Singular Value Decomposition on Sparse Dating Service Dataset

Zhihua Gu
University of California, San Diego
z4gu@ucsd.edu
A99110780

Tian Yang
University of California, San Diego
tiyang@ucsd.edu
A99109363

ABSTRACT

With the increasing demand to optimize online social network user experience, recommender systems have grown in variety and effectiveness. In this paper, we attempt to evaluate algorithms for a recommender system used to predict user ratings in a large data set taken from a online dating website. We analyzed some basic properties of the data set, and researched different approaches to predict ratings including global, memory-based collaborative filtering and Singular Value Decomposition algorithms. Results show that Singular Value Decomposition with Stochastic Gradient Descent significantly outperform others when the dataset is sparse, and remain a suitable algorithm as the dataset grows denser.

1 INTRODUCTION

1.1 Dataset Used to Study

We obtained a dating agency user rating dataset. The dataset records the numerical ratings each user gives to a picture of another profile. The ratings are contained in a single file in the format of UserID, ProfileID ,Rating where UserID is the user who provided rating, ProfileID is the user who has been rated, and Ratings are integers on a 1–10 scale where 10 indicates the highest rating, and 1 the lowest.

Listing 1: First 5 lines of the dataset

```
1,133,8
1,720,6
1,971,10
1,1095,7
1,1616,10
...
```

The large dataset only includes users who provided at least 20 ratings. It also excludes potentially invalid samples such as users who provide constant ratings. [9]

1.2 Dataset Exploratory Analysis

The dataset contains 17359346 ratings of 168791 profiles (ProfileID s range between 1 and 220970 as not every one are rated) by 135359 distinct users.

Table 1 and Figure 1 contain some properties of the dataset. As we can see in the Figure 1(a), the average rating given by different users roughly resembles normal distribution, and by Figure 1(b), we see that there is a high amount of 1s and 10s given, while the rest ranges around 5–8. These figures indicate that even though there is a large amount of low and high ratings given (1 or 10), most users still have an average rating around the 5–6. This further shows that people tend to agree on the extreme cases, which establishes our assumption that there are user preference patterns that could be traced through similarities among users.

Table 1: Data statistics on the entire dataset

Number of ratings	17359346
Number of users	135359
Number of profiles that have been rated	168791
The most amount of ratings provided by a user	25042
The least amount of ratings provided by a user	20
The average amount of ratings provided by a user	128.247
The most amount of ratings towards a profile	33389
The least amount of ratings towards a profile	1
The average amount of ratings towards a profile	102.845
Density of data	0.000290
Global average rating	5.938

We also found that despite the fact that the average rating provided by a user and the average rating received by a profile are over 100, as show in Figure 1(c), and Figure 1(d), the overall matrix is very sparse with density only 0.0002901903129504924.

2 RELATED LITERATURE

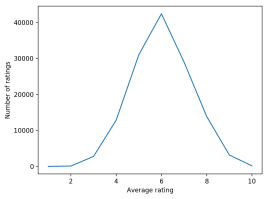
We obtained the dataset from a website operated by Vaclav Petricek, a researcher that is affiliated with the Charles University in Prague, Czech Republic. The dataset was dumped from LibimSeTi, a Czech dating site, on April 4, 2006 and anonymized. [9]

In a paper published in 2007 [3], Lukas Brozovsky, along with Vaclav Petricek, noting the lack of literature on the application of recommender systems for dating preferences, performed a quantitative comparison of two global algorithms and two collaborative filtering algorithm run on a July 2005 version of the same dataset (which we were not able to obtain). Specifically, they implemented Random Algorithm, Mean Algorithm [2], User-User algorithm [5], and Item-Item algorithm [12]. In evaluating their results, they set up different benchmarks such as AllButOne Validation, GivenRandomX Validation, and Production Validation with k-sized block insertion. They also focused on tuning the parameters k and min_k [3].

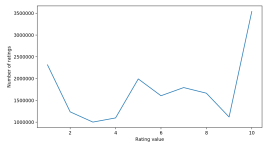
They found that the aforementioned User-User and Item-Item collaborative filtering algorithms significantly outperform the Random and Mean global algorithms which, according to the authors, were apparently used by dating sites at the time [3]. Their conclusion is further verified by conducting a social experiment on 150 subjects who preferred collaborative filtering results to global popularity algorithm.

Other online dating recommendation system research went in a different direction that could improve user experience. In a paper [13], the authors set out to determine the "reciprocal score" of 2 users to match user with another user who are mutually interested in each other as recommendations. They used precision

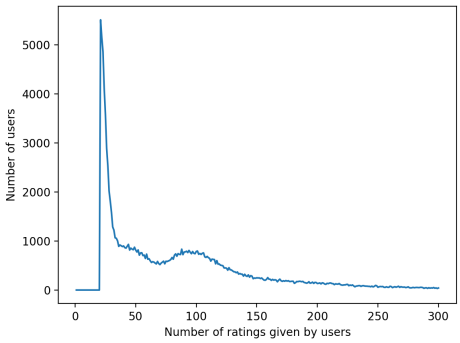
Figure 1: Statistics graphs on the entire dataset



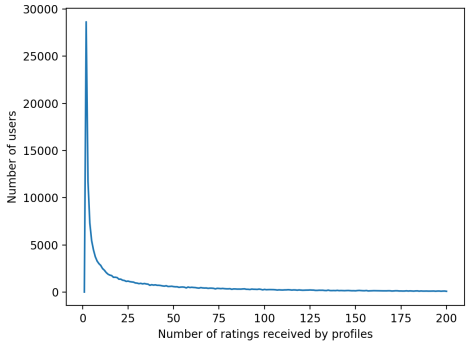
(a) Distribution of average ratings among different users



(b) Distribution of rating values in the whole dataset



(c) Distribution of number of ratings given by a single user



(d) Distribution of number of ratings received by a single profile

and recall as their evaluation metric, and concluded that collaborative filtering-algorithm performed better than content-based algorithm. Although reciprocal matching algorithm would be useful, we cannot validate the effectiveness for such an algorithm with our anonymized dataset which only contains one-sided ratings.

3 PREDICTIVE TASK

As we learned in class, User-User and Item-Item algorithms are slow given a huge enough dataset and they are of no use for new users and new items. We would like to perform an analysis of the Singular Value Decomposition algorithm (of the Netflix Prize fame) [8] on this sparse dating preference dataset. We would also like to compare it with the aforementioned collaborative filtering algorithms which are used in [3].

We were unable to obtain the source code of ColFi recommender system used in [3], so we implemented our version in Python with Python packages Surprise [6], Pandas [10], Matplotlib [7], and Cython [1].

We load the dataset with Pandas. The dataset was split in half as separate training and validation datasets. For each algorithms we train with the training set and use the trained model to predict ratings in the validation set.

We initially encountered out of RAM problem when calculating Pearson similarities due to the large number of ratings available and the lack of a complex recommender system infrastructure like ColFi, so we chose to randomly sample 100000, 150000, 200000, 250000, and 300000 ratings independently from the dataset and use these to perform our analysis.

The sampling resulted in very sparse datasets. Statistics of these datasets are shown in Table 2.

4 MODEL SELECTION

The algorithm we chose are Singular Value Decomposition, as well as User-User and Item-Item as described in [3]. We use global mean as our baseline.

4.1 Mean

As a baseline algorithm, the Mean algorithm always returns the global mean as prediction.

$$\hat{r_{ui}} = \frac{\sum_{k=1}^{N} r_k}{N}$$

where N = the total number of ratings.

4.2 User-User

As one of the most well-know collaborative filtering methods, User-User algorithm predicts the rating that user u will give profile i, $\hat{r_{ui}}$, by computing the similarity between user u and its k nearest neighbors who also rated profile i. We then use these similar users' preference for the profile i to predict the rating $\hat{r_{ui}}$.

The User-User algorithm we use is using the Pearson's correlation coefficient [11] for similarity since our dataset contains numerical ratings:

$$\text{pearson\_sim}(u, v) = \frac{\sum_{i \in I_{uv}} (r_{ui} - \mu_u) \cdot (r_{vi} - \mu_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \mu_u)^2} \cdot \sqrt{\sum_{i \in I_{uv}} (r_{vi} - \mu_v)^2}}$$

and the prediction is set as:

$$\hat{r_{ui}} = \frac{\sum_{v \in N_i^k(u)} \text{pearson\_sim}(u, v) \cdot (r_v i - \mu_v)}{\sum_{v \in N_i^k(u)} \text{pearson\_sim}(u, v)}$$

where $N_i^k(u)$ is the k nearest neighbors of user u that have rated item i.

4.3 Item-Item

Similarly to User-User algorithm, the Item-Item algorithm follows the k-NN approach, and computes the similarity between profile i

Table 2: Data statistics on the different sized dataset

Number of ratings	100000	150000	200000	250000	300000
Number of users	50582	64565	75009	83337	89973
Number of profiles that have been rated	38844	47922	54960	60561	65302
The most # of ratings provided by a user	142	218	280	362	435
The least # of ratings provided by a user	1	1	1	1	1
The average # of ratings provided by a user	1.977	2.323	2.666	3.000	3.334
The most # of ratings towards a profile	195	289	380	483	582
The least # of ratings towards a profile	1	1	1	1	1
The average # of ratings towards a profile	2.574	3.130	3.639	4.128	4.594
Density of data	3.343×10^{-6}	5.015×10^{-6}	6.687×10^{-6}	8.358×10^{-6}	10.030×10^{-6}
Global average rating	5.910	5.916	5.919	5.923	5.927

and its k nearest neighbors who were also rated by user u. We then use the preference that the user u had for the similar profiles' to predict the rating $\hat{r_{ui}}$.

The Item-Item algorithm we use is also using the Pearson's correlation coefficient for similarity:

$$\text{pearson\_sim}(i,j) = \frac{\sum_{u \in U_{ij}} (r_{ui} - \mu_i) \cdot (r_{uj} - \mu_j)}{\sqrt{\sum_{u \in U_{ij}} (r_{ui} - \mu_i)^2} \cdot \sqrt{\sum_{u \in U_{ij}} (r_{vj} - \mu_j)^2}}$$

and the prediction is set as:

$$\hat{r_{ui}} = \frac{\sum_{j \in N_u^k(i)} \text{pearson\_sim}(i,j) \cdot (r_u i - \mu_j)}{\sum_{j \in N_u^k(i)} \text{pearson\_sim}(i,j)}$$

where $N_u^k(i)$ is the k nearest neighbors of itmes i that have rated user u.

Parameters used for User-User and Item-Item:

Our User-User algorithms implementation requires a minimum of 10 common items rated by both user u and its neighbor v to calculate similarity(u, v), while the Item-Item algorithm requires minimum of 10 common users rating both item i and its neighbor j to calculate similarity(i, j). We also set $k = 50$, so a maximum of 50 neighbors is used during the computation. This choice is supported by [3], which stated that "further experiments did confirm that more than 50 neighbors do not significantly improve the performance".

4.4 Singular Value Decomposition with Stochastic Gradient Descent

For the Singular Value Decomposition algorithm, the prediction is as follows:

$$\hat{r_{ui}} = \mu + b_u + b_i + q_i^T p_u$$

The model is trained using Stochastic Gradient Descent on the regularized squared error objective since we only have a partially observed matrix:

$$\sum_{r_{ui} \in R_{train}} (r_{ui} - \hat{r_{ui}})^2 + \lambda(b_i^2 + b_u^2 + ||q_i||^2 + ||p_u||^2)$$

$$b_u \leftarrow b_u + \gamma((r_{ui} - \hat{r_{ui}}) - \lambda b_u)$$
$$b_i \leftarrow b_i + \gamma((r_{ui} - \hat{r_{ui}}) - \lambda b_i)$$
$$p_u \leftarrow p_u + \gamma((r_{ui} - \hat{r_{ui}}) \cdot q_i - \lambda p_u)$$
$$q_i \leftarrow q_i + \gamma((r_{ui} - \hat{r_{ui}}) \cdot p_u - \lambda q_i)$$

where γ is the learning rate and λ is the regularization term.

To optimize the hyper-parameters (the learning rate γ and the regularization term λ), we conducted a grid search on the validation set with a list of different learning rates and regularization terms. For given values, the search exhaustively considers all parameter combinations to determine the best combination.

4.5 Evaluations

We chose Normalized Mean Absolute Error [4] as the metric of prediction quality just like the authors of [3].

$$\text{NMAE} = \frac{\frac{1}{n} \sum_{i=1}^{n} |p_i - r_i|}{r_{max} - r_{min}}$$

It is independent of the scale of the data, so can be used to compare forecasts across data sets with different scales.

Unlike [3], however, we did not exclude any data from calculation. For User-User and Item-Item algorithms, if there are not enough neighbors, the neighbor aggregation is set to 0 so the prediction ends up being equivalent to the mean μ_u or μ_i respectively.

As we can see from Figure 3, Global Mean does not improve when the dataset becomes larger and the dataset becomes denser, which is expected since the algorithm is not personalized in any way. The Mean algorithm serves as a baseline.

We note that the Normalized Mean Absolute Error of predictions of User-User algorithm does not significantly improve. We will discuss possible reasons of this in Section 5.

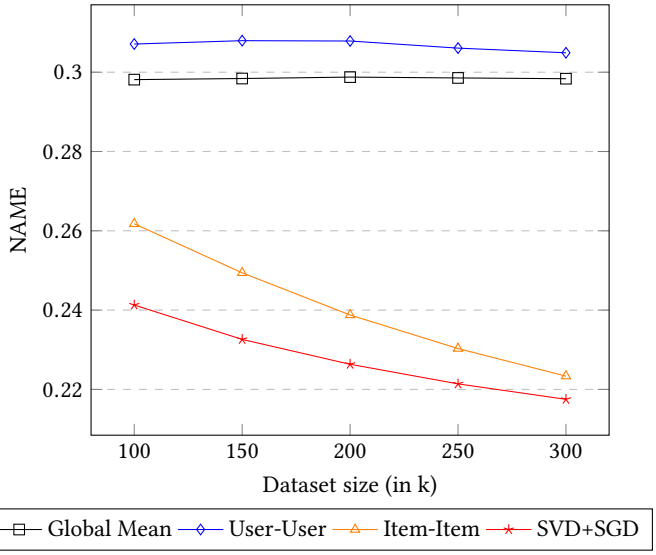
On the other hand, the Item-Item algorithm predictions NMAE notably improves as the dataset becomes larger and the dataset becomes denser, as illustrated by Figure 3.

We also observe that the SVD+SGD algorithm predictions NMAE also improves as the dataset becomes larger and the dataset becomes

Table 3: NMAE of algorithms on different sized dataset

	100000	150000	200000	250000	300000
Global Mean	0.298129	0.298407	0.298767	0.298552	0.298361
User-User	0.307103	0.307953	0.307865	0.306075	0.304895
Item-Item	0.261780	0.249402	0.238768	0.230302	0.223357
SVD+SGD	0.241286	0.232610	0.226339	0.221396	0.217498

Figure 3: NMAE of algorithms on different sized dataset



denser, but not at the same rate as the Item-Item algorithm. However, the SVD+SGD algorithm consistently outperforms the Mean, User-User, and Item-Item algorithms in all of our sampled datasets.

As mentioned above, we were unable to run User-User and Item-Item on the entire dataset due to resource limitation. However, we were able to run Global Mean and SVD+SGD on the entire dataset and obtained NMAEs of 0.297426 and 0.149827, respectively. In [3], the best-performing algorithm for Production Validation is User-User, with a NMAE of roughly 0.145 according to the plot. We will discuss this in Section 5.

5 CONCLUSIONS

We found that Global Mean and User-User algorithm are not suitable for this dating preference dataset. This is likely due to the fact that this dataset is so sparse that for most of the users there are not enough neighbors so the prediction ends up being equivalent to the user mean.

On the other hand, Item-Item algorithm and SVD+SGD algorithm proved to be suitable choices for this dating preference dataset, with SVD+SGD's performance especially stands out when the data is very sparse.

From [3]'s Production Validation Results plot, we know that the User-User algorithm's performance gradually improves as the dataset become denser, overtaking Item-Item in accuracy when the number of ratings provided by active users reachs about 30. We note that it also narrowly beats SVD+SGD on the entire dataset. However, it should not be ignored that SVD+SGD runs significantly faster than the User-User algorithm. SVD+SGD thus remains a suitable algorithm as the dataset grow relatively dense.

We faced scalability problem with User-User and Item-Item algorithms, which is in part caused by the sparsity of the data matrix. Even with our 100k dataset, these algorithms take a long time to compute. A possible direction for improvement is to consider possible optimization to conserve memory in memory-based collaborative filtering.

Another possible direction is to explore other model based Collaborative Filtering algorithms, such as Probabilistic Latent Semantic Analysis.

REFERENCES

[1] Stefan Behnel, Robert Bradshaw, Craig Citro, Lisandro Dalcin, Dag Sverre Seljebotn, and Kurt Smith. 2011. Cython: The Best of Both Worlds. *Computing in Science & Engineering* 13, 2 (2011), 31–39. https://doi.org/10.1109/MCSE.2010.118 arXiv:http://aip.scitation.org/doi/pdf/10.1109/MCSE.2010.118

[2] John S. Breese, David Heckerman, and Carl Kadie. 1998. *Empirical Analysis of Predictive Algorithms for Collaborative Filtering.* Technical Report. Microsoft Research, One Microsoft Way, Redmond, WA 98052.

[3] Lukas Brozovsky and Vaclav Petricek. 2007. Recommender System for Online Dating Service. In *Proceedings of Conference Znalosti 2007.* VSB, Ostrava. http://www.occamslab.com/petricek/papers/dating/brozovsky07recommender.pdf

[4] J.L. Herlocker, J.A. Konstan, L.G. Terveen, and J.T. Riedl. 2004. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)* 22, 1 (Jan. 2004), 5–53. https://doi.org/10.1145/963770.963772

[5] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. 1999. An Algorithmic Framework for Performing Collaborative Filtering. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '99).* ACM, New York, NY, USA, 230–237. https://doi.org/10.1145/312624.312682

[6] Nicolas Hug. 2017. Surprise, a Python library for recommender systems. http://surpriselib.com. (2017).

[7] John D. Hunter. 2007. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering* 9, 3 (2007), 90–95. https://doi.org/10.1109/MCSE.2007.55 arXiv:http://aip.scitation.org/doi/pdf/10.1109/MCSE.2007.55

[8] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30âĂŞ37. https://doi.org/10.1109/mc.2009.263

[9] LibimSeTi and Lukas Brozovsky. 2006. (Apr 2006). http://www.occamslab.com/petricek/data/

[10] Wes McKinney. 2010. Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference*, Stéfan van der Walt and Jarrod Millman (Eds.). 51 – 56.

[11] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. Group Lens: An open architecture for collaborative filtering of netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work.* 175–186. https://doi.org/10.1145/192844.192905

[12] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the 10th International Conference on World Wide Web (WWW '01).* ACM, New York, NY, USA, 285–295. https://doi.org/10.1145/371920.372071

[13] Peng Xia, Benyuan Liu, Yizhou Sun, and Cindy Chen. 2015. Reciprocal Recommendation System for Online Dating. *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015 - ASONAM 15* (Jan 2015). https://doi.org/10.1145/2808797.2809282