Practical Linear Models for Large-Scale One-Class Collaborative Filtering

Suvash Sedhain,[†‡] Hung Bui,[*] Jaya Kawale,[*] Nikos Vlassis,[*]
Branislav Kveton,[*] Aditya Krishna Menon,[‡†] Trung Bui,[*] Scott Sanner[§]
{[†]Australian National University, [‡]Data61}, Canberra, ACT, Australia
[*] Adobe Research, San Jose, USA; [§] University of Toronto, Toronto, Canada
suvash.sedhain@anu.edu.au,{hubui,kawale,vlassis,kveton}@adobe.com
aditya.menon@nicta.com.au, bui@adobe.com, ssanner@mie.utoronto.ca

Abstract

Collaborative filtering has emerged as the de facto approach to personalized recommendation problems. However, a scenario that has proven difficult in practice is the one-class collaborative filtering case (OC-CF), where one has examples of items that a user prefers, but no examples of items they do not prefer. In such cases, it is desirable to have recommendation algorithms that are personalized, learning-based, and highly scalable. Existing linear recommenders for OC-CF achieve good performance in benchmarking tasks, but they involve solving a large number of a regression subproblems, limiting their applicability to large-scale problems. We show that it is possible to scale up linear recommenders to big data by learning an OC-CF model in a randomized low-dimensional embedding of the user-item interaction matrix. Our algorithm, Linear-FLow, achieves state-of-the-art performance in a comprehensive set of experiments on standard benchmarks as well as real data.

1 Introduction

Personalized recommendation systems are a core component in many modern e-commerce services [Leavitt, 2006]. Collaborative filtering (CF) is the de-facto standard approach for making recommendations based on information in a database of item preferences from a population of users [Goldberg *et al.*, 1992; Sarwar *et al.*, 2001]. Most of the work on CF has considered the explicit feedback setting, where users express both positive and negative preferences for items in terms of ratings or likes/dislikes [Koren *et al.*, 2009]. In contrast, in the one-class collaborative filtering setting (OC-CF) [Pan *et al.*, 2008], we do not have explicit negative preference information. For example, consider recommending items to users of an e-commerce website based on their purchase history. One can assume that a purchase indicates a positive preference for an item. However, the lack of a purchase does not necessarily indicate a negative preference; it may just be that the user is unaware of the item.

In designing a real world recommender system there are various factors to be considered [Adomavicius and Tuzhilin, 2005; Ricci *et al.*, 2011]. First and foremost, a recommender system should produce good recommendations which can be quantified in terms of *relevance*. The relevance of a recommender system is measured using evaluation metrics such as precision@k, recall@k etc. Second, a recommender system should be highly *scalable*. In modern day applications, it is very common to have millions of users and items and billions of recorded interactions. A model should be able to handle the data in a computationally efficient manner as the number of users and items grow to this scale. Third, recommending similar items is very prevalent in real-world recommender systems. For instance, Amazon.com shows product suggestions to customers based on the items they are browsing [Linden *et al.*, 2003a]. Hence, a recommendation algorithm should ideally produce *item similarities*. Also, *interpretability* of the recommendation is very critical in persuading users. By explaining the recommendations, the system becomes more transparent, builds users' trust in the system and convinces them to consume the recommended items. For example, the "why this was recommended" feature on Netflix.com explains to users their recommendations by showing them the similar movies that they liked in the past. A lack of interpretability weakens the ability to persuade the users in their decision making [Vig *et al.*, 2009].

While there is a rich literature on OC-CF (discussed subsequently), to our knowledge, all existing methods lack one or more desiderata. Neighborhood-based methods [Sarwar *et al.*, 2001; Linden *et al.*, 2003b] are scalable, incorporate a similarity metric in the model and give explainable recommendations. However, the relevance of their recommendation is weaker as compared to other methods. Matrix factorization models [Hu *et al.*, 2008] optimize a non-convex objective whose solution is sensitive to initialization and hyperparameters. Matrix factorization models are not competitive in terms of top-k ranking performance [Ning and Karypis, 2011; Sedhain *et al.*, 2016]. Also, the recommendations are not explainable and there is no notion of similarity in the model. On the other hand, Linear recommenders [Ning and Karypis, 2011; Sedhain *et al.*, 2016] are the state-of-the-art in terms of relevance. Furthermore, the model explicitly learns a similarity metric. Also, like neighborhood models, recommendations from linear model are easily explainable. However, current linear methods are computationally expensive which limits their applicability to large scale real world problems.

In this paper we address the computational bottleneck of

Method	Relevance	Scalability	Similarity	Interpretability
Neighborhood	×	✓	✓	✓
MF	×	✓*	×	×
Linear	✓	×	✓	✓
Linear-FLow	✓	✓	✓	✓

Table 1: Comparison of recommendation methods for OC-CF. The * for MF is added because weighted MF, WRMF, is relatively expensive.

Table 2: Commonly used symbols.

Symbol	Meaning	Symbol	Meaning
$\mathcal{U}$	Set of users	$\mathbf{R}$	Purchase matrix
$\mathcal{I}$	Set of items	$\hat{\mathbf{R}}$	Recommendation matrix
m	Number of users	$\mathcal{R}(u)$	Items purchased by u
n	Number of items	$\mathbf{S}$	Item similarity matrix

linear models, enabling them to scale up to large OC-CF problems without compromising performance. Our method, Linear-FLow (Fast Low Rank Linear Model), formulates OC-CF as a regularized linear regression problem with a randomized SVD for fast dimensionality reduction. Although regularized regression and randomized SVD are not new ideas, their combined use in the context of OC-CF is, to the best of our knowledge, novel. Through extensive experiments on known benchmarks and real-world datasets, we demonstrate that Linear-FLow achieves state-of-the-art performance as compared to other methods, with a significant reduction in computational cost. Due to its scalability and performance, Linear-FLow has all the desirable properties of a practical recommender system. Table 1 summarizes the comparison of the existing OC-CF methods with respect to Linear-FLow.

2 Background

Let $\mathcal{U}$ denote a set of users, and $\mathcal{I}$ a set of items, with $m = |\mathcal{U}|$ and $n = |\mathcal{I}|$. In OC-CF, we have a purchase[1] matrix $\mathbf{R} \in \{0,1\}^{m\times n}$. We use $\mathbf{R}_{ui}$ to refer to the purchase status for the user u and item i. We use $\mathbf{R}_{:i}$ to denote the indicator vector of each users' purchase of an item, and similarly $\mathbf{R}_{u:}$ to denote the vector of a user's preference for each item i. We denote by $\mathcal{R}(u)$ the set of the items purchased by user u. The goal in OC-CF is to learn a recommender, which for our purposes is simply a matrix $\hat{\mathbf{R}} \in \mathbb{R}^{m\times n}$. We call $\hat{\mathbf{R}}$ the *recommendation matrix*. The notations are summarized in table 2.

In the rest of this section we review some existing approaches to OC-CF.

2.1 Neighbourhood methods

In (item-based) neighbourhood methods, we produce a recommendation matrix of the form

$$\hat{\mathbf{R}} = \mathbf{RS} \tag{1}$$

where $\mathbf{S} \in \mathbb{R}^{n\times n}$ is some *item-similarity matrix*. Typically, one uses a predefined matrix $\mathbf{S}$ that relies on $\mathbf{R}$. A popular example is cosine similarity [Sarwar *et al.*, 2001; Linden *et al.*, 2003b],

$$\mathbf{S}_{i'i} = \frac{\mathbf{R}_{:i}^T \mathbf{R}_{:i'}}{||\mathbf{R}_{:i}||_2 ||\mathbf{R}_{:i'}||_2}.$$

It is typical to sparsify $\mathbf{S}$ so that its columns only keep the top-k similar items. Neighbourhood methods are attractive for several reasons. They are simple to implement, efficient and interpretable. However, they are unable to adapt to the characteristics of the data as they rely on a fixed $\mathbf{S}$ that is not learned from the new data [Koren, 2008]. Furthermore, recommendation performance can be quite sensitive to the choice of $\mathbf{S}$.

2.2 Matrix Factorization

Matrix Factorization methods are the *de facto* approach to collaborative filtering with explicit feedback. The basic idea is to embed users and items into some shared latent space, with the aim of inferring complex preference profiles for both users and items. Weighted matrix factorization (WRMF) for one class collaborative filtering [Hu *et al.*, 2008; Pan and Scholz, 2009] optimizes

$$\min_{\mathbf{A},\mathbf{B}} \sum_{u\in\mathcal{U}, i\in\mathcal{I}} \mathbf{J}_{ui}(\mathbf{R}_{ui} - \mathbf{A}_{:u}^T \mathbf{B}_{:i})^2 + \frac{\lambda}{2}(||\mathbf{A}||_F^2 + ||\mathbf{B}||_F^2) \tag{2}$$

where, $\mathbf{A} \in \mathbb{R}^{k\times m}$, $\mathbf{B} \in \mathbb{R}^{k\times n}$, and $\mathbf{J} \in \mathbb{R}_+^{m\times n}$ is some pre-defined weighting matrix and

$$\mathbf{J}_{ui} = [\![\mathbf{R}_{ui} = 0]\!] + \alpha[\![\mathbf{R}_{ui} > 0]\!] \tag{3}$$

where α assigns an importance weight to the observed interaction. WRMF uses Alternating Least Squares (ALS) method for optimization. The solution for $\mathbf{A}$ and $\mathbf{B}$ in each step of ALS is given by (4)

$$\begin{aligned} \mathbf{A}_{:u} &= (\mathbf{B}\mathbf{J}^u\mathbf{B}^T + \lambda\mathbf{I})^{-1}\mathbf{B}\mathbf{J}^u\mathbf{R}_{u:}^T \\ \mathbf{B}_{:i} &= (\mathbf{A}\mathbf{J}^i\mathbf{A}^T + \lambda\mathbf{I})^{-1}\mathbf{A}\mathbf{J}^i\mathbf{R}_{:i} \end{aligned} \tag{4}$$

where, $\mathbf{J}^u \in \mathbb{R}^{n\times n}$ and $\mathbf{J}^i \in \mathbb{R}^{m\times m}$ are diagonal matrices such that $\mathbf{J}_{ij}^u = \mathbf{J}_{ui}$ and $\mathbf{J}_{uu}^i = \mathbf{J}_{ui}$. In each iteration of ALS, due to the weighting, we need to compute the inverse for each user and item as shown in (4). This makes WRMF computationally expensive compared to unweighted matrix factorization.

2.3 Linear Recommenders

Linear methods [Ning and Karypis, 2011; Sedhain *et al.*, 2016] learn the similarity metric from the data. SLIM [Ning

[1] We use the word "purchase" simply for the purposes of exposition.

and Karypis, 2011] views the recommendation as learning item-item similarity and learns an item-similarity matrix $\mathbf{W} \in \mathbb{R}^{n \times n}$ via

$$\min_{\mathbf{W} \in \mathcal{C}} ||\mathbf{R} - \mathbf{RW}||_F^2 + \frac{\lambda}{2}||\mathbf{W}||_F^2 + \mu||\mathbf{W}||_1, \quad (5)$$

where $\lambda, \mu > 0$ are appropriate constants, and

$$\mathcal{C} = \{\mathbf{W} \in \mathbb{R}^{n \times n} \colon \text{diag}(\mathbf{W}) = 0, \mathbf{W} \geq 0\}. \quad (6)$$

Here, $|| \cdot ||_1$ denotes the elementwise ℓ_1 norm of $\mathbf{W}$ so as to encourage sparsity, and the constraint $\text{diag}(\mathbf{W}) = 0$ prevents a trivial solution of $\mathbf{W} = \mathbf{I}_{n \times n}$. SLIM is equivalent to an item-based neighbourhood approach where the similarity matrix $\mathbf{S} = \mathbf{W}$ is *learned* from the data. In a related linear model for OC-CF [Sedhain *et al.*, 2016], the authors report very good performance, but the proposed method is computationally expensive, which restricts its applicability in real world problems.

Linear methods are attractive for several reasons. They have superior performance, and unlike neighborhood methods, they adapt with the data as the parameters are learned from data itself. Furthermore, the recommendations are easily interpretable. However, linear methods can be computationally expensive as they require solving a large number of regression subproblems with a big design matrix $\mathbf{R}$. This can scale quadratically with the order of $\mathbf{R}$, or worse.

2.4 Randomized SVD

SVD is the archetypal matrix factorization algorithm and has been widely used in machine learning for dimensionality reduction. However, SVD is computationally expensive and not scalable to large scale datasets. It has been recently shown that SVD can be significantly scaled up, at a negligible cost in performance, by randomization [Halko *et al.*, 2011]. We will describe the randomized SVD algorithm in the next section, in the context of the OC-CF problem.

Randomized SVD has been applied to matrix factorization [Tang and Harrington, 2013] (but not in the OC-CF setting that we are considering). The authors compute the rank-k randomized SVD of the matrix $\mathbf{R}$ (Algorithm 1),

$$\mathbf{R} \approx \mathbf{P}_k \mathbf{\Sigma}_k \mathbf{Q}_k^T \quad (7)$$

where, $\mathbf{P}_k \in \mathbb{R}^{m \times k}$, $\mathbf{Q}_k \in \mathbb{R}^{n \times k}$ and $\mathbf{\Sigma}_k \in \mathbb{R}^{k \times k}$. Given the truncated SVD solution, they initialize the item latent factor with the SVD solution and solve

$$\underset{\mathbf{A}}{\text{argmin}} \left\|\mathbf{R} - \mathbf{A}^T\mathbf{B}\right\|_F^2 + \lambda \left\|\mathbf{A}\right\|_F^2 \quad \text{s.t. } \mathbf{B} = \mathbf{\Sigma}_k^{\frac{1}{2}} \mathbf{Q}_k^T \quad (8)$$

Similarly, if the matrix $\mathbf{A}$ is fixed instead of the matrix $\mathbf{B}$, the objective becomes

$$\underset{\mathbf{B}}{\text{argmin}} \left\|\mathbf{R} - \mathbf{A}^T\mathbf{B}\right\|_F^2 + \lambda \left\|\mathbf{B}\right\|_F^2 \quad \text{s.t. } \mathbf{A} = \mathbf{P}_k \mathbf{\Sigma}_k^{\frac{1}{2}} \quad (9)$$

We refer to (9) and (8) as U-MF-RSVD and I-MF-RSVD respectively. As we will see in the results, the performance of these two models can vary significantly.

3 Large Scale Linear Methods for One-Class Collaborative Filtering

Despite their superior performance, the applicability of Linear methods on real world large scale dataset are constrained by their computational cost. Linear methods involve solving a large number of regression subproblems on a huge design matrix $\mathbf{R}$ making it extremely challenging on real world applications where the number of users and items is in millions.

Here we propose a model and an algorithm for scaling up linear methods to large OC-CF problems. In particular, we are seeking an approximation $\mathbf{R} \approx \mathbf{RW}$ that attempts to capture most of the row space of the matrix $\mathbf{R}$ through a matrix $\mathbf{W}$ that is low-rank *and* has small Frobenius norm. The motivation for such a double regularization is to better control the generalization error of the model, an insight that was proven correct by our experiments. Moreover, it turns out that there is a natural and efficient way to compute such an approximate decomposition of the matrix $\mathbf{R}$ by randomization [Halko *et al.*, 2011], which allows scaling to large problems.

Our model amounts to solving the following optimization problem

$$\underset{\text{rank}(\mathbf{W}) \leq k}{\text{argmin}} \left\|\mathbf{R} - \mathbf{RW}\right\|_F^2 + \lambda \left\|\mathbf{W}\right\|_F^2 \quad (10)$$

where, typically, $k \ll n$. For $\lambda = 0$, the optimal solution is given by the Eckart-Young theorem (see, e.g., [Halko *et al.*, 2011])

$$\mathbf{W} = \mathbf{Q}_k \mathbf{Q}_k^T \quad (11)$$

where $\mathbf{Q}_k$ is an orthogonal matrix computed by a truncated SVD as in Equation 7. Similarly, if we drop the low-rank constraint in (10), the optimal matrix $\mathbf{W}$ is given by the solution of a standard regression problem

$$\mathbf{W} = (\mathbf{R}^T\mathbf{R} + \lambda \mathbf{I})^{-1} \mathbf{R}^T\mathbf{R} \quad (12)$$

which involves the inverse of the original matrix $\mathbf{R}$ and therefore does not scale to large problems.

However, under both a low-rank constraint and $\lambda > 0$ in (10), finding the optimal $\mathbf{W}$ involves solving a hard nonconvex problem with no analytical solution in general. Nonetheless, an analytical solution is possible for a certain parametrization of $\mathbf{W}$ as we explain next. We first compute an approximate orthogonal basis $\mathbf{Q}_k$ of the row space of $\mathbf{R}$, i.e.,

$$\mathbf{R} \approx \mathbf{RQ}_k \mathbf{Q}_k^T \quad (13)$$

using randomized SVD. In Algorithm 1, we outline the randomized SVD algorithm (We refer to [Halko *et al.*, 2011] for more details.) Then we re-parametrize the matrix $\mathbf{W}$ as

$$\mathbf{W} = \mathbf{Q}_k \mathbf{Y} \quad (14)$$

for some matrix $\mathbf{Y}$. Note that through this parametrization the rank of $\mathbf{W}$ is automatically controlled, no optimality is lost when $\lambda = 0$, and the optimization problem (10) reads

$$\underset{\mathbf{Y}}{\text{argmin}} \left\|\mathbf{R} - \mathbf{RQ}_k\mathbf{Y}\right\|_F^2 + \lambda \left\|\mathbf{Q}_k\mathbf{Y}\right\|_F^2 \quad (15)$$

Since $\mathbf{Q}_k$ is orthogonal, we have $\|\mathbf{Q}_k\mathbf{Y}\|_F = \|\mathbf{Y}\|_F$, and (15) becomes

$$\underset{\mathbf{Y}}{\operatorname{argmin}} \|\mathbf{R} - \mathbf{R}\mathbf{Q}_k\mathbf{Y}\|_F^2 + \lambda \|\mathbf{Y}\|_F^2 \tag{16}$$

The latter can be solved analytically to give

$$\mathbf{Y} = (\mathbf{Q}_k^T\mathbf{R}^T\mathbf{R}\mathbf{Q}_k + \lambda\mathbf{I})^{-1}\mathbf{Q}_k^T\mathbf{R}^T\mathbf{R}$$

Note that this inversion involves a $k \times k$ matrix, and hence it is tractable.

The choice of $\mathbf{W} = \mathbf{Q}_k\mathbf{Y}$ is motivated by the following observation. When $\lambda = 0$, the solution to our problem is $\mathbf{W} = \mathbf{Q}_k\mathbf{Q}_k^T$. This is also the solution to the new formulation of our problem for $\mathbf{Y} = \mathbf{Q}_k^T$. When λ is close to zero, we believe that sufficiently good solutions lie close to the span of $\mathbf{Q}_k$. Therefore, we choose $\mathbf{W} = \mathbf{Q}_k\mathbf{Y}$. We demonstrate that this choice performs well empirically in the experimental section. Furthermore, Linear-FLow can be seen as an autoencoder collaborative filtering model [Sedhain *et al.*, 2015] with linear activation and $\mathbf{Q}_k$ as input-hidden weights.

We refer to (15) as I-Linear-FLow as it corresponds to item-item model. Similarly, we can define a user-user model, U-Linear-FLow

$$\underset{\mathbf{Y}}{\operatorname{argmin}} \left\|\mathbf{R} - \mathbf{Y}\mathbf{P}_k^T\mathbf{R}\right\|_F^2 + \lambda \|\mathbf{Y}\|_F^2 \tag{17}$$

As we discussed earlier, recommending similar items is very prevalent in real-world recommender systems. In I-Linear-FLow model, the item-item similarity is explicitly given by the matrix $\mathbf{W} = \mathbf{Q}_k\mathbf{Y}$.

Algorithm 1 Given $\mathbf{R} \in \mathbb{R}^{m\times n}$, compute approximate rank-k SVD; $\mathbf{R} \approx \mathbf{P_k}\mathbf{\Sigma}_k\mathbf{Q}_k$

procedure RSVD($\mathbf{R}$, k)
  Draw $n \times k$ Gaussian random matrix $\mathbf{\Omega}$
  Construct $n \times k$ sample matrix $\mathbf{A} = \mathbf{R\Omega}$
  Construct $m \times k$ orthonormal matrix $\mathbf{Z}$, such that $\mathbf{A} = \mathbf{ZX}$
  Constuct $k \times n$ matrix $\mathbf{B} = \mathbf{Z}^T\mathbf{R}$
  Compute the SVD of $\mathbf{B}$, $\mathbf{B} = \hat{\mathbf{P}}_k\mathbf{\Sigma}_k\mathbf{Q}_k$
  Compute $\mathbf{P}_k = \mathbf{Z}\hat{\mathbf{P}}_k$
  return ($\mathbf{P}_k$, $\mathbf{\Sigma}_k$, $\mathbf{Q}_k$)
end procedure

4 Experiments and Evaluation

We now present an extensive set of experiments where we compare the recommendation performance of the proposed method and all the baselines described under Section 2 on several real-world datasets.

4.1 Datasets

For quantitative evaluation, we used two publicly available datasets and two proprietary datasets. In all of our datasets, we remove users with fewer that 3 corresponding items and vice-versa. Table 3 summarizes statistics of the 4 datasets and they are described in details next.

ML10M The MovieLens 10M dataset[2] is a standard benchmark for collaborative filtering tasks. Following the "Who Rated What" KDD Cup 2007 challenge [Bennett *et al.*, 2007], we created a binarized version of the dataset suitable for evaluating implicit feedback methods. From the original rating matrix $\tilde{\mathbf{R}} \in \{0, 1, \ldots, 5\}^{m\times n}$, we created a binarized preference matrix $\mathbf{R}$ with $R_{ui} = [\![\tilde{\mathbf{R}}_{ui} \geq 4]\!]$.

Table 3: Summary of datasets used in evaluation.

Dataset	m	n	$\lvert\mathbf{R}_{ui} > 0\rvert$
ML10M	69,613	9,405	5,004,150
LASTFM	992	88,428	800,274
PROPRIETARY-1	26,928	14,399	120,268
PROPRIETARY-2	264,054	57,214	1,398,332

LASTFM The LastFM dataset[3] [Celma, 2008] contains the play counts of ∼1000 users on ∼170,000 artists. As per ML10M we binarized the raw play counts.

PROPRIETARY-1 & PROPRIETARY-2 are two real but anonymized purchase datasets. PROPRIETARY-1 dataset consists of ∼27,000 users, ∼14,000 items and ∼120,000 item purchases. Similarly, PROPRIETARY-2 dataset consists of ∼264,000 users, ∼57,000 items and ∼1 million item purchases.

4.2 Evaluation Protocol

We split the datasets into random 90%-10% train-test set and hold out 10% of the training set for hyperparamater tuning. We report the mean test split performance, along with standard errors corresponding to 95% confidence intervals. To evaluate the performance of the various recommenders, we report precision@k and recall@k for $k \in \{3, 5, 10, 20\}$ (averaged over test users), and mean average precision (mAP@20).

4.3 Methods compared

We compared the proposed method to a number of baselines:

- User- and item-based nearest neighbour (U-KNN and I-KNN). For each dataset, we use Jaccard and Cosine similarity metric and picked the best performing one.
- PureSVD of [Cremonesi *et al.*, 2010]. Instead of exact SVD, we use randomized SVD for efficiency.
- Weighted matrix Factorization (WRMF) as defined in Eq. (2).
- MF-RSVD of [Tang and Harrington, 2013]. We ran this method with user and item based initialization, U-MF-RSVD and I-MF-RSVD, as discussed in Eq. (9) and (8) respectively.
- SLIM, as per Eq. (5). For computational convenience, we used the SGDReg variant [Levy and Jack, 2013], which is identical to SLIM except that the nonnegativity constraint is removed. We did not evaluate SLIM with nonnegativity directly as [Levy and Jack, 2013] reports superior performance to SLIM, and is considerably faster to train.

We do not compare against LRec [Sedhain *et al.*, 2016] due to its memory complexity on a large dataset. For instance

[2] http://grouplens.org/datasets/movielens/

[3] http://ocelma.net/MusicRecommendationDataset

on the PROPRIETARY-2 dataset, LRec requires ∼260GB of memory.

4.4 Performance Evaluation

Tables 4 – 7 summarize the results of our methods and the various baselines. The results demonstrate that in terms of the quality of recommendation, the linear methods (Linear-FLow and SLIM) always outperform other methods in all four datasets. Linear-FLow and SLIM perform equally well and there is little separation between them in terms of the quality of the recommendation. However, as we will show in section 4.5, Linear-FLow is much more efficient than SLIM. Also, unlike other methods, the performance of Linear-FLow is not sensitive to the choice of user vs. item based formulation. From these tables, we make the following additional observations:

- Among the matrix factorization methods, those that use Randomized SVD [Tang and Harrington, 2013] consistently performed the best. This is possibly due to the fact that SVD provides a good initialization whereas matrix factorization optimizes a highly nonconvex bilinear objective and is sensitive to the initialization and hyperparameters. In Table 4 – 6, we observe that the performance of MF-RSVD varies significantly based on whether we initialize user or item latent factors. Also, the factorization methods do not directly provide item-to-item or user-to-user similarity measures. Hence they are not applicable when a recommendation of similar items or users is needed.
- We observe that the neighborhood models yield inferior results compared to the Linear models. Also, the performance varies with the user and item based models. While they perform competitively in the PROPRIETARY-2 and PROPRIETARY-1 datasets, they perform poorly in LASTFM and ML10M dataset. Hence, we conclude that neighborhood methods are not consistent with their performance.

4.5 Runtime Evaluation

To compare the training times of the various algorithms, we choose PROPRIETARY-2 and ML10M , the two largest datasets for analysis. We benchmarked the training time of the algorithms by training the model on a workstation with 128 GB of main memory and Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz with 32 cores. All of the methods exploit multi-core enabled via numpy linear algebra library, whereas SLIM and WRMF attains parallelism via multiprocessing. For a fair comparison, we ran SLIM and WRMF in parallel to use all available cores. In Table 8 we compare the runtime of the proposed method with the baseline methods.

The results demonstrate that while Linear-FLow offers the same quality of recommendation as SLIM, its training time is an order of magnitude faster than SLIM. SLIM is computationally expensive and is the slowest among the baselines. Neighborhood methods are computationally cheap as they only involve sparse linear algebra, however as demonstrated previously, their recommendation quality is not consistent and lagging behind the other methods. For factorization approaches, those that use randomized SVD similar computational footprints as Linear-FLow while WRMF is much more computationally expensive.

Table 8: Training times of various methods on PROPRIETARY-2 and ML10M Dataset.

	PROPRIETARY-2	ML10M
I-KNN	2.5 sec	10.7 sec
U-KNN	46.9 sec	-
PureSVD	3 min	1 min 27 sec
WRMF	27 min 3 sec	12 min 38 sec
U-MF-SVD	3 min 10 sec	1 min 38 sec
I-MF-SVD	3 min 8 sec	1 min 39 sec
SLIM	32 min 37 sec	7 min 40 sec
U-LRec-Low	3 min 27 sec	1 min 44 sec
I-Lrec-Low	3 min 32 sec	1 min 42 sec

4.6 Qualitative Analysis of Learned Similarities

In this section, we provide a qualitative evaluation of the item-item similarities learned by our I-Linear-FLow model. We use PROPRIETARY-3 , a dataset from a major stock image market site[4]. The data provides whether a given user has clicked on a particular image category, and from these our model can infer the similarity measure between the image categories. Note that the similarity is inferred solely based on the click patterns, without doing any analysis of the textual content of the category names. We choose this dataset for the qualitative evaluation mainly because the category names and are much easier to interpret compared to the other datasets.

In Table 9, we show some examples[5] of top-5 similar items learned by I-Linear-FLow model. We observe that the model discovers meaningful and explainable similarities, hence making it applicable in similar item recommendations.

5 Conclusion

We proposed Linear-FLow, a fast low-dimensional regularized linear model for One-Class Collaborative Filtering, which addresses the computational bottleneck of other linear models for this problem, enabling scaling up to large problem instances while retaining the same performance. In our experiments, we illustrated that the proposed method is computationally superior to the state of the art (an order of magnitude faster than SLIM) and yields competitive performance. In future work, we will explore the possibility of incorporating additional user and item side-information to the model, and further improving the computational and memory footprints by exploring more efficient dimensionality reduction techniques. The main computational bottleneck for Linear-FLow is currently in the computation of randomized SVD, which heavily depends on the linear algebra library, and hence we expect significant speedups by using a GPU back-end linear algebra library [Voronin and Martinsson, 2015].

[4]The dataset sharing agreement with the provider restricts us from reporting the statistics and quantitative results. Hence, we do not report summary statistics and quantitative results on this dataset

[5]Visit http://ssedhain.com/demos/Item-Item.html for interactive visualization

Table 4: Results on the PROPRIETARY-1 dataset. Reported numbers are the mean and standard errors across test folds.

	prec@3	prec@5	prec@10	prec@20	recall@3	recall@5	recall@10	recall@20	mAP@20
I-KNN	0.0393 ± 0.0007	0.0291 ± 0.0005	0.0186 ± 0.0002	0.0115 ± 0.0002	0.0797 ± 0.0011	0.0973 ± 0.0015	0.1232 ± 0.0021	0.1521 ± 0.0028	0.0725 ± 0.0008
U-KNN	0.0514 ± 0.0008	0.0386 ± 0.0005	0.0249 ± 0.0003	0.0153 ± 0.0001	0.1068 ± 0.0035	0.1321 ± 0.0037	0.1679 ± 0.0029	0.2052 ± 0.0036	0.0969 ± 0.0033
PureSVD	0.0376 ± 0.0013	0.0267 ± 0.0009	0.0160 ± 0.0005	0.0094 ± 0.0003	0.0776 ± 0.0018	0.0906 ± 0.0018	0.1073 ± 0.0026	0.1247 ± 0.0030	0.0692 ± 0.0022
WRMF	0.0397 ± 0.0008	0.0293 ± 0.0013	0.0183 ± 0.0008	0.0113 ± 0.0004	0.0787 ± 0.0021	0.0955 ± 0.0047	0.1186 ± 0.0045	0.1467 ± 0.0037	0.0707 ± 0.0025
U-MF-RSVD	0.0503 ± 0.0007	0.0381 ± 0.0004	0.0247 ± 0.0003	0.0155 ± 0.0001	0.1003 ± 0.0024	0.1301 ± 0.0017	0.1693 ± 0.0032	0.2069 ± 0.0039	0.0961 ± 0.0026
I-MF-RSVD	0.0480 ± 0.0010	0.0360 ± 0.0008	0.0234 ± 0.0004	0.0144 ± 0.0002	0.0976 ± 0.0030	0.1211 ± 0.0033	0.1550 ± 0.0038	0.1886 ± 0.0037	0.0889 ± 0.0030
SLIM	$\mathbf{0.0519 \pm 0.0010}$	0.0395 ± 0.0004	0.0258 ± 0.0004	0.0160 ± 0.0002	$\mathbf{0.1069 \pm 0.0034}$	0.1341 ± 0.0026	0.1729 ± 0.0039	0.2117 ± 0.0033	$\mathbf{0.0976 \pm 0.0029}$
U-Linear-FLow	0.0518 ± 0.0013	0.0391 ± 0.0005	0.0259 ± 0.0004	0.0160 ± 0.0002	0.1060 ± 0.0035	0.1350 ± 0.0032	0.1711 ± 0.0026	0.2118 ± 0.0029	0.0970 ± 0.0030
I-Linear-FLow	0.0520 ± 0.0012	$\mathbf{0.0398 \pm 0.0004}$	$\mathbf{0.0262 \pm 0.0003}$	$\mathbf{0.0162 \pm 0.0002}$	0.1062 ± 0.0032	$\mathbf{0.1362 \pm 0.0026}$	$\mathbf{0.1758 \pm 0.0026}$	$\mathbf{0.2145 \pm 0.0033}$	0.0971 ± 0.0028

Table 5: Results on the PROPRIETARY-2 dataset. Reported numbers are the mean and standard errors across test folds.

	prec@3	prec@5	prec@10	prec@20	recall@3	recall@5	recall@10	recall@20	mAP@20
I-KNN	0.0873 ± 0.0003	0.0641 ± 0.0002	$0.0400 \pm 2.4970 \times 10^{-5}$	$0.0236 \pm 2.2155 \times 10^{-5}$	0.1743 ± 0.0005	0.2097 ± 0.0007	0.2566 ± 0.0007	0.2982 ± 0.0008	0.1591 ± 0.0007
U-KNN	0.0836 ± 0.0005	0.0605 ± 0.0003	0.0365 ± 0.0002	$0.0207 \pm 7.6396 \times 10^{-5}$	0.1711 ± 0.0011	0.2029 ± 0.0013	0.2407 ± 0.0013	0.2695 ± 0.0014	0.1532 ± 0.0007
WRMF	0.0579 ± 0.0005	0.0437 ± 0.0004	0.0283 ± 0.0001	$0.0175 \pm 8.6682 \times 10^{-5}$	0.1145 ± 0.0013	0.1417 ± 0.0015	0.1801 ± 0.0012	0.2184 ± 0.0012	0.1053 ± 0.0011
PureSVD	0.0335 ± 0.0004	0.0244 ± 0.0002	0.0153 ± 0.0002	$0.0094 \pm 9.7863 \times 10^{-5}$	0.0686 ± 0.0006	0.0823 ± 0.0005	0.1018 ± 0.0009	0.1233 ± 0.0010	0.0624 ± 0.0004
U-MF-RSVD	0.0699 ± 0.0003	0.0502 ± 0.0002	$0.0305 \pm 6.9209 \times 10^{-5}$	$0.0178 \pm 2.4425 \times 10^{-5}$	0.1408 ± 0.0007	0.1660 ± 0.0008	0.1985 ± 0.0005	0.2282 ± 0.0004	0.1277 ± 0.0006
I-MF-RSVD	0.0880 ± 0.0003	0.0652 ± 0.0001	$0.0408 \pm 6.5965 \times 10^{-5}$	$0.0242 \pm 3.4697 \times 10^{-5}$	0.1720 ± 0.0008	0.2199 ± 0.0008	0.2685 ± 0.0010	0.3203 ± 0.0011	0.1582 ± 0.0005
SLIM	$\mathbf{0.0893 \pm 0.0004}$	$\mathbf{0.0671 \pm 0.0003}$	$\mathbf{0.0433 \pm 9.5600 \times 10^{-5}}$	$\mathbf{0.0264 \pm 4.3581 \times 10^{-5}}$	$\mathbf{0.1796 \pm 0.0010}$	$\mathbf{0.2213 \pm 0.0013}$	$\mathbf{0.2790 \pm 0.0012}$	$\mathbf{0.3336 \pm 0.0009}$	$\mathbf{0.1654 \pm 0.0006}$
U-Linear-FLow	0.0887 ± 0.0004	0.0666 ± 0.0002	$0.0430 \pm 8.6598 \times 10^{-5}$	$0.0258 \pm 4.0894 \times 10^{-5}$	0.1763 ± 0.0012	0.2214 ± 0.0010	0.2739 ± 0.0013	0.3289 ± 0.0010	0.1611 ± 0.0008
I-Linear-FLow	0.0885 ± 0.0003	0.0656 ± 0.0002	$0.0429 \pm 8.6598 \times 10^{-5}$	$0.0256 \pm 4.0894 \times 10^{-5}$	0.1783 ± 0.0012	0.2202 ± 0.0012	0.2719 ± 0.0010	0.3259 ± 0.0010	0.1598 ± 0.0006

Table 6: Results on the LASTFM dataset. Reported numbers are the mean and standard errors across test folds.

	prec@3	prec@5	prec@10	prec@20	recall@3	recall@5	recall@10	recall@20	mAP@20
I-KNN	0.5904 ± 0.0068	0.5600 ± 0.0067	0.5068 ± 0.0053	0.4390 ± 0.0054	0.0187 ± 0.0004	0.0284 ± 0.0002	0.0497 ± 0.0007	0.0824 ± 0.0016	0.3280 ± 0.0055
U-KNN	0.5434 ± 0.0083	0.5127 ± 0.0038	0.4619 ± 0.0044	0.4009 ± 0.0028	0.0169 ± 0.0004	0.0260 ± 0.0011	0.0448 ± 0.0011	0.0747 ± 0.0014	0.2894 ± 0.0029
WRMF	0.6277 ± 0.0071	0.5899 ± 0.0049	0.5308 ± 0.0049	0.4577 ± 0.0032	0.0198 ± 0.0006	0.0300 ± 0.0009	0.0516 ± 0.0015	0.0842 ± 0.0013	0.3562 ± 0.0036
PureSVD	0.3993 ± 0.0121	0.3690 ± 0.0082	0.3321 ± 0.0069	0.2880 ± 0.0069	0.0128 ± 0.0003	0.0194 ± 0.0004	0.0338 ± 0.0006	0.0567 ± 0.0018	0.1723 ± 0.0065
U-MF-RSVD	0.5176 ± 0.0042	0.4865 ± 0.0068	0.4423 ± 0.0028	0.3859 ± 0.0038	0.0141 ± 0.0005	0.0217 ± 0.0008	0.0387 ± 0.0005	0.0653 ± 0.0013	0.2810 ± 0.0023
I-MF-RSVD	0.5780 ± 0.0063	0.5447 ± 0.0050	0.4901 ± 0.0011	0.4236 ± 0.0029	0.0190 ± 0.0008	0.0292 ± 0.0006	0.0496 ± 0.0011	0.0811 ± 0.0009	0.3146 ± 0.0020
SLIM	$\mathbf{0.6319 \pm 0.0069}$	$\mathbf{0.6002 \pm 0.0055}$	$\mathbf{0.5384 \pm 0.0048}$	0.4639 ± 0.0056	$\mathbf{0.0223 \pm 0.0006}$	$\mathbf{0.0346 \pm 0.0004}$	$\mathbf{0.0592 \pm 0.0007}$	$\mathbf{0.0972 \pm 0.0017}$	0.3587 ± 0.0065
U-Linear-FLow	0.6229 ± 0.0081	0.5912 ± 0.0067	0.5337 ± 0.0027	0.4627 ± 0.0020	0.0205 ± 0.0009	0.0315 ± 0.0014	0.0540 ± 0.0004	0.0881 ± 0.0010	0.3563 ± 0.0025
I-Linear-FLow	0.6229 ± 0.0103	0.5913 ± 0.0046	0.5337 ± 0.0021	$\mathbf{0.4653 \pm 0.0023}$	0.0203 ± 0.0011	0.0310 ± 0.0010	0.0529 ± 0.0006	0.0874 ± 0.0015	$\mathbf{0.3615 \pm 0.0014}$

Table 7: Results on the ML10M dataset. Reported numbers are the mean and standard errors across test folds.

	prec@3	prec@5	prec@10	prec@20	recall@3	recall@5	recall@10	recall@20	mAP@20
I-KNN	0.1753 ± 0.0011	0.1506 ± 0.0009	0.1179 ± 0.0006	0.0883 ± 0.0003	0.0994 ± 0.0004	0.1393 ± 0.0009	0.2121 ± 0.0012	0.3070 ± 0.0012	0.1216 ± 0.0003
U-KNN					Out of Memory				
WRMF	0.1832 ± 0.0007	0.1561 ± 0.0006	0.1205 ± 0.0002	0.0889 ± 0.0002	0.1024 ± 0.0003	0.1428 ± 0.0008	0.2139 ± 0.0009	0.3061 ± 0.0010	0.1255 ± 0.0003
PureSVD	0.1216 ± 0.0013	0.1054 ± 0.0006	0.0836 ± 0.0005	0.0634 ± 0.0002	0.0730 ± 0.0014	0.1030 ± 0.0011	0.1572 ± 0.0009	0.2264 ± 0.0011	0.0836 ± 0.0010
U-MF-RSVD	0.2229 ± 0.0007	0.1895 ± 0.0007	0.1456 ± 0.0008	0.1069 ± 0.0004	0.1273 ± 0.0003	0.1755 ± 0.0005	0.2592 ± 0.0014	0.3656 ± 0.0018	0.1586 ± 0.0005
I-MF-RSVD	0.2232 ± 0.0009	0.1902 ± 0.0010	0.1461 ± 0.0005	0.1027 ± 0.0004	0.1246 ± 0.0007	0.1745 ± 0.0010	0.2602 ± 0.0008	0.3675 ± 0.0017	0.1590 ± 0.0007
SLIM	0.2208 ± 0.0011	0.1888 ± 0.0011	0.1464 ± 0.0004	0.1080 ± 0.0004	0.1258 ± 0.0003	0.1748 ± 0.0012	0.2611 ± 0.0009	0.3690 ± 0.0016	0.1579 ± 0.0007
U-Linear-FLow	$\mathbf{0.2270 \pm 0.0012}$	$\mathbf{0.1927 \pm 0.0009}$	$\mathbf{0.1477 \pm 0.0006}$	$\mathbf{0.1083 \pm 0.0004}$	$\mathbf{0.1290 \pm 0.0004}$	$\mathbf{0.1777 \pm 0.0008}$	$\mathbf{0.2624 \pm 0.0013}$	$\mathbf{0.3701 \pm 0.0016}$	$\mathbf{0.1601 \pm 0.0006}$
I-Linear-FLow	0.2242 ± 0.0009	0.1909 ± 0.0010	0.1468 ± 0.0005	0.1077 ± 0.0004	0.1276 ± 0.0007	0.1765 ± 0.0010	0.2609 ± 0.0008	0.3676 ± 0.0017	0.1592 ± 0.0007

Table 9: Top-5 similar items learned by I-Linear-FLow model.

Item	Chemistry	Chilling out	Workers	Unemployment	Divorce and Conflict	Museums
Similar items	Test and Analysis	Beach Holidays	Construction	Job Search	Depression	Painting
	Drug and Pills	Tourism	Teamwork	Tax and Accounting	Getting upset	Statues
	Health Care	Relaxing	Manufacturing	Breaking the law	Crying	Artistic monuments
	Scientists	Hiking	Service industry	Money	Loneliness	Paris
	Medical Equipments	Consumer service	Beaches	Workers	Rage	Italy
Item	Dance	Vinegar	Pearls	Graduation	Aging	Homelessness
Similar items	Exercise	Olive Oil	Wealth	High School	Patients	Depression
	Running And Jumping	Spice	Wedding	School	Grand Parenting	Loneliness
	Disco And Clubs	Salads	Accessories	Exams	Disability	Crying
	Circus And Performing	Garlic	Gold	Job Search	Health Care	Getting Upset
	Gymnastics	other	Make Up	E-Learning	Doctors	Risk And Danger

References

[Adomavicius and Tuzhilin, 2005] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, 2005.

[Bennett *et al.*, 2007] James Bennett, Charles Elkan, Bing Liu, Padhraic Smyth, and Domonkos Tikk. KDD cup and workshop 2007. *SIGKDD Explorations Newsletter*, 9(2):51–52, December 2007.

[Celma, 2008] Òscar. Celma. *Music Recommendation and Discovery in the Long Tail.* PhD thesis, Universitat Pompeu Fabra, Barcelona, 2008.

[Cremonesi *et al.*, 2010] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *ACM Conference on Recommender Systems (RecSys)*, RecSys '10, pages 39–46, New York, NY, USA, 2010. ACM.

[Goldberg *et al.*, 1992] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative fil-

tering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, December 1992.

[Halko *et al.*, 2011] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[Hu *et al.*, 2008] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM '08*, pages 263–272, Washington, DC, USA, 2008. IEEE Computer Society.

[Koren *et al.*, 2009] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[Koren, 2008] Yehuda Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *KDD '08*, pages 426–434, New York, NY, USA, 2008. ACM.

[Leavitt, 2006] Neal Leavitt. Recommendation technology: Will it boost e-commerce? *Computer*, 39(5):13–16, May 2006.

[Levy and Jack, 2013] Mark Levy and Kris Jack. Efficient top-n recommendation by linear regression. In *RecSys Large Scale Recommender Systems Workshop*, 2013.

[Linden *et al.*, 2003a] Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.

[Linden *et al.*, 2003b] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, January 2003.

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

[Pan and Scholz, 2009] Rong Pan and Martin Scholz. Mind the gaps: Weighting the unknown in large-scale one-class collaborative filtering. In *KDD '09*, pages 667–676, New York, NY, USA, 2009. ACM.

[Pan *et al.*, 2008] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N. Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. One-class collaborative filtering. In *IEEE International Conference on Data Mining (ICDM)*, pages 502–511, Washington, DC, USA, 2008. IEEE Computer Society.

[Ricci *et al.*, 2011] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Introduction to recommender systems handbook*. Springer, 2011.

[Sarwar *et al.*, 2001] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW '01*, pages 285–295, New York, NY, USA, 2001. ACM.

[Sedhain *et al.*, 2015] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, 2015.

[Sedhain *et al.*, 2016] Suvash Sedhain, Aditya Menon, Scott Sanner, and Darius Braziunas. On the effectiveness of linear models for one-class collaborative filtering. *Proceedings of the 30th Conference on Artificial Intelligence (AAAI-16)*, 2016.

[Tang and Harrington, 2013] Lei Tang and Patrick Harrington. Scaling matrix factorization for recommendation with randomness. In *Proceedings of the 22nd international conference on World Wide Web companion*, pages 39–40. International World Wide Web Conferences Steering Committee, 2013.

[Vig *et al.*, 2009] Jesse Vig, Shilad Sen, and John Riedl. Tagsplanations: Explaining recommendations using tags. In *Proceedings of the 14th International Conference on Intelligent User Interfaces*, IUI '09, pages 47–56, New York, NY, USA, 2009. ACM.