Noise Contrastive Estimation for One-Class Collaborative Filtering

Ga Wu[*][†]
University of Toronto
wuga@mie.utoronto.ca

Maksims Volkovs
Layer6 AI
maks@layer6.ai

Chee Loong Soon
University of Toronto
cheeloong.soon@mail.utoronto.ca

Scott Sanner[†]
University of Toronto
ssanner@mie.utoronto.ca

Himanshu Rai
Layer6 AI
himanshu@layer6.ai

ABSTRACT

Previous highly scalable One-Class Collaborative Filtering (OC-CF) methods such as Projected Linear Recommendation (PLRec) have advocated using fast randomized SVD to embed items into a latent space, followed by linear regression methods to learn personalized recommendation models per user. However, naive SVD embedding methods often exhibit a strong popularity bias that prevents them from accurately embedding less popular items, which is exacerbated by the extreme sparsity of implicit feedback matrices in the OC-CF setting. To address this deficiency, we leverage insights from Noise Contrastive Estimation (NCE) to derive a closed-form, efficiently computable "depopularized" embedding. We show that NCE item embeddings combined with a personalized user model from PLRec produces superior recommendations that adequately account for popularity bias. Further analysis of the popularity distribution of recommended items demonstrates that NCE-PLRec uniformly distributes recommendations over the popularity spectrum while other methods exhibit distinct biases towards specific popularity subranges. Empirically, NCE-PLRec produces highly competitive performance with run-times an order of magnitude faster than existing state-of-the-art approaches for OC-CF.

KEYWORDS

One-Class Collaborative Filtering; Noise Contrastive Estimation

ACM Reference Format:
Ga Wu, Maksims Volkovs, Chee Loong Soon, Scott Sanner, and Himanshu Rai. 2019. Noise Contrastive Estimation for One-Class Collaborative Filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '19), July 21–25, 2019, Paris, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3331184.3331201

[*]This work has been primarily completed while the author was working at Layer6 AI.
[†]Affiliate to Vector Institute of Artificial Intelligence, Toronto

1 INTRODUCTION

In an era of virtually unlimited choices, recommender systems are necessary to assist users in finding items they may like. Collaborative filtering (CF) is the de-facto standard approach for making such personalized recommendations based on automated collection of item interaction data from a population of users [23]. However, in many cases, these interactions lack explicit negative signals, e.g., clicks on a website or purchases of a book. In these cases, a lack of interaction should not be construed as implicitly negative; indeed, it could simply be that a user was unaware of the item's existence. This recommendation setting where only positive (and typically very sparse) interactions are observed is known as the One Class Collaborative Filtering (OC-CF) problem [20].

One approach to tackle OC-CF is to factorize a large sparse *implicit* matrix into smaller latent matrices of user and item representations [10, 20]. However, matrix factorization requires optimizing a non-convex objective, resulting in local optima and the need for substantial hyperparameter tuning for good practical performance [13]. An alternative scalable solution is to first reduce the dimensionality of the matrix, then learn the importance of different latent projected features using linear regression. Methods such as [24], which we refer to as Projected Linear Recommendation (PLRec) precompute the item embeddings through fast randomized Singular Value Decomposition (SVD) [6] and train separate linear regression models for each user on top of these embeddings. This separation enables parallelization across users and reduces the optimization to a convex objective that is globally optimized in closed-form [14]. However, naive SVD embedding methods exhibit a popularity bias that skews their ability to accurately embed less popular items [21].

In this paper, we propose a novel projected linear recommendation algorithm called Noise Contrastive Estimation PLRec (NCE-PLRec). Instead of explicitly treating unobserved interactions as negative feedback, we leverage insights from the NCE framework [5] that attempt to discriminate between observed interactions and a noise model; NCE has been previously used extensively in high-quality word embeddings for natural language [15, 18]. Specifically, we first transform the implicit matrix into a de-popularized matrix that optimally re-weights the interactions in closed-form according to the NCE objective. Then we extract item embeddings by projecting items onto the principal components of this de-popularized matrix obtained via SVD. We can then leverage the standard PLRec framework with these NCE item embeddings in the novel and highly scalable NCE-PLRec method for OC-CF problems.

An analysis of recommendation popularity distribution demonstrates that NCE-PLRec uniformly distributes its recommendations over the popularity spectrum while other methods exhibit distinct biases towards specific popularity subranges. Overall, our results show that NCE-PLRec is an order of magnitude faster and highly competitive with existing state-of-the-art OC-CF methods.

2 NOTATION AND BACKGROUND

Before proceeding, we define notation used throughout this paper:

- R: $m \times n$ is an *implicit* feedback matrix. Entries r_{ij} are either 1 (observed interaction) or 0 (no interaction). $\mathbf{r}_i$ represents all implicit feedback from user $i \in \{1 \cdots m\}$, and $\mathbf{r}_{:,j}$ represents all user feedback for item $j \in \{1 \cdots n\}$. We use $|\mathbf{r}_{:,j}|$ to represent the count of observed interactions for item j.
- U, V: Latent representations (or embeddings) of users and items respectively. U is $m \times k$, V is $n \times k$. We use $\mathbf{u}_i$ to represent the ith user representation (row of U), and $\mathbf{v}_j$ to represent the jth item representation (row of V).
- $D = UV^T$: An inner product of user and item embeddings that has the same shape as the implicit feedback matrix R.
- $Q = RV$: $m \times k$ is a projected feedback matrix, obtained by projecting R onto item embeddings V.

2.1 Matrix Factorization

Matrix factorization (MF) aims to uncover latent features of users and items that explain observations in the feedback matrix. Implicit feedback is reconstructed through a function g that operates on the corresponding user and item latent representations $\mathbf{u}_i$ and $\mathbf{v}_j$:

$$\underset{U,V}{\operatorname{argmin}} \sum_{i,j} (r_{ij} - g(\mathbf{u}_i, \mathbf{v}_j))^2 + \lambda(\|\mathbf{u}_i\|_2^2 + \|\mathbf{v}_j\|_2^2),$$

where g is typically a dot product $\mathbf{u}_i^T \mathbf{v}_j$, or a neural network [8]. This objective doesn't perform well in the OC-CF setting as unobserved interactions skew the representations. Weighted Regularized MF (WRMF) [10] addresses this by adding a weight on the observed interactions. In WRMF an additional hyper parameter α is introduced to derive a weight for each user-item pair $c_{ij} = 1 + \alpha r_{ij}$ which is incorporated into the squared error objective $\sum_{i,j} c_{ij}(r_{ij} - g(\mathbf{u}_i, \mathbf{v}_j))^2$. Although WRMF performs well in the OC-CF setting, it requires expensive iterative optimization using Alternating Least Squares (ALS) to minimize an upper bound on the reconstruction error. ALS can be replaced with Stochastic Gradient Descent (SGD), however, this further requires sampling "negative" items (i.e., sparsity cannot be exploited like in ALS) and typically slows down convergence.

2.2 Linear Recommenders

An alternative approach to low rank factorization is to learn the similarity matrix directly via linear regression [25]. Sparse LInear Method (SLIM) [19] achieves this by minimizing the constrained linear regression objective:

$$\underset{W}{\operatorname{argmin}} \sum_{i,j} (r_{i,j} - \mathbf{r}_i \mathbf{w}_j)^2 + \beta \left\|\mathbf{w}_j\right\|_2^2 + \lambda \left\|\mathbf{w}_j\right\|_1,$$
$$W \geq 0, \qquad diag(W) = 0$$

where W is the similarity matrix to be learned. The constraints and regularizers prevent the trivial solution where W is the identity matrix I. SLIM and its variants such as LRec [25] also suffer from scalability issues as they require storing a large similarity matrix that grows quadratically with the number of users or items, which is not fully mitigated even with SLIM's L_1-regularized sparsification approach. This is impractical for real-world problems with millions of users and items. Moreover, learning a large number of parameters relative to sparse observations is ill-formed since it leads to more unknowns than available equations.

Linear Flow [24], which we refer to as Projected Linear Recommendation (PLRec) for technical clarity to relate it to LRec, addresses these issues by first reducing the dimensionality of the implicit matrix followed by Linear Regression:

$$\underset{W}{\operatorname{argmin}} \sum_{i,j} (r_{i,j} - \mathbf{r}_i V \mathbf{w}_j^T)^2 + \lambda \left\|\mathbf{w}_j\right\|_2^2, \tag{1}$$

where V is an item embedding matrix obtained from the truncated SVD decomposition of the implicit matrix R ($R = U\Sigma V^T$), often computed scalably on large matrices via fast randomized SVD approaches [6]. Since the item embedding dimension $k \ll min\{m, n\}$, training PLRec requires substantially fewer parameters to learn compared to SLIM or LRec. Moreover, this formulation is fully parallelizeable across users and reduces the optimization to a convex objective that is globally optimized in closed-form [14]. This makes PLRec one of the most efficient and scalable recommender models.

However, PLRec also has several deficiencies that affect its performance. First, directly decomposing the raw implicit feedback matrix with an SVD makes the model highly biased to the large number of unobserved ratings. This can result in poor item embeddings, especially for less popular items [21]. Second, since the SVD is the optimal decomposition under a squared error objective, the optimal W learned from PLRec would be close to V (and hence little gain over the original SVD) if regularization were ignored. To understand this issue, we substitute $\mathbf{r}_i \approx \mathbf{u}_i \Sigma V^T$ (approximate since this is a truncated SVD reconstruction) in the PLRec objective function and obtain the following:

$$\begin{aligned}
&\underset{W}{\operatorname{argmin}} \sum_{i,j} (r_{i,j} - \mathbf{r}_i V \mathbf{w}_j)^2 + \lambda \left\|\mathbf{w}_j\right\|_2^2 \\
&\approx \underset{W}{\operatorname{argmin}} \sum_{i,j} (r_{i,j} - \mathbf{u}_i \Sigma V^T V \mathbf{w}_j)^2 + \lambda \left\|\mathbf{w}_j\right\|_2^2 \\
&= \underset{W}{\operatorname{argmin}} \sum_{i,j} (r_{i,j} - \mathbf{u}_i \Sigma \mathbf{w}_j)^2 + \lambda \left\|\mathbf{w}_j\right\|_2^2 \quad \text{since } V^T V = I.
\end{aligned}$$

With $\lambda = 0$, the optimal $\mathbf{w}_j$ is the $\mathbf{v}_j$ given by SVD since the first term is the SVD objective. Empirically, we find that PLRec often performs similar to the PureSVD [4] algorithm for this reason and thus suffers from the popularity bias of SVD embeddings.

3 NCE-PLREC

PLRec stands out as one of the most scalable OC-CF methods (as our runtime results later verify). However, the SVD item embeddings used by PLRec suffer from popularity bias that can significantly affect performance [21]. In this work we leverage ideas from Noise-Contrastive Estimation (NCE) to derive improved item embeddings for PLRec. We begin by revisiting recommendation from a probabilistic perspective. Specifically, we fit a model parameterized by the user and item embeddings to maximize the probability of observed feedback. Instead of explicitly treating unobserved interactions as negative feedback, our NCE approach learns to discriminate between observed interactions and unobserved noise.

3.1 Noise Contrastive Estimation in Recommendation

Noise-Contrastive Estimation (NCE) [5] learns to discriminate between the observed data and some artificially generated noise. Given a set of observations $X = \{\mathbf{x}_1 \cdots \mathbf{x}_n\}$ and artificially generated noise $Y = \{\mathbf{y}_1 \cdots \mathbf{y}_n\}$, NCE aims to separate observations from noise:

$$\sum_j \log(g(\mathbf{x}_j;\theta)) + \log(1 - g(\mathbf{y}_j;\theta)), \tag{2}$$

where $g(\cdot)$ is a (possibly unnormalized) probability density function, and θ are model parameters to estimate. NCE is normally used to estimate the observation probability where the partition function is hard to estimate due to either computational cost or lack of observed negative samples. In implicit feedback recommendation tasks, we only explicitly observe positive observations for each user, which makes NCE an ideal tool to estimate user preferences without explicitly assuming unobserved interactions are negative samples as done in most OC-CF methods. The simple idea driving NCE is that it adversarially trains to maximize prediction probability of the observed user preferences while minimizing the prediction probability of negative samples drawn from a (usually) popularity-biased noise distribution.

To apply NCE to OC-CF we first note that in the probabilistic formulation the objective of recommendation is to train a model that maximizes the probability of observed interactions $p(r_{i,j} = 1|i,j)$. Motivated by the log-odds ratio derived from a Bernoulli Distribution [2], we define this positive interaction probability as the following logistic sigmoid function:

$$p(r_{i,j} = 1|i,j) = \sigma(\mathbf{u}_i^T\mathbf{v}_j) = \frac{1}{1+e^{-\mathbf{u}_i^T\mathbf{v}_j}}. \tag{3}$$

Since the negative feedback is not observed in OC-CF, we could artificially generate negative samples by sampling from some item distribution q. A natural choice for q is the popularity distribution $q(j) = |\mathbf{r}_{:,j}|/\sum_k^n |\mathbf{r}_{:,k}|$. Popular items are more likely to be encountered by the user so the absence of an interaction with these items is more likely to be indicative of negative feedback. This leads to the following NCE objective for each user i, where the goal is to maximize the probability of observed interactions and minimize it for sampled "noise" items:

$$\underset{\mathbf{u}_i, V}{\operatorname{argmax}} \sum_j r_{i,j}\left[\log\sigma(\mathbf{u}_i^T\mathbf{v}_j) + \log\sigma(-\mathbf{u}_i^T\mathbf{v}_{j'})\right], \tag{4}$$

where we sample a noise item j' according to $q(j')$ for each observed user interaction j. Here we remark that (a) the $r_{i,j}$ indicators restrict us to only contrast observed positive interactions with the noise distribution and that (b) observing the logistic sigmoid identity $1-\sigma(\mathbf{u}_i^T\mathbf{v}_{j'}) = \sigma(-\mathbf{u}_i^T\mathbf{v}_{j'})$ allows us to the see the equivalence with the general form of NCE in (2).

If we take the expectation of the above objective w.r.t. distribution $q(j')$, we see that we can rewrite the objective in the following expected form by exploiting linearity of expectation:

$$\underset{\mathbf{u}_i, V}{\operatorname{argmax}} \sum_j r_{i,j}\left[\log\sigma(\mathbf{u}_i^T\mathbf{v}_j) + E_{q(j')}[\log\sigma(-\mathbf{u}_i^T\mathbf{v}_{j'})]\right]. \tag{5}$$

In the multi-user collaborative filtering setting, the full objective function ℓ corresponds to a summation over each independent user, where the item embeddings are shared by all users:

$$\underset{U, V}{\operatorname{argmax}} \underbrace{\sum_i\sum_j r_{i,j}\left[\log\sigma(\mathbf{u}_i^T\mathbf{v}_j) + E_{q(j')}[\log\sigma(-\mathbf{u}_i^T\mathbf{v}_{j'})]\right]}_{\ell}. \tag{6}$$

Globally optimizing (6) with respect to both user and item representations in closed-form is intractable due to the shared item embeddings and the nonlinear relationship between user and item embeddings. Therefore, we optimize (6) with respect to the dot product $d_{i,j} = \mathbf{u}_i^T\mathbf{v}_j$ directly to simplify the objective into a convex optimization problem. Such a procedure is often referred to as *lifting* in the optimization literature; once we solve for $d_{i,j}$, we will then be able to recover a suitable $\mathbf{u}_i$ and $\mathbf{v}_j$. Solving for the optimal $d_{i,j}$ for positive observations ($r_{i,j} = 1$), we obtain the following:

$$\frac{\partial\ell}{\partial d_{i,j}} = \sigma(-d_{i,j}) - \frac{|r_{:,j}|}{\sum_{j'}|r_{:,j'}|}\sigma(d_{i,j}) \tag{7}$$

$$d^*_{i,j} = \log\frac{\sum_{j'}|r_{:,j'}|}{|r_{:,j}|} \qquad \forall r_{i,j} = 1. \tag{8}$$

For the unobserved interactions, the optimal solution is simply zero:

$$d^*_{i,j} = 0 \qquad \forall r_{i,j} = 0. \tag{9}$$

The resulting sparse matrix D maintains the same number of non-zero entries and shape from the original implicit matrix. The difference is that the entries are now replaced with the optimal inner product of user and item representations, $D^* = U^*V^{*T}$.

Finally, we recover all $\mathbf{u}_i$ and $\mathbf{v}_j$ embeddings by projecting the sparse D^* using truncated SVD [6] as it exploits sparsity in the matrix. Then

$$U^* \approx U_D\Sigma_D^{\frac{1}{2}} \qquad V^* \approx V_D\Sigma_D^{\frac{1}{2}}, \tag{10}$$

where U_D, V_D and Σ_D come from the truncated SVD $D^* \approx U_D\Sigma_D V_D^T$.

This concludes our NCE-based derivation of user and item embeddings, where we see that in contrast to the standard SVD method of embeddings, NCE provides a slight variation we call NCE-SVD. That is, instead of taking the SVD of the implicit feedback matrix R directly, the NCE embedding instead takes the SVD of the D matrix as defined in (8) and (9), which is a logarithmic and popularity renormalized version of R. In short, a visual inspection of D indicates that prior to performing the SVD, NCE suggests that items j with more positive observations – i.e., a larger denominator in (8) – should be downweighted prior to deriving the SVD-based user and item embeddings. In this sense, we can view the NCE-SVD embedding as a "depopularized" approach that places more emphasis on accurate SVD reconstruction of less popular items.

3.2 Relation to the Neural Word Embeddings

Noise Contrastive Estimation was first brought to the attention of the Machine Learning community from the literature on word embeddings [7, 15, 17, 18]. However, to the best of our knowledge, it has not been widely applied in the recommendation setting even though the implicit negative problem of word embeddings is precisely the same as the implicit negative problem in OC-CF.

Conceptually, our proposed objective is similar to word embeddings such as Word2Vec [18] (that can be intepreted as a special case of NCE), where we analogize users as word contexts and items as words. One key difference from word embeddings is that we assume the users (contexts) are unique and that the user interactions (words) are independent with a uniform discrete distribution.

3.3 An NCE Item Embedding Hyperparameter

The optimal solution of NCE as shown in Equation (8) penalizes the influence of popular items on the user and item representation. In other words, it is inversely proportional to the popularity of an observed item. However, this relies heavily on a good estimate of the popularity of observed items $p(j')$. Since the data is sparse, there is high uncertainty on the popularity estimate and this uncertainty propagates to Equation (8).

To alleviate this, we introduce a hyperparameter β into the denominator, which adjusts the penalty on high frequency items. We rewrite Equation (8) to include β as follows:

$$d_{i,j} = \max(\log \sum_{j'} |r_{:,j'}| - \beta \log |\mathbf{r}_{:,j}|, 0) \quad \forall r_{i,j} = 1, \tag{11}$$

where we add a $d_{i,j} \geq 0$ constraint to guarantee the positive feedback is more significant compared to the unobserved feedback in Equation (9). Empirically, this hyperparameter aids generalization on the test set as shown in Figure 4.

3.4 NCE Projected Linear Recommendation

Using optimal user U^* and item V^* embeddings from Equation (10), we can predict unobserved interactions with a simple dot product U^*V^{*T}. Hence, the simple method of NCE-SVD can be used as a recommendation algorithm by itself.

As noted previously, NCE-SVD effectively de-popularizes the dataset by rescaling entries in R inversely proportional to their popularity. However, popularity bias can still be important in terms of ranking performance depending on the dataset [3]. Therefore, following the approach of PLRec, we further perform the Linear Regression of (1) on top of NCE-SVD for the model to learn the importance of different latent features for each user. It also has the side benefit of correcting for approximation error in fast randomized truncated methods used for large-scale SVD [6]. We call this final solution, NCE-PLRec, as it performs NCE followed by PLRec [25].

We note that the static latent representation of NCE-SVD is unable to capture non-uniform weighting of users and items, such as done for drifting user preferences [12] or to improve OC-CF matrix factorization [10]. For example, in the latter case of [10], the authors propose to use a user-item weighting matrix C of the same dimensions R with elements $c_{i,j}$ defined as follows:

$$c_{i,j} = 1 + \alpha r_{i,j}, \tag{12}$$

where hyperparameter $\alpha \geq -1$ modulates the loss weighting differential of positive and negative examples (with $\alpha = 0$ leading to all preferences being weighted equally). We will support this form of C in the following user-item loss weighted extension of basic NCE-PLRec.

To leverage the user-item weights of C in NCE-PLRec, we project the original implicit matrix R onto the learned item representation V^*. This projection produces the user representation, $Q = RV^*$, which is the sum over all item representations of the user's interaction history. Then, we maximize a user-item reweighted version of the PLRec objective as follows:

$$\underset{W}{\operatorname{argmin}} \sum_{i,j} c_{i,j}(r_{i,j} - \mathbf{q}_i \mathbf{w}_j)^2 + \lambda \left\|\mathbf{w}_j\right\|_2^2, \tag{13}$$

where $\mathbf{q}_i$ is the user representation, $c_{i,j}$ is described previously, and W is the linear regression coefficient matrix to estimate. This leads to the most general form of NCE-PLRec that we will define in this paper; if specific user-item weights are not needed, one can simply set $\alpha = 0$.

3.5 NCE-PLRec for Cold-Start Test Users

Referring to Equation (13), the trained weights $\mathbf{w}_j$ for each item j are shared and trained over all training users. Given a cold-start test user i', whose ratings $\mathbf{r}_{i'}$ were not observed during training, we can recommend the top-K items from the projection of $\mathbf{r}_{i'}$ onto the item features and weights learned by NCE-PLRec on the training users. Specifically, a vector of NCE-PLRec predictions for cold-start test user i' is given by $\mathbf{r}_{i'}V^*W^T = \mathbf{q}_{i'}W^T$.

3.6 Algorithm

We summarize the general user-item weighted Noise-Contrastive Estimation Projected Linear Recommender (NCE-PLRec) required to optimize Equation (13) in Algorithm 1.

Algorithm 1 NCE-PLRec

```
1: procedure TRAIN($R$, $\alpha = 0$, $\beta = 1$)
2:     $D^* \leftarrow NCE(R, \beta)$                              ▷ Construct D matrix
3:     $U_D, \Sigma_D, V_D^T \leftarrow$ Truncated SVD($D^*$)
4:     $Q \leftarrow RV_D\Sigma_D^{\frac{1}{2}}$                    ▷ Project implicit matrix
5:     for $i \in range(1, m)$ do                                ▷ Loop over users
6:         $C^j \leftarrow diag(1 + \alpha \mathbf{r}_{:,j})$
7:         $\mathbf{w}_j \leftarrow (Q^T C^j Q + \lambda I)^{-1} Q^T C^j \mathbf{r}_{:,j}$
8:     return $QW^T$                                             ▷ Prediction
```

With user-item loss weighting given by C, the optimization shown in Algorithm 1 appears to be closed-form only with respect to each user. However, when $\alpha = 0$, we note that the NCE-PLRec algorithm has a globally closed-form solution for this special case, i.e., $W = (Q^TQ + \lambda I)^{-1}Q^TR$.

Since user-item loss weighting for implicit feedback recommendation has already been well-studied in the literature [8–11], we do not explore tuning α in our experiments, but simply define NCE-PLRec as in Algorithm 1 for full generality in cases where varying α is desired. Hence, we will set $\alpha = 0$ in all experiments.

Table 1: Summary of datasets used in evaluation.

Dataset	m	n	$\|r_{i,j} > \vartheta\|$	Sparsity
MovieLens-20m	138,493	27,278	12,195,566	3.47×10^{-3}
Netflix Prize	2,649,430	17,771	56,919,190	1.2×10^{-3}
YahooR1 Data	1,948,882	46110	48,817,561	5.43×10^{-4}

4 EXPERIMENTS AND EVALUATION

In this section, we evaluate the proposed NCE-PLRec model by comparing to a variety of state-of-the-art OC-CF algorithms on three real-world datasets with at least 10 million interactions. We compare across a variety of performance metrics and settings including an evaluation of top-k recommendation performance, running time performance, and the distribution of recommended item popularity.

We ran our experiments on a GPU compute cluster with the Slurm workload manager system. Each computation node has one 4-core CPU, 32GB RAM, and one Nvidia Titan XP GPU. Implementation is done with Python 2.7 and includes Tensorflow 1.4 [1]. Code to reproduce all results is available on Github.[1]

4.1 Datasets

We evaluate the candidate algorithms on three publicly available rating datasets: Movielens-20M, Netflix Prize, and Yahoo R1. Each dataset contains more than 10 million interactions. Thus, we are only able to compare with state-of-the-art models that are able to run on these large-scale datasets. For each dataset, we binarize the rating dataset with a rating threshold, ϑ, defined to be the upper half of the range of the ratings. We do this so that the observed interactions correspond to positive feedback. To be specific, the threshold is $\vartheta > 3$ for Movielens-20M and Netflix Prize, and $\vartheta > 70$ for Yahoo R1. Table 1 summarizes the properties of the binarized matrices.

We split the data into train, validation and test sets based on timestamps given by the dataset (when available) to provide a recommendation evaluation setting closer to production use [27]. For each user, we use the first 50% of data as the train set, 20% data as validation set and 30% data as the test set. For the Yahoo dataset, we split the dataset randomly as it does not contain timestamps.

4.2 Evaluation Metrics

We evaluate the recommendation performance using five metrics: Precision@K, Recall@K, MAP@K, R-Precision, and B-NDCG, where R-Precision is an order insensitive metrics, NDCG is order sensitive, and Precision@K as well as Recall@K are semi-order sensitive due to the K values given.

4.3 Candidate Methods

We compare the proposed algorithm with nine state-of-the-art models from classical matrix factorization to the latest Collaborative Metric Learning approaches. These models all scale to the large size of our datasets.

- POP: Most popular items – not user personalized but an intuitive baseline to test the claims of this paper.

[1]https://github.com/wuga214/NCE_Projected_LRec

Table 2: Hyper-parameters tuned on the experiments.

name	Range	Functionality	Algorithms affected
r	{50, 100, 200, 500}	Latent Dimension	PLRec, PureSVD WRMF, AutoRec, CML NCE-SVD, NCE-PLRec BPR, CDAE, VAE-CF
α	{-0.5, -0.4 $\cdots$ -0.1} $\cup$ {0, 0.1, 1, 10, 100}	Loss Weighting	WRMF, NCE-PLRec
β	{0.7, 0.8 $\cdots$ 1.3}	Popularity Sensitivity	NCE-PLRec
λ	{1e-4, 1e-3 $\cdots$ 1e4}	Regularization	PLRec, WRMF, AutoRec CML, BPR, NCE-PLRec CDAE, VAE-CF
ρ	{0.1, 0.2 $\cdots$ 1}	Corruption	CDAE, VAE-CF

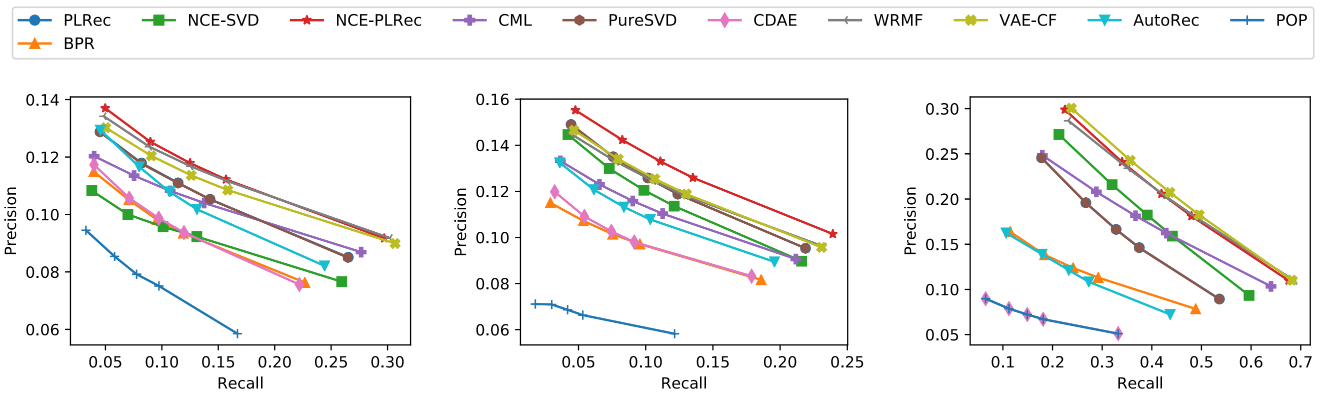
- PureSVD [4]: A similarity based recommendation method that constructs a similarity matrix through SVD decomposition of implicit matrix R.
- WRMF [10]: Weighted Regularized Matrix Factorization.
- AutoRec [26]: A neural Autoencoder based recommendation system with one hidden layer and ReLU activation function.
- CDAE [29]: Collaborative Denoising Autoencoder, which is specifically optimized for implicit feedback recommendation tasks.
- VAE-CF [16]: Variational Autoencoder for Collaborative Filtering – a state-of-the-art deep learning based recommender system.
- BPR [22]: Bayesian Personalized Ranking. One of the first recommendation algorithms that explicitly optimizes pairwise rankings.
- CML [9]: Collaborative Metric Learning. A state-of-the-art metric learning based recommender system.
- PLRec [24]: Also called Linear-Flow. This is the baseline projected linear recommendation approach. This is one ablation of NCE-PLRec.
- NCE-SVD: Inner product of SVD-decomposed item and user representation learned from NCE. This is the second ablation of NCE-PLRec without PLRec's learned linear models.
- NCE-PLRec: The full version of the proposed model.

We tune the hyper-parameters for the candidate algorithms by evaluating on the validation datasets through grid search. The candidate parameter ranges are shown in Table 2. The best hyper-parameter settings found for each algorithm and domain are listed in Table 3.

4.4 Ranking Performance Evaluation

Tables 4, 5 and 6 show the general performance comparison between the proposed model with the nine existing methods on all metrics. From the results, we notice the following observations:

(a) In one of the three domains, the proposed NCE-PLRec model outperforms all candidate methods on all metrics in the experiments. In the other two domains, the NCE-PLRec model also shows to be strongly competitive with the state-of-the-art VAE-CF deep learning model.
(b) NCE-PLRec as a projected linear model shows a substantial performance improvement compared to PLRec, which indicates the benefit of NCE optimized embeddings.



(a) Movielens20m **(b) Netflix** **(c) Yahoo**

Figure 1: Precision-recall curve based on number of items being recommended. Larger area underneath the curve is better. For all three domains, PLRec overlaps with PureSVD. NCE-PLRec performs strongly across the entire precision-recall spectrum.

Table 3: Best hyper-parameter setting for each algorithm.

Domain	Algorithm	r	α	λ	Iteration*	ρ	β
Movielens20m	PLRec	50	1	10000	10	0	1
	BPR	50	1	0.00001	30	0	1
	NCE-SVD	50	1	1	10	0	1
	NCE-PLRec	50	1	1	10	0	1
	CML	100	1	0.01	30	0	1
	PureSVD	50	1	1	10	0	1
	CDAE	50	1	0.00001	300	0.2	1
	WRMF	200	1	100	10	0	1
	VAE-CF	200	1	0.001	300	0.5	1
	AutoRec	50	1	0.00001	300	0	1
Netflix	PLRec	50	1	1	10	0	1
	BPR	50	1	0.00001	30	0	1
	NCE-SVD	50	1	1	10	0	1
	NCE-PLRec	100	1	1	10	0	1.1
	CML	50	1	0.00001	30	0	1
	PureSVD	50	1	1	10	0	1
	CDAE	50	1	0.00001	300	0.2	1
	WRMF	200	10	1000	10	0	1
	VAE-CF	100	1	0.0001	300	0.5	1
	AutoRec	50	1	0.00001	300	0	1
Yahoo	PLRec	50	1	10000	10	0	1
	BPR	50	1	0.00001	30	0	1
	NCE-SVD	50	1	1	10	0	1
	NCE-PLRec	50	1	1000	10	0	1.2
	CML	100	1	0.0001	30	0	1
	PureSVD	50	1	1	10	0	1
	CDAE	50	1	0.1	300	0.2	1
	WRMF	100	100	1000	10	0	1
	VAE-CF	200	1	0.001	300	0.5	1
	AutoRec	50	1	0.00001	300	0	1

* For PureSVD, PLRec, NCE-SVD, and NCE-PLRec, iterations in this table means number of randomized SVD iterations. For WRMF, iteration shows number of alternative close-form optimizations. For AutoRec, BPR, CDAE, and VAE-CF, iteration shows number of epochs that processed over all users.

(c) NCE-SVD alone is not sufficient to provide good performance, indicating the importance of using the PLRec correction to the NCE-SVD embeddings.
(d) WRMF is another strong competitor in terms of general performance. We notice that WRMF outperforms NCE-PLRec when a large number of items are retrieved. This reflects its wide usage as the first stage algorithm in multi-stage recommendation tasks [28], where the first stage retrieves a high recall candidate list to be refined by other models precise at lower ranks in a second stage.
(e) CDAE is inconsistent as it performs well on Movielens-20m and Netflix, but performs poorly on Yahoo R1.
(f) PLRec and PureSVD show similar performances across all three datasets. This observation supports our theoretical claim that PLRec should learn a near-optimal weight $W \approx V$ that is close to the SVD decomposition given by PureSVD.

Figure 1 provides further detail of the performance through the precision-recall curve with $K \in [5, 10, 15, 20, 50]$. It shows that NCE-PLRec has consistently good performance over the number of items being retrieved. Especially, in the Netflix dataset, NCE-PLRec shows significantly better performance than the other candidates. We also observe that the performance of AutoRec and CDAE drops considerably when the data becomes sparse with large user-item dimensions. We will discuss reasons for this in Section 4.6, which explores the popularity distribution of recommendations.

4.5 Performance vs. User Interaction Level

We now investigate conditions where the proposed algorithm works better compared to the strongest baselines. We categorize users based on the number of interactions they made in the training set into 4 categories. The categories come from the 25%, 50%, 75%, and 100% quantiles of the number of training interactions, which indicate how often the user rated items in the training set.

Figure 2 shows comparative results in regard to the four quantiles for MovieLens-20M. In general, NCE-PLRec shows strong performance across all quantiles of user interaction volume. While VAE-CF shows better performance over the user categories with a lower number of ratings, its performance drops considerably with a

Table 4: Results of Movielens-20M dataset with 95% confidence interval. Hyper-parameters are chosen from the validation set.

model	R-Precision	NDCG	MAP@5	MAP@50	Precision@5	Precision@50	Recall@5	Recall@50
POP	0.068±0.0005	0.1194±0.0007	0.1011±0.0011	0.0739±0.0005	0.0945±0.0009	0.0585±0.0004	0.0327±0.0004	0.167±0.0011
AutoRec	0.0931±0.0006	0.1693±0.0009	0.1388±0.0012	0.1018±0.0006	0.1294±0.001	0.0821±0.0005	0.0455±0.0005	0.2442±0.0013
BPR	0.0846±0.0006	0.154±0.0008	0.1207±0.0011	0.0927±0.0006	0.1149±0.001	0.0763±0.0004	0.0397±0.0005	0.2267±0.0012
CDAE	0.084±0.0006	0.1536±0.0008	0.1259±0.0012	0.0932±0.0006	0.1173±0.001	0.0756±0.0005	0.0399±0.0005	0.2219±0.0012
CML	0.0906±0.0006	0.177±0.0008	0.1246±0.0011	0.1024±0.0006	0.1203±0.001	0.0869±0.0005	0.04±0.0005	0.2766±0.0013
PLRec	0.0955±0.0006	0.1785±0.0008	0.1376±0.0012	0.1042±0.0006	0.1288±0.001	0.0851±0.0004	0.0452±0.0005	0.2652±0.0012
PureSVD	0.0954±0.0006	0.1783±0.0008	0.1375±0.0012	0.1041±0.0006	0.1287±0.001	0.085±0.0004	0.0451±0.0005	0.2647±0.0012
VAE-CF	0.1008±0.0006	**0.1973±0.0009**	0.1371±0.0012	0.1075±0.0006	0.1302±0.001	0.0899±0.0004	**0.0503±0.0006**	**0.3067±0.0014**
WRMF	0.1±0.0006	0.1962±0.0008	0.1433±0.0012	0.1106±0.0006	0.1342±0.001	**0.0918±0.0004**	0.0485±0.0005	0.3021±0.0014
NCE-PLRec	**0.102±0.0006**	0.1957±0.0009	**0.1456±0.0012**	**0.1113±0.0006**	**0.137±0.001**	0.0917±0.0005	0.0498±0.0005	0.2971±0.0014
NCE-SVD	0.0809±0.0005	0.1638±0.0008	0.115±0.0011	0.0911±0.0005	0.1083±0.0009	0.0766±0.0004	0.0379±0.0005	0.2594±0.0013

Table 5: Results of Netflix dataset with 95% confidence interval. Hyper-parameters are chosen from the validation set.

model	R-Precision	NDCG	MAP@5	MAP@50	Precision@5	Precision@50	Recall@5	Recall@50
POP	0.0486±0.0002	0.0853±0.0003	0.0747±0.0005	0.065±0.0003	0.0711±0.0004	0.0582±0.0002	0.0179±0.0002	0.1215±0.0005
AutoRec	0.0876±0.0003	0.1454±0.0004	0.14±0.0006	0.1074±0.0003	0.1324±0.0005	0.0894±0.0003	0.0361±0.0002	0.1958±0.0006
BPR	0.0757±0.0002	0.1312±0.0003	0.1197±0.0006	0.096±0.0003	0.115±0.0005	0.0816±0.0002	0.0291±0.0002	0.1859±0.0006
CDAE	0.0797±0.0003	0.1316±0.0004	0.1251±0.0006	0.0979±0.0003	0.1198±0.0005	0.0832±0.0002	0.0323±0.0002	0.1788±0.0006
CML	0.0878±0.0003	0.1511±0.0004	0.1398±0.0006	0.1091±0.0003	0.1332±0.0005	0.0906±0.0003	0.0365±0.0002	0.2117±0.0006
PLRec	0.0994±0.0003	0.1645±0.0004	0.1591±0.0007	0.118±0.0003	0.149±0.0006	0.0953±0.0003	0.0445±0.0003	0.2189±0.0006
PureSVD	0.0994±0.0003	0.1644±0.0004	0.159±0.0007	0.118±0.0003	0.149±0.0005	0.0953±0.0003	0.0445±0.0003	0.2188±0.0006
VAE-CF	0.1017±0.0003	0.1705±0.0004	0.1559±0.0007	0.1176±0.0003	0.1465±0.0005	0.0957±0.0003	0.0467±0.0003	0.2309±0.0006
WRMF	0.0985±0.0003	0.1681±0.0004	0.1531±0.0007	0.117±0.0003	0.1447±0.0006	0.096±0.0003	0.045±0.0003	0.2325±0.0007
NCE-PLRec	**0.1049±0.0003**	**0.1764±0.0004**	**0.1654±0.0007**	**0.1247±0.0003**	**0.1552±0.0006**	**0.1015±0.0003**	**0.0477±0.0003**	**0.2392±0.0007**
NCE-SVD	0.0917±0.0003	0.158±0.0004	0.1559±0.0007	0.1129±0.0003	0.1446±0.0006	0.0897±0.0002	0.0422±0.0003	0.216±0.0006

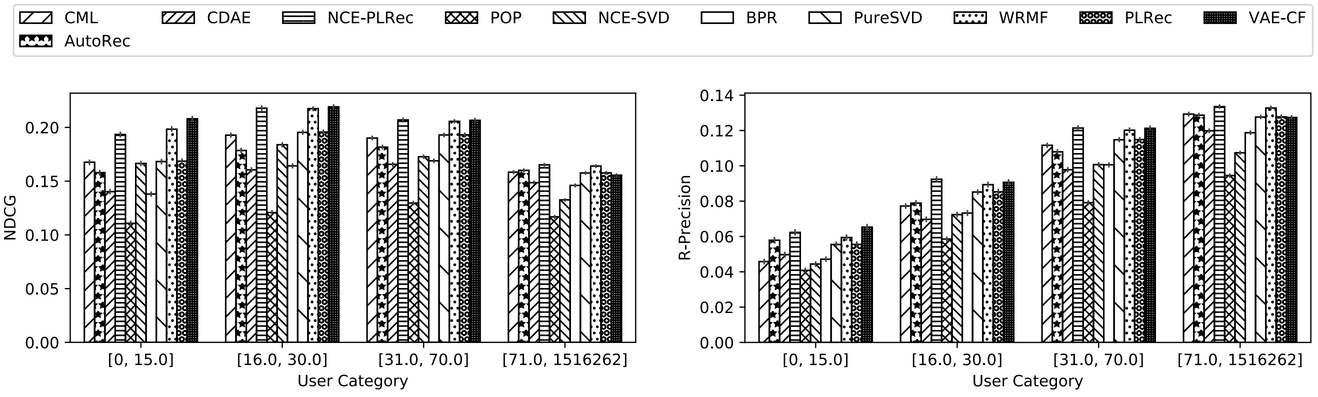
Table 6: Results of Yahoo dataset with 95% confidence interval. Hyper-parameters are chosen from the validation set.

model	R-Precision	NDCG	MAP@5	MAP@50	Precision@5	Precision@50	Recall@5	Recall@50
POP	0.0795±0.0004	0.1843±0.0005	0.1022±0.0006	0.0672±0.0003	0.0897±0.0005	0.051±0.0002	0.0652±0.0005	0.3322±0.0009
AutoRec	0.1338±0.0005	0.2711±0.0007	0.1845±0.0008	0.1093±0.0004	0.1625±0.0007	0.0723±0.0003	0.1065±0.0006	0.437±0.001
BPR	0.1377±0.0005	0.288±0.0006	0.1822±0.0008	0.1129±0.0004	0.1637±0.0007	0.0784±0.0003	0.1146±0.0006	0.4883±0.001
CDAE	0.0795±0.0004	0.1843±0.0005	0.1022±0.0006	0.0672±0.0003	0.0897±0.0005	0.051±0.0002	0.0652±0.0005	0.3323±0.0009
CML	0.2101±0.0007	0.4046±0.0007	0.2754±0.001	0.162±0.0005	0.2484±0.0008	0.1034±0.0003	0.1795±0.0007	0.6401±0.0009
PLRec	0.2051±0.0007	0.3673±0.0008	0.2838±0.001	0.1496±0.0005	0.2456±0.0008	0.0893±0.0003	0.1779±0.0007	0.5363±0.001
PureSVD	0.205±0.0007	0.367±0.0008	0.2836±0.001	0.1495±0.0005	0.2455±0.0008	0.0893±0.0003	0.1778±0.0007	0.5358±0.001
VAE-CF	**0.2701±0.0008**	**0.4698±0.0008**	0.3423±0.001	**0.1849±0.0005**	**0.3005±0.0009**	**0.1102±0.0003**	**0.238±0.0008**	**0.6824±0.0009**
WRMF	0.2612±0.0008	0.4633±0.0008	0.3244±0.001	0.1795±0.0005	0.2866±0.0008	0.1094±0.0003	0.2321±0.0008	0.6881±0.0009
NCE-PLRec	0.2562±0.0008	0.4595±0.0008	**0.344±0.0011**	0.1841±0.0005	0.2989±0.0009	0.1098±0.0004	0.2249±0.0008	0.6757±0.0009
NCE-SVD	0.2332±0.0007	0.412±0.0008	0.313±0.001	0.1624±0.0005	0.2713±0.0008	0.0934±0.0003	0.2129±0.0008	0.5959±0.001

higher number of ratings. With more than 71 ratings, VAE-CF performs even worse than the simpler AutoRec model. We conjecture that VAE blindly increases the certainty of prediction given more observed user interactions. While it seems reasonable in the case that the user rates items with coherent properties, this assumption does not hold if there are the thousands of items being rated by the user. WRMF as another strong competitor also shows robust performance where we observe very similar performance for all four rating quantiles comparing to the NCE-PLRec. CML is competitive when the number of observed ratings is higher than 71. This is reasonable because CML learns better distance metrics with more observations.

4.6 Popularity Distribution

We analyze the sensitivity of the candidate methods' recommendations on popular items as shown in Figure 3. In general, most of the candidate learning methods show strong personalization (recommendations of less popular items) except AutoRec and CDAE, which tend to recommend popular items. While recommending popular items in a relatively dense and small dataset achieves acceptable performance, the performance drops quickly with increasing data sparsity and dimensionality as shown in Figure 1. BPR does not show strong personalization in this Figure due to the insufficient pairwise sampling given limited training epochs. While we are able to tune the sampling size for a small dataset such as Movielens



(a) NDCG

(b) R-Precision

Figure 2: Performance comparison for different quantiles of user activity (number of ratings) for MovieLens-20M. Error bars show standard derivation. All figures share the legend.

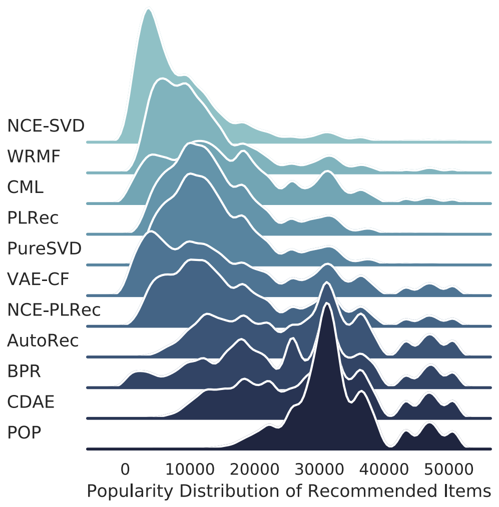


Figure 3: Popularity distribution of items recommended by the candidate algorithms for all the users. Higher value in x-axis indicates less personalization. Algorithms are sorted by the median of their popularity distributions respectively.

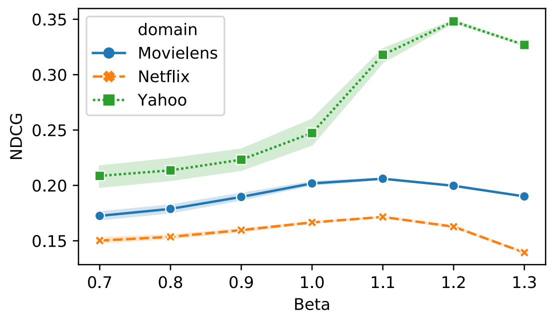


Figure 4: The effect of tuning the hyper-parameter β on the three datasets.

1m, it is hard to estimate the pairwise sampling size that maintains a good performance on very large datasets considering the stochasticity of performance contributed by the noise. On the other hand, NCE-SVD learns to only recommend unpopular items since the NCE embedding is de-popularized. Impressively, NCE-PLRec spreads its recommendations over the popularity spectrum compared to other algorithms and this proves to be beneficial in terms of its overall ranking performance previously observed in Tables 4, 5 and 6.

4.7 Hyper-parameter Tuning

Figure 4 shows the effects of tuning hyper-parameter β for NCE-PLRec defined in Equation (11) on NDCG in all datasets (performance on other metrics was similar). A higher value indicates higher popularity. While only moderate in the Movielens-20M and Netflix datasets, we observe a remarkable performance improvement by adjusting the weighting of the noise contrastive term in the Yahoo dataset. This observation corresponds to our conjecture that this adjustment of the level of depopularization is critical for working with extremely sparse recommendation data as found in this dataset.

4.8 Training Time and Scalability

Figure 5 shows the total time taken for training the candidate methods on the Netflix dataset. We compare only the training time since the prediction and evaluation step require similar operations for all algorithms and take approximately the same time. The result shows the significant efficiency improvements from the linear models compared to neural network and alternating least squares training.

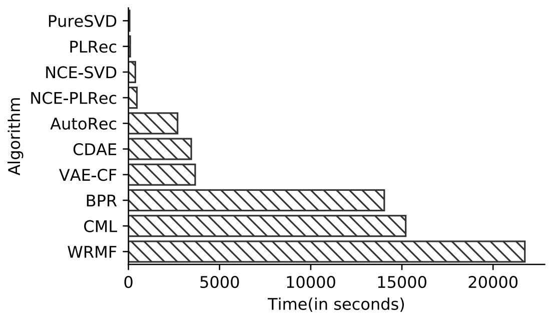


Figure 5: Training times in seconds of the various methods on Netflix. AutoRec, CDAE, and VAE-CF are neural networks that optimized through gradient descent. BPR and CML requires pairwise sampling to learn the ranking. Alternating optimization of WRMF requires the inversion of large matrices twice per iteration.

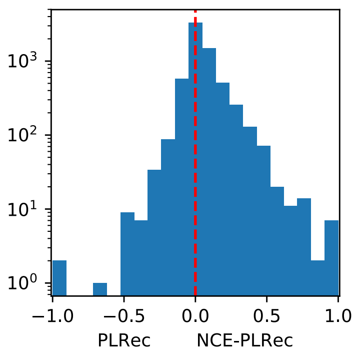


Figure 6: Cold-start comparison histogram for Recall@50 of NCE-PLRec minus Recall@50 for PLRec. Positive values show NCE-PLRec has higher Recall whereas negative shows PLRec has higher Recall. The significant skew of area to the right side of the dotted 0.0 red line indicates that more cold-start users benefitted from NCE-PLRec.

All PLRec methods including NCE-PLRec *easily* scale to these very large datasets.

4.9 Cold-Start Test Users Case Study

Among the recommendation methods, PureSVD, PLRec and NCE-PLRec are able to handle cold-start recommendations without leveraging additional side information. Since PLRec and PureSVD behave similarly, we only compare NCE-PLRec to PLRec for our user cold-start case study.

Figure 6 shows a more comprehensive pairwise comparison between NCE-PLRec and PLRec for the cold-start test users evaluation. In this experiment, we randomly remove 5% of the users from the training dataset and use the remaining users for training. Then, we use the trained model to recommend items to the 5% of held-out cold-start test users and evaluate performance. Clearly, most of the users received better cold-start recommendations from NCE-PLRec compared to PLRec in terms of Recall@50.

5 CONCLUSION

We proposed a novel and highly efficient linear recommendation algorithm called Noise Contrastive Estimation Projected Linear Recommendation (NCE-PLRec) that leverages item embeddings learned from NCE to make predictions using the highly scalable PLRec approach. NCE-PLRec addresses the strong popularity bias of existing SVD-based embedding approaches for recommendation that are exacerbated by the extreme sparsity of implicit feedback matrices in the OC-CF setting. On three large OC-CF recommendation datasets, we showed that NCE-PLRec outperforms several robust and scalable recommendation methods (or performs comparably to the best performing algorithms) in almost all metrics. Furthermore, NCE-PLRec is highly efficient during training, personalized with little popularity bias, and is able to effectively handle cold-start user recommendation without leveraging side information.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. http://tensorflow.org/ Software available from tensorflow.org.
[2] Arindam Banerjee. 2007. An analysis of logistic models: exponential family connections and online performance. In *Proceedings of the 2007 SIAM International Conference on Data Mining*. SIAM, 204–215.
[3] Rocío Cañamares and Pablo Castells. 2018. Should I Follow the Crowd? A Probabilistic Analysis of the Effectiveness of Popularity in Recommender Systems. *SIGIR 18 The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval* (2018), 415–424.
[4] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 39–46.
[5] Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. 297–304.
[6] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. 2011. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review* 53, 2 (2011), 217–288.
[7] Tatsunori B Hashimoto, David Alvarez-Melis, and Tommi S Jaakkola. 2016. Word embeddings as metric recovery in semantic spaces. *Transactions of the Association for Computational Linguistics* 4 (2016), 273–286.
[8] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 173–182.
[9] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. 2017. Collaborative metric learning. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 193–201.
[10] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. Ieee, 263–272.
[11] Christopher C Johnson. 2014. Logistic matrix factorization for implicit feedback data. In *Advances in Neural Information Processing Systems*.
[12] Yehuda Koren. 2009. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 447–456.
[13] Amy N Langville, Carl D Meyer, and Russell Albright. 2006. Initializations for the nonnegative matrix factorization. Proceedings of the twelfth ACM SIGKDD international conference on knowledge discovery and data mining.
[14] M. Levy and K. Jack. 2013. Efficient top-n recommendation by linear regression. In *In RecSysâĂŽ13 Large Scale Recommender Systems Workshop*.
[15] Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*. 2177–2185.
[16] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. *arXiv preprint arXiv:1802.05814* (2018).

[17] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[18] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[19] Xia Ning and George Karypis. 2011. Slim: Sparse linear methods for top-n recommender systems. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*. IEEE, 497–506.

[20] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. 2008. One-class collaborative filtering. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. IEEE, 502–511.

[21] Arkadiusz Paterek. 2007. Improving regularized singular value decomposition for collaborative filtering.

[22] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 452–461.

[23] Badrul M Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2002. Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In *Proceedings of the fifth international conference on computer and information technology*. Vol. 1.

[24] S. Sedhain, H. Bui, J. Kawale, N. Vlassis, B. Kveton, A. Menon, T. Bui, and S. Sanner. 2016. Practical Linear Models for Large-Scale One-Class Collaborative Filtering. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI-16)*. New York, USA.

[25] S. Sedhain, A. Menon, S. Sanner, and D. Braziunas. 2016. On the Effectiveness of Linear Models for One-Class Collaborative Filtering. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*. Phoenix, USA.

[26] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web*. ACM, 111–112.

[27] Guy Shani and Asela Gunawardana. 2011. Evaluating recommendation systems. In *Recommender systems handbook*. Springer, 257–297.

[28] Maksims Volkovs, Himanshu Rai, Zhaoyue Cheng, Ga Wu, Yichao Lu, and Scott Sanner. 2018. Two-stage model for automatic playlist continuation at scale. In *Proceedings of the ACM Recommender Systems Challenge 2018*. ACM, 9.

[29] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. ACM, 153–162.