Non-popular Items for Accurate & Diverse Linear Auto-Encoding Recommenders

Farhan Khawar*
Amazon
Toronto, Canada
fkhawar@connect.ust.hk

Adam Roegiest
Zuva
Toronto, Canada
adam@zuva.ai

ABSTRACT

High-dimensional linear models are some of the best performing collaborative filtering models today. They learn full-rank item embeddings by inverting the Gram matrix calculated from the input user-item matrix. The entries of this Gram matrix are item co-occurrence counts which are unbounded. As a result, the Gram matrix is dominated by larger co-occurrence counts of popular items. In this paper, we propose to alleviate this issue by incorporating cosine similarity with the co-occurrence counts. We show that this increases the recommender diversity and more non-popular items are recommended. We also show that this increase in diversity correlates with an increase in accuracy signifying that the newly recommended non-popular items are relevant. Finally, we also present a more efficient procedure to obtain the parameters of linear auto-encoding recommender and show that it reduces the running time by at least half on the three standard publicly available datasets used for this line of research.

CCS CONCEPTS

• **Information systems → Collaborative filtering**; **Recommender systems**.

KEYWORDS

Collaborative Filtering, Linear Auto-Encoders, Implicit Feedback, Recommender Systems

ACM Reference Format:
Farhan Khawar and Adam Roegiest. 2022. Non-popular Items for Accurate & Diverse Linear Auto-Encoding Recommenders . In *Proceedings of the 2022 ACM SIGIR International Conference on the Theory of Information Retrieval (ICTIR '22), July 11–12, 2022, Madrid, Spain.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3539813.3545132

1 INTRODUCTION

Binary implicit feedback is one of the most simple signals of user preference. In this type of data, a 1 denotes that the user has interacted with the item and a 0 denotes the absence of an interaction. By utilizing the implicit feedback from multiple users we can employ collaborative filtering techniques to make recommendations of new items to a user.

Recently, auto-encoder based collaborative filtering methods [2, 4–6] have shown state-of-the-art performance for the recommendation task. These work by taking a user's implicit feedback vector over the items as the input and then performing some operations (linear or non-linear) to attempt to recreate this vector at the output. These auto-encoders have mechanisms that discourage the model from learning the identity function and that force the model to learn the underlying item interactions. These interactions are then used to generate the output.

A subset of auto-encoder recommenders are linear full-rank auto-encoders [5, 6]. They perform the linear operation of multiplying the input with a full-rank item interaction matrix B to subsequently recreate the input. This simple method is recently one of the best implicit feedback recommenders [5, 6].

To obtain the item interaction matrix B, existing methods [5, 6] rely on the Gram matrix. The i, j entry of Gram matrix is the number of users that have consumed items i and j together. However, this can bias the recommender to recommend popular items because items with larger co-occurrence counts dominate the Gram matrix. Like the Gram matrix, the cosine similarity matrix also relies on the co-occurrence counts, but it normalizes to account for the popularity bias. In this paper, we show that by combining the cosine similarity with the Gram matrix we can recommend more non-popular items and hence increase the overall diversity of the recommender. In addition, we show that this diversity comes at no accuracy cost, and the accuracy increases in proportion to the increase in diversity.

Linear auto-encoders, like Balen and Goethels [6], calculate the inverse of the Gram matrix followed by its Cholesky decomposition. These operations can be computationally intensive for larger datasets. We show that an alternative procedure to perform the Cholesky decomposition followed by forward substitution leads to faster wall clock times while being theoretically equivalent.

The main contributions of this paper are as follows:

- We illustrate the usefulness of injecting the cosine similarity signal along with the co-occurrence signal and show that it increases the recommender diversity by recommending non-popular items.
- We show that recommending non-popular items increases the accuracy of the recommendation.
- We provide of more efficient implementation of linear auto-encoders that decrease the running time by at least half.

*This work was done prior to joining Amazon while the author was affiliated with Zuva.

ICTIR '22, July 11–12, 2022, Madrid, Spain

ACM ISBN 978-1-4503-9412-3/22/07…$15.00
https://doi.org/10.1145/3539813.3545132

Algorithm 1: Training eCHOL in Python 3

Input: data Gram-matrix $G := X^\top X \in \mathbb{R}^{N \times N}$,
L2-norm regularization-parameter $\lambda \in \mathbb{R}^+$.
Output: weight-matrix B with zero diagonal.

```
diagIndices = numpy.diag_indices(G.shape[0])
G[diagIndices] += λ
Q = scipy.linalg.cholesky(G, lower=True)
P = scipy.linalg.cho_solve((Q,True), np.eye(Q.shape[0]))
B = P / (-numpy.diag(P))
B[diagIndices] = 0
```

2 LINEAR AUTO-ENCODERS

Let the input implicit feedback of M user and N items be represented by an $M \times N$ binary matrix X. Linear auto-encoders aim to learn the parameter matrix $B \in R^{N \times N}$ to solve the following least-squares problem:

$$\|X - XB\|_F^2 + \lambda \|B\|_F^2 \tag{1}$$

The closed form solution to this is called the EASE[5] recommender:

$$B = I - PD^{-1} \tag{2}$$

where, I is the identity matrix, D is the diagonal matrix that contains the diagonal entries of P and P is the regularized inverse of the Gram (co-occurrence) matrix, $G = X^T X$, that is:

$$P = (X^T X + \lambda I)^{-1} \tag{3}$$

An alternative procedure to solve the objective function of equation 1 was proposed in [6] in which they obtain B as follows:

$$B = \beta I - PD^{-1} = (\beta D - P)D^{-1} = AD^{-1} \tag{4}$$

where, $A = \beta D - P$ and β is a scalar to make A positive semi-definite. Then the Cholesky decomposition of A is performed (i.e., $A = L^T L$, where L is a lower triangular matrix) to yield:

$$B = LL^T D^{-1} \tag{5}$$

And this is the CHOL [6] recommender. The CHOL recommender performs this Cholesky decomposition to increase the recommendation diversity [6](i.e., the number of unique items recommended). However, it requires a larger computational time as it involves calculating the inverse of the regularized Gram matrix $(G + \lambda I)$ followed by the Cholesky decomposition of A and both these matrices are $N \times N$.

In the following sections, we first show how we can make this procedure more efficient, and then we show that adding the cosine similarity can increase the diversity without compromising accuracy.

2.1 Efficient Implementation

Instead of calculating the inverse of G first and then performing the Cholesky decomposition of P, we first perform the Cholesky decomposition of $G = QQ^T$, where Q is a lower triangular matrix. Then we calculate P by solving $QQ^T P = I$ via backward substitution [3]. Since performing the backward substitution is $O(N)^2$ compared to $O(N)^3$ for calculating the inverse, this is a more efficient approach. The Python code of eCHOL is given in Algorithm 1.

Table 1: Statistics of the datasets.

	ML20M	Netflix	MSD
# of users	136,677	463,435	571,355
# of items	20,108	17,769	41,140
# of interactions	10.0M	56.9M	33.6M
% of interactions	0.36%	0.69%	0.14%
# of val./test users	10,000	40,000	50,000

We observe that $P = G^{-1} = Q^{T^{-1}} Q^{-1} = Q^{-1^T} Q^{-1}$. Let $R^T = Q^{-1}$, then $P = RR^T$. Substituting this into equation 2 results in $B = I - RR^T D^{-1}$. Since we are concerned with ranking *non-consumed* items, this is equivalent to $RR^T D^{-1}$ which is the same form as the CHOL solution from equation 5.

2.2 Incorporating Cosine Similarity

Linear auto-encoders rely on the information that is available in the item co-occurrence matrix G. A popular item will have high co-occurrence counts compared to an unpopular item. These higher counts dominate G, and, as a result, some relevant but non-popular items will not be recommended.

To cater for this imbalance we can normalize each column of X with its L_2-norm. That is, $\bar{X} = XZ$ where Z is a diagonal matrix that contains the L_2-norm of each column of X. This results in each entry $[\hat{X}^T \hat{X}]_{i,j} \in [0, 1]$ corresponding to the cosine similarity between items i and j.

Let B_1 and B_2 be the outputs of Algorithm 1 when $X^T X$ and $\bar{X}^T \bar{X}$ are passed as inputs respectively. Then, the final item interaction matrix used for eCHOLc will be $B = B_1 + B_2$.

3 EXPERIMENTAL SETUP

We compare the accuracy of eCHOLc in terms of NDCG@100, Recall@20 and Recall@50 (similar to [2, 5, 6]) with linear auto-encoders [5, 6], three non-linear auto-encoders Mult-vae [4], Mult-dae [4] and Sw-dae [2], and weighted matrix factorization [1]. The evaluation is done on three commonly used datasets [2, 4–6]: Movielens20M (ML-20M), Netflix and MSD. The details of the datasets are shown in Table 1. For direct comparison, we use the strong generalization setup that was used in [2, 4–6]: both validation and test sets contain users that were not seen in the train set. We used the same dataset splits (using the same random seed) to split the data into train, validation and test sets.

4 RESULTS & DISCUSSIONS

4.1 Running time of eCHOL

Table 3 shows the training times in seconds for eCHOL along with eCHOLc, CHOL and EASE. We see that the implementation of eCHOL outlined in Algorithm 1 improves the running time by at least two times compared to CHOL. In addition, it is also faster than EASE. Finally, as expected the running time of eCHOLc is twice that of eCHOL, however it is still faster than CHOL and EASE. The only exception is the MSD dataset where EASE is faster than eCHOLc.

Table 2: Comparison between eCHOLc with various baselines on the test set. Results for the non-linear auto-encoders and Wmf are consistent with [2].

(a) ML-20M

	Recall@20	Recall@50	NDCG@100
eCHOLc	0.393	0.527	0.424
CHOL	0.391	0.521	0.420
EASE	0.391	0.521	0.420
Mult-vae	0.395	0.537	0.426
Mult-dae	0.387	0.524	0.419
Sw-dae	**0.410**	**0.549**	**0.442**
Wmf	0.360	0.498	0.386

(b) Netflix

	Recall@20	Recall@50	NDCG@100
eCHOLc	0.363	0.447	0.395
CHOL	0.362	0.445	0.393
EASE	0.362	0.445	0.393
Mult-vae	0.351	0.444	0.386
Mult-dae	0.344	0.438	0.380
Sw-dae	**0.370**	**0.458**	**0.404**
Wmf	0.316	0.404	0.351

(c) MSD

	Recall@20	Recall@50	NDCG@100
eCHOLc	**0.334**	**0.428**	**0.390**
CHOL	0.333	0.428	0.389
EASE	0.333	0.428	0.389
Mult-vae	0.266	0.364	0.316
Mult-dae	0.266	0.363	0.313
Sw-dae	0.317	0.416	0.372
Wmf	0.211	0.312	0.257

Table 3: Training time in seconds of eCHOL compared with other linear auto-encoders. eCHOL is atleast 2 times faster than CHOL.

Dataset	EASE	CHOL	eCHOL	eCHOLc
ML-20M	269.4	182.11	85.1	171.1
Netflix	275	330.82	117.65	251.34
MSD	1090.3	1472.2	719.24	1485.6

4.2 Performance of eCHOLc

Table 2 shows the performance of eCHOLc and the baselines on the three datasets with respect to Recall@20,50 and NDCG@100.[1] Among the linear auto-encoders, eCHOLc performs the best across all three datasets. This suggests that incorporating cosine similarity information helped the recommendation. Secondly, we observe that the relative performance improvement of eCHOLc over EASE/CHOL is the most on ML-20M and the least on MSD. We will see how the diversity of the recommendation can provide more insights into these observations.

Sw-dae is the strongest baseline as it provides the best results on two out of the three datasets. It is a non-linear auto-encoder that relies on a sparse and wide bottleneck layer to encode item relationships. But this bottleneck layer is not as wide as the full-rank embeddings obtained by linear models and as a result on the largest dataset (MSD) all linear full-rank auto-encoders outperform it by a considerable margin and eCHOLc provides the best accuracy.

[1]Since eCHOL is equivalent to CHOL its performance is not shown. The ablation study of Table 5 provides the results for eCHOL.

Table 4: The total number of unique items recommended to all test users. We see that eCHOLc increases the recommender coverage from 0.6-39%.

Dataset	CHOL	eCHOLc	% Increase
ML-20M	4291	5969	39.1
Netflix	9206	11402	23.85
MSD	40033	40276	0.61

4.3 Non-popular Items and Accuracy

We refer to the recommender diversity (or coverage) as the total number of unique items recommended across all users. Table 4 shows the diversity of eCHOLc versus CHOL for all three datasets. We see that eCHOLc increased the recommendation diversity between 0.6% and 39% compared to CHOL depending on the dataset. This increase is due to eCHOLc incorporating the additional information from cosine similarity which CHOL does not exploit. But what is more interesting is that the increased diversity correlates with the increase in accuracy in Table 2. For example, for ML-20M we see the greatest increase in diversity and for MSD we see the least. Looking closely at Table 4 we notice that CHOL is already recommending almost all the items hence there is little room for improvement, but for ML-20M there is much more room for improvement in diversity. This also suggests that the issue of popularity bias is more prevalent in ML-20M and the Gram matrix based CHOL suffers for it.

To see what type of additional items are recommended by incorporating the cosine similarity, we show the frequency of the 39% new items recommended by eCHOLc for ML-20M in green in Figure 1. The other recommended items are shown in blue. The y-axis shows the number of times each item appeared in the training set and the x-axis denotes the item ID. Figure 1(b) is the same as Figure 1(a) but its y-axis is in log scale. We see that all the new items recommended due to incorporating cosine similarity are non-popular (i.e., consumed less than 600 times in the training set). This conforms with our intuition that by incorporating the cosine similarity we can encourage the recommendation of the non-popular (long-tail) items. Thus, eCHOLc can increase accuracy by recommending more diverse items which are non-popular. This is in contrast to the traditional notions of the accuracy-diversity trade-off.

4.4 Ablation Study

We perform an ablation study to see the effect of the Gram and Cosine similarity matrix components of eCHOLc. Table 5 shows the NDCG and Recall scores for:

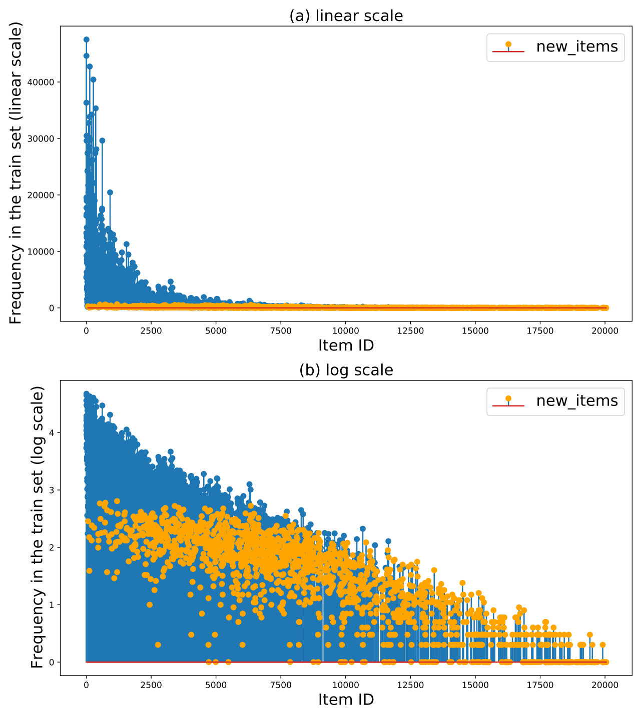

Figure 1: The train set frequency of the additional items recommended by eCHOLc due to the addition of cosine similarity is shown in green. The frequency of the items recommended in the absence of the cosine similarity signal is in blue. We see that all the additional items recommended by including cosine similarity were non-popular.

Table 5: Ablation study of the components of eCHOLc on the ML-20M dataset. While CHOLc alone is not as good as eCHOL, their combined performance yields the best results.

ML-20M	eCHOL	CHOLc	eCHOLc
Recall@20	0.391	0.382	0.393
Recall@50	0.521	0.517	0.527
NDCG@100	0.420	0.382	0.424

- eCHOL which has shown equivalent performance but faster training time compared to CHOL,
- CHOLc which is the same as eCHOL except that the input is the cosine similarity matrix instead of the Gram matrix,
- eCHOLc which combines both eCHOL and CHOLc.

We first observe that the accuracy of eCHOL is the same as CHOL from Table 2. This is expected as both procedures are equivalent. Secondly, we see from comparing eCHOL and CHOLc that cosine similarity by itself does not yield superior performance compared the to Gram matrix. However, by combining the information from both the cosine and Gram matrix we can get the best combination of popular and unpopular items and hence better performance.

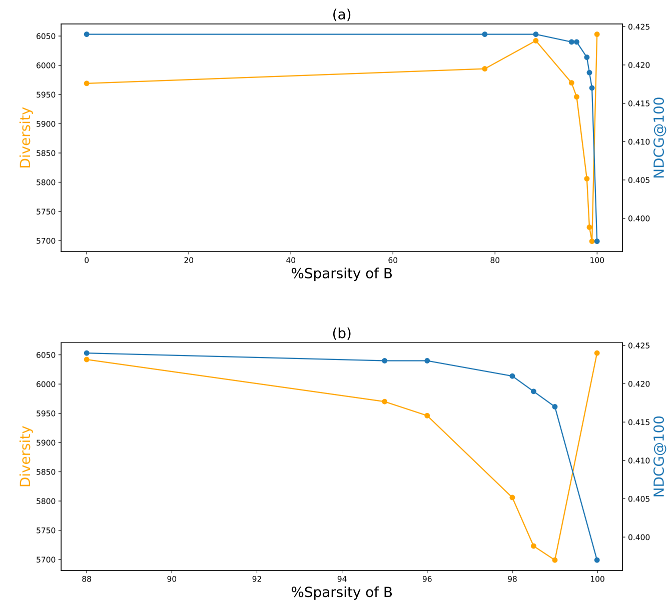

Figure 2: (a) The effect of sparsifying B on NDCG@100 and the recommendation diversity for eCHOLc for the ML-20M dataset. At 98.5% sparsity the NDCG@100 is at par with CHOL after which the diversity and NDCG@100 drop. A zoomed in version of (a) is shown in (b).

4.5 Making B Sparse

The parameter matrix B learned by eCHOLc (and other linear auto-encoders) is full-rank, but they have many entries close to 0. We can prune these entries smaller (in absolute value) than a threshold to yield a much sparser parameter matrix B. This aids in reducing the memory footprint which is especially useful if the number of items is large. We explore the robustness of the recommendation performance as the sparsity level of B is increased and plot the diversity and the NDCG@100 in Figure 2. At 95% sparsity, there is minimal loss in accuracy or diversity, and at 98.5% the performance of eCHOLc is effectively the same as that of CHOL. After this point, B becomes too sparse, and valuable signals are lost leading to decreased accuracy. This is accompanied by an increase in diversity as random noisy items start to get recommended.

We also observe an interesting phenomenon of diversity increasing at 88%. Upon examination, we found that the new items were non-popular items with a median frequency of 27 in the training dataset. It might be the case that these items would have been relevant but since they appear so seldom in the datasets (both train and test), they don't impact the accuracy metrics.

5 CONCLUSION

In this paper, we presented an efficient linear auto-encoder and showed how it can be extended to incorporate cosine similarity to mitigate the popularity bias that the existing linear auto-encoders are susceptible to. We argued that this is because existing linear auto-encoders rely on the Gram matrix which is dominated by large co-occurrence counts from popular items. We showed that since cosine similarity is normalized to be between 0 and 1, it can alleviate this problem and can increase the diversity of the recommender by recommending non-popular items. Finally, we showed that this diversity increase is proportional to an increase in accuracy which is counterintuitive to the traditional accuracy-diversity trade-off.

REFERENCES

[1] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE international conference on data mining*. Ieee, 263–272.
[2] Farhan Khawar, Leonard Poon, and Nevin L Zhang. 2020. Learning the structure of auto-encoding recommenders. In *Proceedings of The Web Conference 2020*. 519–529.
[3] Aravindh Krishnamoorthy and Deepak Menon. 2013. Matrix inversion using Cholesky decomposition. In *2013 signal processing: Algorithms, architectures, arrangements, and applications (SPA)*. IEEE, 70–72.
[4] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 world wide web conference*. 689–698.
[5] Harald Steck. 2019. Embarrassingly shallow autoencoders for sparse data. In *The World Wide Web Conference*. 3251–3257.
[6] Jan Van Balen and Bart Goethals. 2021. High-dimensional sparse embeddings for collaborative filtering. In *Proceedings of the Web Conference 2021*. 575–581.