1 Introduction

- **Group members**
  Aw Young Qingzhuo, Ola Kalisz, and Riley Patterson
- **Team name**
  Not (Chicken [3]) Hotdog
- **Code**
  `https://github.com/veniversum/cs155-projects`, see especially `project2/src`
- **Division of labour**
  We all independently did the "basic" visualizations before deciding on a final view of them to present here.

  Ola implemented the "off the shelf" SVD and looked into the difference between using U, V from $Y \sim= U\Sigma V$ vs. $Y \sim= UV$ and into the test error for this vs. the HW5 model.

  Qingzhuo worked with models incorporating biases and created visualizations colored by age and genre, including an interactive one.

  Riley worked with models using his hw5 solution and wrote code for finding/labeling outliers in the 2D projections as well as coloring by age, genre, and buckets of genres.

2 Overview

- **Models and techniques tried**

  We visualized a 2D projection of the movies data using each of the following:

  - Matrix Factorization as implemented in HW5, where $Y \sim= UV^T$, i.e. $U = U_{SVD}\sqrt{\Sigma}$ and $V = V_{SVD}\sqrt{\Sigma}$.
  - Matrix Factorization models including bias. Implementation using Surprise SVD, where a single prediction of users u rating on movie m is $p_{um} = \mu + b_u + b_m + p_m^T q_u$, μ is a global bias and b_u, b_m are respectively user and movie specific biases, and q_u is the user factors and p_m is the movie factors.
  - SVD from SciPy, where we used U_{SVD}, V_{SVD} in $Y \sim= U_{SVD}\Sigma V_{SVD}$.

- **Work timeline**

  - **Feb 21:** Riley: code and results from basic visualizations, some helper code for loading features of the movie dataset and querying them
  - **Feb 21:** Ola: also worked on some basic visualizations and experimented with results from HW5 implementation
  - **Feb 21:** Qingzhuo: worked on SVD with biases using `surprise` library

- **Feb 22:** All: worked on various matrix factorization models as described in "division of work," producing many of the plots to be used in the report and exploring lots of possible correlations using interesting colorings and labelings of the graph.
- **Feb 23:** All: Produced final visualizations and report using learnings from previous day.
- **Feb 23:** Qingzhuo: Created the interactive visualizations for different matrix factorization methods.

3 Approach

- **Data processing and manipulation** We loaded the movies into a dictionary that allowed more intuitive querying and comparison between genres and other data implementations for movie features. `pandas` was used to load the datasets into dataframes. This allowed easy manipulation of data via slicing, group by, and aggregations. For some of our models, we had to shift the id spaces by 1 to get a 0-indexed space rather than a 1-indexed one.

- **Details of models and techniques**: We decided to experiment with three different implementations of matrix factorizations described below. In our experiments we were the most interested in examining how using regularization and bias terms affect the prediction.

  - **HW5 Implementation:**
    We focused on examining how different regularization λ affect the predictions. We were mostly comparing the predictions by visualizing them. We used the learning rate $\eta = 0.03$, and the stopping condition we used was the relative loss reduction compared to the first epoch is less than $\epsilon = 0.0001$ or the number of iteration is greater than 300.
  - **Surprise SVD with and without bias:**
    We left most of the parameters as default for SVD and SVD++. `n_factors` was set to 20 and we used 50 epochs. We tried naive SVD models without the bias term, as well as ones incorporating global and user/movie biases. The regularized error to minimize is $\sum_{r_{ui} \in R_{train}} (r_{ui} - \hat{r}_{ui})^2 + \lambda \left(b_i^2 + b_u^2 + ||q_i||^2 + ||p_u||^2\right)$ and SGD is used as the optimizer. SVD++ [1] is an extension of SVD which takes into account implicit ratings, that is the fact that an user rated a movie, regardless of what rating they gave. The prediction is given by $\hat{r}_{ui} = \mu + b_u + b_i + q_i^T \left(p_u + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_j\right)$
  - **SVD from SciPy:**
    SciPy offers a very simple implementation of SVD. It does not include any regularization or bias terms. We decided to compare the results achieved with SciPy SVD to the results for HW5 implementation. SciPy returns a prediction of user/movie matrix X_{pred} in the form $X_{pred} = U \Sigma V^T$. To be able to apply the SVD to the matrix, using the user/movie matrix Y we create a train matrix X_{train} where x_{ij} is the rating of user i of film j for the known ratings and 0 if the rating is unknown. This approach is not expected to give a high quality prediction as we 'imputate/pad' the train matrix with 0 [2]. The test error that we received with this method reflects the low quality of the prediction.

4 Visualizations and Results

Basic Visualizations

- **Histogram of All Ratings in the MovieLens Dataset**

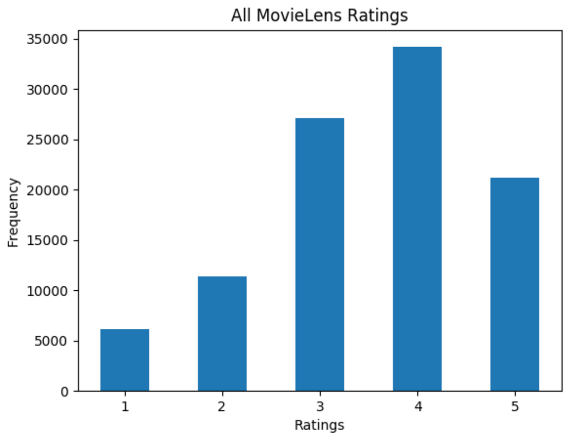


Figure 1: Here we see a distribution biased towards higher ratings, which suggests either that people have a systematic bias to think all movies are above average or that there's a selection bias in the movies that they end up seeing and rating.

- **Histogram of Ratings for top 10 most frequently rated movies in the MovieLens Dataset**

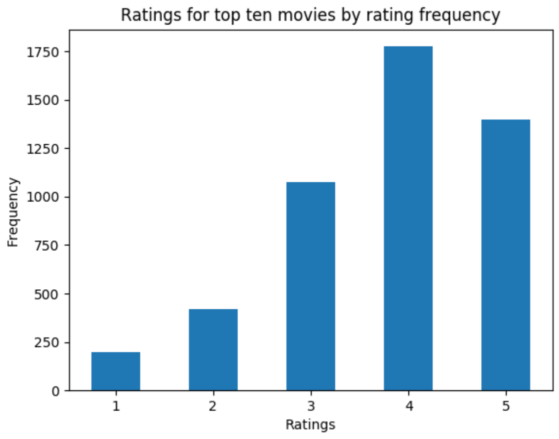


Figure 2: Here we see a distribution even more biased towards higher ratings than for all movies, which suggests some form of positive feedback where a movie which receives higher ratings on average also ends up getting more viewers/raters to rate it.

- **Histogram of Ratings for top 10 movies by average rating in the MovieLens Dataset**

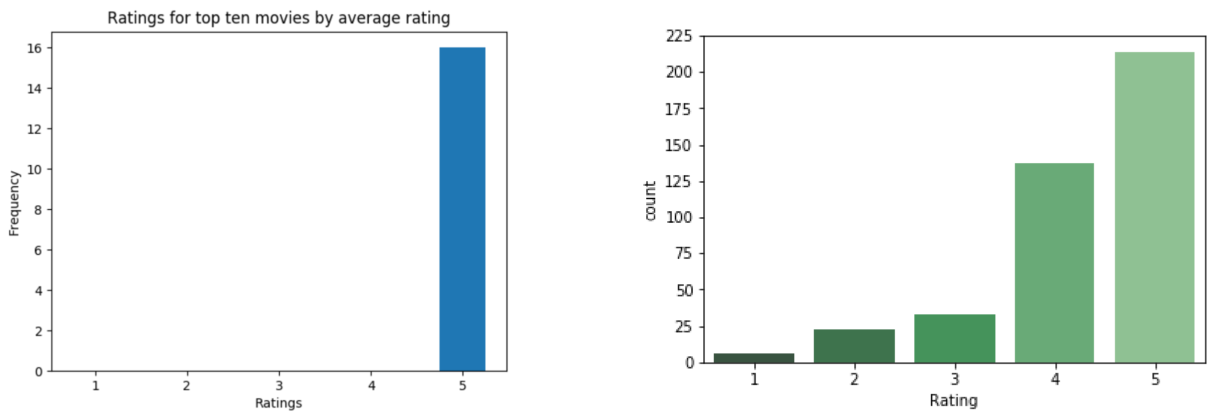


Figure 3: When we look at movies that strictly match the definition of highest average rating, we find movies that have been exclusively rated 5 stars, but with very small numbers of ratings each (sometimes just one). We also looked at the top ten movies *having at least 50 ratings*, for which we found a bit more interesting results but still with an obvious strong bias towards higher ratings compared to the whole dataset.

- **Histograms of Ratings for all movies in three different genres**

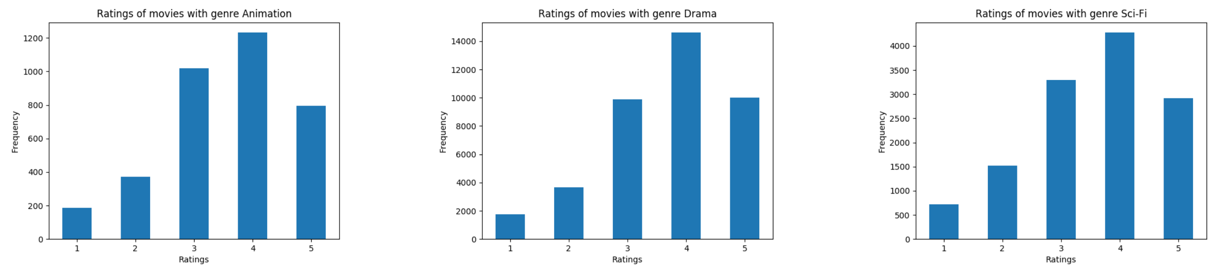


Figure 4: Looking at movies from three different genres, we see that Drama movies have the highest proportion of positive (4- and 5-star) reviews among the three, and Sci-Fi movies, while having a larger proportion of 5-star reviews than Animation movies, also have notably greater proportions of 1- and 2-star reviews, exhibiting higher variance of quality and/or taste.

Matrix Factorization

Comparison of Models

- **HW5 Implementation** HW5 Implementation Out-of-Sample Mean Squared Error: 0.605. For the HW5 latent factor model implementation, we found that a regularization of 0 with 300 epochs produced the most interpretable plots, and a learning rate of 0.03 gave good results, though varying this within reason had very little impact on the error or the interpretablility of the plots. The regularization impact on interpretability is probably mostly due to the fact that 90% of the movies in our plots are from the training dataset, so overfitting may end up showing stronger results for those at the expense of the other 10%, resulting in an overall more interpretable plot.

- **Surprise SVD with and without Bias**
  Surprise SVD Unbiased Out-of-Sample Error: 0.481
  Surprise SVD w/ Global and Term Bias Out-of-Sample Error: 0.476
  Surprise SVD++ Out-of-Sample Error: 0.418
  With Surprise SVD implementations, we mostly experimented with the impact of bias and SVD++. We found modest improvements in test error by including bias terms, and slightly better improvement using SVD++ rather than SVD. However, these improvements didn't translate into substantially more interpretable projections of the data.

- **SVD from SciPy**

  SciPy SVD Out-of-Sample Mean Squared Error: 3.13. In comparison with other models use the SVD SciPy error is high, as expected since we 'padded/imputatated' the training matrix with 0s for unobserved user, movie rating pairs. However, we have noticed that plotting matrix V from $X_{pred} = U \Sigma V^T$ as a movie prediction with 20 latent factors (projected to 2 dimensions) gives us quite interpretable results. We found plotting the results from SciPy SVD more interpretable than plotting the results from HW5 implementation.

Visualizations

- **Ten movies chosen by us**

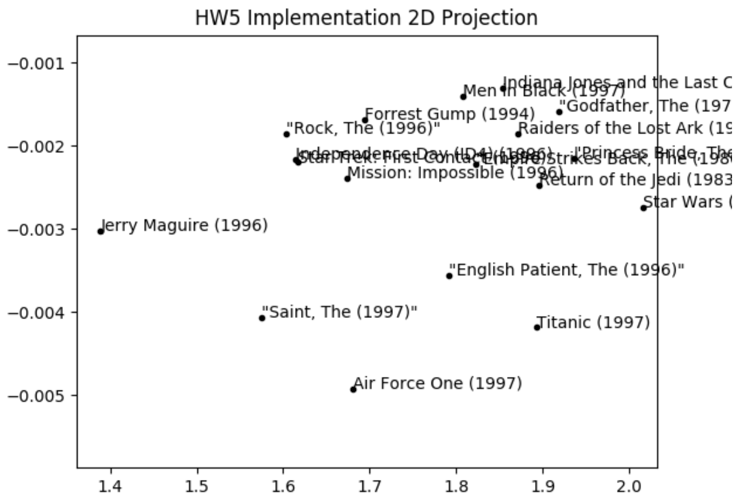


Figure 5: Action and Romance films projected using V from the HW5 implementation of matrix factorization. Here we can see many of the Romance movies in the bottom half of the plot (with the exception of Air Force One), and a cluster of Action movies at the top, suggesting the difference between these genres is correlated with the dimension represented in the y-axis.

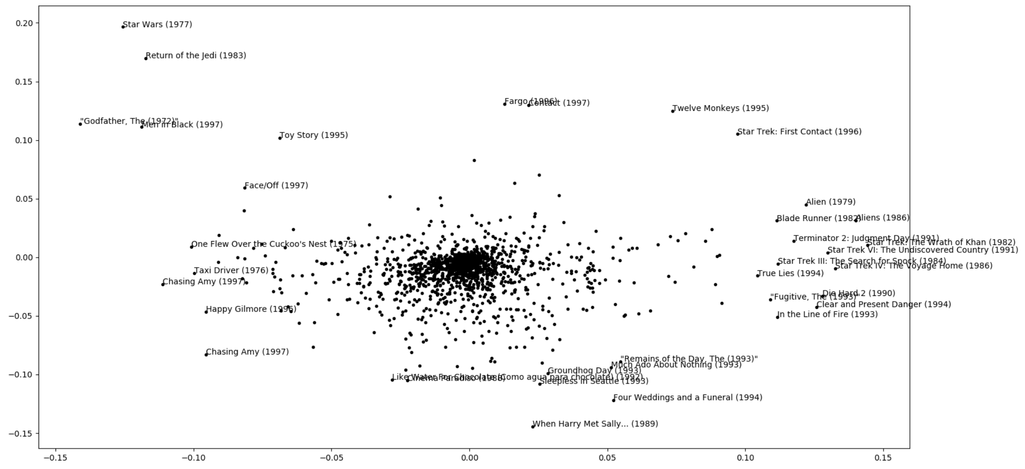
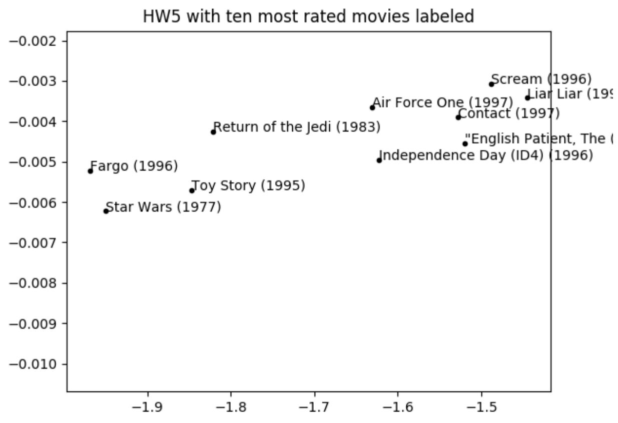


Figure 6: All movies projected using V from SciPy SVD, with the most distance movies from the origin labeled. Here we see a number of clusters of similar movies, including comedies in the bottom left, Sci-Fi in the right, and more "artsy" movies in the bottom.

- **Top ten most frequently rated movies**



Top ten most frequently rated movies plotted with the projection of V from the HW5 implementation. We see notably less interpretable structure here than with the SciPy SVD implementation (below).

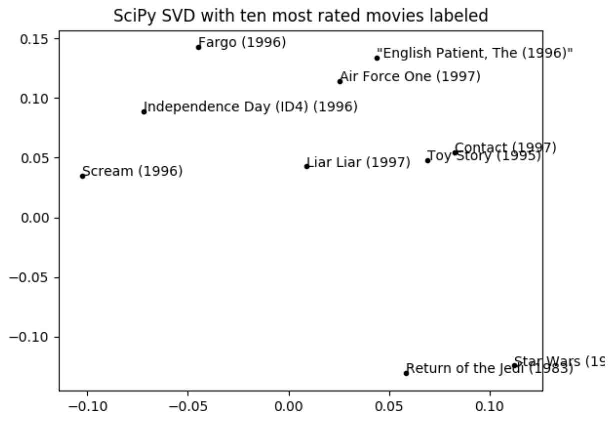


Figure 7: Top ten most frequently rated movies plotted with the SciPy SVD projection. Here we see a clear cluster in the bottom left where the Star Wars movies appear together, and some distance from the romance movies on top.

- **Highest average rated movies**

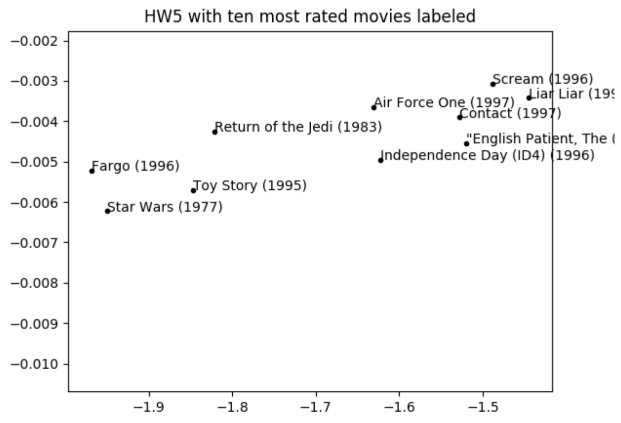


Figure 8: Top ten most frequently rated movies plotted with the projection of V from the HW5 implementation. Here it is difficult to find a correlation, likely due to the small number of ratings (as few as one) for these movies results in less data to distinguish them.

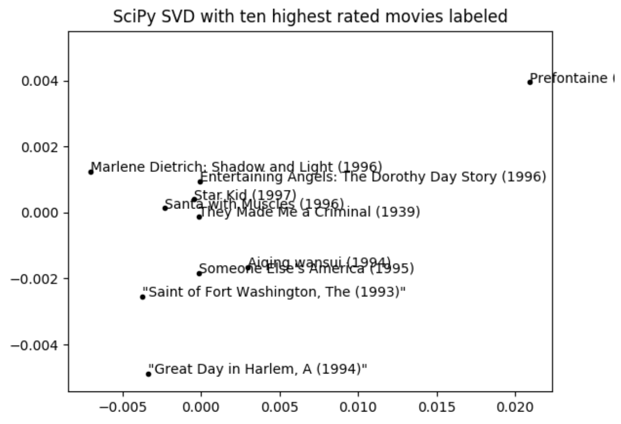


Figure 9: Top ten rated movies plotted with the SciPy SVD projection. As above, it is difficult to interpret the meaning of the dimensions here due to low amount of data for these movies.

- **Ten movies from three different genres**

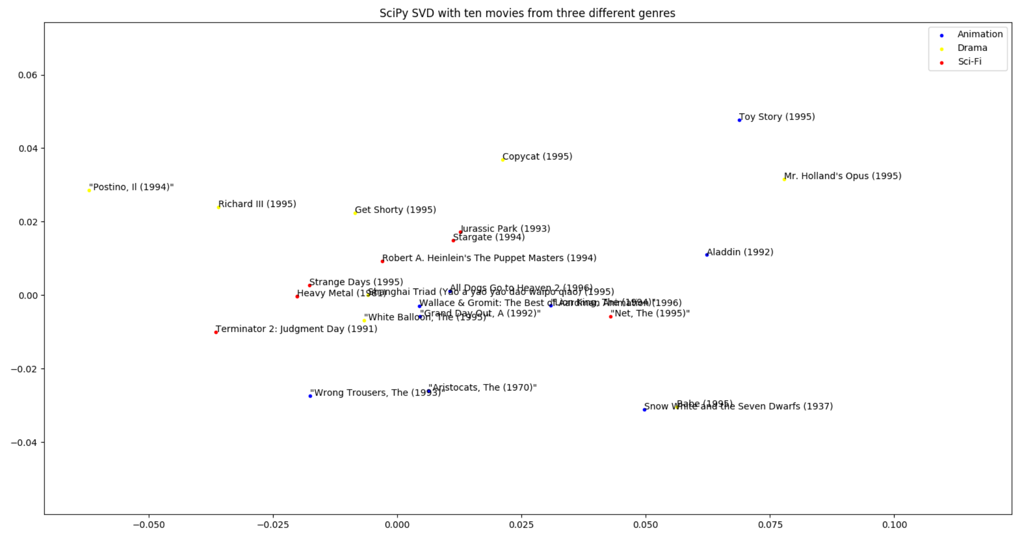


Figure 10: Ten movies each from Animation, Drama, and Sci-Fi, plotted with the SciPy SVD projection. Here we see that most of the dramas are higher on the y-axis, while Sci-Fi is largely lower on the x-axis than Animation.

For the genre plots we notice that the movies in the same genre are generally clustered close to each other when we project the prediction to 2-dimensions. In every genre there are some single outliers but the general trend that we notice is as described, which indicates that the movies in the same genre are similar, as expected.

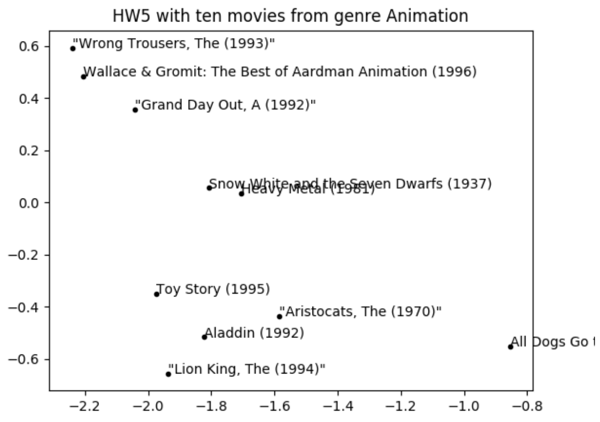


Figure 11: Ten movies from Animation, projected using V from the HW5 implementation. hHere we see a cluster of Disney moves towards the bottom and a cluster of non-Disney movies towards the top.

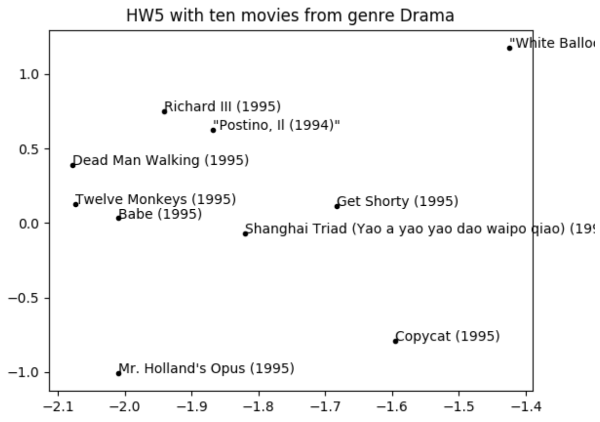


Figure 12: Ten movies from Drama, projected using V from the HW5 implementation. There's a small cluster of historical movies, but otherwise not too much in the way of interpretable results here.

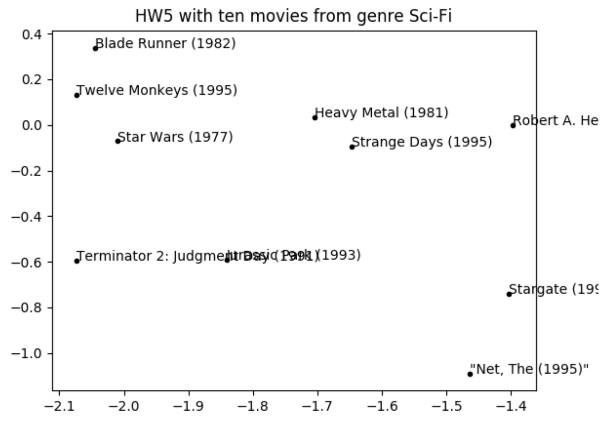


Figure 13: Ten movies from Sci-Fi, projected using V from the HW5 implementation. There appears to be a trend on the x-axis with more popular Sci-Fi to the left (like *Star Wars* and *Terminator*), and nerdier Sci-Fi to the right.

- **Other**

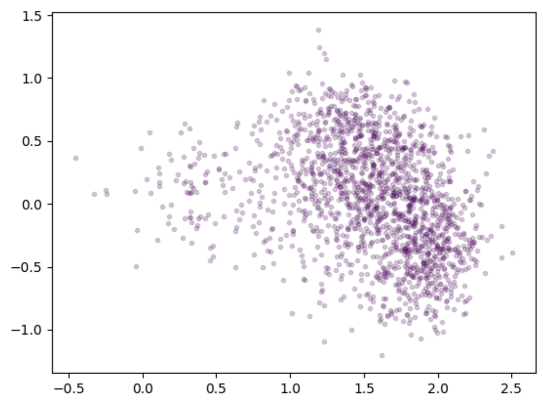


Figure 14: All movies from the HW5 implementation-based 2D projection, colored with more recent movies being darker. Here we see a correlation with age on the x-axis.

Surprise SVD with and without Bias, SVD++

The interactive visualization for the different Surprise SVD implementations can be found here: `https://veniversum.me/cs155-projects/`.

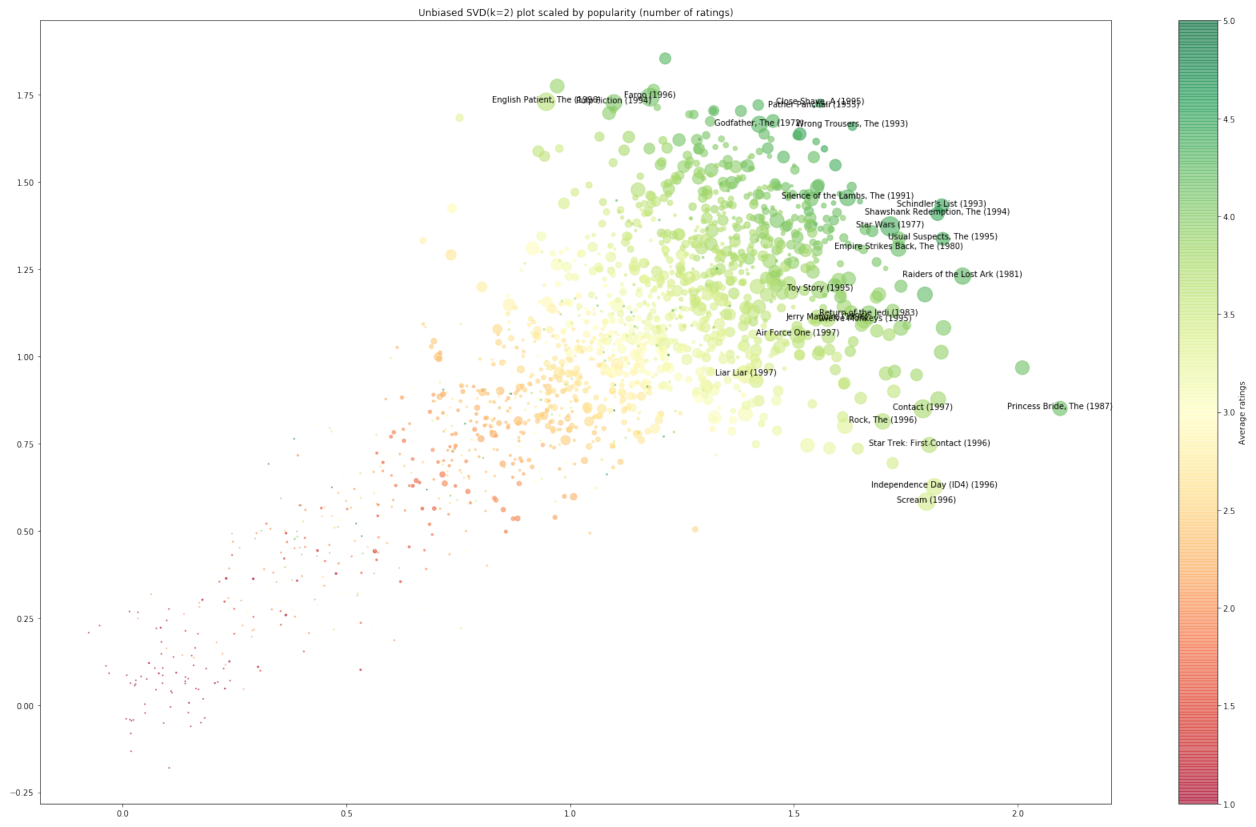


Figure 15: All movies colored by average rating and scaled by number of ratings, as projected by Surprise SVD, without using any bias terms.

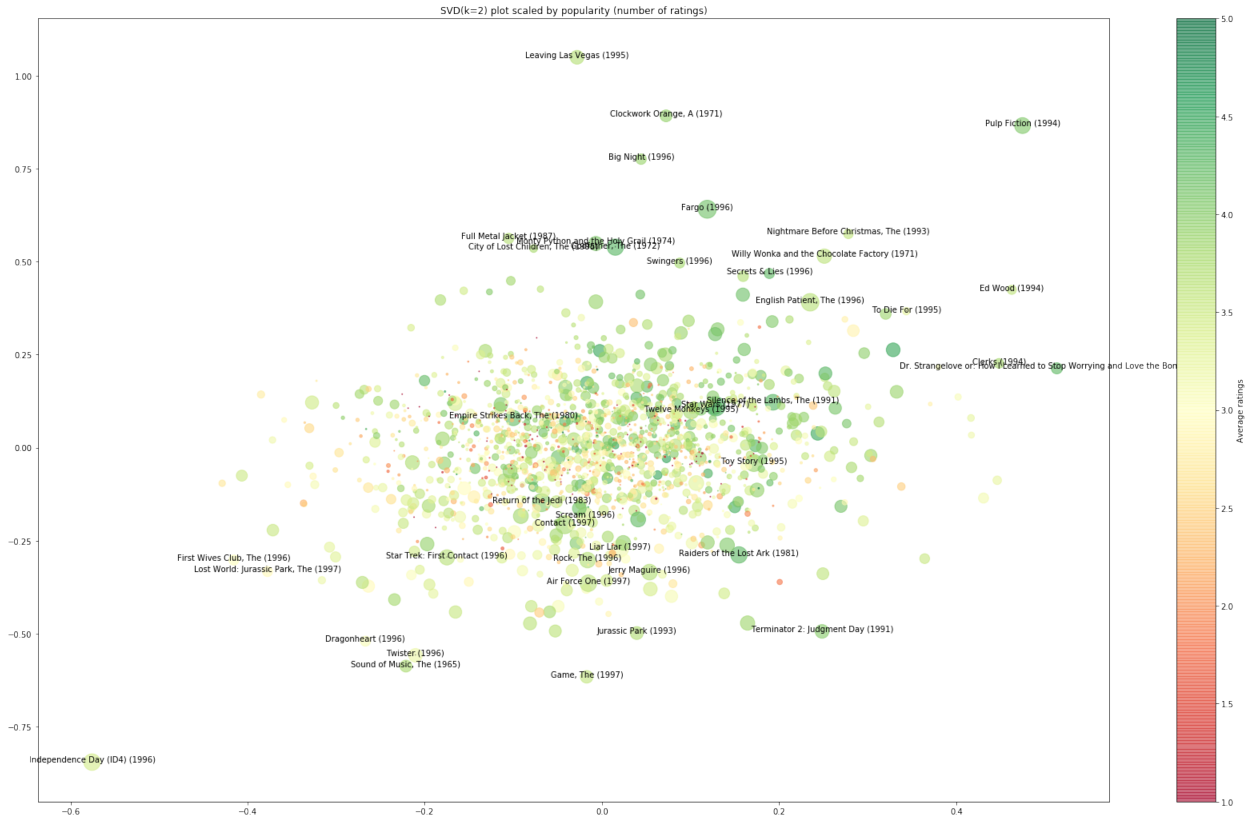


Figure 16: All movies colored by average rating and scaled by number of ratings, as projected by Surprise SVD with global bias as well as user/movie specific bias which are deviations from the global bias.

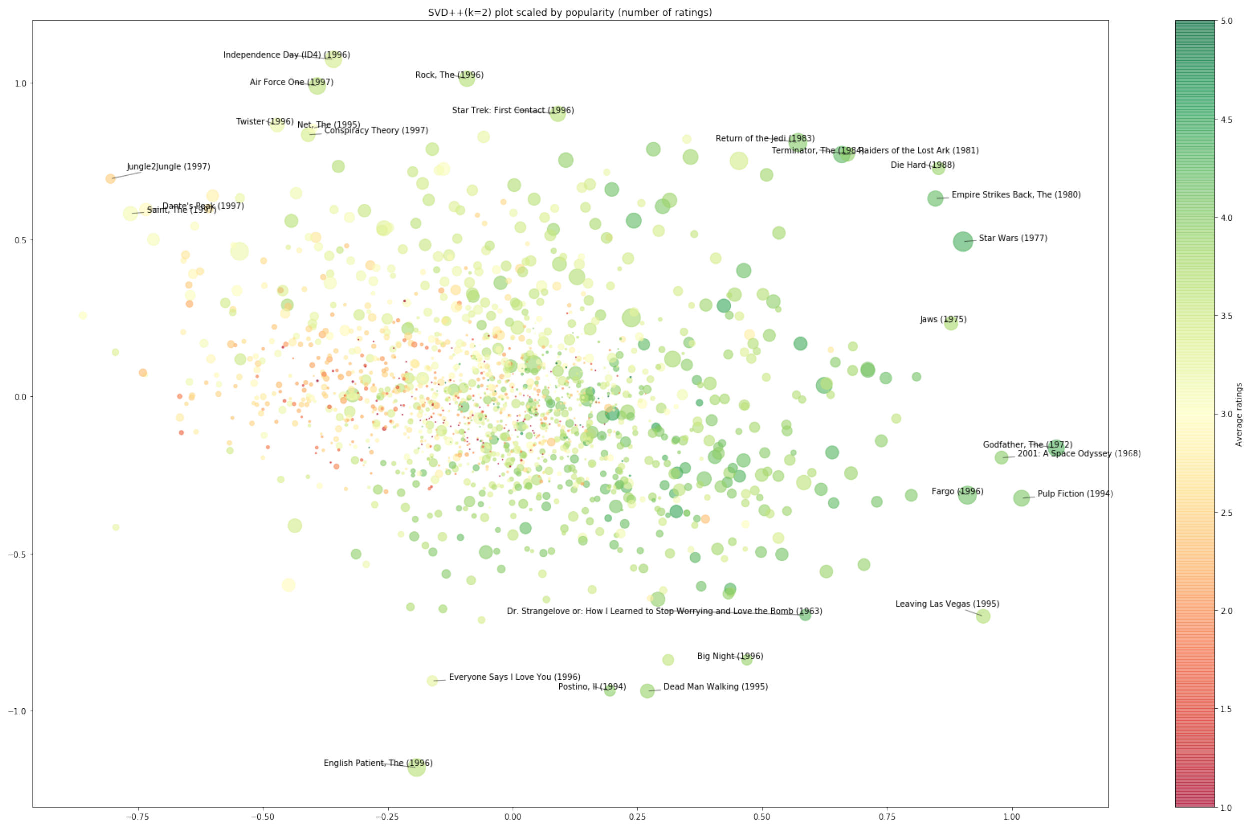


Figure 17: All movies colored by average rating and scaled by number of ratings, as projected by Surprise SVD++ with global bias as well as user/movie specific bias which are deviations from the global bias.

5 Conclusion

- **Discoveries**

  Different SVD implementations gives quite different results. Notably, the `surprise` package and the homework implementation use SGD with mean square error, while `scipy` implementation use the ARPACK eigensolver which appears to perform a different optimization. This difference meant that MSE comparisons involving the SciPy implementation had less meaning (because it is not optimizing for MSE), and the plots produced with SciPy were quite different from those with SGD on mean square error.

  Additionally, we learned that introducing more complexity to models in the form of global and local bias terms, or by using a different model altogether with SVD++ and its use of implicit user signal, allowed for slightly lower error but had a slightly negative effect on interpretability of visualizations.

- **Challenges**

  With all the models we found out that interpreting the visualizations is challenging. It is hard to determine if the latent factors of the model clusters movies to some logically justifiable categories. We found it difficult to draw connections between movies and their closest neighbors, but it might just be because we don't watch many movies.

Handling indexing was difficult in a few respects:

– Our use of zero-indexing in some models and one-indexing elsewhere led to several confusing bugs.

– Surprise performs and id-space remapping internally and also makes it difficult to deal with splits of data where the training set does not contain examples for ids found in the test set. Not applying this mapping correctly sometimes led to errors in labeling and coloring our graphs by various movie features.

- **Concluding Remarks**
  Matrix factorization clearly provides a powerful way to decrease dimensionality of data and thereby reason about it more easily. Using it effectively, however, is not trivial – projections are very sensitive to small changes in models and parameters, and sometimes the dimensions being captured are ones that could easily be extracted directly from the data, as was the case with age of movie and average rating correlations that we found with some of our plots. When used carefully, it can help surface clusters that would otherwise be difficult to extract from the input data.

References

[1] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008.

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

[3] Doug Zongker. Chicken chicken chicken: Chicken chicken. *Annals of Improbable Research*, 12(5):16–21, 2006.