Methods of classification and dimensionality reduction - Report 1

April 12, 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 $\tilde{Z}_r = W_rH_r$, 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 $\|Z - W_rH_r\|^2$ 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).

It is an iterative method and work this way: set some W and H,

(1) sample one pair (i, j) from I,

(2) let $\tilde{w}_i^T = w_i^T - \eta \cdot \big(2(z_{ij} - w_i^T h_j)h_j + 2\lambda w_i^T\big)$ and $\tilde{h}_j = h_j - \eta \cdot \big(2(z_{ij} - w_i^T h_j)w_i^T + 2\lambda h_j\big)$,

(3) the rests of matrices W and H stay unchanged, so $\tilde{w}_k^T = w_k^T$ for $k \neq i$ and $\tilde{h}_l = h_l$ for $l \neq j$,

(4) take $W = \tilde{W}$ and $H = \tilde{H}$,

and repeat.

η is a parameter that tells us how big steps we want to do. The method stops after a certain number of steps, or it can be given a stop condition.

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.

Quality of the system. We want somehow judge how well each method work, and which one works the best. So before performing algorithms we split our data into two parts: train set and test set. The train set will be used to build the programs. And the test set is intended to evaluate how our programs work.

Of course, we don't want to construct optimal programs for only one split of data, we want it to be as good as possible for every data. So we will have to repeatedly split our data, find the best possible model for every split, and then average them somehow.

To give our programs enough information about every user we split the data so that the train set contains 90% of ratings of each user (and the test set the remaining ones).

How exactly we want to evaluate how good our program is? We will use the distance measure called **root-mean square error** (RMSE).

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 $\boldsymbol{T}$. Assume that our algorithm return a matrix $\boldsymbol{Z}'$. Then the quality of our programs is computed as

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

where $\mathcal{T}$ contains pairs (u, m) from test set. So it is a distance between matrices Z and Z' on only elements from $\mathcal{T}$.

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 5 different ways, we replace missing values with:

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

We may expect that the closer to reality we impute the missing data, the better results we will obtain. That's why we expect that filling missing data with 0 will give worse result than the rest methods. Similarly, probably since filling data with global mean doesn't differentiate the users or the movies

Performing methods. The first thing to do is to actually implement our methods. In case of SVD1 and NMF we just perform the method described in first section on the filled matrix $\boldsymbol{Z}$. SVD2 we also perform on the filled $\boldsymbol{Z}$ and in every iteration step we fill the elements from train set with real values. We make this correction, beacause ... SGD we perform only on values from train set.

3. Parameters tuning

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 η and λ. The iteration methods need maximum of possible iterations or a stop condition.

What's more, for the methods that need filled data we want to choose optimal parameter α in the imputation method with weighted means.

SVD1.

Optimizing r. For a start, let's consider only imputation methods that don't need estimation of α, so replacing missing values with: 0, global mean, column mean, row mean and weighted row and column mean with $\alpha = \frac{1}{2}$.

For these methods we only need to find optimal r and we will call them *basic*. So for each of them and for every r from 1 to 100 we perform SVD1. Below, we present a graph showing results.

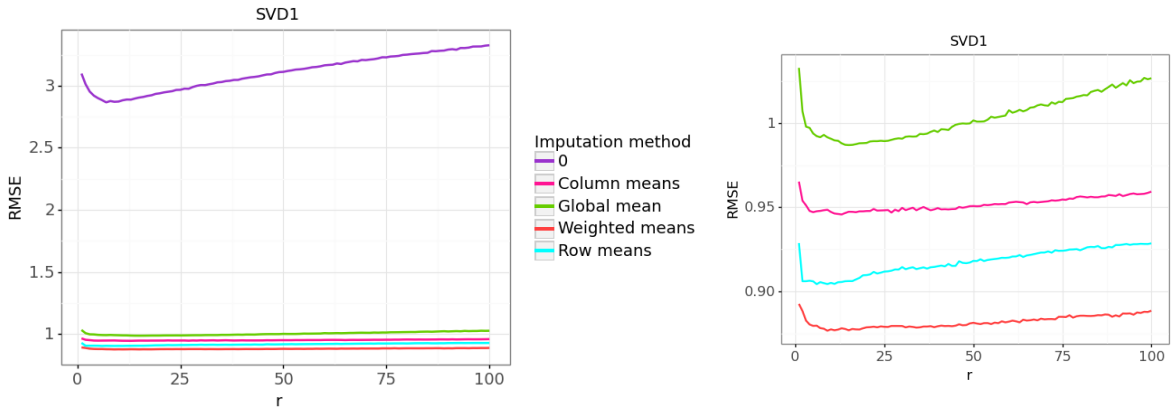

FIGURE 1. RMSE of SVD1 for basic imputation methods and $r = 1, \ldots, 100$

Let's denote that the results in the graph above are computed for the same split of data into train and test set. That's why we can compare them. Also, these results can be much different if we take different split.

Of course we look for the lowest RMSE obtained for each imputation method and the optimal r. So below we present a table containing this information.

	0	column means	global mean	weighted means	row means
r	7	13	15	9	6
RMSE	2.8660	0.9458	0.9870	0.8767	0.9043

TABLE 1. The lowest RMSE and optimal r for SVD1 with basic imputation methods

First of all, from both the graph and the table we can observe that as we expected the choice of the imputation method affects the RMSE. It can be most clearly seen on an example of

data filled with zeros. For the best r RMSE there is around 2.9 that is, it is about 3 times larger than for other imputation methods. Other methods also differ. The lowest RMSE is obtained for the data filled with weighted data. But the result for data filled with row means is also quite good. That's why we may suspect that optimizing α in our weighted imputation method can give even better results.

Optimizing α. To get optimal result we perform optimization with respect to two parameters: α and r. As we can see on the graph 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 results of optimization.

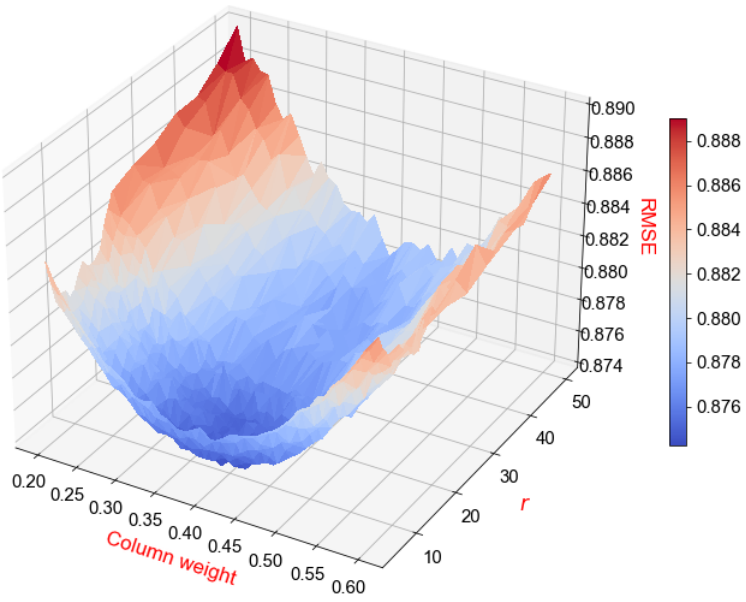

FIGURE 2. RMSE of SVD1 for weigthed imputation method for different α and r

Below we present also table with 5 lowest RMSE and pairs (α, r) that gave them.

α	r	RMSE
0.39	10	0.8740
0.38	10	0.8742
0.42	10	0.8743
0.36	10	0.8744
0.39	11	0.8745

TABLE 2. 5 lowest RMSE of SVD1 for weigthed imputation method and (α, r) that gave them

Obtained best RMSEs are similar, but they differ in fourth decimal place. We can observe also that the pair $(0.39, 10)$ seems to be optimal in this case and all other pairs are close to it.

Is this means that we choose the weighted imputation method with $\alpha = 0.39$ and $r = 10$ to perform SVD1? No, because until now, we have considered only one data split into train and test set. To find the best parameters in our method we have to average optimal α and r over different splits. So we consider 20 different splits and the results are as follows

- mean value of the best (α, r) is $(0.4055, 12.2)$,
- median of the best (α, r) is $(0.41, 13)$,
- in 15 of 20 cases the best pair is $(0.41, 13)$.

So $\alpha = 0.41$ and $r = 13$ are the parameters we use in our method SVD1 with weighted means as the imputation method.

SVD2. We are going to proceed as in SVD1 case, but before we have to choose some stop condition.

Stop condition. We decided to use the stop condition of following form: if the difference of Z obtained in previous step is enough close to Z obtained in this step we stop the algorithm. Precisely, we stop algorithm if $\|Z_{n+1} - Z_n\| < \varepsilon$, where $\|\cdot\|$ is the Frobenius norm, Z_n is the matrix obtained in nth step and ε is a parameter to be set.

To find optimal ε we perform an optimization with respect to r and ε on the data filled with weighted means with $\alpha = \frac{1}{2}$. Below we present the graph with results.

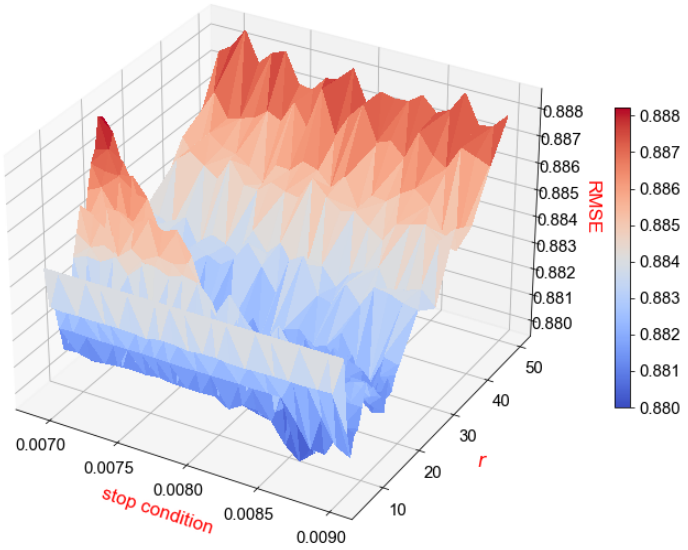

FIGURE 3. RMSE of SVD2 for weigthed imputation method with $\alpha = \frac{1}{2}$ for different r and ε

The minimum of RMSE on the graph above is taken for $r = 9$ and $\varepsilon = 0.0086$. So we decided to take $= 0.0086$ in this method.

Optimizing r. After choosing the stop condition we can proceed exactly as in SVD1 case. So first we present a graph showing dependence of RMSE on r and on the imputation method for basic imputation methods.

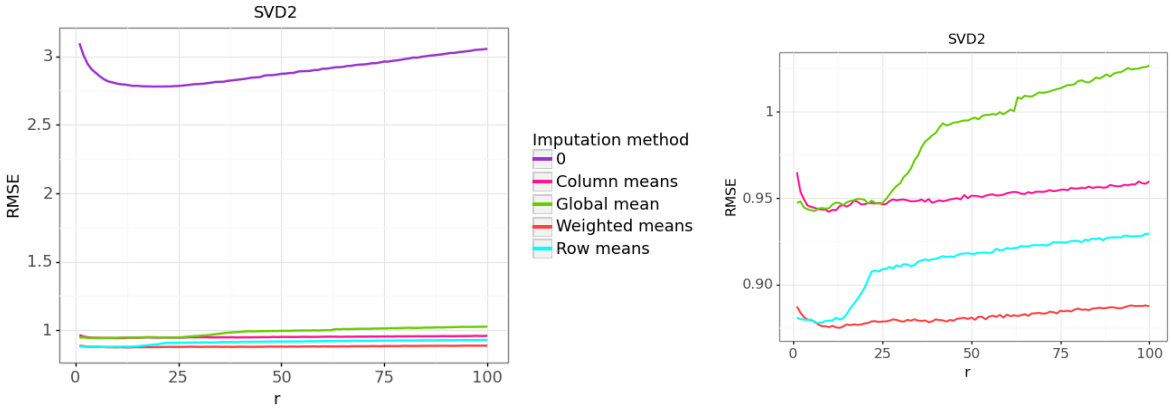

FIGURE 4. RMSE of SVD2 for basic imputation methods and $r = 1, \ldots, 100$

Comparing it with analogous graph for SVD1 we observe bigger „jumps" for imputation methods using global means and row means. And for these methods best RMSEs are improved. But after jumps the graph is very similar to the previous one - results and trends are similar. In case of imputation methods with column means and weighted means doesn't look like the trajectories are changed much.

Now we present a table showing the best r and RMSE for every imputation method.

	0	column means	global mean	weighted means	row means
r	19	10	6	13	7
RMSE	2.7789	0.9420	0.9425	0.8749	0.8778

TABLE 3. The lowest RMSE and optimal r for SVD2 with basic imputation methods

Firstly, we can observe that SVD2 improved the results of SVD1. We expected that since it is SVD1 just performed many times. So every result is smaller, differences between results are also smaller, but the order which methods are better or worse stayed the same.

We can observe again that only r between 0 and 50 gives reasonable results. Though the best r chosen by SVD2 in all cases differ a lot from r chosen by SVD1.

Optimizing α. Moving on to the weighted imputation method, we present a graph showing the results of optimization with respect to α and r.

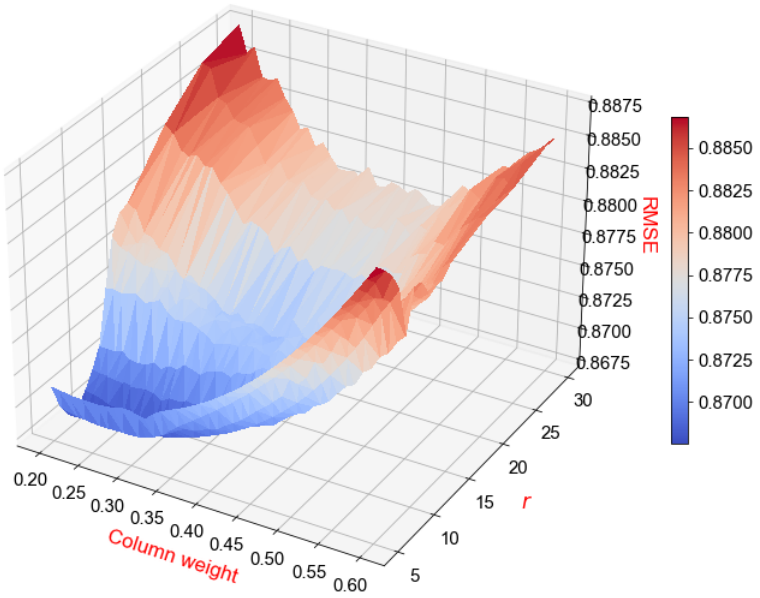

FIGURE 5. RMSE of SVD2 for weigthed imputation method for different α and r

The graph differs a lot from the analogous graph for SVD1. This time the minimum is taken for smaller α and smaller r. To find exact values of minimum let's look at the results in table.

α	r	RMSE
0.25	8	0.8674
0.26	8	0.8674
0.24	8	0.8674
0.27	8	0.8674
0.28	8	0.8675

TABLE 4. 5 lowest RMSE of SVD2 for weigthed imputation method and (α, r) that gave them

This time method took the best 5 results for the same r. Also in this case, the differences between results are smaller. It is intuitive for iterative method because they usually converge to some specific model and that is why we get clear results.

After repeating this optimization for 20 different splits we get that:

- the mean value of the best (α, r) is $(0.259, 8)$,

- the median of the best (α, r) is $(0.255, 8)$.

So $\alpha = 0.26$ and $r = 8$ are parameters we use in our SVD2 with weighted means as the imputation method.

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.

Optimizing r. So firstly we present a graph showing dependence of RMSE on r and on the imputation method for basic imputation methods.

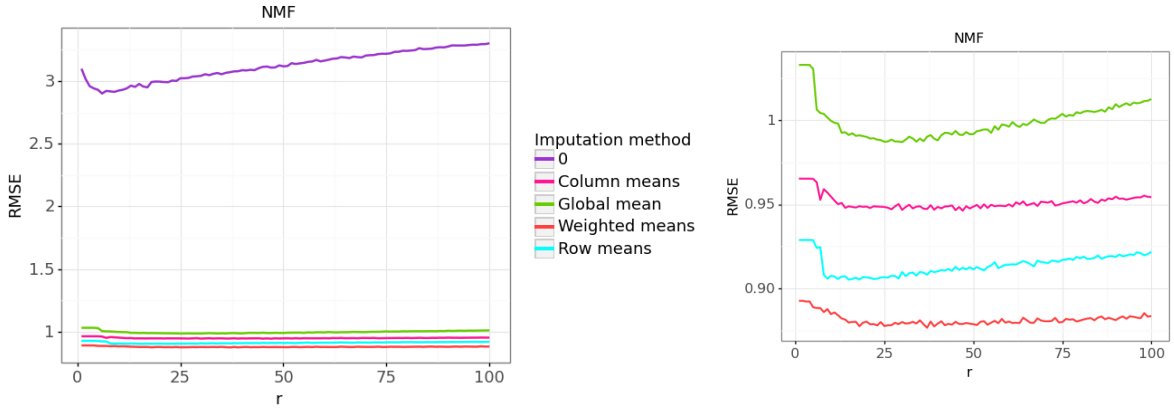

FIGURE 6. RMSE of NMF for basic imputation methods and $r = 1, \ldots, 100$

Comparing this graph to the graph for SVDs we observe that the previous graphs where smoother. This time trajectories oscillate a lot, which means that for similar r it gives quite different RMSE. So this method seem more unstable than the previous ones.

What's more, here it is not so obvious where to look for optimal r. For instance for imputation method with weighted means RMSE don't grow much with greater r.

Nevertheless, the order of methods (so which one is the best and so on) is the same.

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	2.8997	0.9462	0.9870	0.8766	0.9053

TABLE 5. The lowest RMSE and optimal r for NMF with basic imputation methods

As we can see the parameters r are in general bigger than in previous cases. The RMSEs are very similar to those obtained using SVD1.

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

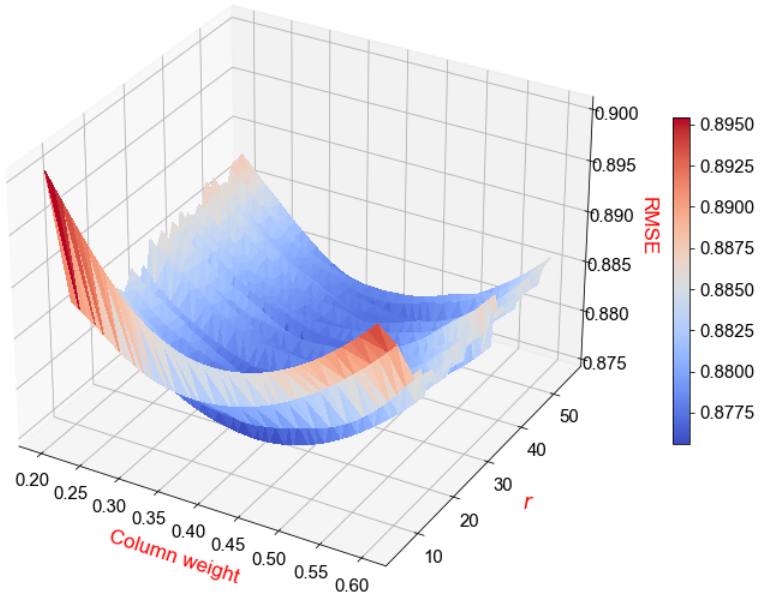

FIGURE 7. RMSE of NMF for weigthed imputation method for different α and r

Comparing this graph to previous analogous graphs this one is completely different. RMSE in here doesn't grow much with growth of r. Similarly, as in previous graph for NMF, this graph oscillates a lot at the axis of r. To find where exactly is the minimum of RMSE taken we have to look at the results in table.

α	r	RMSE
0.40	37	0.8748
0.41	37	0.8748
0.39	18	0.8748
0.39	37	0.8748
0.40	18	0.8748

TABLE 6. 5 lowest RMSE of NMF for weigthed imputation method and (α, r) that gave them

This time something unusual happened, two far values of r gave very close RMSEs. wnioski, że te r są takie różne, a wyniki bardzo bliskie

After repeating this optimization for 20 different splits we get that:

- the mean value of the best (α, r) is $(0.393, 32.25)$,
- the median of the best (α, r) is $(0.39, 37)$,
- 15 times the best r is 37 and 5 times the best r is 18.

So $\alpha = 0.39$ and $r = 37$ are parameters we use in our NMF with weighted means as the imputation method.

SGD. In case of this method we have to set parameters r, η and λ, but also set some stop condition.

Stop condition. We chose to use a similar stop condition as in the case of SVD2. So we stop algorithm if $\|Z_{n+1} - Z_n\| < \varepsilon$, where Z_n is the matrix obtained in nth step and ε is a parameter to be set. This time we used different norm less time-consuming, so $\|\cdot\|$ is L^1 matrix norm.

Performing similar analysis as in the case of SVD2 this time we decided to use as small ε as possible so that it can be computed. So in this case $\varepsilon = 10^{-11}$.

Optimizing r, η and λ. Then we performed optimization with respect to r, η and λ. The result are as follows ...

So $r = 5$, $\eta = 0.007$ and $\lambda = 0.01$ are parameters we use in our SGD.

5. Results

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 ...