Methods of classification and dimensionality reduction - Report 1

April 2, 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.

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).

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

We decided to impute the data in ... different ways:

- putting 0 everywhere,

-

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

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

Choosing parameters.

SVD1. r

SVD2. r?, czy tutaj warunek zatrzymania

NMF. r

SGD. η, warunek początkowy

3. Results