Recommender system

Bartosz Makaruś

Our goal is to build and store a recommender system for movies, that works by predicting ratings for movies which given user has not seen (rated) yet. We have access to dataset of `ratings`, which contains 100836 ratings from 610 users, of 9724 different films.

We consider matrix Z of size 610×9724, containing the ratings. The matrix is sparsely filled ($\approx 98.3\%$ of the terms are NaN), so we have to impute the missing data. I have opted for using the k Nearest Neighbours classifier for imputation, so as to in some way take into account that some movie ratings might be correlated, or more generally, users with similar tastes are likely to rate the same movie similarly. The KNN imputation, for each user (row z_i of Z) finds k other users who are the closest in the feature space, where distance between rows i, j is measured by

$$\sqrt{\frac{d}{d_1} \sum_{m \neq NaN} (z_{im} - z_{jm})^2},$$

where d_1 is the number of features that are not NaN for both rows. Next, algorith performs majority voting among the neighbours to choose what values to impute in the i-th row.

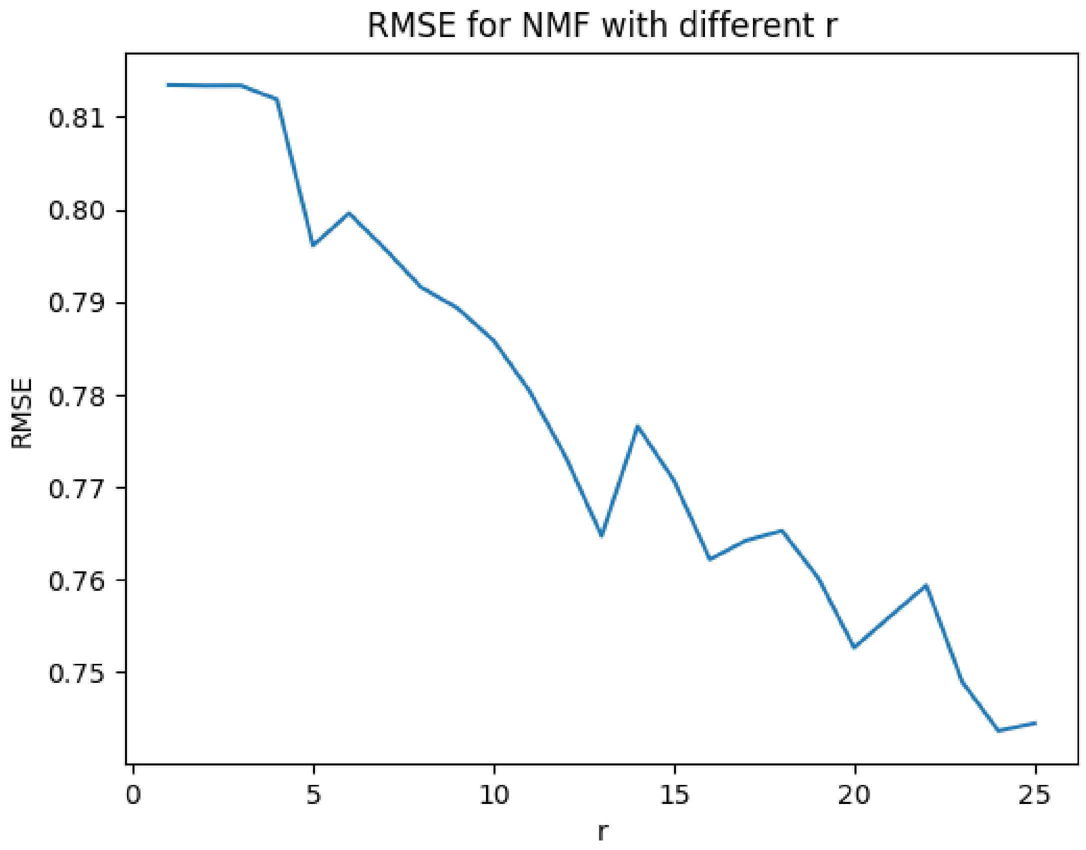
Then, we perform dimensionality reduction on the matrix Z using four methods.

Non-negative Matrix Factorisation

The NMF method approximates Z of size $n \times d$ with a product

$$Z = WH,$$

where W is of size $n \times r$, and H, $r \times d$, with r being a parameter. We choose r based on the $RMSE$ for models with different parameters.



We see that there is first dip for $r = 5$, which will also result in a small amount of parameters stored.

Singular Value Decomposition

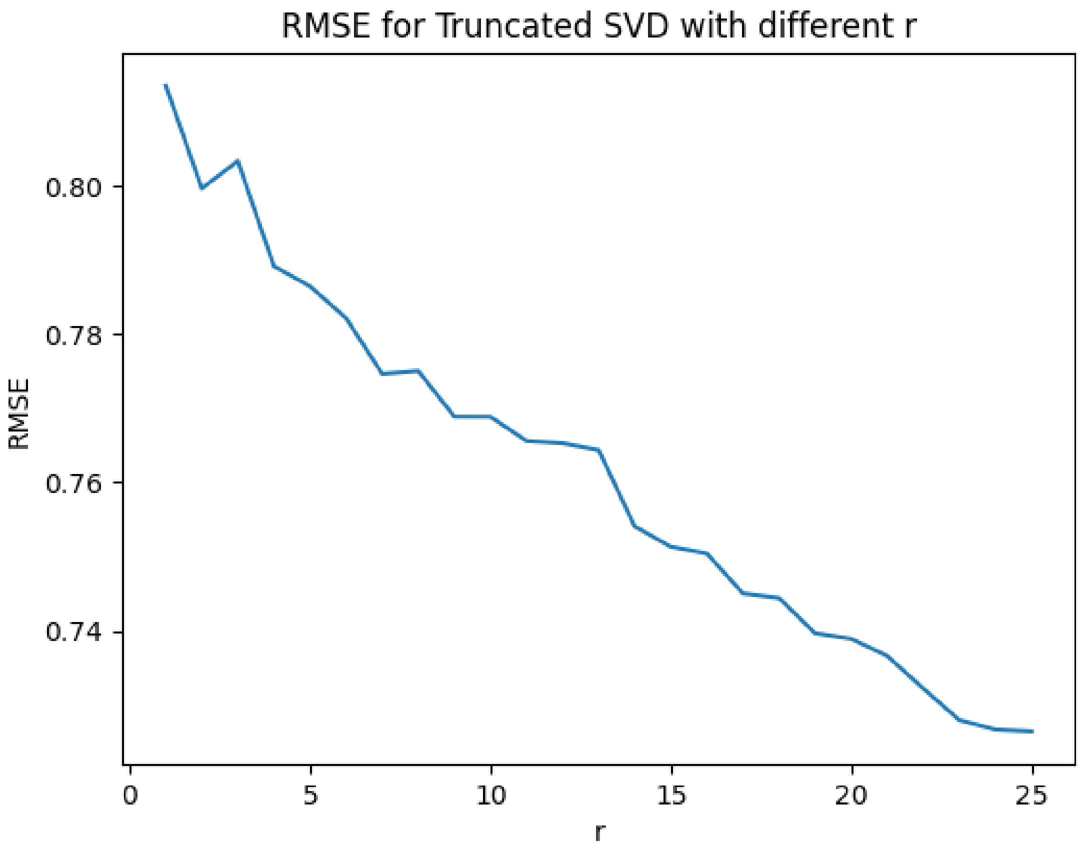
In this case, we perform the singular value decomposition for matrix Z,

$$Z = U\Lambda^{\frac{1}{2}}V^T.$$

Then, lower rank approximation is obtained by Truncated SVD, that is we calculate

$$Z \approx U_r\Lambda_r^{\frac{1}{2}}V_r^T,$$

where U_r, V_r are matrices of first r columns of U, V, and Λ_r is the first r rows and columns of Λ. We choose the parameter r by comparing the root mean squared error for models with different r.



There is again sharper decline for $r \leq 5$ so the value 5 will be used in the models.

SVD2

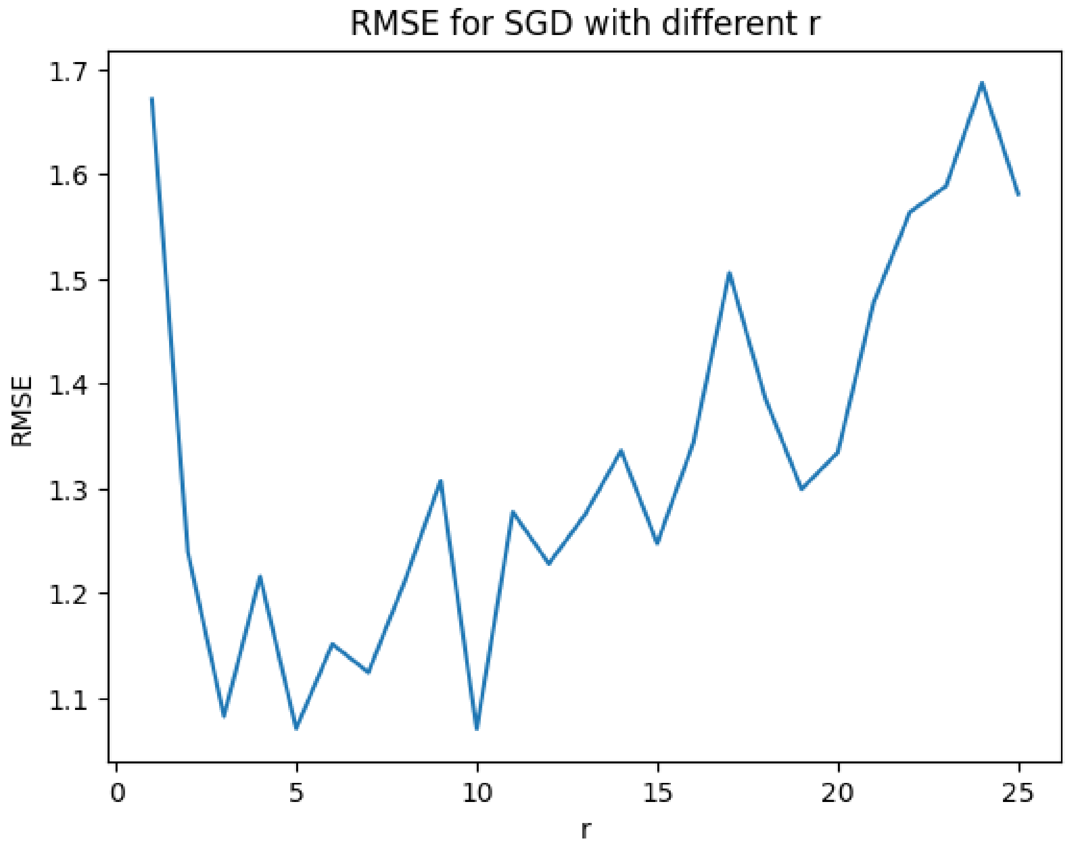
In this algorithm, we first perform Truncated SVD on the matrix Z with r and replace the previously imputed values in Z with approximates from SVD. Then, we perform SVD on Z again, and we repeat the procedure n times.

Stochastic Gradient Descent

This method does not require imputation of data in matrix Z. Here, we simply use SGD (calculated on batches) to optimize matrices W, H as parameters. The loss function we use is

$$l(W, H) = \sum_{(i,j):z_{ij} \neq NaN} (z_{ij} - w_i^T h_j)^2 + 10(||w_i||^2 + ||h_j||^2),$$

where w_i^T is the i-th row of W and h_j is the j-th column of H.



$RMSE$ seems to reach lowest values around $r = 5$, which we will use in the model.