Comparing Recommendation Systems

Nuno Gonçalves

January 2023

Singular Value Decomposition and Low-Rank Factorization

Let $A \in \mathbb{R}^{m \times n}$ be a real (possibly rectangular) matrix with $m \leq n$. Suppose that :

$$A = U\Sigma V^\top$$

is the singular value decomposition (SVD) of A. U and V are then orthogonal matrices, and Σ is an $m \times n$ diagonal matrix with entries $(\sigma_1, \sigma_2, \cdots, \sigma_m)$ such that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$. Using the function from numpy, *numpy.linalg.svd* we can directly obtain the mentioned decomposition.
Eckart-Young Theorem states that the best rank-k approximation to A in certain norms, is given by:

$$A_k := \sum_{i=1}^{k} \sigma_i u_i v_i^\top$$

where u_i and v_i denote the ith column of U and V, respectively. This theorem can be proven for different matrix norms, for instance: spectral L^2 norm = σ_1 , Frobenius norm which is equal to the square root of $\sigma_1^2 + \sigma_2^2 + ... + \sigma_n^2$; nuclear/trace norm = $\sigma_1 + \sigma_2 + ... + \sigma_n$ To check if the algorithm implemented was correct it was tested the full reconstruction of the matrix on rectangular and squares up to 6x6 matrices which succesfully recovered the original matrix in all cases. We also plotted the error of reconstruction for the frobenius, spectral and nuclear norm for a random 100x50 matrix:

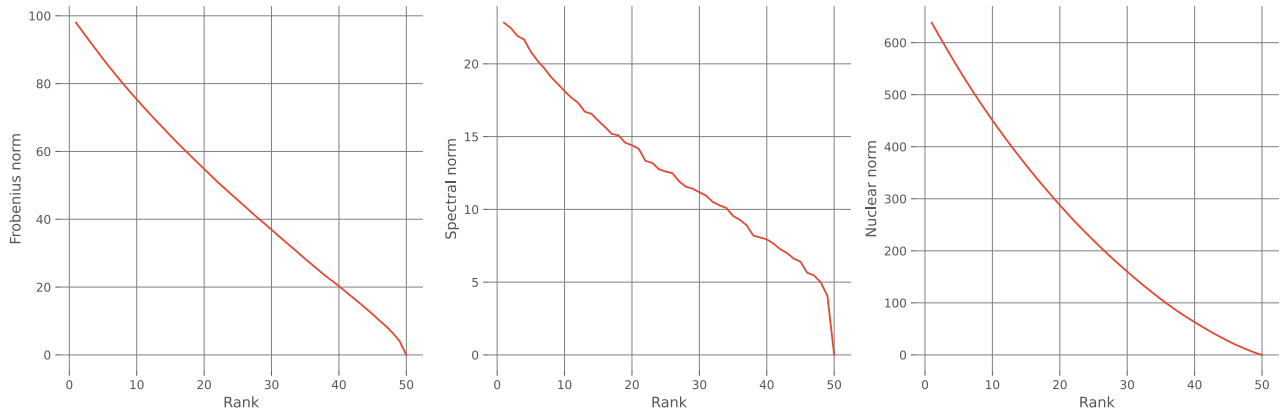


Figure 1: Reconstruction error for different norms of a 100x50 matrix.

As expected as we increase the rank we get lower and lower errors for the three norms and finally when the rank is the same as the original matrix the error is zero.

MovieLens Dataset and SVD

Matrix factorization models map both users and items to a joint latent factor space of dimensionality k, such that movie-user interactions are modeled as inner products in that space. Accordingly, each movie i is associated with a row vector

$M_i \in \mathbb{R}^k$, and each user j is associated with a column vector $(U^T)_j \in \mathbb{R}^k$. For a given movie i, the elements of M_i measure the extent to which the item possesses those factors, positive or negative (though we can make the restriction for them to be positive, doing so would be called Non-negative Matrix Factorization). For a given user j, the elements of U_j measure the extent of interest the user has in items that are high on the corresponding factors, again, positive or negative. The resulting dot product, $M_i(U^t)_j$, captures the interaction between user j and movie i—the user's overall interest in the movie's characteristics. This tries to approximate what would that user give to that specific movie, which is denoted by r_{ui}, leading to the estimate:

$$\hat{r}_{ij} = M_i U_j^t$$

So our goal is to approximate the matrices M and U and with that done all we have to do to predict the rating of a (movie, user) pair is to do the dot product between the corresponding latent vectors.
Our first model will be a singular value decomposition (SVD). Tough applying SVD requires the knowledge of the matrix to be complete, that is, we can't have missing values like we have in ours, given that not every user rated every movie and not every movie was rated by every user. Though we can "cheat" our way around by imputing the values that are missing[1]. We can impute the data in many different ways: as zeros, the mean of the user or movie respectively or a combination of both, using k-nearest neighbours (kNN). We will later see than even though this works, it leads to very poor results and there are many better and robust techniques available. As suggested by the project statement we will imput them as zeros.
Using the R matrix constructed only with the training data and the rest zeros, we will now build the SVD for different k's and evaluate the performance of the algorithm based on the error it obtains on the test data.

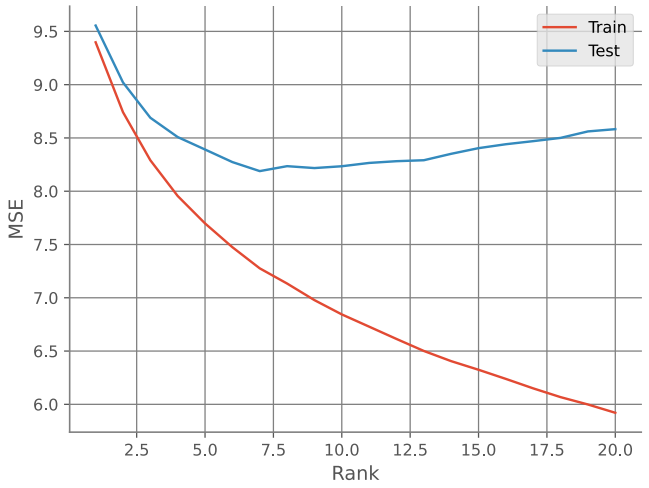


Figure 2: SVD error obtained for different ranks (k), for the training and test set.

This goes as predicted, initially as we increase k we get better generalizations, we allow the matrix factorization to have more features therefore better predictive power, as we can see through the decrease of the test error. But, as we increase k more and more we are allowing the factorization to overfit, as seen through the increase of the test error and decrease of the training error. This happens because as we increase the k, the factorization of the SVD is "memorizing" the training data which leads to a worse predictive power.
In short, the problem of doing imputation to fill in missing ratings and make the matrix dense is that innacurate imputation might disort the data considerably.[2] If we had a way to reliably fill in the missing entries, we wouldn't technically need to use SVD at all. We'd just give recommendations based on the filled in entries.

Another problem with this procedure is that imputation can be very expensive as it significantly increases the amount of data.
As a direct proof that imputation highly influences the predictions let's now try to impute with the mean of the movie.

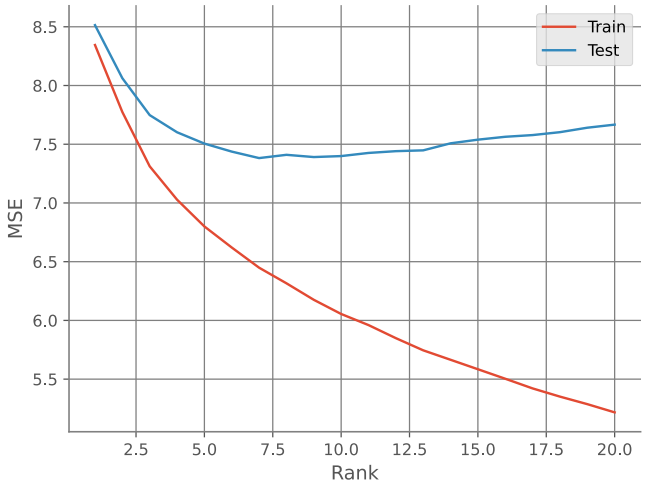


Figure 3: SVD error obtained for different ranks (k), for the training and test set when imputing with the mean of the movies in the missing entries.

As we can see, the test error achieves considerably better values by just imputing the mean of the movie instead of zeros.

Non-negative Matrix Factorization

We've seen that imputing values leads to a bias in the predictions, so now we are going to model only the **observed ratings**, while avoiding overfitting with the assistance of a regularized model. So to learn the factors M and U, we will minimize the regularized squared error on the set of known ratings:

$$\min_{M,U} \sum_{(i,j)\in\kappa} \left(r_{ij} - M_i U_j^T\right)^2 + \lambda \left(\|M_i\|^2 + \|U_j\|^2\right) \tag{1}$$

where, κ is the set of the (i,j) pairs for which r_{ij} is known (the training set).
Though, because we are allowing the factors of M and U to assume any number, we will certainly obtain positive and negative factors. This, however, leads to a difficult interpretation of the results as the factors can cancel each other out. To solve this issue we can make a similar formulation so that we only get non negative factors.
Non-negative matrix factorization (NMF) (Bro and de Jong, 1997; Lee and Seung, 1999)[3][4] is a method for finding such a representation. In this context, given a non-negative data matrix, NMF finds an approximate factorization $R \approx MU$ into non-negative factors M and U. The non-negativity constraints make the representation purely additive (allowing no subtractions) which makes the encoding easy to interpret.

First Method - NMF (Lee and Seung, 1999)[4] According Lee and Seung, the multiplicative update rule for M and U are as follows :

$$M \leftarrow M \cdot \frac{RU}{MU^\top U} \tag{2}$$

$$U \leftarrow U \cdot \frac{R^\top M}{UM^\top M} \tag{3}$$

However, since R is a sparse matrix, we need to update each M_i according to existing ratings of the movie i. Similarily, we need to update U_j according to existing ratings of the user j. Hence :

$$M_{i,k} \leftarrow M_{i,k} \cdot \frac{\sum_{j\in M_i^*} U_{j,k} \cdot r_{i,j}}{\sum_{j\in M_i^*} U_{j,k} \cdot \hat{r}_{i,j} + \lambda |M_u^*| M_{i,k}} \tag{4}$$

$$U_{j,k} \leftarrow U_{j,k} \cdot \frac{\sum_{u \in U_i^*} M_{i,k} \cdot r_{i,j}}{\sum_{u \in U_j^*} M_{i,k} \cdot \hat{r}_{i,j} + \lambda |U_j^*| U_{i,k}} \tag{5}$$

Where:

- $M_{i,k}$ is the k^{th} latent factor of M_i
- M_i^* the set of users who rated movie i
- $U_{j,k}$ is the k^{th} latent factor of U_j
- U_j^* the set of movies rated by user j

Here we define one epoch as one complete iteration through the training set, that is, every triplet (movie, user, rating) in our training set. Now running the algorithm we get:

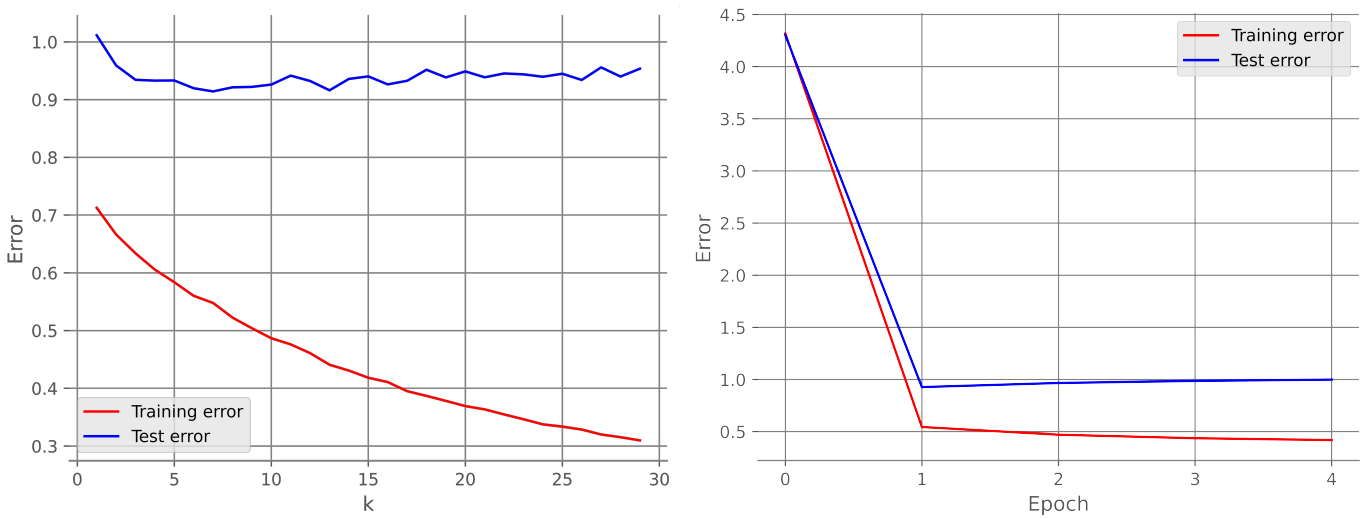


Figure 4: On the left we have the mean squared error as we increase the number of features, k. On the right we have for k = 7, the train and test error for each epoch. $\lambda = 0.1$

Immediately we get extremely better results ($\approx 833\%$) than the SVD procedure, proving once again that imputing the data can lead to a bad recommendation system. We also notice that this algorithm converges really fast and is actually guaranteed to converge as showed by Lee and Seung[4] using the multiplicative rule update. For a basis of comparison we added an epoch 0 which corresponds to the error obtained when we initialize M and U randomly to show how quickly the algorithm converges.
As for the error, for increasing k's we get, as expected, similar results to before. As we increase the k in the beginning the system obtains a better predictive power, given by the lowering of the test error, but if we keep increasing k's, soon enough the system will being to overfit, as given by the decrease of the train error while the test error plateaus (and actually starts increasing a little, even though, it is not as much as the SVD case much likely because of the regularization added).

Second Method - NNLS-ALS (Bro and De Jong, 1997)[3] Because both M_i and U_j are unknowns, equation 1 is not convex. However, if we fix one of the unknowns, the optimization problem becomes quadratic and can be solved optimally. Thus, Alternating Least Squares (ALS) techniques, as the name suggest, rotate between fixing the M_i's and fixing the U_j's. When all M's are fixed, the system recomputes the U's by solving a least-squares problem, and vice versa. This ensures that each step decreases equation 1 until convergence[2]. However, this by itself doesn't lead to non negative factors. Therefore, if we want to restrain M and U to positive values, we have to use a least squared method that ensures that Ax=b leads to x being positive. This idea was put into practice by Bro and De Jong[3].
The NNLS (non-negative least squares) algorithm is an active set algorithm that works by iteratively identifying and removing inactive constraints until the true active set is found. The algorithm starts with an initial feasible set of

regression coefficients and iteratively computes new estimates of the coefficients until the solution converges. At each iteration, the algorithm computes the residual vector and identifies the indices corresponding to the largest magnitude components of the residual vector. The algorithm then computes the unconstrained least-squares solution over the variables not in the active set and identifies the indices of the variables with negative regression coefficients. These indices represent the active set of constraints. The algorithm then computes the optimal step size by solving a quadratic program subject to the active set of constraints and updates the regression coefficients and the active set accordingly. This process is repeated until the active set is equal to the true active set or until a maximum number of iterations is reached. For a more detailed description see Bro and De Jong[3]. Combining this with the ALS we get the following algorithm:

Algorithm 1: Non-negative ALS

repeat

  for $i \leftarrow 1$ **to** n **do**

    Compute NNLS for $\underbrace{\left(\sum_{r_{ij} \in r_{i*}} U_j U_j^\top + \lambda I_k\right)}_{\text{A}} M_i = \underbrace{\sum_{r_{ij} \in r_{i*}} r_{ui} U_j}_{\text{b}};$

  end

  for $j \leftarrow 1$ **to** m **do**

    Compute NNLS for $\underbrace{\left(\sum_{r_{ij} \in r_{*j}} M_i M_i^\top + \lambda I_k\right)}_{\text{A}} U_j = \underbrace{\sum_{r_{ij} \in r_{*j}} r_{ui} M_i}_{\text{b}};$

  end

until *convergence criterion is met*;

where r_{i*} is the set of all the ratings given to movie i and r_{*j} is the set of all the rating the user j actually gave. The results obtained were the following:

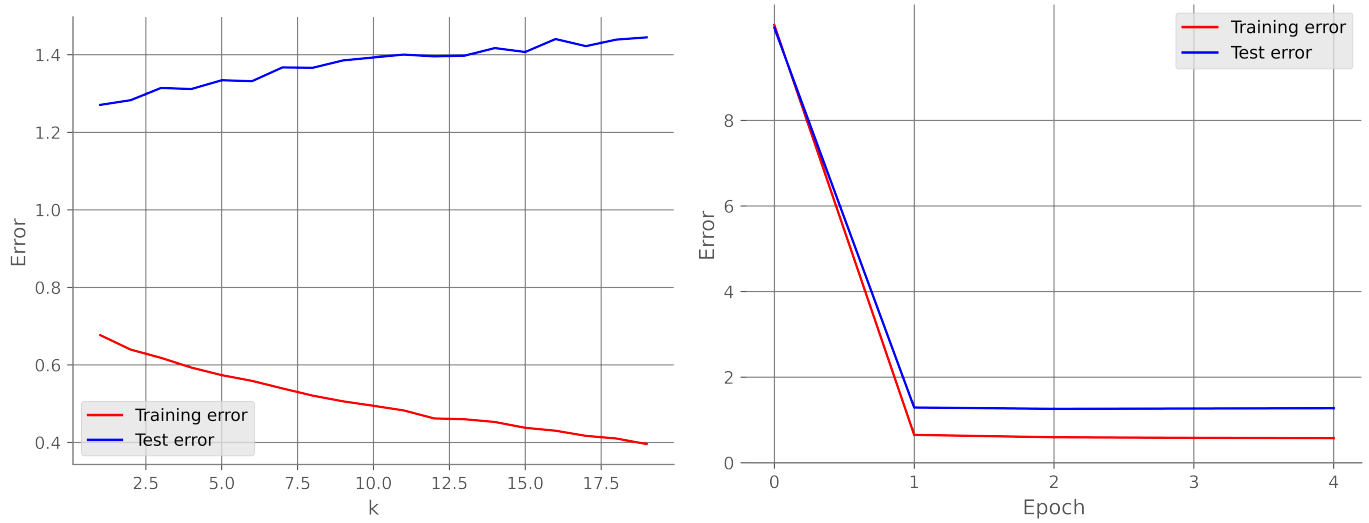


Figure 5: On the left we have the mean squared error as we increase the number of features, k. On the right we have for k = 2, the train and test error for each epoch. $\lambda = 0.2$

As we can see we achieved way better results when compared with the SVD procedure, though, we got a little bit of a worse performance when compared to the previous algorithm. As we can also see, the algorithm is overfitting right from the beginning for different k's which probably means that the regularization λ was not high enough (or possibly that the higher the nº features, the more regularization we need in order to counterbalance the overfitting!).
As to show that the factors M and U are really non negative let's plot an histogram of the values of the factors in both

algorithms:

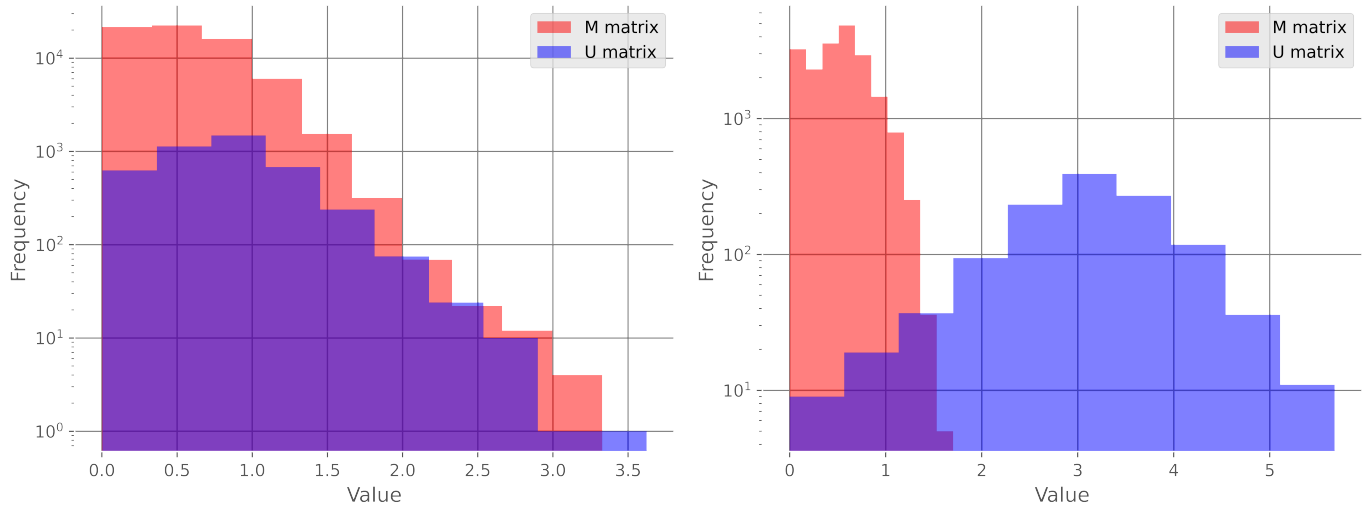


Figure 6: On the left we have the histogram of the feature matrices for the Multiplicative Rule and on the right for the ALS.

It's interesting to notice how the values of the first algorithm are more well and evenly distributed when compared to the ALS algorithm.

Some visualization

Now let's project some movies into a 2D space to visually analize the recommending system we achieved. Using the ALS the results for 50 movies (in two different images for a easier visualization) were:

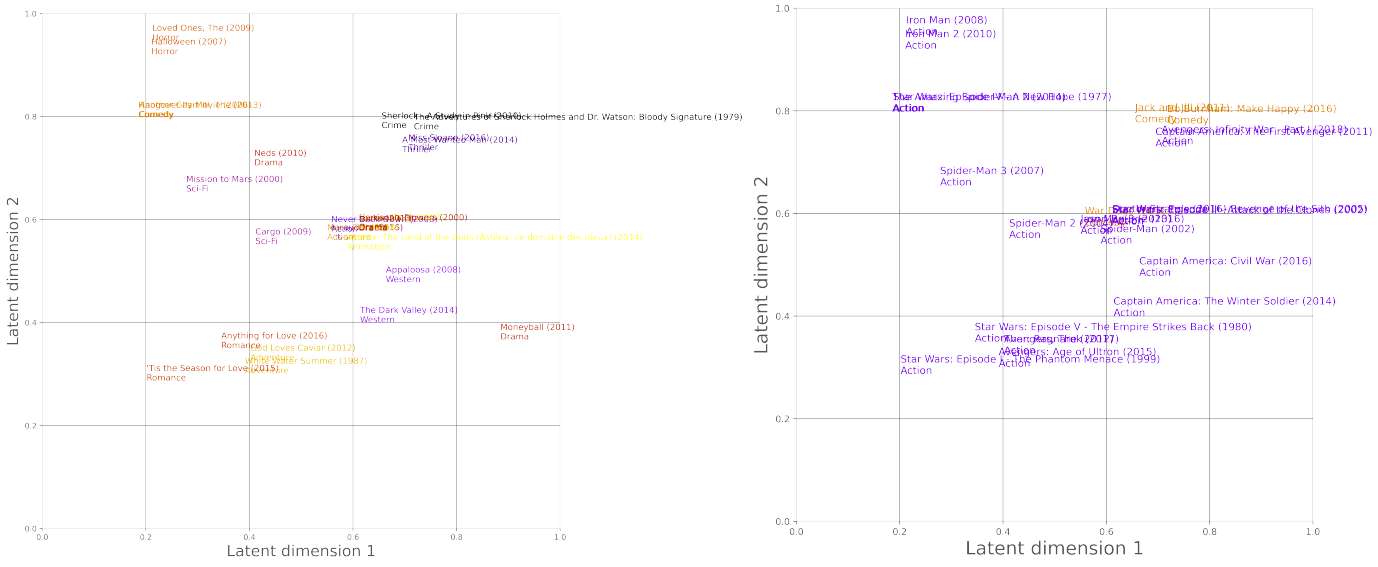
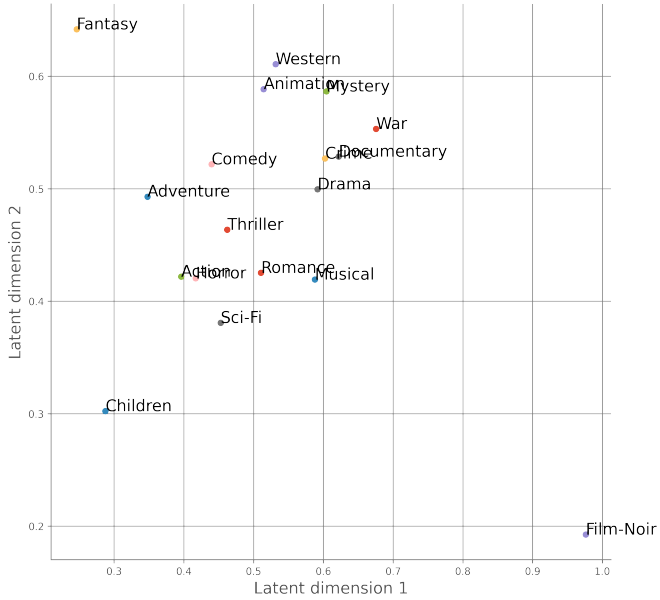


Figure 7: Projection of 50 movies on the movie latent space.

This results are very satisfactory and go within my expectations. On the left we see movies that belong to the same category (same coloring) appear to be clustered together. This is the case of horror movies on the top left, comedy a little bit under, romance, sci-fi... Another interesting fact is, for example, the movies that are about *Sherlock Holmes* are very close together, showing that they have very similar characteristics as one would expect. On the right we see more of this phenomenon, on the top left we see the movies *Iron Man* 1 and 2 very close together, in the middle we see the two *Captain America* movies again close to each other and finally, we see the original trilogies of the *Star Wars* also

clustered together. This goes to show that similar movies, in our human subjectiveness, show in the system has having very similar features. Let's look at this in another way by plotting the mean position for each genre:



Again we get some useful insight into the the recommending system. Firstly, we can notice that very niche movies like children, film-noir and fantasy are very far away from the rest of the genres given that they are for a particular audience and usually don't appeal to everyone. Also, we can see that the genres: documentary, war and crime are very close together which makes sense given that this genres are usually grouped together, that is, a lot of documentaries are related to war and crime. This goes to show that movies with similar genres often share common themes, motifs, and styles that are reflected in their features and latent factors. Therefore, it is expected that these movies would have similar latent factor values and be grouped together in the 2D projection.

Honorable Mention

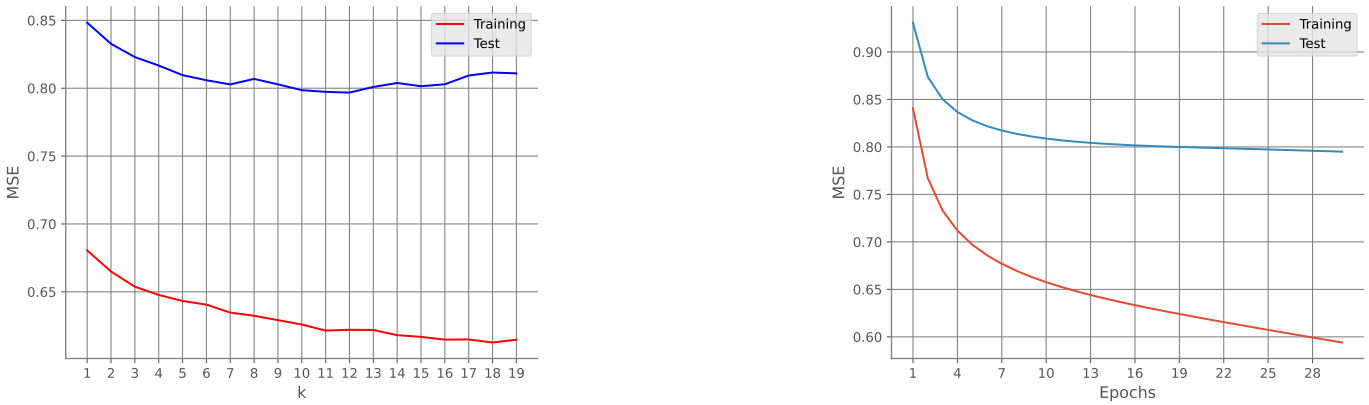


Figure 8: SGD method for non negative matrix factorization. On the left we can see the evolution of the errors as we increase the rank k. On the right the error per epoch for k=12

Another method that worked well was an NMF approximation using projected SGD (stochastic gradient descent). That is, we use gradient descent to approximate the entries of the matrix M and U using only the training data and, everytime, one of the factors is negative we clip it to zero. As we can see by the figure above, the test errors achieve really good values and it actually reaches a minimum for k=12 (which is something one would expect that we didn't get in the other methods). However, in theory, there is no guarantee of convergence.
Though is a interesting method particularly because this method can be seen as an auto-encoder with linear activation! https://andre-martins.github.io/docs/dsl2019/lecture_04.pdf

References

[1] Badrul Munir Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. Application of dimensionality reduction in recommender system - a case study. 2000.

[2] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[3] Rasmus Bro and Sijmen de Jong. A fast non-negativity-constrained least squares algorithm. *Journal of Chemometrics*, 11, 1997.

[4] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS'00, page 535–541, Cambridge, MA, USA, 2000. MIT Press.