Machine Learning & Data Mining
Miniproject 2
Caltech CS/CNS/EE 155

1. Introduction

1.1. Group members

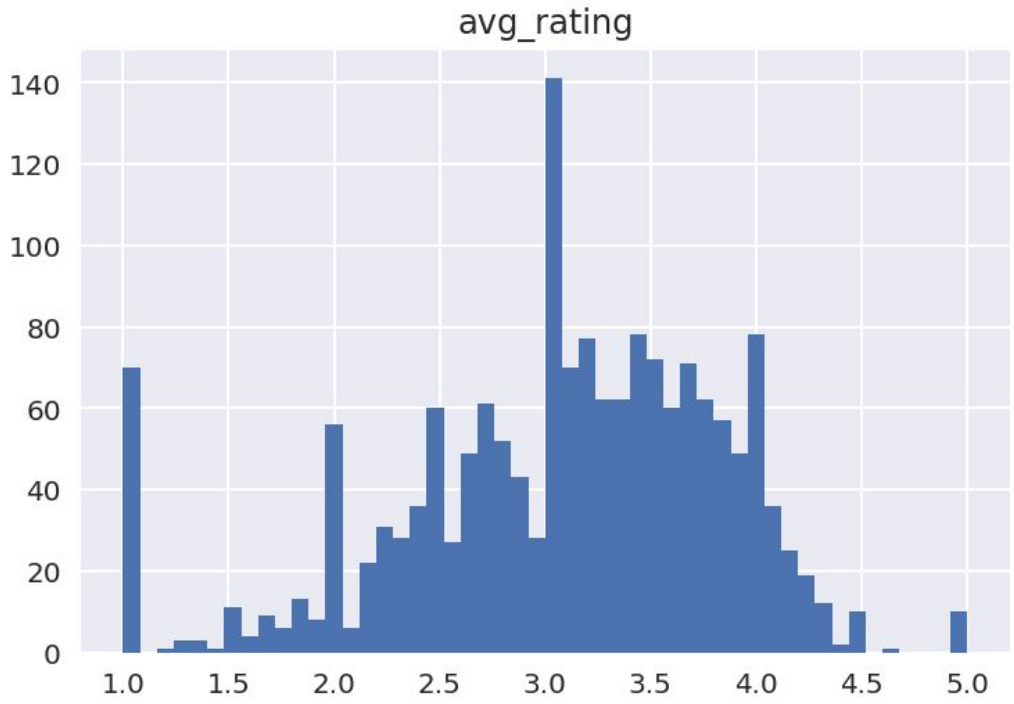
Jialiang Lu
Eduardo Beltrame
David Brown

1.2. Team name

Central Nervous System /ˌsō͞opərˌkaləˌfrajəˌlistikˌekspēˌ aləˈdōSHəs/ 💥🌽🐶
GitHub: https://github.com/dibidave/central-nervous-system

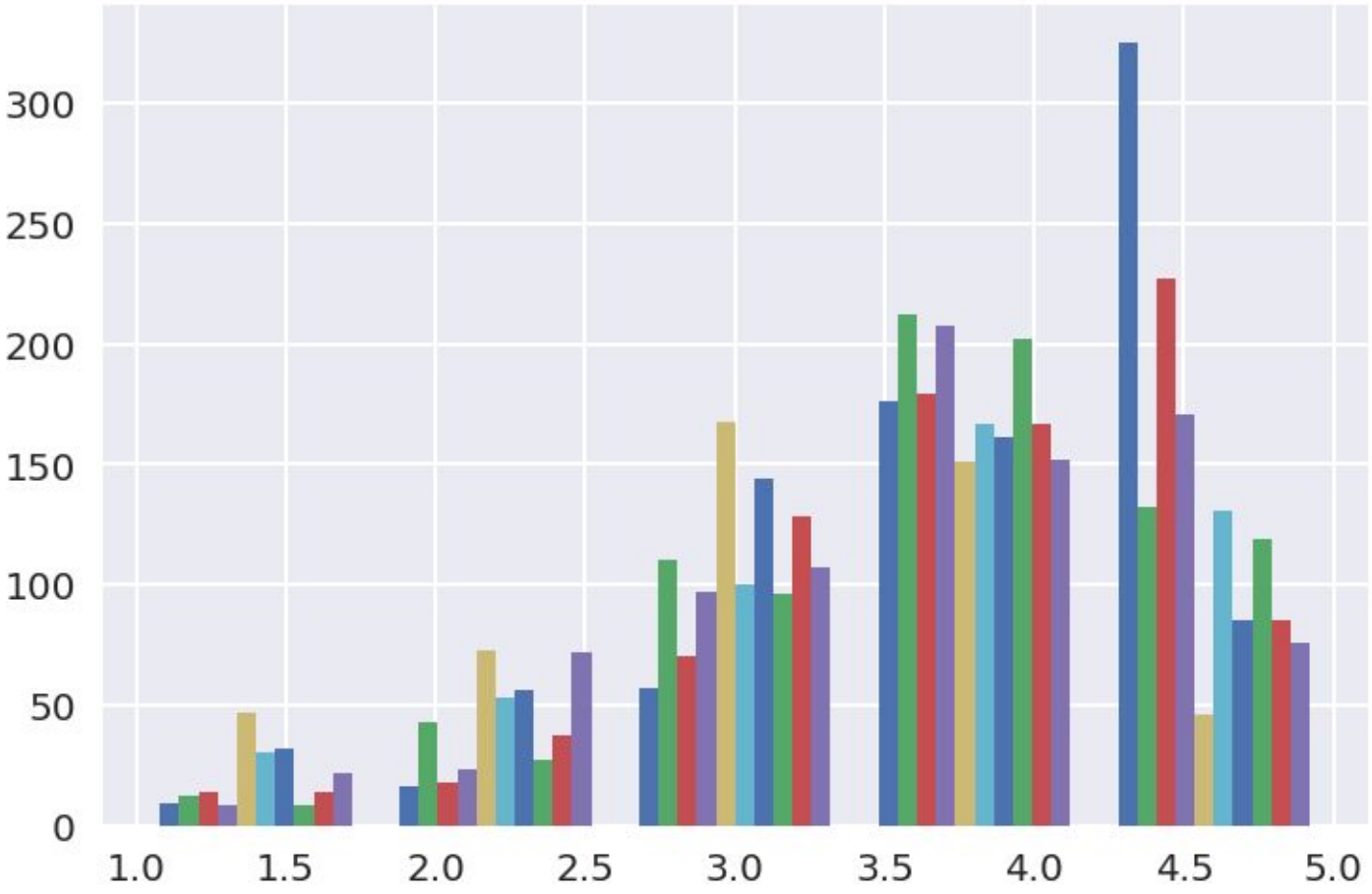
2. Basic Visualizations

2.1 All ratings



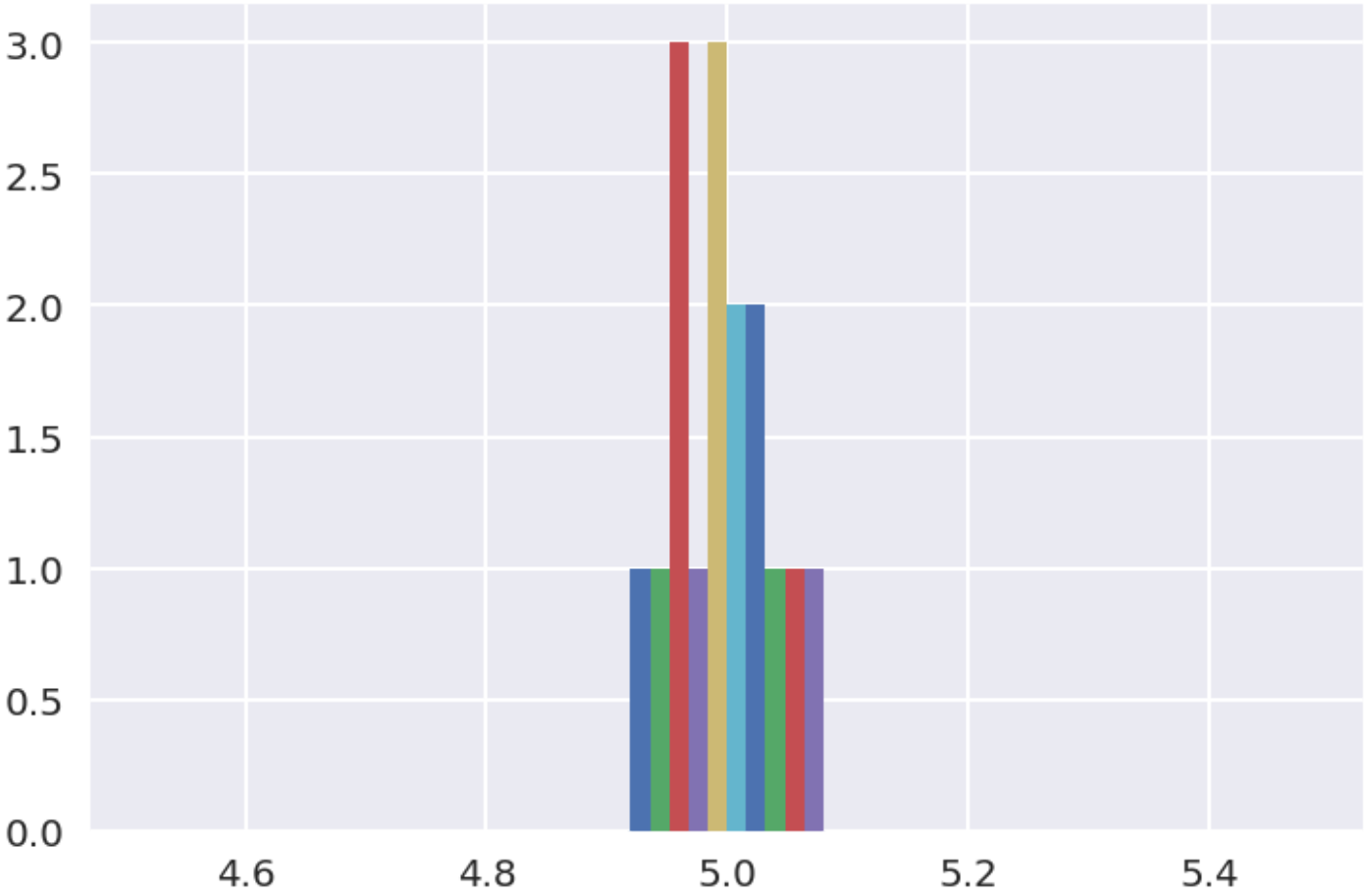
Movie ratings overall seem to be skewed slightly to the right, with a median rating of 4. There is also a much smaller difference between the number of 1 and 2 ratings than there are between other pairs of ratings. These observations concur with the general sense from reading reviews: people tend to be more giving of praise (they only give low ratings when a movie is REALLY bad), and once a movie is bad they are less hesitant to drop all the way to a 1 (“this was the worst movie ever”).

2.2 Ratings of the 10 most popular movies



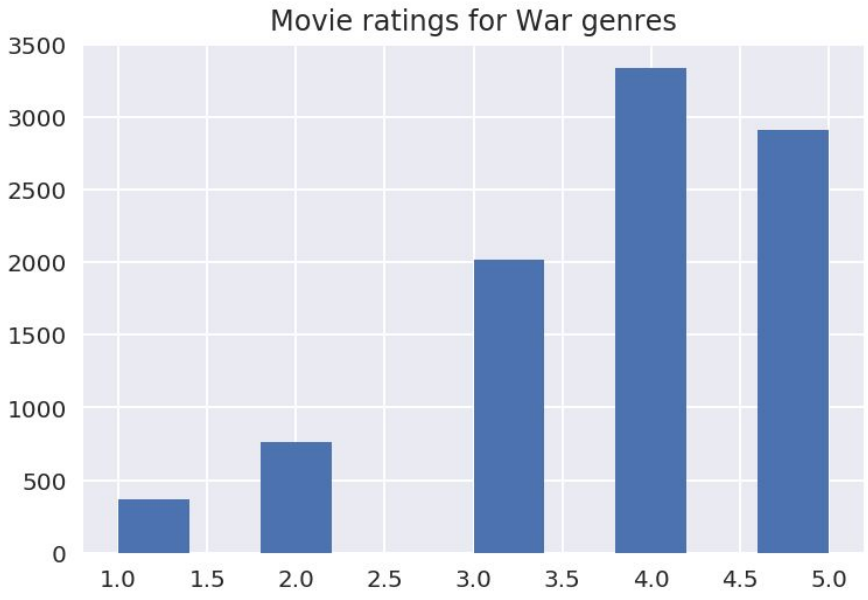
The most popular movies (as defined by having the largest number of reviews) seem to have a similar distribution of ratings as the general grouping of movies. This is intuitive - the most popular movies dominate the histograms of the dataset overall, so their distribution is likely to be similar.

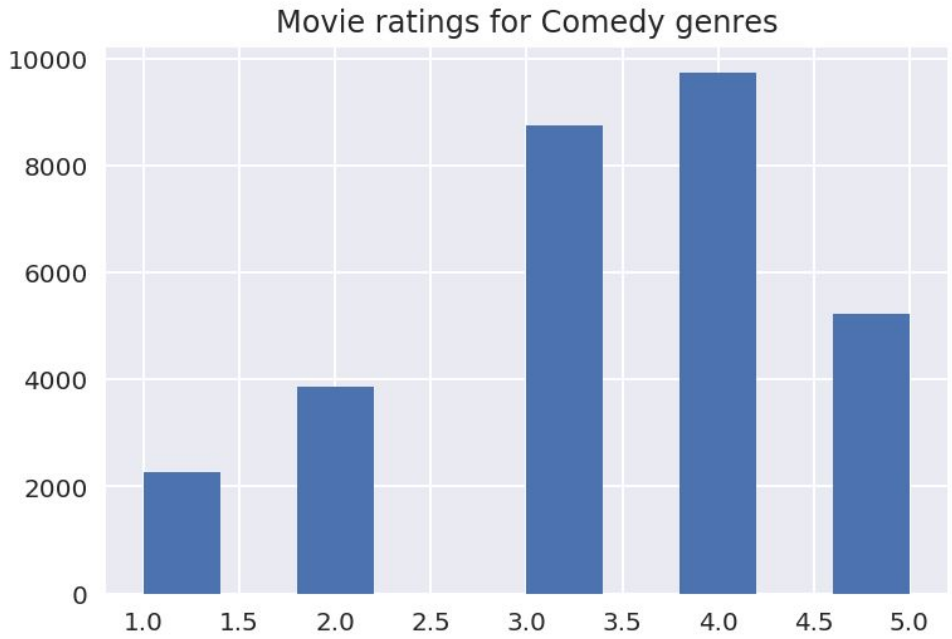
2.3 Ratings of the 10 most best movies

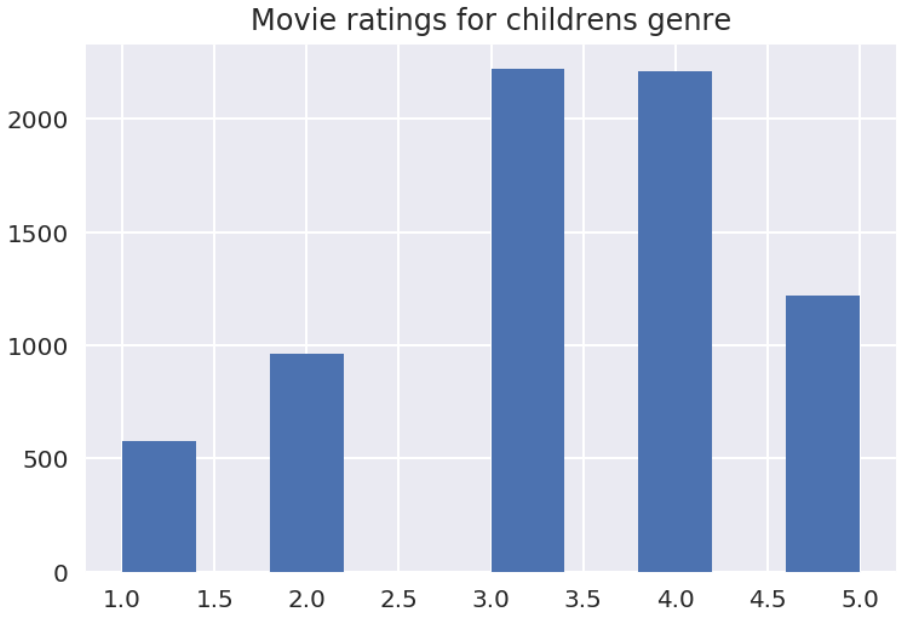


All of the 10 best rated movies had average ratings of 5, and all their ratings were 5. However, these movies had very few ratings overall (3 or less). A better metric would be a joint average and popularity rating.

2.4 Ratings from 3 genres







The ratings for movies in the war and comedy genre followed a similar distribution to all movies. However, the children's genre had a markedly lower number of "5" ratings. This is not surprising, as children's movies are designed to a appeal to a child audience, but it is often the parents leaving the reviews. Children's movies are known to often be of lower caliber, serving the role of babysitter more than being an artistic production.

3. Matrix Factorization

3.1. Theory

Matrix factorization is one method for collaborative filtering. It works under the assumption that the movies and the preferences of the user for the movies, represented by a collection of vectors called latent vectors, live in a common low-dimensional space called joint latent factor space, and the rating of each user for each movie can be approximated by the dot product of the user with the movie.

Let Y be the m x n matrix containing the ratings from m users to n movies, then we used matrix factorization to decompose Y into the product of U^TV where U denotes the latent factor U of size k x m representing each user in the common space and the the latent factor V of size k x n representing each move in the common space. In the extreme case where k equals to the minimum between m and n, the decomposition simply memorizes all different combinations of users and movies and the resulted ratings, with zero prediction error. However, when k decreases, the model squeezes the latent factors into a lower-dimensional space while trying to preserve most of the interactions between users and movies. The learning objective is to minimize the error, here simply the squared error, between the predicted rating and the actual rating of each entry. Optimization is achieved by stochastic gradient descent.

One limitation of the simple matrix factorization is that the interaction between the users and movies captured by the algorithm includes also variations between individual movies or users. We could model the individual preference of a user or the individual quality of a movie by introducing a bias term for each user and movie, and express our prediction as the sum of the product of latent factors plus the bias term for the user and the movie plus a global bias that is the mean rating of all entries. In this way, we effectively “centered” the data and can potentially capture more unbiased interaction between users and movies in the low-dimensional space.

Finally, there are a lot of off-the-shelf packages available for matrix factorization. We tested the python package *Surprise*, of which the *matrix_factorization.SVD* module follows the same algorithm described above.

3.2. Results and performance

To compare the performance of each algorithm, we trained them on the training dataset and tested on the test dataset. For visualization in the following sections, the entire dataset is trained.

Model	**Mean squared test error**
Simple SVD	0.458
Biased SVD	0.418
Surprise SVD	0.815

As explained above, the biased matrix factorization fits the data better than the matrix factorization without the biased term. The external package seems to have a worse performance, probably because we used some default internal parameters that have not been fine tuned.

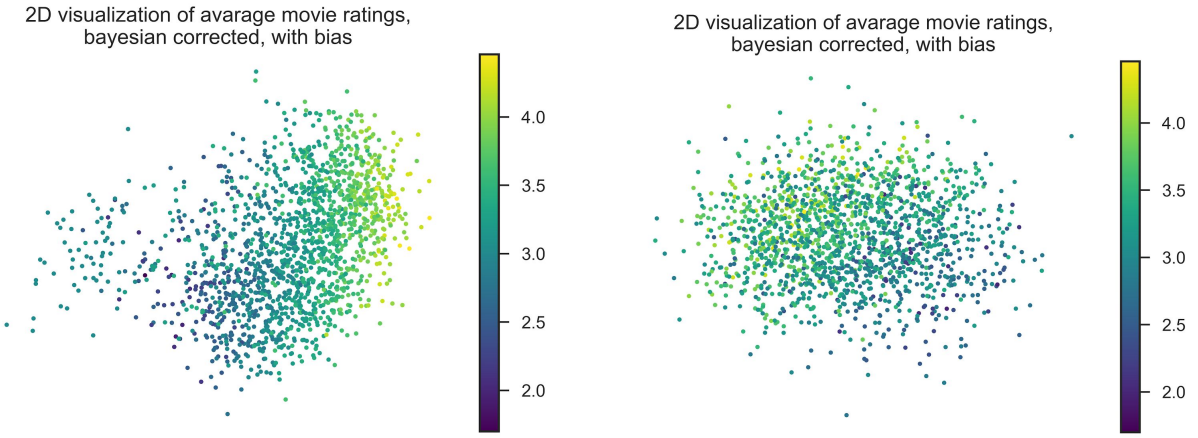
4. Matrix Factorization Visualizations

4.1. Theory

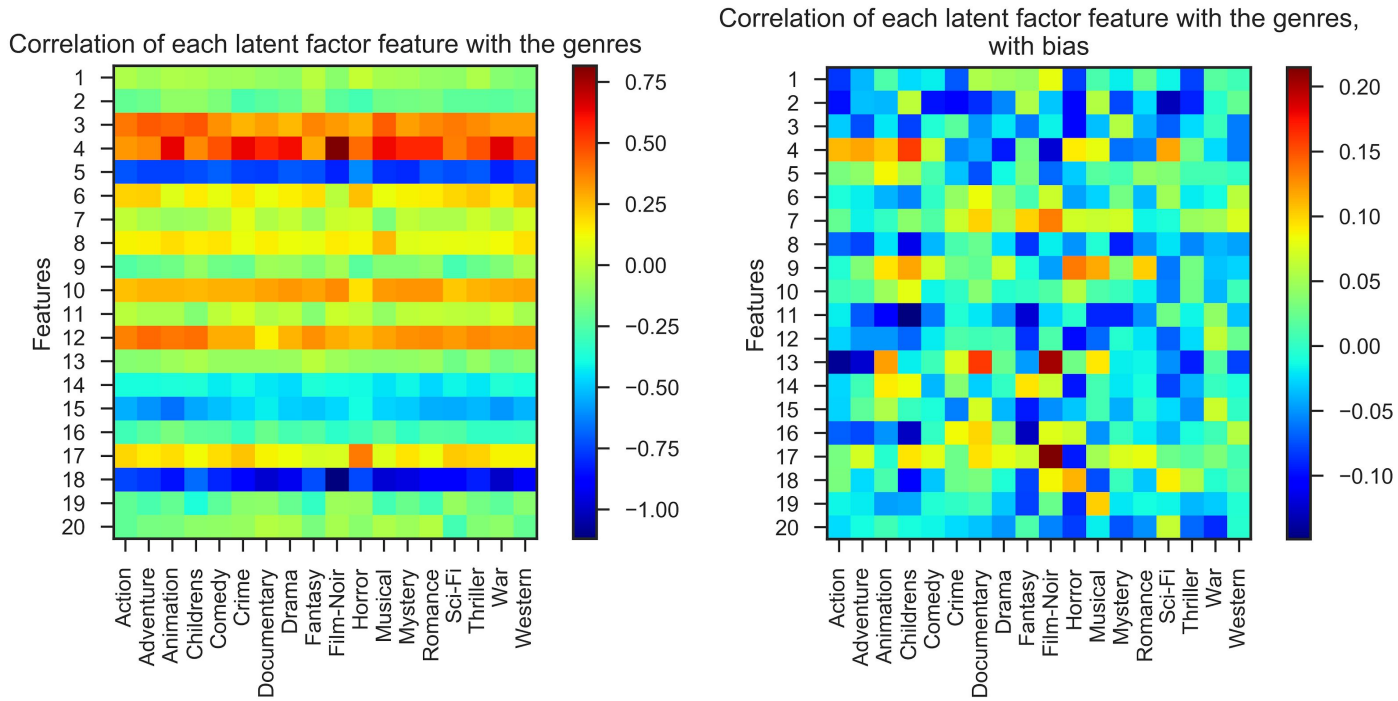
In this section, we focus on the visualization of the movies and the associated latent factor, which is more interesting than the users with no further information. To visualize the 20-D latent factor V for movies, we performed singular value decomposition on V and used the first two left-singular vectors to project V to a 2-D

space. Since these two vectors are associated with the two highest singular values, we were able to capture most of the variance within V in the 2-D space.

4.2. Comparison between simple and biased SVD

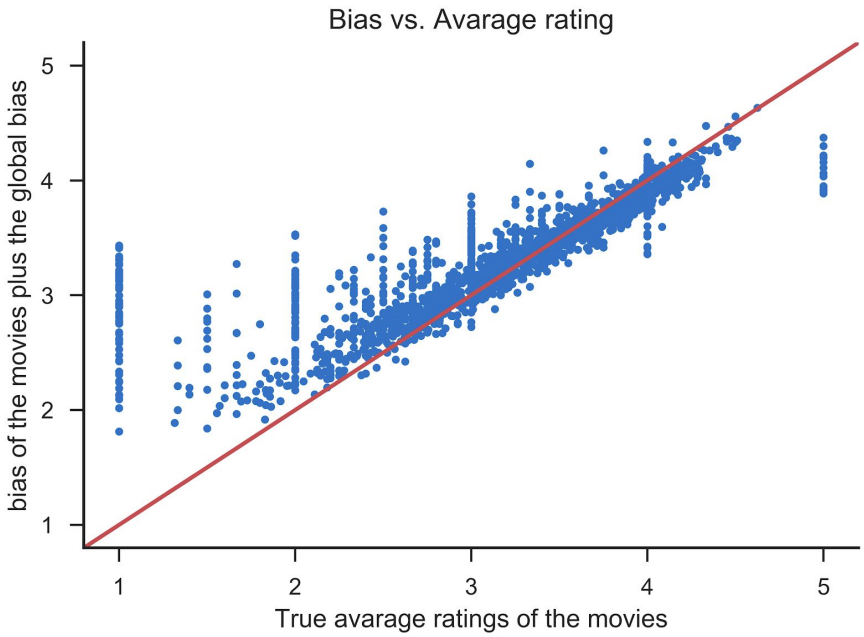


The above two plots shows the 2-D projection of the movie latent factors, either without (left) or with (right) the bias term, color-coded for the bayesian corrected average ratings. It is obvious that the first two principal components of the 20-D latent factors without the bias term account mostly for the average rating. In fact, the first PC (the x-axis) encodes the rating almost completely. In contrast, the two PCs of the 20-D latent factors with the bias term are uncorrelated with the ratings. Both good and bad movies are scattered around the origin.



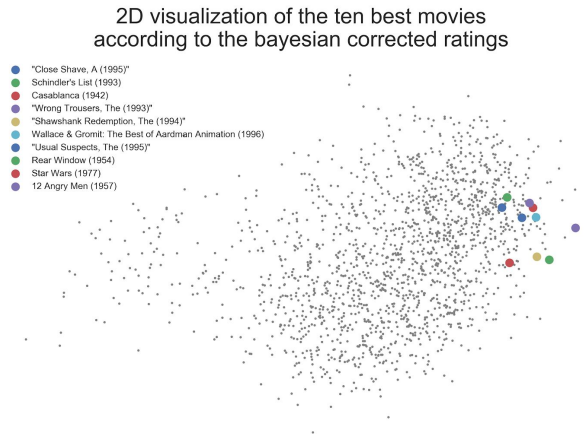
So, what do the 20 features of the 20-D latent factors with the bias term encode. The two plot above compares the correlation between the 20 features and the 18 genres (excluding the *unknown* genre), for the unbiased and biased latent factors. Clearly, the unbiased version is poorly correlated, if any, with the genres of the movies, while the biased version maximally correlated with different combination of genres and points toward

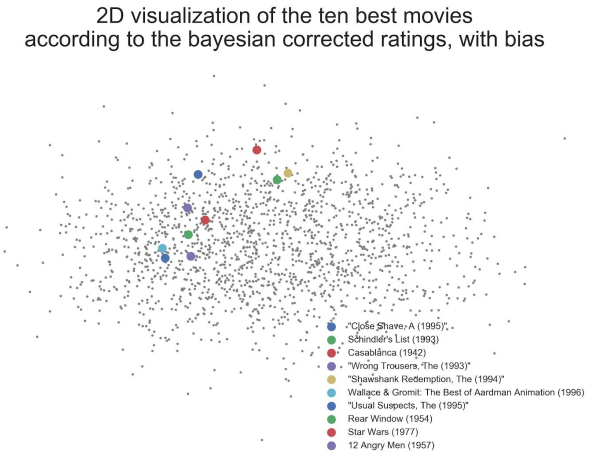
very different direction in the genre space. Therefore, the biased latent factors are able to capture more the style of each movie and the taste of each user and capture their interaction.



Being extracted from the latent factor features, the average rating of the movies is encoded in the bias term for the biased SVD. This figure demonstrates the difference between the actual average rating of the movies and the sum of the bias term for the movies plus the global bias. To some extent, the bias term can be viewed as the expected value of the average rating, therefore approximated the average rating very well when the number of ratings is high. Interestingly, the bias for movies with very low number of ratings was closer to the global bias than the actual average. This was because the global bias was the “prior” knowledge for any movie.

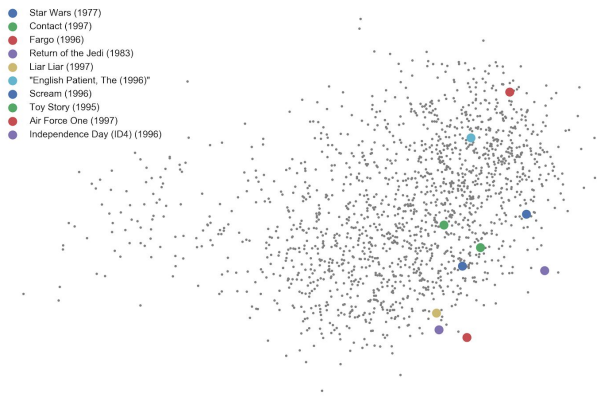
4.3. Best vs. popular movies



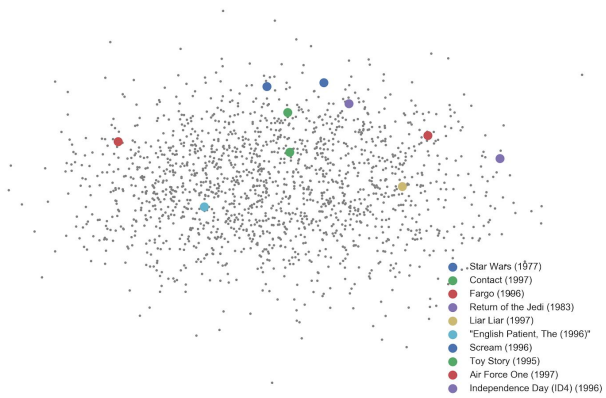


Consistent with the explanation in the previous paragraph, the ten best movies appears on the very end of the population for the simple matrix factorization, while rather randomly distributed in the biased matrix factorization.

2D visualization of the ten most popular movies

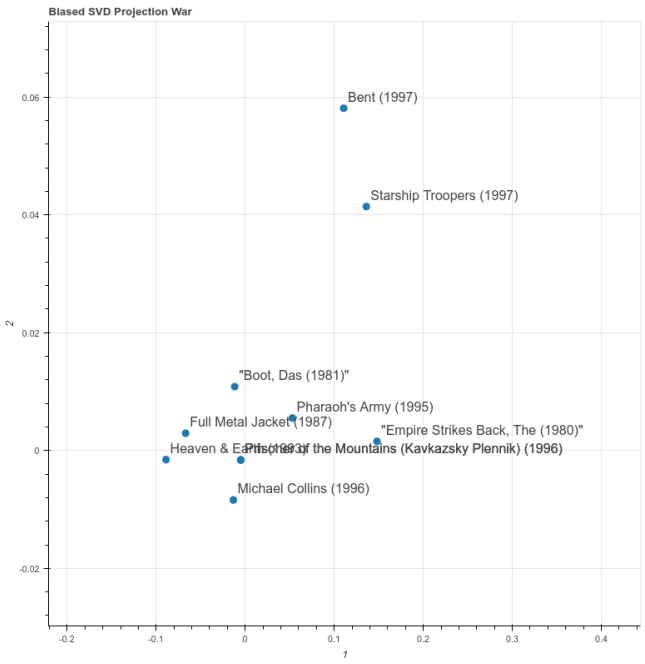


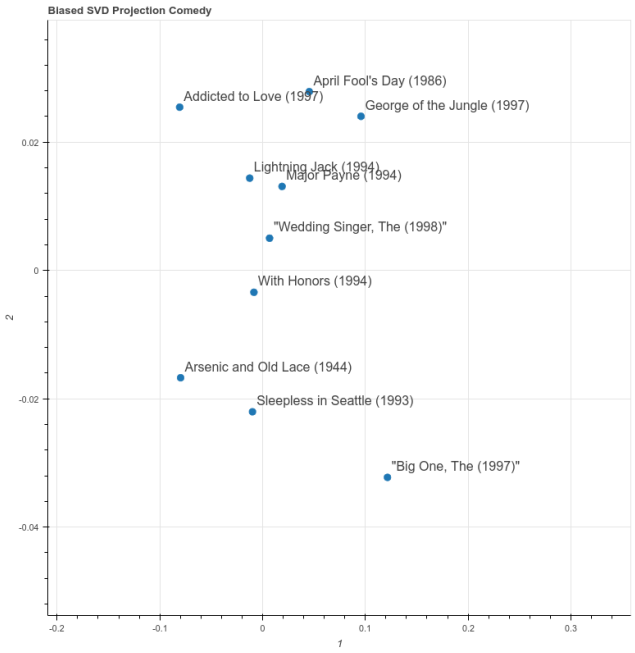
2D visualization of the ten most popular movies, with bias

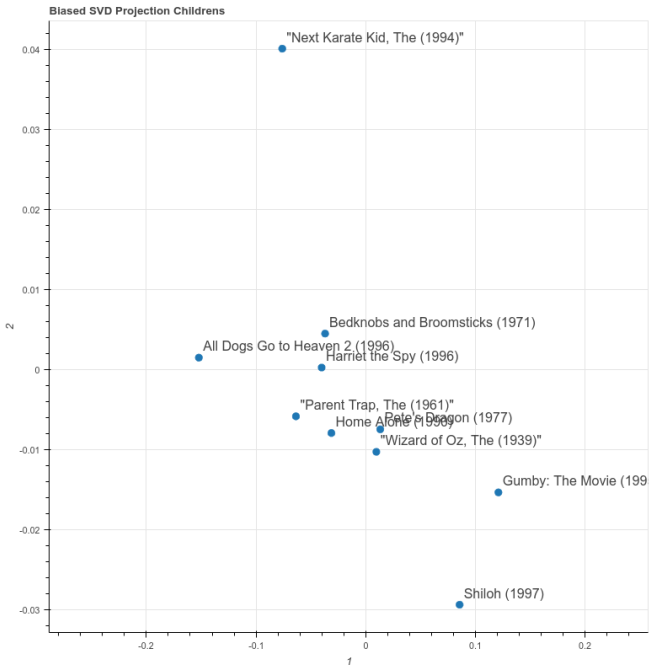


In the simple matrix factorization, popularity and rating seems to be weakly correlated, because the popular movies seem to be on the same part of the population as the best movies. For the biased matrix factorization, it remains uncorrelated.

4.4. Ten movies and three genres selected







We looked at projections of 10 random movies from 3 different genres: War, Comedy, and Children's movies. There don't seem to be any noticeable patterns that come out that are consistent between genres.

5. Conclusion

Matrix factorization is a great tool to model the interaction between "users" and "items" in a simple and logical way and to generate meaningful visualization of the features of the resulting latent factors. Introducing the bias

term to the matrix factorization removes the individual general bias of users and items and is able to fit the testing data even better, since it reduces the effect of overfitting.