Cleaned Similarity for Better Memory-Based Recommenders

Farhan Khawar
fkhawar@cse.ust.hk
Department of Computer Science & Engineering
The Hong Kong University of Science and Technology
Hong Kong

Nevin L. Zhang
lzhang@cse.ust.hk
Department of Computer Science & Engineering
The Hong Kong University of Science and Technology
Hong Kong

ABSTRACT

Memory-based collaborative filtering methods like user or item k-nearest neighbors (kNN) are a simple yet effective solution to the recommendation problem. The backbone of these methods is the estimation of the empirical similarity between users/items. In this paper, we analyze the spectral properties of the Pearson and the cosine similarity estimators, and we use tools from random matrix theory to argue that they suffer from noise and eigenvalues spreading. We argue that, unlike the Pearson correlation, the cosine similarity naturally possesses the desirable property of eigenvalue shrinkage for large eigenvalues. However, due to its zero-mean assumption, it overestimates the largest eigenvalues. We quantify this overestimation and present a simple re-scaling and noise cleaning scheme. This results in better performance of the memory-based methods compared to their vanilla counterparts.

CCS CONCEPTS

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

KEYWORDS

Collaborative filtering; Memory-based methods; Random matrix theory; Cosine similarity; Pearson correlation; Noise reduction.

ACM Reference Format:
Farhan Khawar and Nevin L. Zhang. 2019. Cleaned Similarity for Better Memory-Based Recommenders. 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, 5 pages. https://doi.org/10.1145/3331184.3331310

1 INTRODUCTION

Collaborative Filtering (CF) methods are one type of recommendation techniques that use the *past* interactions of other users to filter items for a single user. Broadly speaking, CF methods are generally characterized into memory-based and model-based methods. Memory-based methods are known for their simplicity and yet having competitive performance. Recently, they have been successfully used for session-based recommendations[4] and they are still used as a part of the recommendation solution in industry[2].

SIGIR '19, July 21–25, 2019, Paris, France

ACM ISBN 978-1-4503-6172-9/19/07…$15.00
https://doi.org/10.1145/3331184.3331310

Memory-based methods like user-kNN and item-kNN extract user (or item) similarities which are used to form user (or item) neighborhoods by taking the k-nearest neighbors. These neighborhoods are then used to filter items for a user.

Calculating the similarity effectively is of great importance in these methods. One of the most commonly used similarity metrics is cosine similarity. Formally, the cosine similarity between two users x and y can be defined as:

$$\sigma = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}}, \tag{1}$$

where, n is the total number of samples (items in this case) and x_i and y_i represent the preferences of user x and user y on the i-th item respectively. The similarity between two items is defined in a similar manner. If the data is centered then the cosine similarity is equivalent to the empirical correlation which is calculated by:

$$\sigma = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n} (y_i - \bar{y})^2}}, \tag{2}$$

where, $\bar{x}$ is the sample mean i.e., $\frac{1}{n}\sum_{i=1}^{n} x_i$, and analogously for $\bar{y}$.

The empirical correlation, and hence the cosine similarity, is a good estimation of the true correlation when the number of samples is large. However, in practice the number of users is of the same order as the number of items and the ratio of the number of users to the number of items is not very small compared to 1. In this case, the empirical correlations are dominated by noise and care should be taken while using them as similarities.

The correlations between users (or items) can be viewed as an empirical correlation matrix where each entry denotes the empirical correlation of the entities represented by its index e.g., the entry at the index $(1, 5)$ of the user empirical correlation matrix would be the correlation between user 1 and user 5. Results from random matrix theory (RMT) can then be used to understand the structure of the eigenvalues and eigenvectors of this empirical correlation matrix. The main contributions of this paper are as follows:

- We analyze the structure and spectral properties of the Pearson and cosine similarity.
- We argue that Cosine similarity possesses the desirable property of eigenvalue shrinkage.
- We quantify the overestimation of cosine similarity's largest eigenvalue.
- We show that the theoretical results regarding the distribution of eigenvalues of random matrices can be used to clean the noise from the empirical user/item correlation matrix.

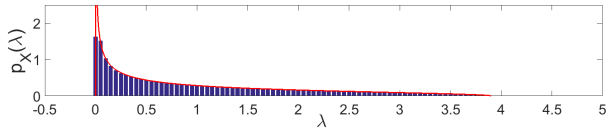

Figure 1: The solid line shows the plot of the MP-law density from Equation 3. The histogram obtained from eigenvalues of a random matrix follows the MP-law distribution.

2 PRELIMINARIES OF RMT

RMT theorems attempt to make statements about the spectral properties of large random correlation matrices. They are applied in the case when an $n \times m$ random matrix $\mathbf{X}$ with independent and identically distributed (i.i.d.) random entries of zero-mean is such that $m, n \to \infty$ and the ratio $m/n \to q \in (0, 1]$.

Interestingly, the eigenvalue distribution of the empirical correlation matrix of $\mathbf{X}$ is known exactly under these conditions and given by the Marčenko Pastur law (MP-law):

$$\rho_{\mathbf{X}}(\lambda) = \frac{1}{2\pi q\lambda}\sqrt{(\lambda_{max} - \lambda)(\lambda - \lambda_{min})}, \tag{3}$$

where the eigenvalue $\lambda \in [\lambda_{max}, \lambda_{min}]$ and $\lambda_{max} = (1 + \sqrt{q})^2$ and $\lambda_{min} = (1 - \sqrt{q})^2$.

This result implies that there should be no eigenvalues outside the interval $[\lambda_{max}, \lambda_{min}]$ for a random noise correlation matrix. A plot of the density of Equation 3 is shown in Figure 1 along with the eigenvalue distribution of a random item correlation matrix formed by randomly permuting the entries of each column of a user-item feedback matrix. As we can see the histogram follows the theoretical MP-law distribution quite accurately.

3 CLEANING THE CORRELATION MATRIX

Using the result where a pure noise correlation matrix has an eigenvalue distribution similar to MP-law in the limiting case, we can clean the user (or item) correlation matrix by comparing its empirical eigenvalue distribution with that of the MP-law. If the bulk of the eigenvalues are within the range $[\lambda_{max}, \lambda_{min}]$ and their distribution resembles the MP-law then it is most probably due to noise and can be ignored.

A simple strategy is to remove all eigenvalues between RMT "noise bulk" range i.e., $[\lambda_{min}, \lambda_{max}]$ by setting them to 0, and retaining the rest of the eigenvalues. However, in practice the eigenvalue distribution in the noise bulk range does not follow the MP-law exactly. Therefore, a cutoff point near λ_{max} is used instead of λ_{max}. This cutoff point λ_{cut} is usually searched within a range near λ_{max}. This strategy is known as eigenvalue clipping [1].

3.1 Eigenvalue spreading

The empirical correlation estimator of Equation 2, also known as the Pearson or the sample correlation matrix is a common estimator of the true user or item correlation matrix. When we have a much larger number of datacases compared to the number of features i.e., $q \to 0$ then this estimator approaches the true correlation matrix. However, when the number of datacases and the number of features are of the same order i.e., $q = O(1)$, the MP-law states that the empirical correlation estimate becomes a noisy estimate of the true correlation matrix. This is because if the true correlation matrix is an identity matrix (pure noise) then the distribution of the eigenvalues of the empirical correlation is not a single spike at 1, but rather it is spread out as shown in Figure 1. This spreading out is dependent on q itself and given by the MP-law stated in Equation 3. The spectrum gets more spread out (noisier) as q increases. This tells us that when we have a data sample in the regime $q = O(1)$ then the small eigenvalues are *smaller* and the large eigenvalues are *larger* compared to the corresponding eigenvalues of the true correlation matrix. Therefore, the cleaning strategy should take this into account and shrink the estimated eigenvalues appropriately.

3.2 Zero-mean assumption

The Pearson estimator is more general as it assumes that the data is not-zero mean, which is often the case in practice. However, the data in collaborative filtering are large and sparse, and applying the Pearson correlation estimator on this data would imply making this large user-item matrix $\mathbf{X}$ dense (by removing the mean from each entry of the matrix). This is problematic from both the memory and computational points of view.

The MP-law was stated for the zero-mean data. The Pearson estimator standardizes the data to make it zero-mean, therefore we can use the MP-law results. In this subsection, we show that we can use the findings from MP-law for the case when the data is not zero-mean. This is because any matrix $\mathbf{X}$ can be written as $\tilde{\mathbf{X}} = \mathbf{X} - \mathbf{M}$, where, $\tilde{\mathbf{X}}$ is the demeaned version of $\mathbf{X}$ and $\mathbf{M} = \mathbf{1}_n \times \mathbf{m}$ is an $n \times m$ matrix, where each row is equal to the vector $\mathbf{m}$. Additionally, $\mathbf{m}$ is a $1 \times m$ row vector that contains the column mean of the corresponding columns of $\mathbf{X}$ and $\mathbf{1}_n$ is a $1 \times n$ vector of all 1's. Then we can rewrite the Pearson correlation estimation as:

$$\mathbf{E}_p = \frac{1}{n}\tilde{\mathbf{X}}^T\tilde{\mathbf{X}} = \frac{1}{n}(\mathbf{X}^T\mathbf{X} - \mathbf{M}^T\mathbf{M}), \tag{4}$$

where, w.l.o.g., for simplicity of notation, we assume that data has unit variance. It is trivial to see that $\mathbf{M}^T\mathbf{M}$ is of rank 1 and has one eigenvalue ξ, which is a positive number. We know from the subadditivity property of rank that:

$$rank(\mathbf{X}^T\mathbf{X}) = rank(\tilde{\mathbf{X}}^T\tilde{\mathbf{X}} + \mathbf{M}^T\mathbf{M}) \tag{5}$$

$$\leq rank(\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}) + rank(\mathbf{M}^T\mathbf{M}) \leq N + 1, \tag{6}$$

where, $rank(\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}) = N$ and it can also be shown [3] that since $rank(\mathbf{M}^T\mathbf{M}) = 1$ then:

$$rank(\mathbf{X}^T\mathbf{X}) = rank(\tilde{\mathbf{X}}^T\tilde{\mathbf{X}} + \mathbf{M}^T\mathbf{M}) \geq N - 1, \tag{7}$$

therefore, the rank of the correlation matrix ($\frac{1}{n}\mathbf{X}^T\mathbf{X}$) of data will change by at most 1, if at all, compared with the rank of the correlation matrix of the demeaned data. As we will see next, the eigenvalue ξ is positive and large, so it will *only* affect the top eigenvalues of the correlation matrix of the original data.

In Figure 2 we plot the *difference* in the eigenvalue magnitudes of the user correlation matrices of the original data and the demeaned data for the Movielens1M dataset, where the eigenvalues of both matrices are sorted in the ascending order of magnitude. We can see a huge positive spike at the largest eigenvalue, signifying that the largest eigenvalue of the original data correlation matrix is

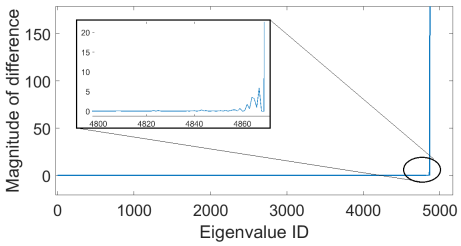

Figure 2: The magnitude of the difference in the corresponding eigenvalues of the original data correlation matrix and demeaned data correlation matrix is shown on the y-axis, against the ID of the eigenvalue on the x-axis.

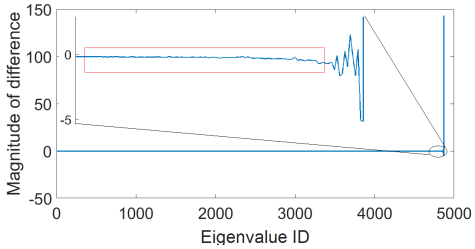

Figure 3: The the magnitude of the difference in the corresponding eigenvalues of the Pearson and Cosine correlation matrices is shown. The negative slope, marked by the red box, signifies the shrinkage property of cosine similarity.

overestimated, and a couple of relatively negligible spikes. From the discussion in the previous subsection, the largest eigenvalue of the demeaned data correlation matrix is already overestimated and the effect of not removing the mean exaggerates it further. Therefore, the effect of not removing the mean from the data is that the largest eigenvalue of the correlation matrix is overestimated.

In the context of recommender systems, where the data are sparse and large, this means that we can still operate on the sparse data matrices if we correct for this overestimation. Moreover, since not demeaning the data effectively just changes the top eigenvalue, we can still use the eigenvalue clipping strategy and other insights based on the MP-law.

3.3 Quantifying the overestimation

Interestingly this overestimation can be quantified by the eigenvalue of $\frac{1}{n}\mathbf{M}^T\mathbf{M}$. The sum of the difference shown in Figure 2 is exactly equal to ξ. This is trivially true since the trace of the data correlation matrix is to be preserved.

We do not need to do the eigenvalue decomposition of $\frac{1}{n}\mathbf{M}^T\mathbf{M}$ to get ξ. This is because, firstly, the eigenvalue of a rank 1 matrix is equal to its trace by the following argument; $\frac{1}{n}\mathbf{M}^T\mathbf{M} = uv^T$ is an $m \times m$ rank 1 matrix, where u, v are $m \times 1$ vectors. Since $m \geq 1$ the matrix is singular and has 0 as its eigenvalue. We know if μ is the eigenvector associated with ξ then $(uv^T)\mu = \xi\mu$. Since $(v^T\mu)/\xi$ is a scalar, u is also an eigenvector associated with ξ. Then, it follows that $u(v^Tu) = \xi u$, and as $u \neq 0$ we have $\xi = (v^Tu) = \sum_{i=1}^{m} v_i u_i = Tr(\frac{1}{n}\mathbf{M}^T\mathbf{M})$. Secondly, the trace of $\frac{1}{n}\mathbf{M}^T\mathbf{M}$ is non-zero by the construction of the matrix $\mathbf{M}$.

The matrix $\frac{1}{n}\mathbf{M}^T\mathbf{M}$ is dense and when m is large calculating this matrix gets unfeasible. However, we notice that we are only interested in the diagonal of the above matrix and not the complete matrix. Therefore, the above trace can efficiently be calculated by:

$$Tr(\frac{1}{n}\mathbf{M}^T\mathbf{M}) = \sum_{i=1}^{m} n\tilde{m}_i^2, \tag{8}$$

where, $\tilde{m}_i = m_i/\sqrt{n}$ and m_i is the i-th element of $\mathbf{m}$. This quantity represents the overestimation in the top eigenvalue of $\mathbf{X}^T\mathbf{X}$.[1]

[1]This can generalize to the case when $\mathbf{X}$ has non-unit variance columns if we divide each column of $\mathbf{X}$ and $\mathbf{M}$ by the standard deviation of the corresponding column of $\mathbf{X}$.

3.4 Eigenvalue shrinkage

Before we outline our cleaning procedure we briefly talk about cosine similarity. Cosine similarity assumes that the data is zero mean, however, this is not true in general. Moreover, it does not make the correction for this by scaling the largest eigenvalue.

However, when we plot the difference in the eigenvalues of the cosine similarity and the Pearson correlation, we find some interesting results. As seen in Figure 3, we have a large spike at the top eigenvalue as before which is expected since cosine similarity does not remove the mean. This is followed by some oscillations, but these oscillations are negative too. This can be due to the difference in variance. Finally, and more importantly, unlike before, the difference between the magnitude of eigenvalues of cosine similarity and Pearson correlation for all the other top eigenvalues is not very close to 0. In fact, we can see a gradual upward slope, in the zoomed-in plot in Figure 3, which was not visible before.

This negative slope signifies that the top eigenvalues of cosine similarity (except the maximum eigenvalue) are shrunk compared to the eigenvalues of the Pearson correlation. Therefore, the cosine similarity implicitly does eigenvalue shrinkage.

The reason for this shrinkage is that the column variances of the data calculated in the Pearson correlation and cosine similarity are not the same. This can been seen from the denominators of Equation 1 and Equation 2. When this is the case we cannot write a simple expression like Equation 4 since both matrices on the right-hand side will have different column variances. Consequently, the effect of not demeaning will be more complex and appear in the form of shrinkage of the top eigenvalues.

3.5 Cleaning algorithm

Below we outline a linear time and memory efficient similarity matrix cleaning strategy that explicitly shrinks the top eigenvalue, inherits the shrinkage property of cosine similarity for other eigenvalues[2] and removes noise by clipping the smaller eigenvalues.

where, "." denotes element-wise operation on vectors and matrices. S_i and V_j denote the i-th and j-th row of the matrices respectively, s_{top} is the largest singular value, λ_{top} is the largest eigenvalue and n_{nz} is the number of non-zeros.

Clean-KNN starts by calculating the mean and sum of each column of $\mathbf{X}$ and then it normalizes $\mathbf{X}$ in line 6 to form $\check{\mathbf{X}}$. This

[2]This shrinkage(both explicit and inherent) is not present in vanilla SVD/PCA.

Algorithm 1 Clean-KNN(X,F)

Inputs: Sparse user-item matrix $\mathbf{X}$,, number of top eigenvalues F

1: **procedure** LEARN ITEM-ITEM SIMILARITY
2: *One-pass over non-zero entries:*
3: Calculate column mean vector $\mathbf{m}$;
4: Calculate column sum vector σ;
5: *One-pass over the non-zero entries* x_{ij} *of* $\mathbf{X}$:
6: $\check{\mathbf{X}} = [x_{ij}/\sigma_j]_{ij}$, divide each x_{ij} by its column sum σ_j to form $\check{\mathbf{X}}$;
7: *Get the top* F *singular value matrix* $\mathbf{S}$ *and right-singular vector matrix* $\mathbf{V}$:
8: $[\mathbf{V}, \mathbf{S}] \leftarrow$ svds($\check{\mathbf{X}}$) via Lanczos algorithm in roughly $O(n_{nz})$ time;
9: *Adjust maximum eigenvalue*:
10: $\mathbf{m} \leftarrow \mathbf{m}./(\sigma.\sqrt{n})$;
11: $s^2_{top} \leftarrow s^2_{top} - \sum_{i=1}^{n} n m_i^2$; $\lambda_{top} = \sqrt{s^2_{top}}$;
12: *Get the cleaned, low-dimensional similarity representation:*
13: $\mathbf{S} \leftarrow \mathbf{V} \times (\mathbf{S}.^2)$;
14: For item i and j the similarity/correlation $c_{ij} = S_i \times V_j^T$

is so that $\check{\mathbf{X}}^T\check{\mathbf{X}}$ is equal to cosine similarity matrix of $\mathbf{X}$. Since for real matrices the square of the *singular values* of $\check{\mathbf{X}}$ is equal to the *eigenvalues* of $\check{\mathbf{X}}^T\check{\mathbf{X}}$ while the eigenvectors are the same, Clean-KNN just calculates the right-singular vectors and singular values of $\check{\mathbf{X}}$ in line 8. In line 11 the top eigenvalue is shrunk. Finally, we get the low-dimensional similarity representation in line 13. We note that Clean-KNN($\mathbf{X}^T$,F) can also give us the user-user similarity.

4 EXPERIMENTS

We aim to answer the following questions via quantitative evaluation: i) Is noise removed by removing the bulk of the eigenvalues? ii) Does the shrinkage of λ_{top} improve performance?

For our experiments we used Movielens1M dataset[3](ML1M) and converted it to implicit feedback by ignoring the rating magnitudes. We used four evaluation metrics namely, recall@50 (R@50), NDCG@50, AUC and diversity@50 (D@50). D@N is the total number of distinct items in the top-N list across all users.

4.1 Baselines and Parameters

Weighted user-KNN (WUkNN) and weighted item-KNN (WIkNN) were used as the base recommenders, with the similarity function defined by Equation 1. We also compare our performance with a well know item recommender SLIM[5], and the vanilla SVD recommender which used the same number of factors F as Clean-KNN. We performed 5-fold cross-validation to select the parameters. We searched for λ_{cut} by incrementing F by 10 when $10 \leq F \leq 100$ and in increments of 100 afterwards till we reach close to λ_{max}.

5 RESULTS

The results are shown in Table 1. We note that Clean-KNN improves the performs over the vanilla kNN. We also see that it is better than vanilla SVD with the same number of factors.

5.1 Is noise removed?

For both datasets, the table is divided into subsections by dashed horizontal lines. In each subsection we want to highlight two scenarios: (a) the best base KNN recommender, and (b) the noise removed Clean-KNN recommender of Algorithm 1. We can see that the performance of the scenario (b) is better than scenario (a). This signifies

[3]https://grouplens.org/datasets/movielens/1m/

Table 1: Performance of Clean-KNN w.r.t. four metrics shows that it outperforms its vanilla counterparts.

Movielens1M	NDCG@50	AUC	R@50	D@50
(a)WUKNN(k = 500)	0.345	0.905	0.346	661
(b)**Clean-UKNN**(k = 500, F = 400)	0.361	0.912	0.364	761
(c)Shrink-UKNN(k = 500)	0.358	0.911	0.361	720
(a)WIKNN(k = 500)	0.356	0.912	0.355	1668
(b)**Clean-IKNN**(k = 500, F = 400)	0.368	0.919	0.378	2187
(c)Shrink-IKNN(k = 500)	0.369	0.917	0.368	1730
SVD(F = 400)	0.236	0.770	0.248	2242
SLIM($L_1 = 10^{-2}$, $L_2 = 10^{-3}$, k = 500)	0.293	0.882	0.300	534

that most of the removed eigenvalues did not carry much useful information and hence can be categorized as noise.

5.2 Does shrinkage of λ_{top} help?

To answer this question we have to compare a base user or item-KNN recommender with a recommender that contains all the eigenvalues but shrinks the top eigenvalue according to Equation 8. Note, that this recommender is created for illustration of the effectiveness of the shrinkage procedure. The performance of this recommender is shown in Table 1 and labeled as (c). We see that the performance of the scenario (c) is always better than scenario (a). This confirms that just by shrinking λ_{top} we get improved performance. In addition, scenario (c) is still outperformed by scenario (b), thus this confirms the utility of the clipping strategy.

6 CONCLUSION

Memory-based recommenders are one of the earliest recommendation techniques which are still being deployed in the industry. In this paper, we analyzed the spectral properties of the Pearson and cosine similarities, and we used insights from MP-law to show that these empirical similarities suffer from noise and eigenvalue spreading. We showed that the cosine similarity naturally performs the eigenvalue shrinkage but it overestimates λ_{top}. We then provided a linear time and memory efficient cleaning strategy, Clean-KNN, that removes noise and corrects for this overestimation. Empirically, we showed that this cleaning strategy is effective as it results in better accuracy and diversity compared to the vanilla kNN recommenders.

Acknowledgment. Research on this article was supported by Hong Kong Research Grants Council under grant 16202118.

REFERENCES

[1] Jean-Philippe Bouchaud and Marc Potters. 2009. Financial applications of random matrix theory: a short review. *arXiv preprint arXiv:0910.1205* (2009).
[2] Renaud Bourassa. [n. d.]. Personalized channel recommendations in Slack. https://slack.engineering/personalized-channel-recommendations-in-slack-3077f10a5e81(version: 2019-02-19).
[3] Amitai Yuval (https://math.stackexchange.com/users/166201/amitai yuval). [n. d.]. The rank after adding a rank-one matrix to a full rank matrix. Mathematics Stack Exchange. https://math.stackexchange.com/q/944751 (version: 2014-09-24).
[4] Dietmar Jannach and Malte Ludewig. 2017. When Recurrent Neural Networks Meet the Neighborhood for Session-Based Recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems (RecSys '17)*. ACM, New York, NY, USA, 306–310. https://doi.org/10.1145/3109859.3109872
[5] 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.