Machine Learning & Data Mining
Miniproject 2: MovieLens Visualization

Caltech CS/CNS/EE 155
Released February 16st, 2022

---

1 Introduction

Our group is composed of Jason Yang, Pierre Walker, James Wang, and Cyrus Fiori. Our team name is *The Navier Folks*.

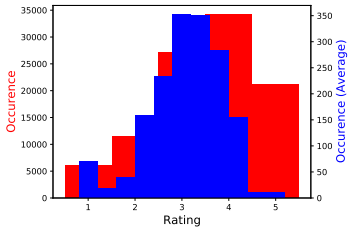
Tasks were divided as follows:

- Jason Yang: Addressed parts 2 and 3 for the matrix factorization visualizations (both using surprise), wrote up relevant sections.
- Pierre Walker: Completed the basic visualizations and wrote up that section and the discussion section.
- James Wang: Addressed part 1 of the matrix factorization visualizations by modifying the code from HW 5, wrote up parameter justifications for methods.
- Cyrus Fiori: Addressed part 1 of the matrix factorization visualisations by modifying the code from HW 5 and wrote up the matrix factorization method description sections, edited report.

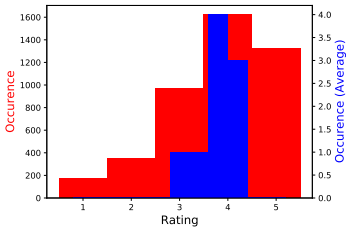
We used the following packages: numpy, matplotlib, sklearn, pandas and surprise.
Link to code

2 Basic Visualizations

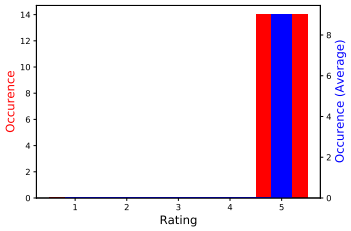
The following plots were generated for this section:



(a) All movies

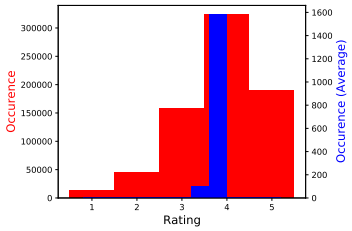


(b) Most popular movies

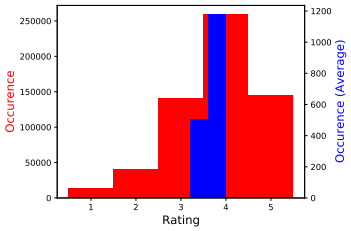


(c) Best movies

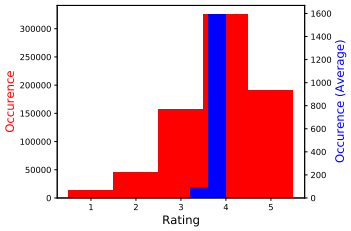
Figure 1



(a) Sci-fi



(b) Comedy



(c) Horror

Figure 2

In general, most of the distributions are skewed towards higher ratings, with the mean typically being between 3 and 4. The only exception to this was the top 10 highest rated movies where, as we separated movies by their ratings, it is unsurprising that all ten movies have a rating of 5. However, if we examine the average ratings per movie, we find that the is a larger number of movies with ratings close to 1.

Nevertheless, examining the most popular movies, we find that the average is slightly more skewed towards higher ratings than when examining all movies. This is not too unsurprising as, logically, if a movie is popular, it would have higher ratings (although a similar argument could be made for controversial movies). Thus, this matches our expectations. This distribution does not match the one for the ten best movies, although the reason for this was addressed in the previous paragraph. Nevertheless, we do point out that most of the top ten highest rated films had just one review, highlighting a severe lack of representation. If, somehow, we had collect all users' ratings on all movies, perhaps we would expect the distribution to taper off around a rating of 5, rather than one distinct peak.

Comparing the three genres we've picked, we find that replicate much of the same trends observed when considering all movies. The distribution of ratings is very similar between all three genres, implying that, perhaps unsurprisingly, that movie quality is independent of genre (don't judge a book by its cover). However, examining the average rating, comedies appear to have a slightly lower average whilst sci-fi and

horror are more skewed towards higher ratings. This may be because sci-fi and horror movies attract a more-niche audience but comedies are more generally accessible, explaining why the ratings are closer to average.

3 Matrix Factorization Methods

Method 1: Modified code for Homework 5

In Homework 5, Problem 2, we were tasked with deriving and implementing a matrix factorization algorithm. The algorithm derives the coefficients of the matrices $U \in \mathbb{R}^{M\times K}$ and $V \in \mathbb{R}^{N\times K}$ by minimizing the regularized square error, which is given as follows

$$\arg\min_{U,V} \frac{\lambda}{2}\left(\|U\|_F^2 + \|V\|_F^2\right) + \frac{1}{2}\sum_{i,j}\left(y_{ij} - u_i^T v_j\right)^2 \tag{1}$$

where u_i^T and v_j^T are the i^{th} and j^{th} rows of U and V, respectively, and λ is a regularization parameter.

$\|X\|_F^2$ is the Frobenius norm of $X = U\Sigma V^T$ and is given by

$$\|X\|_F^2 = \sqrt{\sum_{ij} X_{ij}^2} = \sqrt{\sum_d \sigma_d^2} \tag{2}$$

where σ_d are the diagonal elements in Σ. As such, $Y \in \mathbb{R}^{M\times N} \approx UV^T$, and the *ij*-th element of Y is $y_{ij} \approx u_i^T v_j$. Our algorithm computes U and V via stochastic gradient descent in equation (1) as

$$u_i = u_i - \eta\partial_{u_i} = u_i - \eta\Big(\lambda u_i - \sum_j v_j(y_{ij} - u_i^T v_j)\Big) \tag{3}$$

$$v_j = v_j - \eta\partial_{v_j} = v_j - \eta\Big(\lambda v_j - \sum_i v_i(y_i j - u_i^T v_j)\Big) \tag{4}$$

It then uses alternating least squares error by first fixing U and solving for the related optimal V, then taking this optimal V as fixed and solving for an optimal U. The steps are repeated until convergence is reached. The closed form of the expressions for optimal u_i and v_j (solved for by setting the relevant derivatives = 0) are

$$u_i = \frac{\sum_j v_j y_i j}{(\lambda I - \sum_j v_j v_j^T)} \tag{5}$$

$$v_j = \frac{\sum_j u_i y_i j}{(\lambda I - \sum_j u_i u_i^T)} \tag{6}$$

where I is the identity matrix.

Our code for this method initializes matrices U and V to random values and calculates gradients using alternating least squares to achieve U and V such as to minimize the regularized squared error. To tune the model, different values of regularization parameter λ were used to train the model. The value of λ that minimizes loss is $\lambda = 0.1$. Similarly, the optimal learning rate $\eta = 0.1$. We chose the stopping criteria to be when the loss L of the i'th iteration fulfills $\frac{|L_i - L_{i-1}|}{L_1} \leq \epsilon$, where ϵ is some small value. We chose this stopping criteria because further iterations would not yield large gradients and thus the elements of U and V would be stable to further calculation.

Method 2: Incorporated bias (surprise)

This method differs from that in the previous section by adding vectors containing bias values of a and b for U and V, respectively. The vectors function as deviations from the global bias. For U, a represents user-specific preferences for certain movies (i.e., due to their personal preferences). For V, b represents movie-specific preferences for certain users (i.e., due to movies being targeted towards certain audiences).

The regularized square error thus takes the form

$$\arg\min_{U,V,a,b} \frac{\lambda}{2} \left(\|U\|_F^2 + \|V\|_F^2 + \|a\|_F^2 + \|b\|_F^2 \right) + \sum_{i,j} \left((y_{ij} - \mu) - (u_i^T v_j + a_i + b_i) \right)^2 \tag{7}$$

Aside from this difference, the incorporated bias method is similar to that implemented in Homework 5, Problem 2. The package *surprise* was used for implementation of this method.

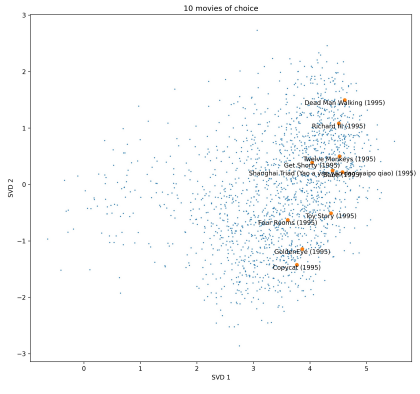
Method 3: Off-the-shelf implementation (surprise)

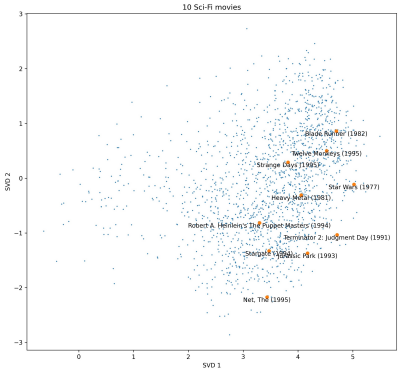
The chosen off-the-shelf implementation method also utilized the package *surprise*, however the bias terms were switched off. As such, this method more closely matched the modified code from Homework 5, with the key difference being the use of the *surprise* package.

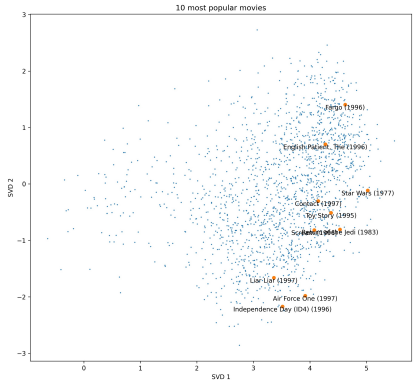
4 Matrix Factorization Visualizations

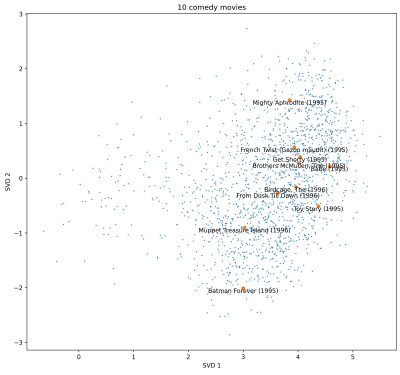
The following visualizations were produced:

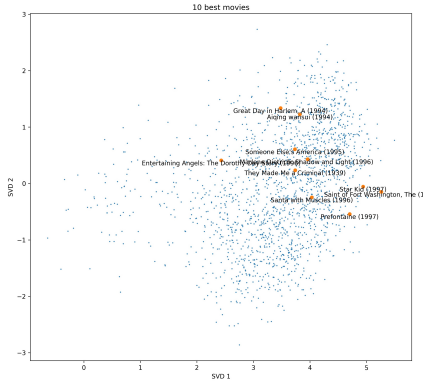
1. Method 1:

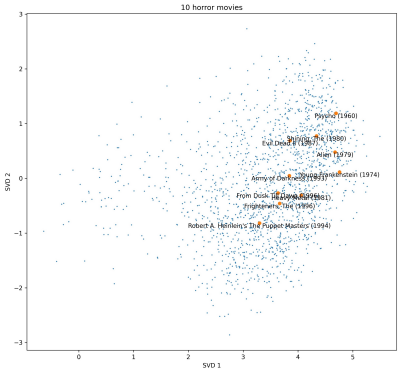








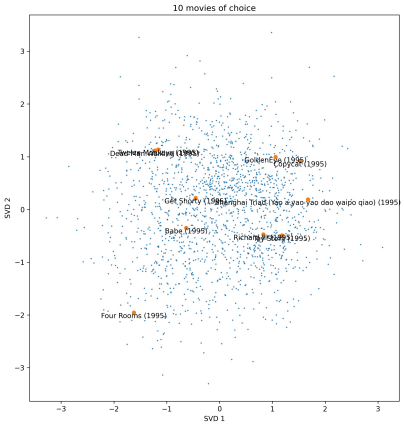


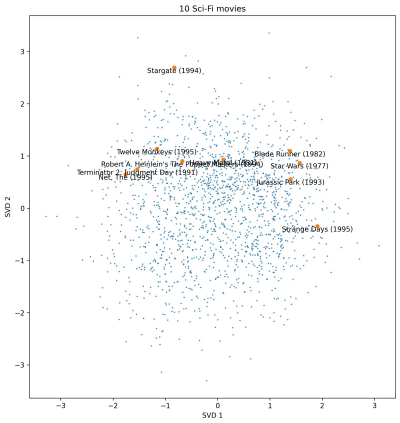


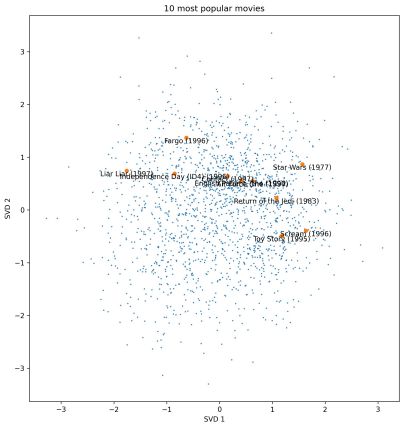
(a) Method 1: selected, popular, and best

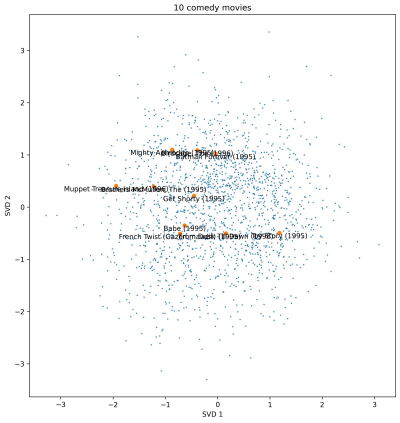
(b) Method 1: genres

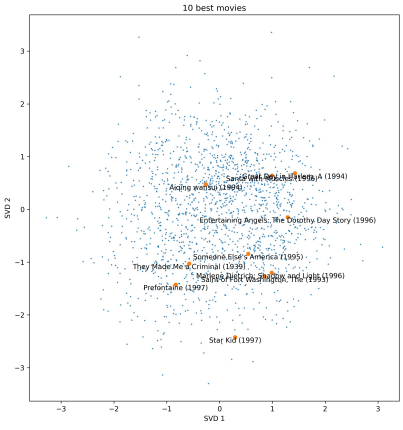
2. Method 2:

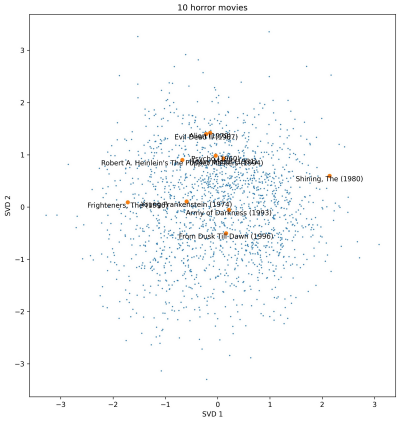








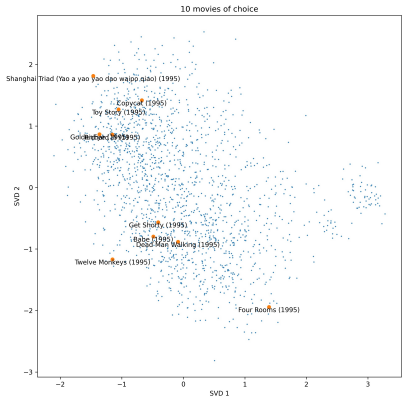


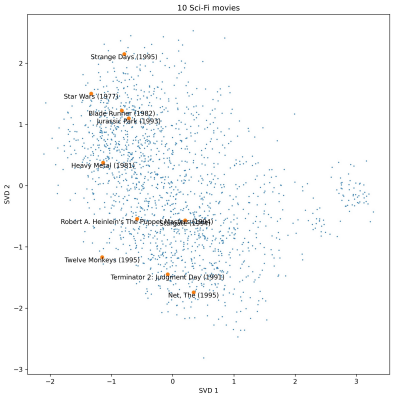


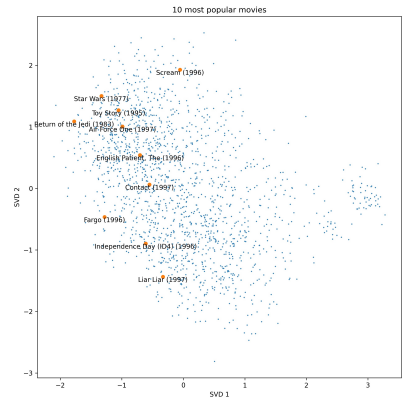
(a) Method 2: selected, popular, and best

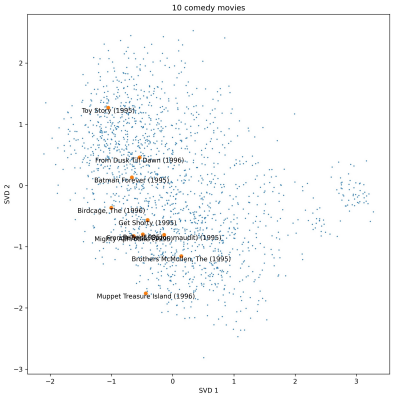
(b) Method 2: genres

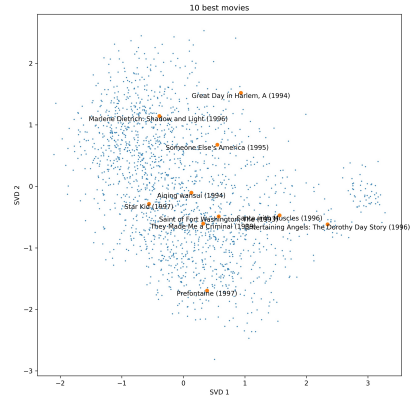
3. Method 3:

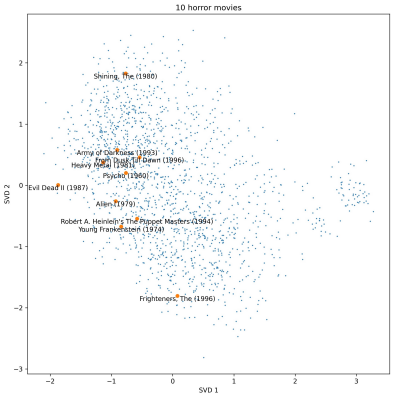












(a) Method 3: selected, popular, and best

(b) Method 3: genres

Analysis

The error on the test sets for methods 1, 2, 3 were 0.42, 6.92, and 0.95, respectively. The error is very high on method 2 because of the added bias term, because the testing data did not have a user-specific bias fit to it (we are assuming their tastes are close to average). It is interesting that our own implementation had a lower error than the off-the-shelf implementation using *surprise*. Perhaps it is because we implemented early stopping in our coding, while the SVD algorithm used by *surprise* trains for a fixed number of epochs. The optimization algorithms between the two methods also differed.

Discussion

Generally, across our visualizations, we see that SVD 2 tends to be more evenly distributed around 0 for most movies. In the case of methods 2 and 3, SVD 1 is also closely distributed around 0 (less so for method 3) whilst for method 1, most movies tend to have large, positive values. This is to be expected as, by the introduction of a bias in method 2, we are shifting our latent factors V. Examining the methods individually, methods 1 and 3 both have a slight positive and negative correlation between SVD 1 and 2, respectively. This is most probably because these methods do not incorporate a bias, removing the shift in the latent factors. We also notice a small cluster for large SVD 1 in method three; examining these movies showed no apparent relations between the movies in this cluster.

The ten best movies tend to cluster the most together in method 1 but slightly more-spread in methods 2 and 3. In contrast, the most popular movies are always spread out in all three methods (perhaps less so in method two). This perhaps indicate that the best movies have more in common than the most popular movies. The reason that the most popular movies are more clustered in method 2 could be due to the use of a bias.

Examining each genre, horror movies/comedy movies tend to cluster together as well. These clustering patterns are expected because highly rated movies and movies of the same genre should be appear similar to each other when projected along the most informative axes. On the other hand, sci-fi movies tend to cluster less intensely. This may indicate that there is a weaker correlation between sci-fi movies than comedies / horror movies (the latter two genres are 'more-general' classes of movies).