Methods of classification and dimensionality reduction - Report 1

April 11, 2022

1. Introduction

Statement of the problem. In this task we have to create a movie recommender system for our users. We have users who rated some movies. Of course, not every user rated every movie and it is our task to fill those gaps. So if one user hasn't seen one movie, we want to predict how he would like it.

For this purpose we build few algorithms using different methods of predicting. Of course different methods will give us different results (errors). Our task is to tune parameters of those methods and try to get the best possible ratings prediction.

Description of methods. In this problem, we use different methods which are subset of PCA methods. They are often used for dimensionality reduction and matrix factorization.

SVD1. This method gets a $n \times d$ dimensional matrix Z and approximate it by a different matrix $\tilde{Z}$. Since we want somehow $\tilde{Z}$ to maintain only "the most important" information from Z, then the rank of $\tilde{Z}$ is to be much smaller than rank of Z. Precisely, we want to find matrix $\tilde{Z}_r$ of rank r ($r < rank(Z)$ and r is a parameter), so that $\|Z - \tilde{Z}_r\|$ is small.

Using SVD decomposition $Z = U\Lambda^{\frac{1}{2}}V^T$ we construct $\tilde{Z}$ as

$$\tilde{Z}_r = U_r \Lambda_r^{\frac{1}{2}} V_r^T,$$

where Λ_r contains r biggest eigenvalues of Z and U_r, V_r contains only columns corresponding to those eigenvalues.

SVD2. It is an iterative method. We perform SVD1 on matrix Z, then on the result of first SVD1 and so on. The algorithm can be stopped after a fixed number of iterations or some stop condition can be established.

NMF. Similarly as in SVD1 the method obtain a $n \times d$ dimensional matrix Z and approximate it by $\tilde{Z}$. This time $\tilde{Z}$ is constructed as $\tilde{Z}_r = W_r H_r$, where W_r and H_r are matrices with non-negative elements (W_r has r columns and H_r has r rows). Precisely, we look for such W_r and H_r that $\|Z - W_r H_r\|^2$ is the smallest, where $\|A\|^2 = \sum_{i,j} A_{ij}^2$.

SGD. This method, similarly as previous ones want to estimate matrix Z with a product of matrices W and H, but not necessarily obtaining the whole matrix Z.

Let's assume that we have only some values of z_{ij} and let call those pairs (i,j) where we know the value of Z as I. We look for

$$\arg\min_{W,H} \sum_{(i,j)\in I} (z_{ij} - w_i^T h_j)^2 + \lambda(\|w_i^T\|^2 + \|h_j\|^2),$$

where h_j is j-th column of h, w_i^T is i-th row of W and $\lambda > 0$ is a parameter. So roughly speaking we look for W and H such that $Z \approx WH$ for elements known in Z, but also we want W and H to have quite small values (it gives us the part of sum with parameter λ).

opis metody...

2. Implementation

Description of the data. Our data contains information 100837 ratings - exactly 610 users rated 9724 movies. The columns are: userId (integer), movieId (integer) and rating (integer), where userId is a unique user id and movieId is a unique movie id.

We keep this data in two-dimensional matrix of size $n \times d$ where n is the number of users and d is the number of movies. In element (i,j) we put the rate of the user i of the movie j. If the user i haven't rated the movie j we leave the element empty.

Performing methods. ??tutaj jakaś intuicja po co dzielić dane??

So to be able to evaluate the quality of the programs we split our data to two parts: train set and test set. The train set is used to build the programs. And the test set is used to evaluate how our programs work.

To give our programs enough information about every user we split the data so that the train set contain 90% of ratings of each user (and the test set the remaining ones). tutaj coś o tym, że będziemy to powtarzać??

Let call the matrix containing the data from the train set as $\boldsymbol{Z}$ and the matrix containing the data from the test set as $\boldsymbol{T}$.

Quality of the system. Assume that our algorithm return a matrix $\boldsymbol{Z}'$. Then the quality of our programs is computed as **root-mean square error**

$$\mathsf{RMSE} = \sqrt{\frac{1}{|\mathcal{T}|} \sum_{(u,m)\in\mathcal{T}} \left(\boldsymbol{Z}'[u,m] - \boldsymbol{T}[u,m]\right)^2}$$

where $\mathcal{T}$ contains pairs (u,m) from test set.

Imputing the missing data. Since three of our methods (SVD1, SVD2 and NMF) are given a full matrix $\boldsymbol{Z}$ then they need the missing data to be imputed before performing.

We decided to impute the data in 4 different ways:

- putting 0 everywhere,
- putting global mean everywhere,
- putting row mean,
- putting weighted row and column mean ($\alpha \cdot \mathsf{col\_mean} + (1-\alpha) \cdot \mathsf{row\_mean}$, where α is a parameter).

tutaj przemyślenia na temat tego czemu niektóre metody działają lepiej i dlaczego

3. Parameters tuning and results

coś o tym, że metody pozostawiaja trochę dowolności?

Before performing our methods and obtaining results we have to set some parameters.

First of all, all the methods need a parameter r, which is the rank of matrices in Z decomposition. SGD needs also learning rate and λ. And iteration methods need maximum of possible iterations or a stop condition.

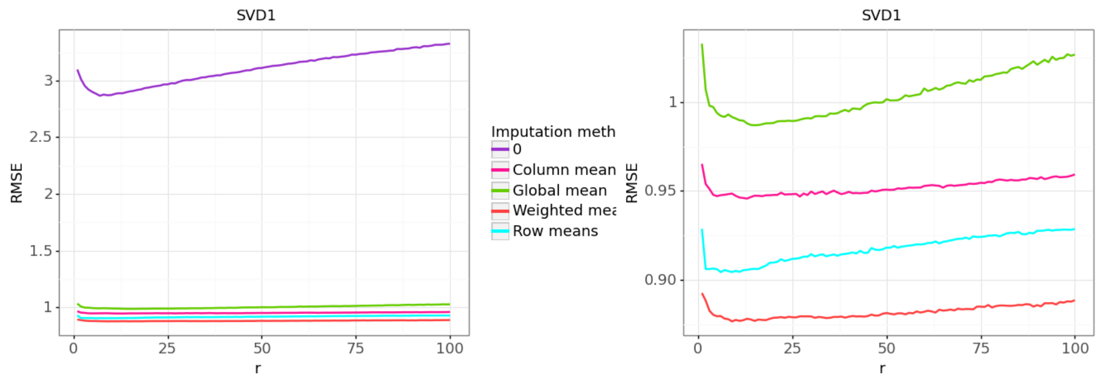
What's more, for all of our methods we want to choose optimal α in the last method of imputing data.

SVD1. co tu napisać

napisać co to jest to weighted

At the beginning we consider only ...

Below, we present a graph showing dependence of RMSE on r.

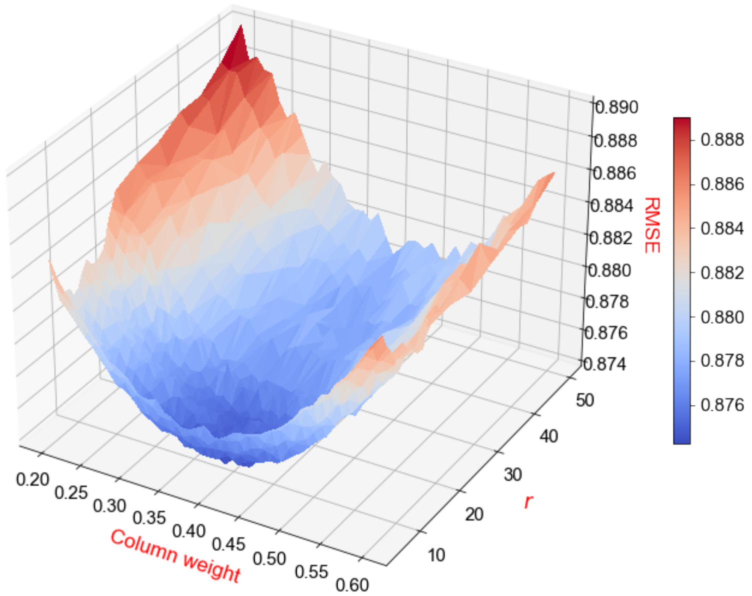


Also we present a table with the lowest RMSE for every imputation method and the parameter r that gave it.

	0	column means	global mean	weighted means	row means
r	7	13	15	9	6
RMSE	2.866	0.946	0.987	0.877	0.904

First of all, we observe that as we expected the imputation method does matter. It is most clearly seen looking at RMSE of data filled with zeros, that for the best r is around 2.9. Other methods also differ a lot. The lowest RMSE obtain the data filled with weighted data. That's why we may suspect that optimizing α can give even better results.

To get optimal result we perform optimization with respect to two parameters: α and r. As we can see on the picture above only r between 0 and 50 give some reasonable results, so we consider only those (we could use all r, but it is time consuming). Below, we present graph showing dependence of RMSE on α and r for data ...



Below we present also table with ... lowest RMSE and pairs (α, r) that gave those.

α	r	RMSE
0.39	10	0.873987
0.38	10	0.874244
0.42	10	0.874274
0.36	10	0.874449
0.39	11	0.874469

As we can see 10 seems to be the best r and 0.39 seems to be the best α. Also in all those results (α, r) are close to them. So $\alpha = 0.39$ and $r = 10$ are parameters that we use to perform this method further in the report.

Since in columns we keep indexes of movies, it means that our filled data take a bit more information from user ratings mean than from the movie ratings mean. That is probably logical ...

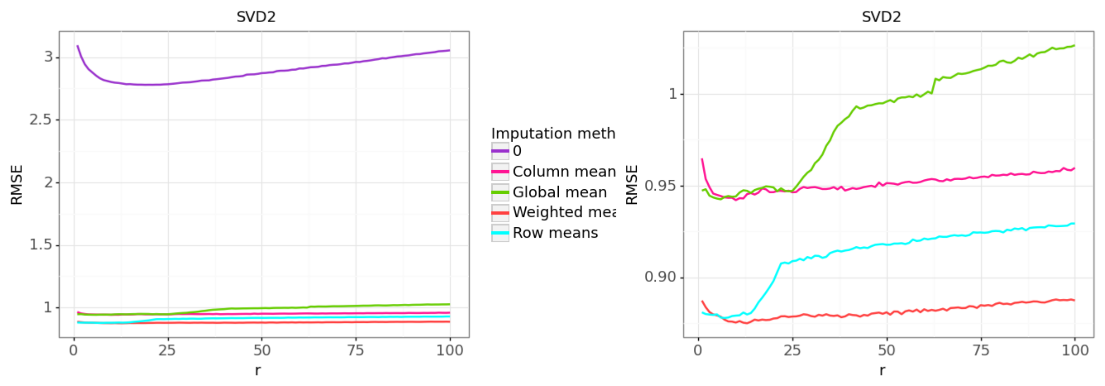
To conclude this subsection we present our best results obtained using these methods. which means what

tabelka z najlepszymi wynikami

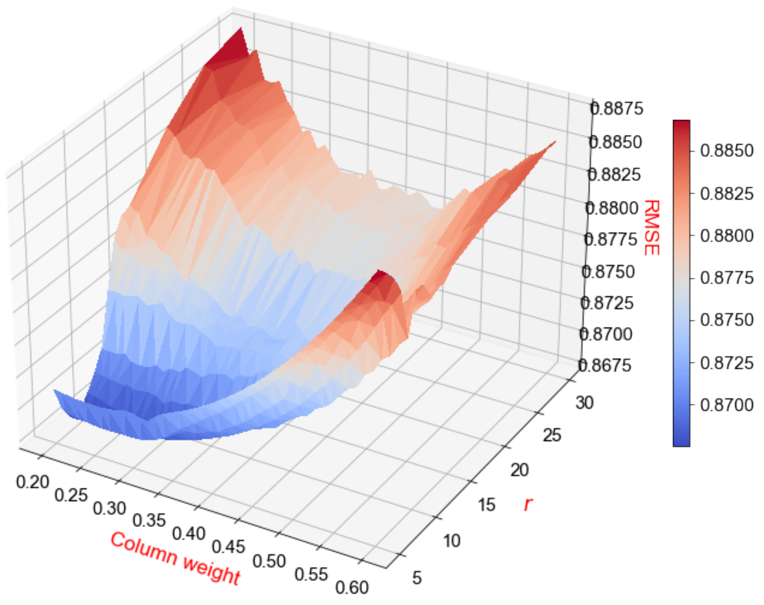
SVD2. In this case we want to proceed as in SVD1 case.

??In SVD2 we make a correction – czy to tu

stop condition

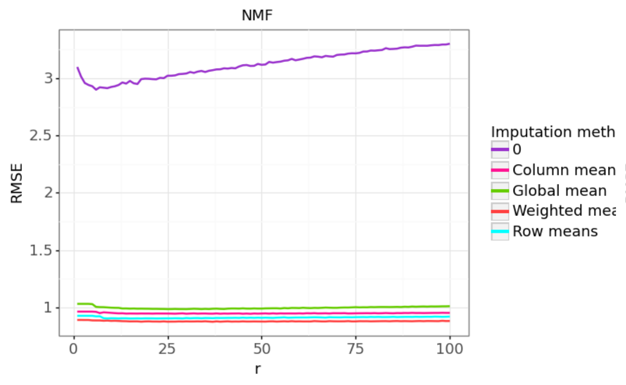


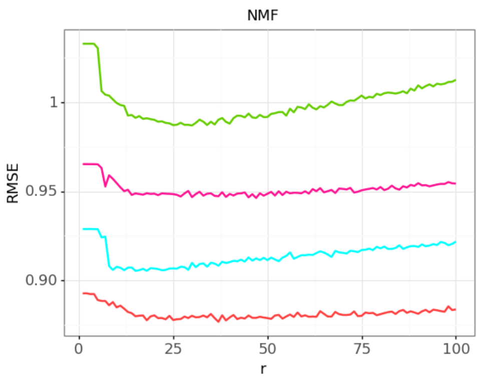
	0	column means	global mean	weighted means	row means
r	19	10	6	13	7
RMSE	2.779	0.942	0.942	0.875	0.878



α	r	RMSE
0.25	8	0.867393
0.26	8	0.867397
0.24	8	0.867402
0.27	8	0.867410
0.28	8	0.867494

NMF. In this case since we have only r and α to find, we proceed in exactly the same way as in the case of SVD. So, firstly we present a graph showing dependence of RMSE on r. in some cases...





Comparing this graph to the graph for SVD we can see that tutaj o tym, że jest bardziej takie postrzępione

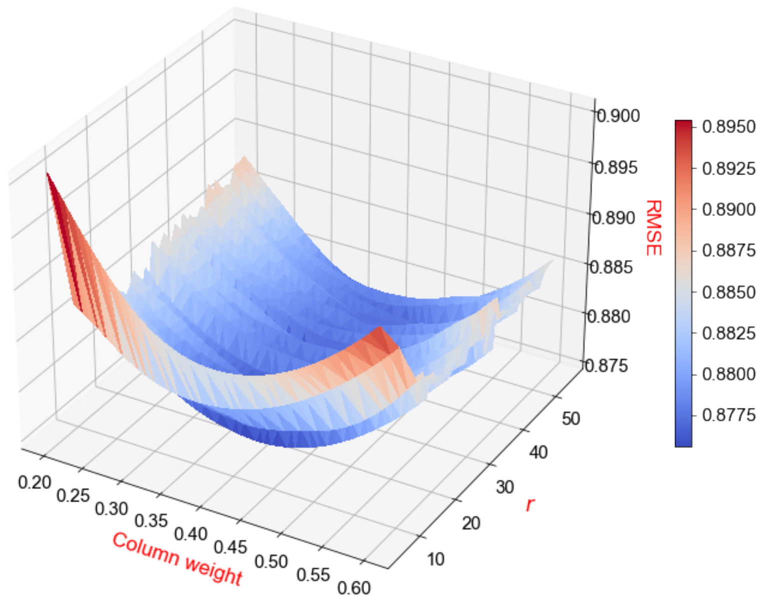
Below we also present a table with the lowest RMSE for every imputation method and the parameter r that gave it.

	0	column means	global mean	weighted means	row means
r	6	47	30	37	15
RMSE	2.900	0.946	0.987	0.877	0.905

α	r	RMSE
0.40	37	0.874794
0.41	37	0.874817
0.39	18	0.874841
0.39	37	0.874848
0.40	18	0.874849

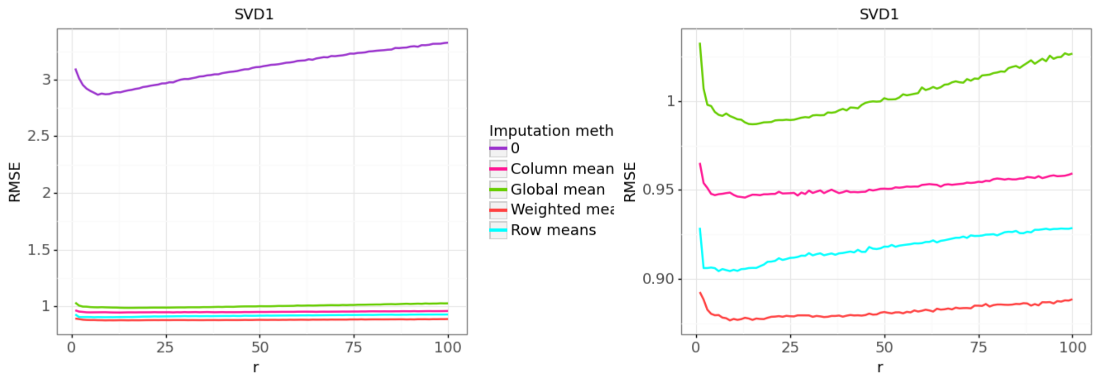
As we can see the parameters r are in general bigger than in previous cases. czy będą bardziej porostrzelane

Now we pefrorm the optimization with respect to α and r and present a graph showing the results.



tabelka z najlepszymi r i alpha jakie r i alpha wybieramy

tabelka z najlepszymi wynikami



SGD.

4

5. Conclusions