Article

Improvising Singular Value Decomposition by KNN for Use in Movie Recommender Systems

Journal of Operations
and Strategic Planning
2(1) 22–34, 2019
© 2019 International Management
Institute, Kolkata
SAGE Publications
sagepub.in/home.nav
DOI: 10.1177/2516600X19848956
http://journals.sagepub.com/home/osp

Sukanya Patra[1]
Boudhayan Ganguly[2]

Abstract

Online recommender systems are an integral part of e-commerce. There are a plethora of algorithms following different approaches. However, most of the approaches except the singular value decomposition (SVD), do not provide any insight into the underlying patterns/concepts used in item rating. SVD used underlying features of movies but are computationally resource-heavy and performs poorly when there is data sparsity. In this article, we perform a comparative study among several pre-processing algorithms on SVD. In the experiments, we have used the MovieLens 1M dataset to compare the performance of these algorithms. KNN-based approach was used to find out *K*-nearest neighbors of users and their ratings were then used to impute the missing values. Experiments were conducted using different distance measures, such as Jaccard and Euclidian. We found that when the missing values were imputed using the mean of similar users and the distance measure was Euclidean, the KNN-based (K-Nearest Neighbour) approach of pre-processing the SVD was performing the best. Based on our comparative study, data managers can choose to employ the algorithm best suited for their business.

Keywords

Recommender systems, e-commerce, KNN (K-Nearest Neighbour), singular value decomposition, Jaccard distance

[1] Department of Information Technology, Government College of Engineering and Leather Technology, Kolkata, West Bengal, India.
[2] Operations and Technology Management Area, International Management Institute, Kolkata, West Bengal, India.

Corresponding author:
Boudhayan Ganguly, Operations and Technology Management Area, International Management Institute, 2/4C, Judges Court Road, Alipore, Kolkata 700027, West Bengal, India.
E-mail: b.ganguly@imi-k.edu.in

Introduction

A recommender system belongs to a category of information filtering mechanism in which a user can predict the liking of an item, which could be a product or service. The outcome of the recommender system is usually a score or preference that the user might give to that item. According to Konstan and Riedl (2012), the modern-day recommender systems interact directly with the users to find content, products, and services by aggregating and analyzing ratings from other users, mean reviews from various experts and users.

According to Ansari, Essegaier, and Kohli (2000), in the context of marketing, recommender systems can act as a powerful promotional and decision-making tool. Further, in the current era of handheld devices and mobile apps, right promotional strategies can be directed towards matching users. Smith (2011) noted that such steps can increase sales of products and improve customer retention among fast-moving items. Recommender systems can also help companies in planning adaptive television programs, personalized online show selection, and product offerings. Thus, recommender systems are used to personalize the online store for each customer.

From the computational perspective, recommendation systems may face multiple challenges. For example, when a new customer joins an e-commerce site for the first time, there is no previous information available about that customer. This is known as the cold start problem. Here some recommender systems such as user-based collaborative filtering fails. Sometimes there is huge information for existing customers according to their ratings and purchases. Size of their dataset drastically increases as customer spend significant time in the system. It leads to increased memory requirement and computation time thus making recommendation difficult. Thus, the field of e-commerce has witnessed the emergence of many kinds of recommender systems trying to address these issues each with its own strengths and weaknesses.

On and above this, one serious drawback of the aforementioned approaches is the fact that they do not give insights into the rating patterns. For example, after looking into several ratings from a user one cannot get an understanding on what might have motivated the user to give such ratings. We believe that the singular value decomposition (SVD) of the user rating matrix will give us insights into such underlying concepts or patterns. However, the SVD does suffer from a drawback that it does not perform well when there are missing values in the rating matrix. To overcome this we have pre-processed the data. At first we have used KNN algorithm to identify the *K*-nearest neighbors of a given user and then based on their ratings we have filled up the missing rating values of the user in question. This article is divided into five sections. After the introduction we have presented the extant literature in the second section followed by methodology and results. In the last section we have provided conclusions and directions for future research.

Literature Review

Most of prevalent collaborative filtering systems are built based on neighborhood of likeminded customers by using, generally, Pearson correlation or cosine similarity as a measure of proximity (Resnick, Iacovou, Suchak, Bergstrom, & Riedl, 1994; Sarwar, Karypis, Konstan, & Riedl, 2000; Shardanand & Maes, 1995). Two types of recommendation emerge out of these proximity calculations. The first set of approaches try predicting how much a customer likes a product depending on the co-rated items between the customer and the calculated neighbors. Nevertheless, some non-personalized prediction schemes are also there like computing the average ratings based on all users rating (Herlocker, Konstan, Terveen, & Riedl, 2004). The other approach use top-N recommendation generation, that is, recommending a list of "N" products for the customer based on the neighbors, the algorithm selects the products which the customers are most likely to consider.

In spite of getting success in several domains, they have shown some severe limitations like that of sparsity. Nearest neighbor algorithms tend to lose coverage and accuracy as they depend upon exact matches (Konstan et al., 1997). Billsus and Pazzani (1998) noted that since correlation coefficient can be computed only for the users who have rated minimum of two same products, many pairs of users are left out. In a real life situation, a large product set is used in which active users may have rated a very few number of products which sometimes results to reduced convergence. In this case Pearson nearest neighbor algorithm may fail to predict recommendation for a specific user.

Moreover, the accuracy is compromised due to the presence of fairly less ratings data. Besides, most of the recommendation systems suffer from serious scalability issues as the computation of nearest neighbors grow in dimension with the increase in either customer or product.

Recommender systems in real world deals with a large amount of sparsity or unobserved value. Since zero values have certain meaning in case of rating matrix, estimating the missing values to zero provides wrong predictions. Thus, an alternative approach would be to use some prediction algorithm for imputation of missing values. The most common approach is to fill up those values with column or row means. The full matrix thus obtained can be then used for SVD calculation although an efficient way would be to re-center the matrix before proceeding with the SVD calculations (Rendle & Schmidt-Thieme, 2008). However, as mean ignores the underlying correlation of data, it sacrifices accuracy (Schmitt, Mandel, & Guedj, 2015).

The objectives of the study is twofold.

First of all, we propose a method for improving the performance of SVD matrix in the context of recommender systems. This is done by using the KNN algorithm for classification.

Our second objective is to test the approach for a range of values of K for which the performance of the improvised method is significantly better than the normal SVD.

Methodology

Data Source

We used the movie ratings dataset MovieLens 1M available at MovieLens, which is also available for public use. To keep the recommender consistent with time and minimum code changes, we used the standard 1 million dataset. It contains 1,000,209 anonymous movie ratings tagged by 6,040 users across 3,900 movies in the year 2000. The user-id ranges between 1 and 6040 while the movie-id ranges between 1 and 3952. The ratings are made on integral values only with a 5-star Likert-scale. Data is available in four files: links.csv, movies.csv, ratings.csv, and tags.csv. We used movies.csv and ratings.csv to execute the recommendation algorithms in this article.

Machine Configuration

The experiments were carried out on a 64-bit Windows 10 machine with 8 GB RAM, and Intel Core i3 4010U processor with 1.70 GHz clock speed. Additionally, we recorded the performance of the algorithms and reported their execution time for performance comparison.

Data Pre-processing

The main aim was to improve the performance of SVD algorithm implemented in recommender lab package in terms of time, precision, and accuracy. For this purpose, first we calculated two distance matrices using two different distance metrics namely, Jaccard and Euclidean.

The Jaccard similarity index (also known as the Jaccard similarity coefficient) is used to compare members of two sets to find distinct and shared members in them. Its value ranges from 0% to 100% and signifies the similarity between the two sets under consideration. It is calculated by

$$Jaccard\ Index = \frac{(the\ number\ of\ shared\ members)}{(total\ number\ of\ members)} \times 100$$

In notation, it is denoted as

$$J(X,Y) = \frac{|X \cap Y|}{|X \cup Y|}$$

Jaccard distance on the other hand measures the degree of dissimilarity between two sets. It is the complement of Jaccard Index. In set notation, it is expressed as

$$D(X,Y) = 1 - J(X,Y)$$

Euclidean distance or Euclidean metric is measured as the distance between two points in the Euclidean space. Therefore, if $p = (p_1, p_2, \ldots, p_n)$ and $q = (q_1, q_2, \ldots, q_n)$ are two points in Euclidean n-space in Cartesian coordinates, the distance (d) between them is given by Pythagorean formula,

$$d(p,q) = d(q,p) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \ldots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

Second, the distance matrices were used to find *K*-nearest neighbors using the KNN algorithm from the fast KNN library. A range of values of *K* was used to test the performance of the algorithm. At first *K* was set to 10, then 20 and finally to 50. In each case, the information of the nearest neighbors were utilized to remove the sparsity of the rating matrix to some extent. For this purpose, mean and mode of the ratings given by the nearest neighbors were used to fill up the empty spaces in the original rating matrix resulting in total four separate rating matrices. In the first two rating matrices the nearest neighbors were calculated using the distance matrix in which Jaccard distance was employed as the distance metric and mean and mode were applied to them, respectively. Similarly, in the next two rating matrices the nearest neighbors were calculated using the distance matrix in which Euclidean distance was employed as the distance metric and mean and mode were applied to them, respectively. SVD Funk algorithm from the recommender library was used separately on the four rating matrices. In this study, we performed a 5-fold cross-validation, which means that out of five subsamples, four are used for training the model, and the remaining one is retained for validation. The entire process is repeated five times, or equal to the number of folds defined in the model.

Table 1. Data Pre-processing for SVD algorithm

#Converting to binary scale for Jaccard
for each row in result do
for each column in result do
if result[row, column] greater than 0 then
result[row, column] = 1
else
result[row, column] = 0
end for
end for
	Removing sparsity in the rating Rating Matrix
	createMatrix(nearestNeighbour[userID, noOfNearestNehigbours])
	# Reducing sparsity using Jaccard distance and mode
	similarityMatrix = createSimilaritymatrix(result, method="jaccard")
	for each userID do
	nearestNeighbour[userID,] = findKNearestNeighbour(k, similarityMatrix)
	end for
	for each row in ratings do
	for each column in ratings do
	if ratings[row, column is empty] then
	ratings[row, column] = mode(ratings[nearest Neighbour[row,]])
	end if
	end for
	end for

(Table 1 Continued)

(Table 1 Continued)

```
# Reducing sparsity using Euclidean distance and mean
similarityMatrix = createSimilaritymatrix(result, method="euclidean")
for each userID do
    nearestNeighbour[userID, ] = findKNearestNeighbour(k, similarityMatrix)
end for
for each row in ratings do
    for each column in ratings do
        if ratings[row, column is empty] then
            ratings[row, column] = mean(ratings[nearestNeighbour[row, ]])
        end if
    end for
end for
```

```
# Reducing sparsity using Euclidean distance and mode
similarityMatrix = createSimilaritymatrix(result, method="euclidean")
for each userID do
    nearestNeighbour[userID, ] = findKNearestNeighbour(k, similarityMatrix)
end for
for each row in ratings do
    for each column in ratings do
        if ratings[row, column is empty] then
            ratings[row, column] = mode(ratings[nearest Neighbour[row, ]])
        end if
    end for
end for
```

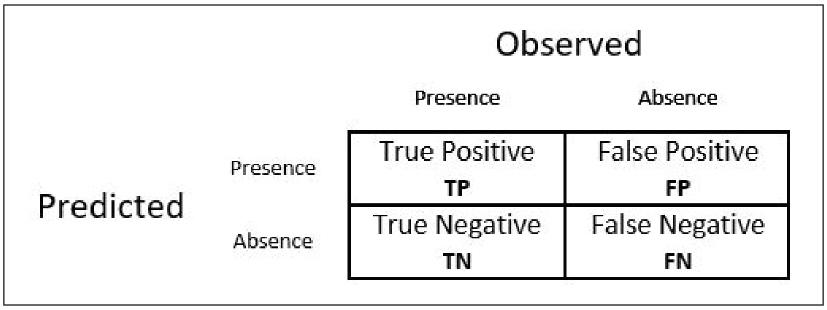
```
# Reducing sparsity using Euclidean distance and median
similarityMatrix = createSimilaritymatrix(result, method="euclidean")
for each userID do
    nearestNeighbour[userID, ] = findKNearestNeighbour(k, similarityMatrix)
end for
for each row in ratings do
    for each column in ratings do
        if ratings[row, column is empty] then
            ratings[row, column] = median(ratings [nearestNeighbour[row, ]])
        end if
    end for
end for
```

Source: Data Pre-processing for SVD algorithm.

Empirical Findings

We explored the recommender data for a count of scores in each rating category, average score per user, average score per item, and a combined average of the ratings. The median score is 3.744 across 6,040 users and 3.33 across 3,900 movies.

After pre-processing of data with Jaccard distance and Euclidian distance the data in terms of rating matrix becomes in a new format which is shown below.



		Observed	
		Presence	Absence
Predicted	Presence	True Positive **TP**	False Positive **FP**
	Absence	True Negative **TN**	False Negative **FN**

Figure 1. Confusion Matrix for the Recommendation Problem

Source: The authors.

A confusion matrix is a specific table that allows visualization of the performance of a machine-learning algorithm, typically a supervised one and often in the problem of statistical classification. Each row represents the instances in an actual class while each column of the matrix represents the instances in a predicted class (or vice versa). The confusion matrix is also known as an error matrix.

$$TPR = \frac{TP}{TP + FN}$$

True-positive rate (TPR) is also known as sensitivity or probability of detection in machine learning.

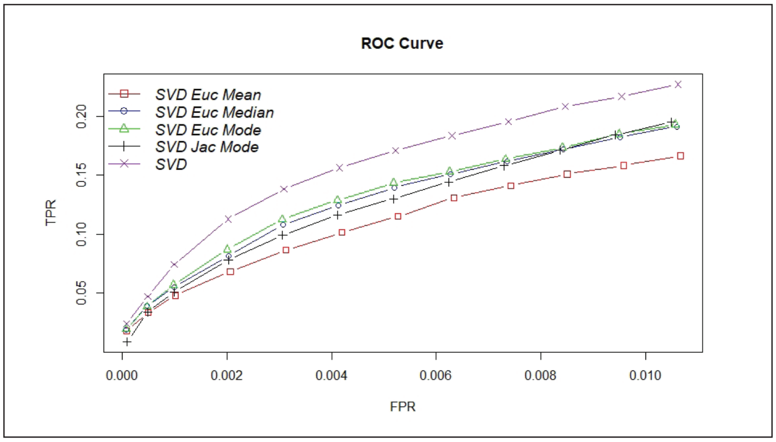


Figure 2a. ROC Curve for $K = 10$

Source: The authors.

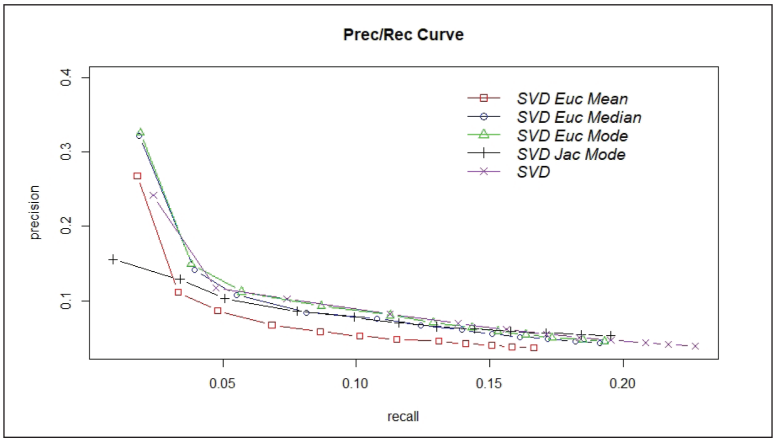


Figure 2b. Precision Recall Curve for *K* = 10

Source: The authors.

$$FPR = \frac{FP}{FP + TN}$$

The false-positive rate (FPR) is also known as the fall-out or probability of false alarm.

The diagnostic ability of a binary classifier system, as its discrimination threshold is varied, is illustrated by a graphical plot in statistics known as a receiver operating characteristic (ROC) curve . At various threshold settings the plot of the TPR against the FPR creates the ROC curve (Bobadilla, Ortega, Hernando, & Gutiérrez, 2013). Thus the ROC curve is the sensitivity as a function of fall-out.

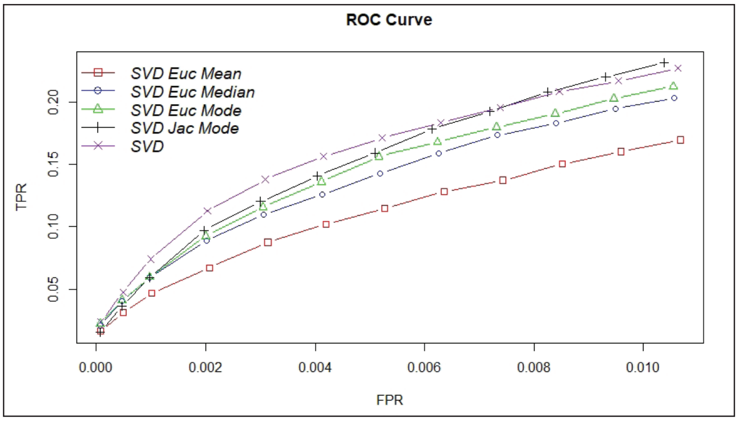


Figure 3a. ROC Curve for *K* = 20

Source: The authors.

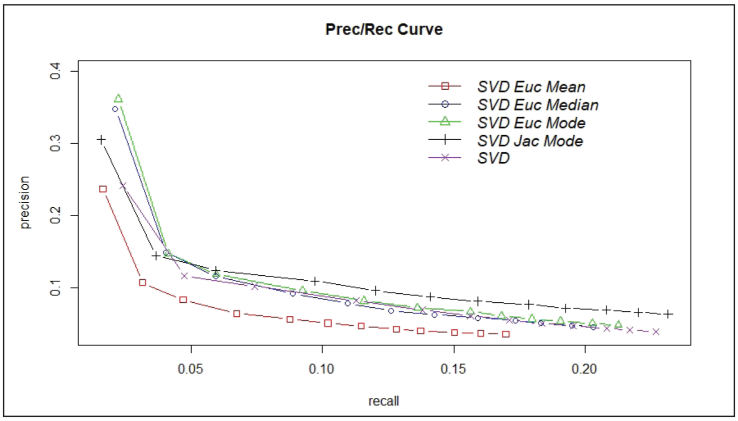


Figure 3b. Precision Recall Curve for *K* = 20

Source: The authors.

We plot the ROC curve in with the area under the ROC curve serving as our performance indicator.

Our results suggest that that "SVD with Jaccard Mode" algorithm performs best in terms of correct movie prediction using user-based similarity. The "SVD with Ecludian Mean" algorithm is also close to it in terms of performance. Thus, from our study it is clear that with improvisation such as pre-processing the user rating matrix with filling up of missing values help us in significantly improving the performance of the SVD algorithm. The performance in terms of accuracy and precision is clearly better with larger values of *K*. So, we believe that with larger datasets the pre-processing of SVD would give even better results. From Table 2 it is also clear that as we increase *K* the time taken by the algorithm increases only minimally.

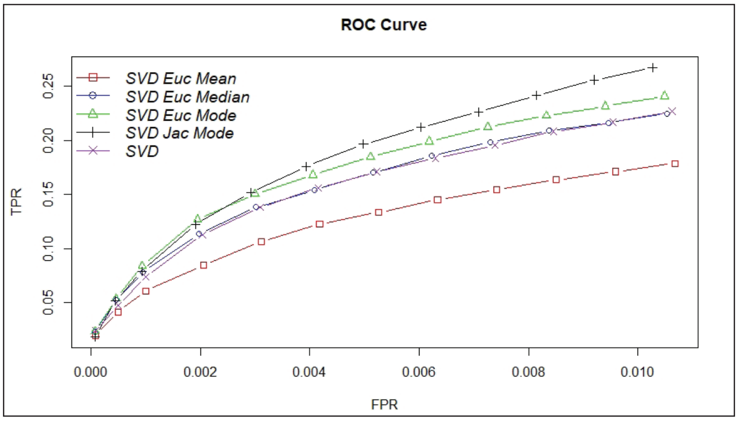


Figure 4a. ROC Curve for *K* = 50

Source: The authors.

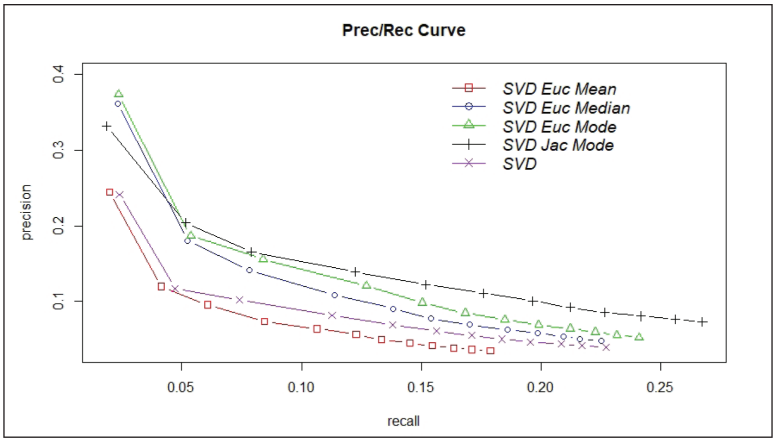


Figure 4b. Precision Recall Curve for *K* = 50

Source: The authors.

Table 2. Performance of the KNN-based Methods of Data Imputation in SVD with Different Values of *K*

Algorithm	*K*	SVDF-run Fold/Sample [Model Time/Prediction Time]
SVD	NA	1 [174.09s/23.24s]
		2 [267.79s/23.46s]
		3 [199.23s/23.11s]
		4 [226.81s/24.65s]
		5 [230.99s/23.35s]
SVD using Euclidian mean	10	1 [301.86s/25.13s]
		2 [238.91s/23.9s]
		3 [271.67s/23.29s]
		4 [224.08s/24.09s]
		5 [216.18s/23.58s]
SVD using Euclidian median	10	1 [287.29s/22.48s]
		2 [241.82s/22.49s]
		3 [306.87s/22.45s]
		4 [297.07s/22.28s]
		5 [262.76s/22.14s]
SVD using Euclidian mode	10	1 [331.88s/23.03s]
		2 [320.89s/22.82s]
		3 [327.8s/22.7s]
		4 [327.64s/22.69s]
		5 [304.04s/24.99s]
SVD using Jaccard mode	10	1 [282.77s/23.5s]
		2 [281.94s/31.8s]
		3 [283.97s/26.7s]
		4 [291.95s/23.78s]
		5 [289.25s/23.67s]

(Table 2 Continued)

(Table 2 Continued)

Algorithm	***K***	**SVDF-run Fold/Sample [Model Time/Prediction Time]**
SVD using Euclidian mean	20	1 [290.3s/24.21s]
		2 [338.76s/23.87s]
		3 [339.34s/23.54s]
		4 [338.6s/23.22s]
		5 [268.96s/23.35s]
SVD using Euclidian median	20	1 [264.68s/23.6s]
		2 [297.14s/23.78s]
		3 [302.3s/22.12s]
		4 [301.35s/23.48s]
		5 [234.84s/23.63s]
SVD using Euclidian mode	20	1 [344.11s/24.07s]
		2 [330.38s/23.94s]
		3 [344.12s/24.71s]
		4 [348.54s/22.55s]
		5 [314.8s/23.03s]
SVD using Jaccard mode	20	1 [311.92s/26.85s]
		2 [289.4s/26.09s]
		3 [283.58s/24.22s]
		4 [289.66s/25.38s]
		5 [314.64s/23.97s]
SVD using Euclidian mean	50	1 [326.95s/25.19s]
		2 [350.32s/25.12s]
		3 [354.43s/24.73s]
		4 [247.5s/24.2s]
		5 [274.14s/25.04s]
SVD using Euclidian median	50	1 [264.56s/23.42s]
		2 [278.54s/23.46s]
		3 [289.24s/23.02s]
		4 [195.92s/23.36s]
		5 [221.67s/23.53s]
SVD using Euclidian mode	50	1 [291.04s/23.08s]
		2 [292s/25.53s]
		3 [312.02s/23.06s]
		4 [291.66s/23.05s]
		5 [266.2s/22.89s]
SVD using Jaccard mode	50	1 [314.39s/23.45s]
		2 [313.48s/24.61s]
		3 [314.17s/23.33s]
		4 [243.12s/23.22s]
		5 [321.14s/23.17s]

Source: The authors.

Managerial Implications and Scope of Future Work

There has been an increase in social media relationship-based research studies that rely heavily on recommender systems. There has also been an increase in development of hybrid techniques which combine different aspects of various types of recommender systems to achieve results that are best suited for different domains. We believe that our reliance on recommender systems shall increase in the years to come. One of the drawbacks of typical recommender system based on user-based collaborations is that they fail to provide insights into the motivation of users for giving movie ratings. The SVD of the rating matrix does help us to identify some latent pattern in the ratings. The first matrix gives us the information about how the users have liked the underlying themes/concepts of the items (movies in our study) that they have rated. The second matrix tells us the strength of the concepts and the third provides us insights on how the concepts are linked with the items/movies. Thus, SVD-based approaches are better suited for recommendation when one's objective is to get insights into the ratings.

However, one drawback of SVD is that of imputing missing value. There have been a number of studies that have tried to address this issue but each of them had their own drawbacks. This article addressed this issue partly by filling the missing values based on what *K*-nearest neighbors of the user have done. The neighbors were identified using the KNN approach. Our approach has given better results compared to the SVD with imputation of user's average.

We further note that SVD is a "resource hungry" algorithm. However, with a believe that with time computational resources will become cheaper and with some improvisations the popularity of SVD will also increase along with those algorithms that are more resource heavy.

Declaration of Conflicting Interests

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

References

Ansari, A., Essegaier, A., & Kohli, R. (2000). Internet recommendation systems. *Journal of Marketing Research*, *37*(3), 363–375.

Billsus, D., & Pazzani, M. J. (1998). Learning collaborative information filters. In *Proceedings of international conference on machine learning* (pp. 46–54). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-Based Systems*, *46*, 109–132.

Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, *22*(1), 5–53.

Konstan, J. A., & Riedl, J. (2012). Recommender systems: From algorithms to user experience. *User Modeling and User-Adapted Interaction, 22*(1), 101–123.

Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., & Riedl, J. (1997). Grouplens: Applying collaborative filtering to Usenet news. *Communications of the ACM*, *40*(3), 77–87.

Rendle, S., & Schmidt-Thieme, L. (2008). Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *Proceedings of the 2008 ACM conference on recommender systems* (pp. 251–258). Lausanne, Switzerland: ACM.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994). Grouplens: An open architecture for collaborative filtering of Netnews. In *Proceedings of the 1994 ACM conference on computer supported cooperative work* (pp. 175–186). New York, NY: ACM.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2000). Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2nd ACM conference on electronic commerce* (pp. 158–167). Minneapolis, Minnesota.

Schmitt, P., Mandel, J., & Guedj, M. (2015). A comparison of six methods for missing data imputation. *Journal of Biometrics and Biostatistics*, *6*(1), 1–6.

Shardanand, U., & Maes, P. (1995). Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of CHI '95* (pp. 210-217). Denver, CO.

Smith, K. T. (2011). Digital marketing strategies that Millennials find appealing, motivating, or just annoying. *Journal of Strategic Marketing*, *19*(6), 489–499.