Report

Maciej Buszka

February 7, 2020

1 Introduction

This project aims to compare different approaches to movie recommendation based on historical ratings of users. Each model is tested in predicting the rating for a given user and movie pair. Available ratings are split into training and testing subsets. Models are scored based on accuracy of their predictions with respect to ratings in the training set. I chose the MovieLens data set as I was also curious how the models scale with amount of data, both in their accuracy and computational requirements.

2 Data set

The data set contains ratings of movies by users, in the form of records `(rating, movieId, userId)`. The ratings are given as a number in range $[0.5, 5.0]$. The data set is available in two sizes: `~100k` ratings and `~25m` ratings. These ratings are distributed among `~53k` different movies and are given by `~283k` users. The records are transformed into sparse utility matrix with ratings as elements and user and movie ids as axes. The sparse storage is dictated by huge dimensions of the matrix coupled with sparsity of ratings (0.2 % non-zero entries in large data set). Aside from using utility matrix as is, I normalized it by subtracting for each user (movie) their average score from non-zero entries in respective row (column) of utility matrix. In order to retain numerical stability in later processing I added small ($\epsilon = 0.001$) amount to each rating, such that no row or column would become all zeroes. Conceptually this normaliztion assigns average rating as the initial prediction. Additionally I checked how restricting the dataset by requiring minimal amount of ratings for a movie affects the algorithms.

```
       user_small  movie_small  user_big  movie_big
count         610         9724    283228      53889
mean          165           10        97        515
min            20            1         1          1
25%            35            1        15          2
50%            70            3        30          7
75%           168            9        95         48
max          2698          329     23715      97999
```

3 Collaborative Filtering

Collaborative Filtering is a technique that attempts to predict unknown ratings by using known ratings of similar items. It comes in two flavours: 1. `Item-item` - rating is predicted based on ratings of other similar movies rated by a given user. 2. `User-user` - rating is predicted based on ratings of a given movie rated by other similar users.

In this project I chose `item-item` approach for two reasons. Firstly, according to literature it gives more stable results over time, as user tastes may change. Secondly, the similarity matrix for movies is about 100 times smaller than for users (which would not fit in memory of my computer). I tested two approaches to aggregating the users: Weighted Average and K-Most Similar Average. Both of them use precomputed similarity matrix. I chose cosine similarity as it is recommended by literature and can be precomputed using sparse matrix multiplication. The result is stored as a dense matrix to facilitate faster access.

3.1 Weighted Average

In this variant all ratings that have been found are aggregated by computing their average weighted by similarity.

$$\begin{aligned} r_m^u =& \frac{\sum_{m' \in M_u} r_{m'}^u s_m^{m'}}{\sum_{m' \in M_u} s_m^{m'}} \\ M_u =& \text{ set of movies rated by } u \\ s_m^{m'} =& \text{ similarity of } m \text{ and } m' \end{aligned}$$

3.2 K-Most Similar

This approach aggregates ratings, by taking average rating of k most similar movies.

4 Singular Value Decomposition

Another approach to ranking prediction is learning latent relationships between users and items. Using Singular Value Decomposition, the utility matrix is decomposed into three matrices of lower dimension, parameterized by k.

$$\begin{aligned} R =& U\Sigma V^T \\ U \in& \mathbb{R}^{m \times k} \\ \Sigma \in& \mathbb{R}^{k \times k} \\ V^T \in& \mathbb{R}^{k \times n} \end{aligned}$$

To predict the rating of a movie m by an user u it suffices to calculate $U_{(m)}\Sigma(V^T)^{(u)}$

5 Results

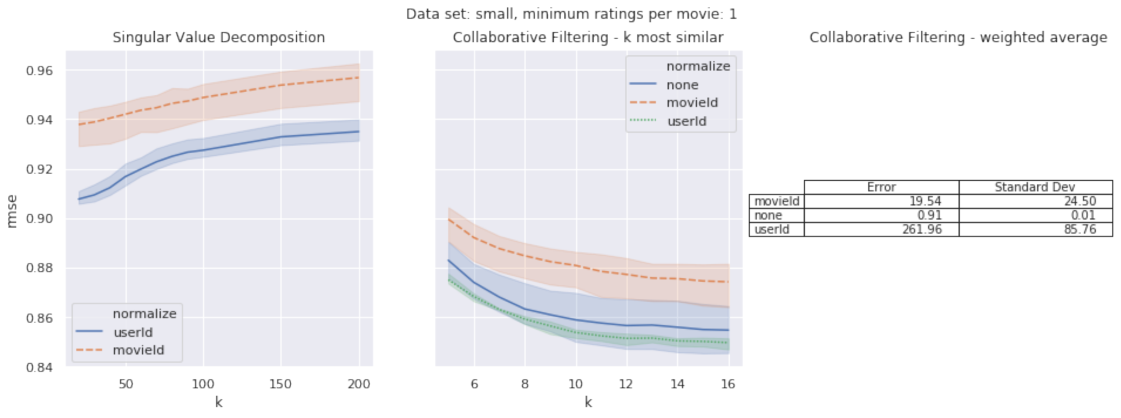
I measured mean squared error of predicted rating compared to given rating in the testing set.

In general Collaborative Filtering with K-Most Similar aggregation performed the best, particularly when ratings were normalized by average movie rating. Unfortunately it was also the slowest method, but I suspect it may be due to suboptimal implementation.

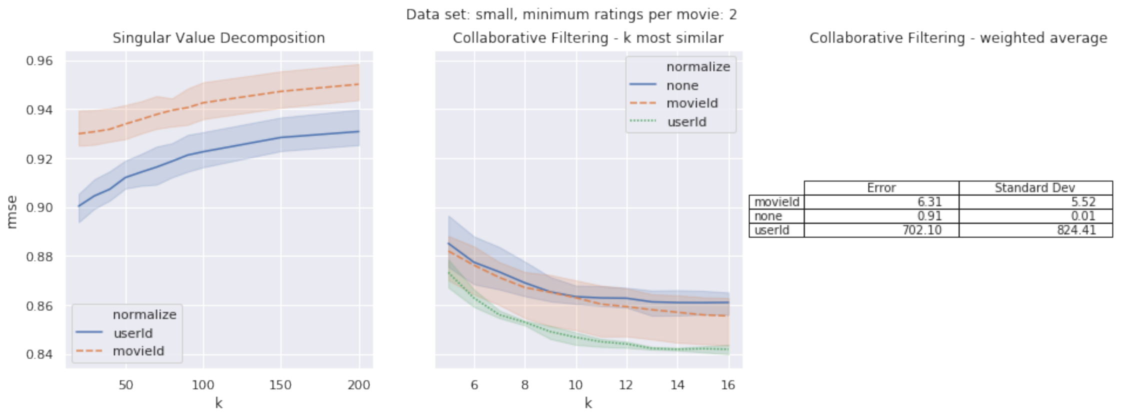
The Weighted Average performed the worst on small data set and didn't work with normalized ratings. I suspect, that it may be due to low amount of ratings per movie (less than 10 for majority of movies) and resulting numerical instabilities in weight calculation. On large data set however, it performed quite well on data normalized by average movie rating.

Singular Value Decomposition gave stable results both for small and large data sets. It's accuracy was worse than KMS but it was the fastest algorithm and it had smallest memory footprint.

Restricting the set of movies based on minimal count of ratings improved the accuracy of all algorithms, especially when the data was also normalized by average movie rating.

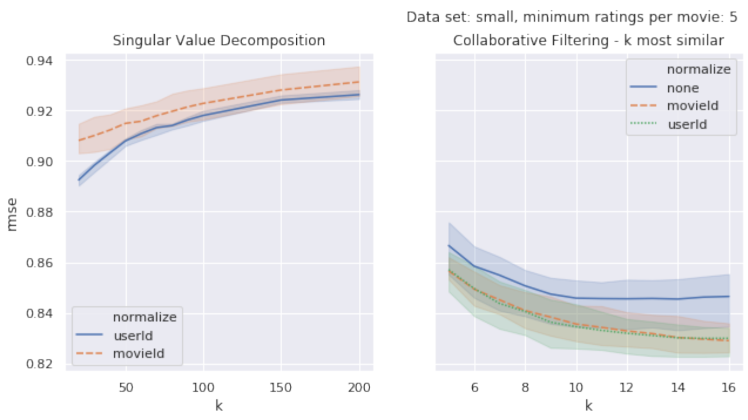


	Error	Standard Dev
movieId	19.54	24.50
none	0.91	0.01
userId	261.96	85.76



	Error	Standard Dev
movieId	6.31	5.52
none	0.91	0.01
userId	702.10	824.41

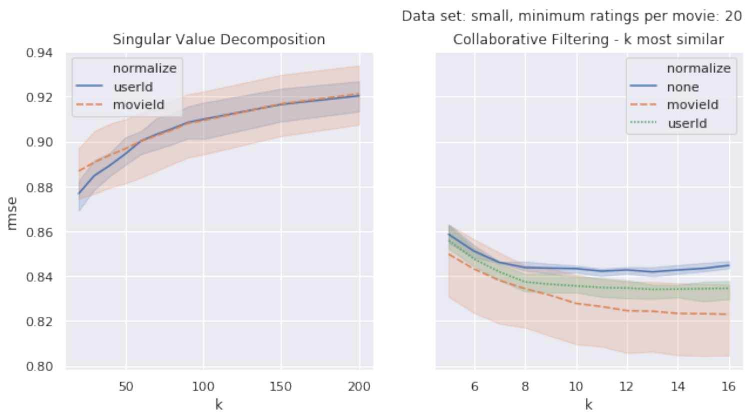
Data set: small, minimum ratings per movie: 5



Collaborative Filtering - weighted average

	Error	Standard Dev
movieId	3.95	2.42
none	0.91	0.01
userId	153.00	170.36

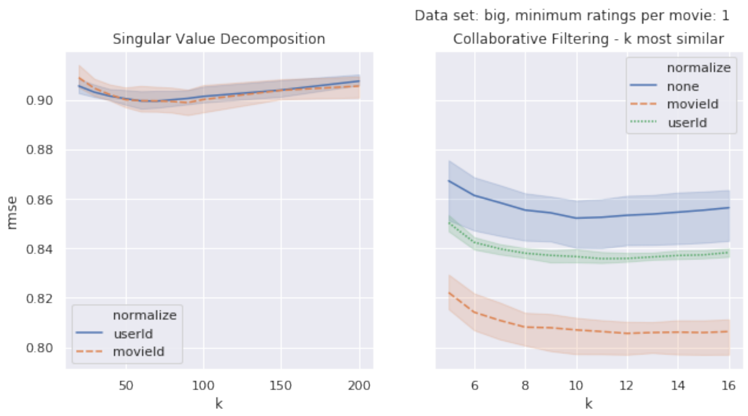
Data set: small, minimum ratings per movie: 20



Collaborative Filtering - weighted average

	Error	Standard Dev
movieId	4.59	1.94
none	0.89	0.01
userId	306.58	188.19

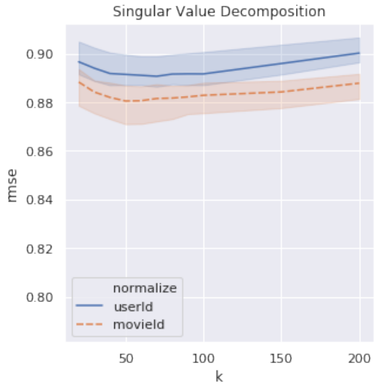
Data set: big, minimum ratings per movie: 1

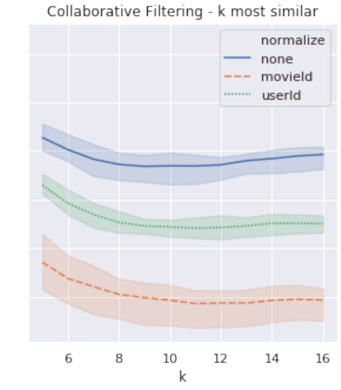


Collaborative Filtering - weighted average

	Error	Standard Dev
movieId	0.85	0.02
none	0.93	0.01
userId	59.84	49.76

Data set: big, minimum ratings per movie: 2

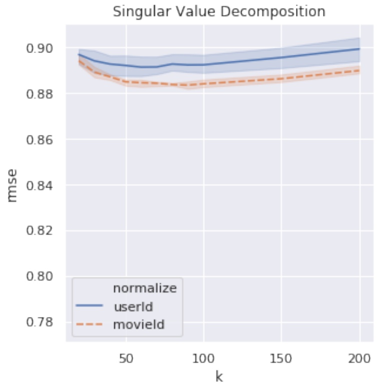


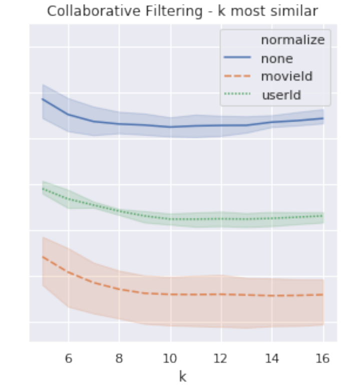


Collaborative Filtering - weighted average

	Error	Standard Dev
movieId	0.83	0.00
none	0.93	0.01
userId	75.46	30.61

Data set: big, minimum ratings per movie: 5

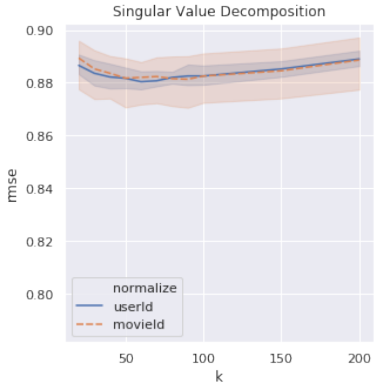


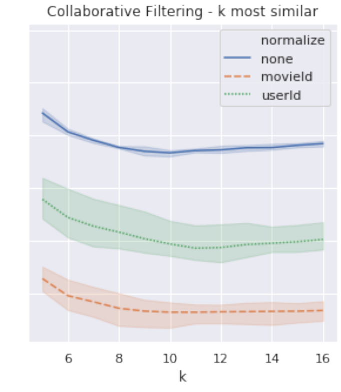


Collaborative Filtering - weighted average

	Error	Standard Dev
movieId	0.85	0.04
none	0.93	0.00
userId	482.97	699.91

Data set: big, minimum ratings per movie: 20





Collaborative Filtering - weighted average

	Error	Standard Dev
movieId	0.83	0.00
none	0.93	0.01
userId	144.27	93.86