0651, DARPA FA8750-14-2-0009, DARPA N66001-15-C-4032, Adobe, and the Sloan Foundation.

References

[1] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD*, pages 19–28, 2009.

[2] A. Almahairi, K. Kastner, K. Cho, and A. Courville. Learning distributed representations from reviews for collaborative filtering. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 147–154, 2015.

[3] T. Bertin-Mahieux, D. P. W. Ellis, B. Whitman, and P. Lamere. The million song dataset. In *Proceedings of the 12th International Society for Music Information Retrieval Conference*, pages 591–596, 2011.

[4] 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.

[5] E. Guàrdia-Sebaoun, V. Guigue, and P. Gallinari. Latent trajectory modeling: A light and efficient way to introduce time in recommender systems. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 281–284, 2015.

[6] F. M. Harper and J. A. Konstan. The MovieLens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(4):19, 2015.

[7] R. He and J. McAuley. VBPR: Visual Bayesian personalized ranking from implicit feedback. In *AAAI Conference on Artificial Intelligence*, 2016.

[8] J. Herlocker, J. A. Konstan, and J. Riedl. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Information retrieval*, 5(4): 287–310, 2002.

[9] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Data Mining, IEEE International Conference on*, pages 263–272, 2008.

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

[11] O. Levy and Y. Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, pages 2177–2185. 2014.

[12] D. Liang, M. Zhan, and D. P. W. Ellis. Content-aware collaborative music recommendation using pre-trained neural networks. In *Proceedings of the 16th International Society for Music Information Retrieval Conference*, pages 295–301, 2015.

[13] J. McAuley and J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Recommender Systems*, 2013.

[14] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[15] X. Ning and G. Karypis. SLIM: Sparse linear methods for top-n recommender systems. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 497–506. IEEE, 2011.

[16] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *Data Mining, IEEE International Conference on*, pages 502–511, 2008.

[17] R. Ranganath, L. Tang, L. Charlin, and D. M. Blei. Deep exponential families. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, AISTATS*, 2015.

[18] S. Rendle. Factorization machines. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, pages 995–1000, 2010.

[19] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 452–461, 2009.

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

[21] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798, 2007.

[22] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

[23] H. Shan and A. Banerjee. Generalized probabilistic matrix factorizations for collaborative filtering. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 1025–1030. IEEE, 2010.

[24] G. Shani, D. Heckerman, and R. I. Brafman. An MDP-based recommender system. *Journal of Machine Learning Research*, 6:1265–1295, 2005.

[25] A. P. Singh and G. J. Gordon. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 650–658. ACM, 2008.

[26] C. Wang and D. Blei. Collaborative topic modeling for recommending scientific articles. In *Knowledge Discovery and Data Mining*, 2011.