Machine Learning & Data Mining
Miniproject 2

Caltech CS/CNS/EE 155
February 2020.

## 1 Introduction [10 points]

### Team name:

Turtle School Son Goku.

### Group member(s):

Liting Xiao.

### Division of labor:

The whole of the project, including data exploration, data processing, basic visualization, matrix factorization, matrix factorization visualization, and the writing of this report.

## 2 Basic Visualizations [20 points]

### 0. Data preprocessing.

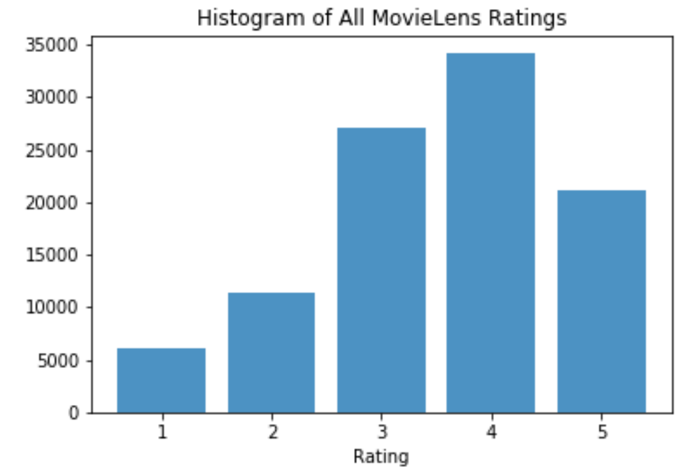
I used *pandas* to read in the data files 'movies.txt' and 'data.txt', and I added in corresponding column names to the dataframes for easier access to both datasets.

There are movies with duplicate titles but different IDs in 'movies.txt'. And user ratings are assigned to different IDs that are actually the same movies. So, I grouped together these duplicate movie entries in 'movies.txt' to record the duplicate IDs. Then in 'data.txt', for each duplicate ID group, I replaced the duplicates with the first (lowest numbered) ID registered to the same movies.

### 1. All ratings in the MovieLens Dataset.

A histogram of ratings of all movies in the MovieLens dataset is plotted below. The mean is 3.53, and the standard deviation is 1.126.

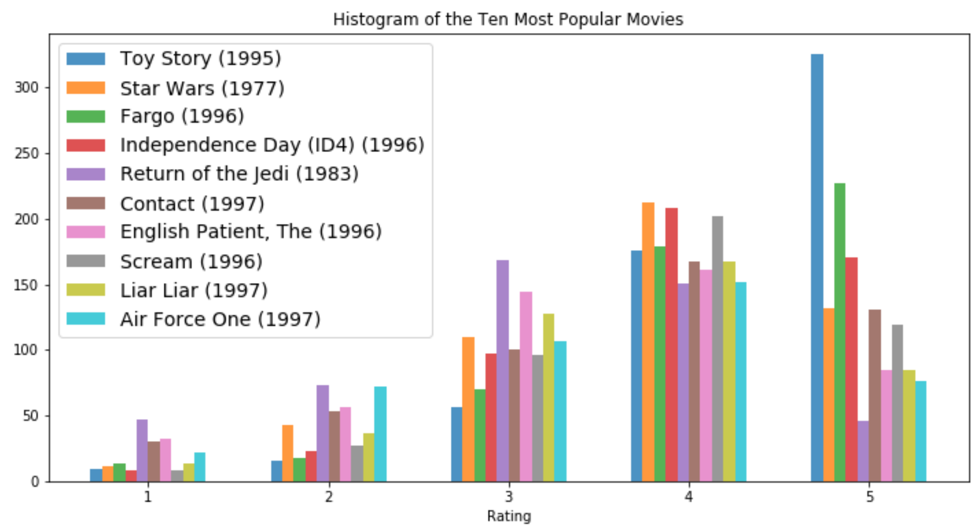
There are more movies with ratings of 3, 4, 5 than of 1, 2. The distribution is skewed towards the right. With the general shape of the all rating distribution in mind, let's move on to more specific aspects of our datasets.



### 2. All ratings of the ten most popular movies (movies which have received the most ratings).

A histogram of ratings of the 10 most popular movies in the MovieLens dataset is plotted below. The mean is 3.77, and the standard deviation is 1.079.

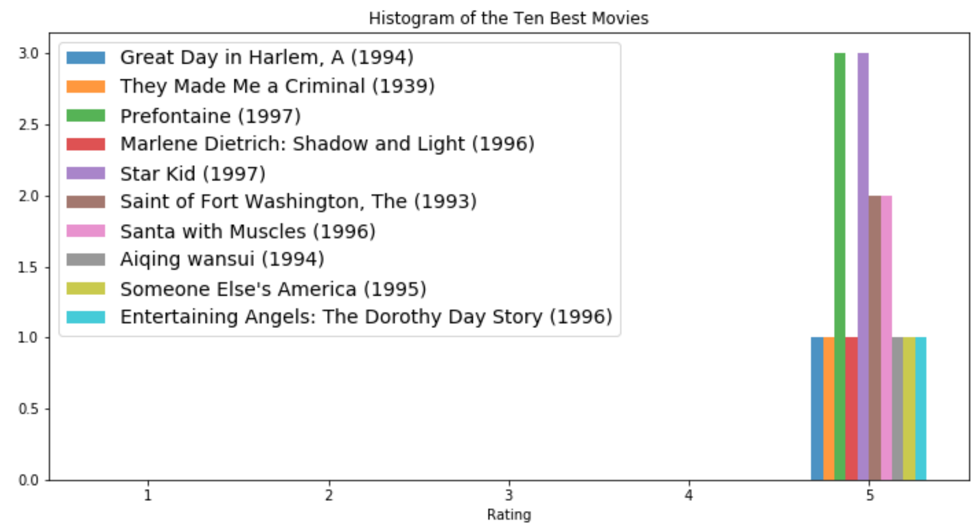
Compared to the histogram of all ratings, the distribution shifts even more towards the higher rating part. This is as expected since these movies are most popular most likely because more people watched them, liked them and recommended them.



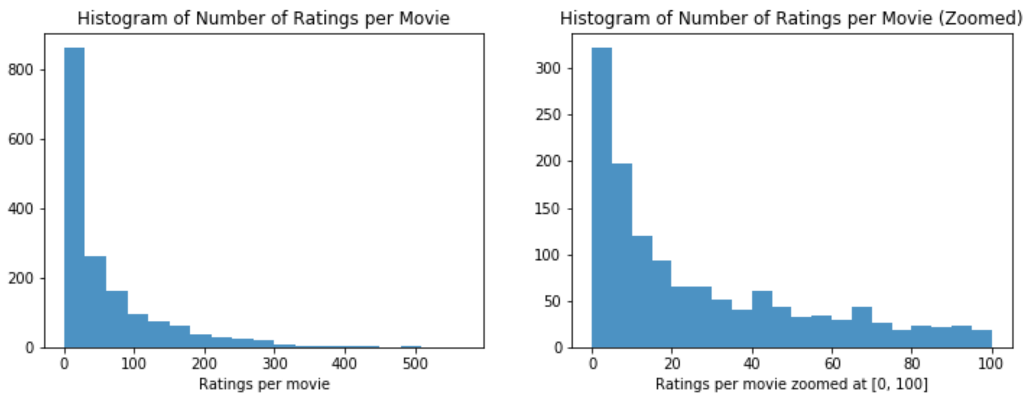
## 3. All ratings of the ten best movies (movies with the highest average ratings).

Naively, a histogram of ratings of the 10 movies with the highest average ratings in the MovieLens dataset is plotted below. The mean is 5.0, and the standard deviation is 0.

These 10 movies only have ratings of 5, but notice that's because only 1-3 people have rated them. This plot is not very informative in a sense that these movies are not very popular and could not be termed as "the best" movies. With the small number of ratings gathered for these movies, the distribution is not meaningful as well with such a small sample size.



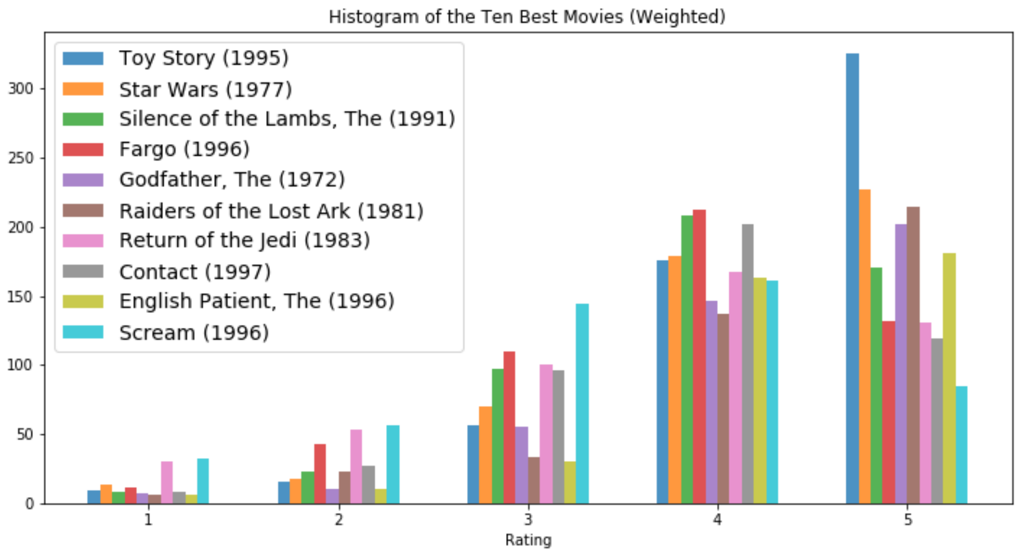
We can take a look at the distribution of number of ratings per movie. And it's noticeable that we have a skewed dataset towards lower number of ratings. Let's also see this by zooming in to [0, 100] ratings per movie range.



Therefore, a more useful plot would be a histogram of all ratings of the ten best movies (movies with the highest average ratings, **weighted** by the number of ratings). Such a histogram is shown below.

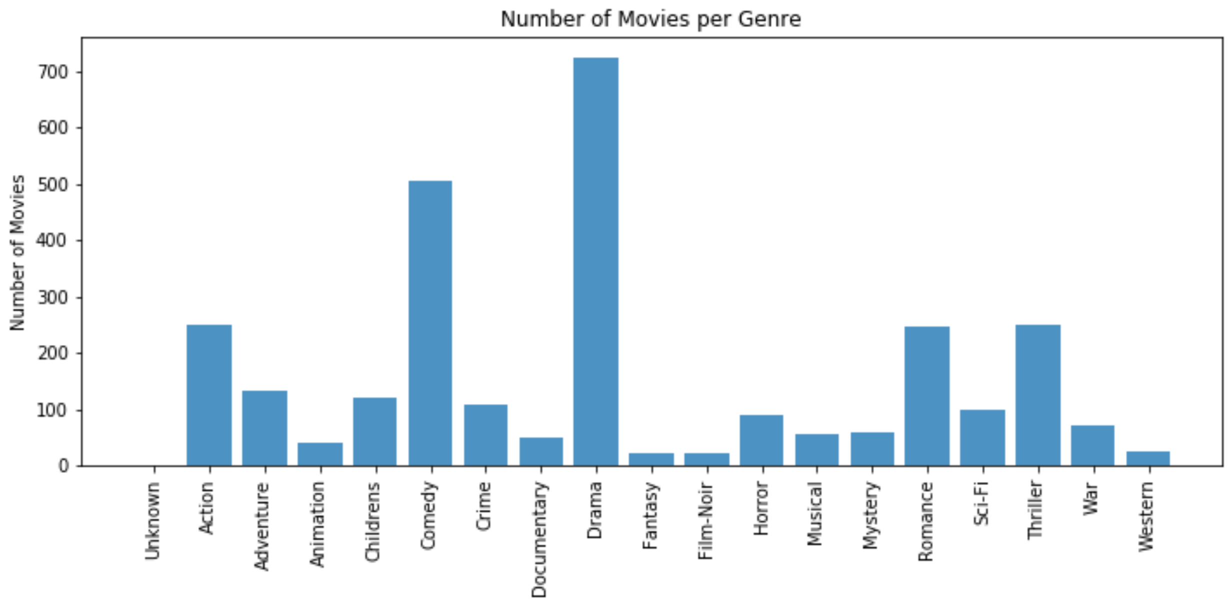
For the actual 10 best movies, the mean is 4.01, and the standard deviation is 1.015.

Compared to the 10 most popular movies, we see that these 2 categories have 7 overlaps: *Toy Story (1995), Star Wars (1977), Fargo (1996), Return of the Jedi (1983), Contact (1997), English Patient, The (1996), Scream (1996).* Thus, as expected, the distribution looks alike, with the 10 best movie rating distribution shifting even further towards the right.



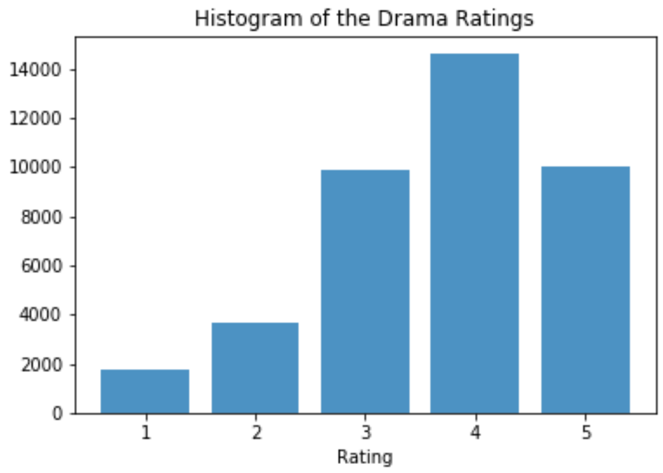
## 4. All ratings of movies from three genres of your choice (create three separate visualizations).

Let's first visualize the number of movies per genre in our dataset and then pick a range of 3 from them.

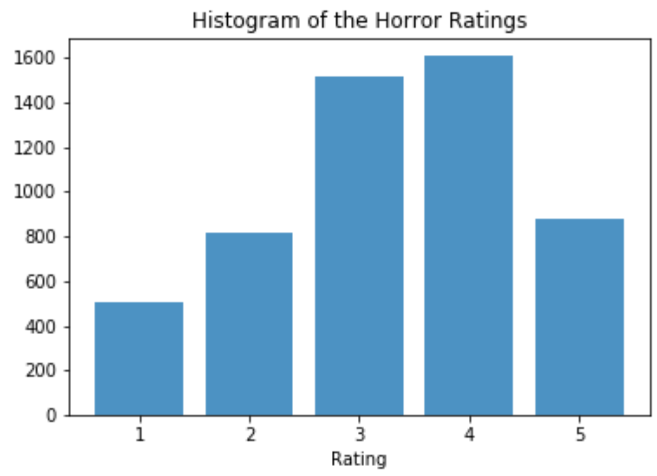


I picked 3 genres with the most, and medium, and few number of movies labeled: *Drama, Horror, Fantasy*. We can see from the following 3 plots that the more a genre is popular, the more likely that the viewers will rate them higher, at least in our MovieLens dataset. Popularity of movies seems to be associated with their viewer ratings.

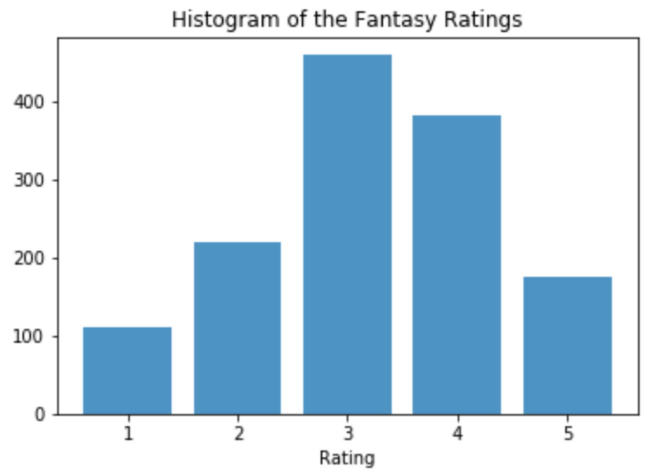
The mean of ratings in the most popular *Drama* genre is 3.69, and the standard deviation 1.079.



The mean of ratings in the medium popular *Horror* genre is 3.29, and the standard deviation 1.187.



The mean of ratings in the less popular *Fantasy* genre is 3.22, and the standard deviation 1.119.



## 3 Matrix Factorization Methods [40 points]

### 1. Re-use code from HW5, not including biases.

The first method is the standard SVD from HW5. The standard loss function with regularization is:

$$J = \frac{\lambda}{2}(||U||^2 + ||V||^2) + \frac{1}{2}\sum_{(i,j)\in S}(y_{ij} - u_i^T v_j)^2.$$

For implementation, I calculate the gradients $\partial_{u_i}$ and $\partial_{v_j}$.

$$\partial_{u_i} = \lambda u_i - \sum_j v_j(y_{ij} - u_i^T v_j),$$
$$\partial_{v_j} = \lambda v_j - \sum_i u_i(y_{ij} - u_i^T v_j).$$

Then we can use the gradients to calculate $U$ and $V$ by stochastic gradient descent.

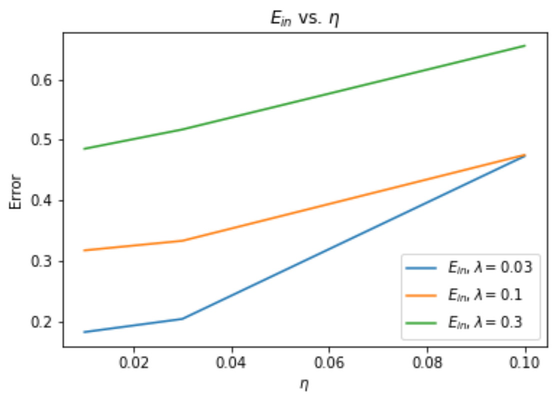
$$u_i = u_i - \eta\partial_{u_i},$$
$$v_j = v_j - \eta\partial_{v_j}.$$

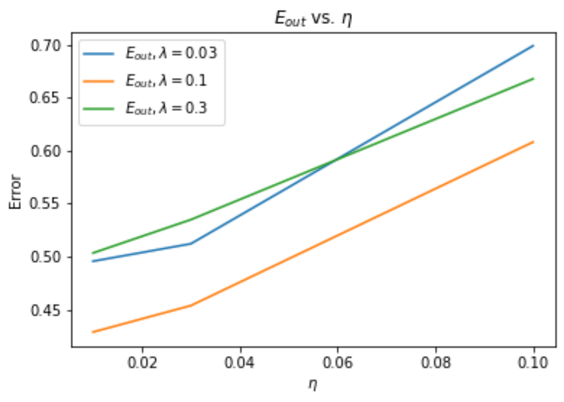
For training, I initialize U and V to be uniform random variables in [-0.5, 0.5]. Our data are 1-indexed, and this is dealt with in the training process. Here, we use $K = 20$ latent factors, maximum epochs of 300. We also use early stopping technique such that when the fraction of loss descent is less than $\epsilon = 0.0001$, we will stop the algorithm.

To determine values of other hyper-parameters (in our case, regularization $\lambda$ and learning rate $\eta$), we loop through these hyper-parameters, train the model, and see the in-sample and out-of-sample performance, $E_{\text{in}}$ and $E_{\text{out}}$.

From HW5, we know that $\lambda = 0.1$ and $\eta = 0.03$ work pretty well already. So, let's try values in the vicinities. I tried $\lambda = [0.03, 0.1, 0.3]$ and $\eta = [0.01, 0.03, 0.1]$ in the interest of time. The following 2 plots show the performance of the method in these hyper-parameter ranges. We can see that $\lambda = 0.1$ and $\eta = 0.01$ work the best with the lowest $E_{\text{out}}$ and no overfitting or underfitting occurs.

Lastly, we train using the whole dataset with $\lambda = 0.1$ and $\eta = 0.01$.

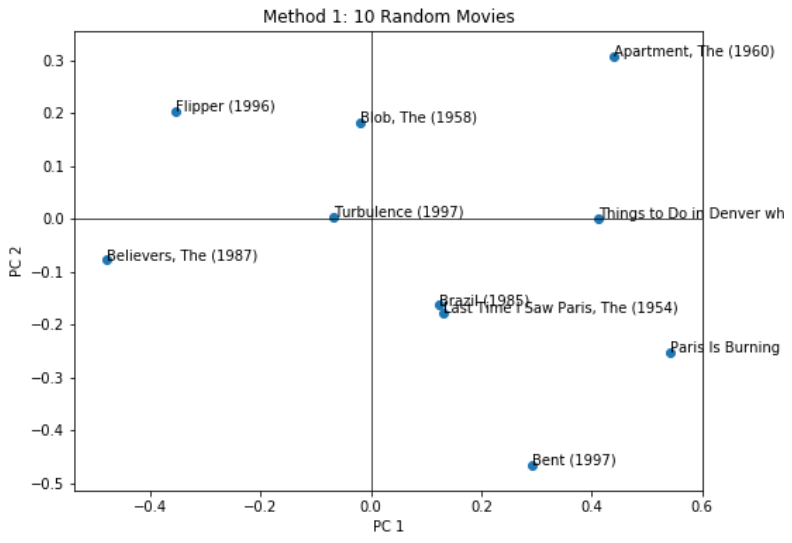




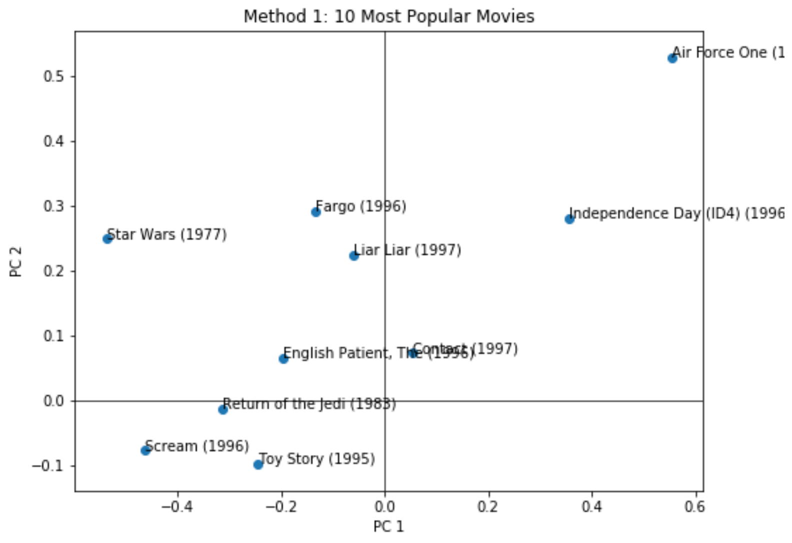
Now, in order to visualize the resulting latent factors, I apply SVD to $V = A\Sigma B$ and use the first 2 columns of $A$ to project $U$ and $V$ into a 2D space. The projection is given by $\tilde{U} = A_{1:2}^T U$ and $\tilde{V} = A_{1:2}^T V$.

Now, we visualize the following:

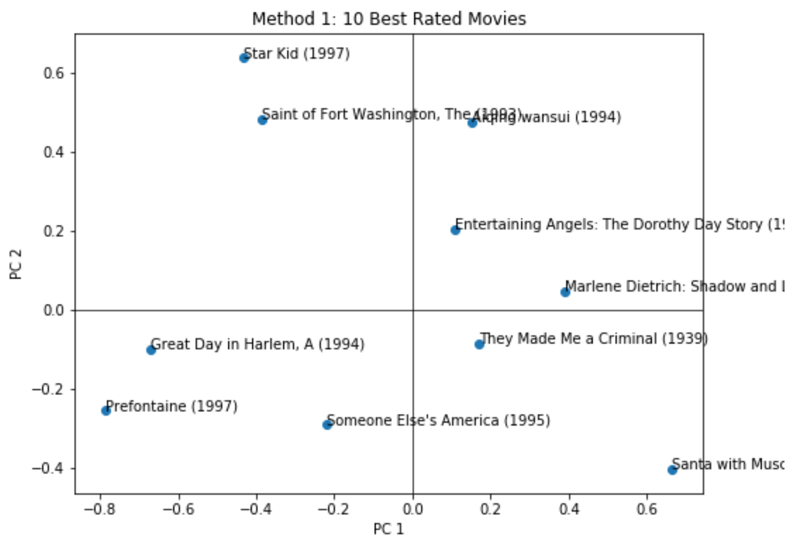
(a) 10 random movies from the MovieLens dataset.



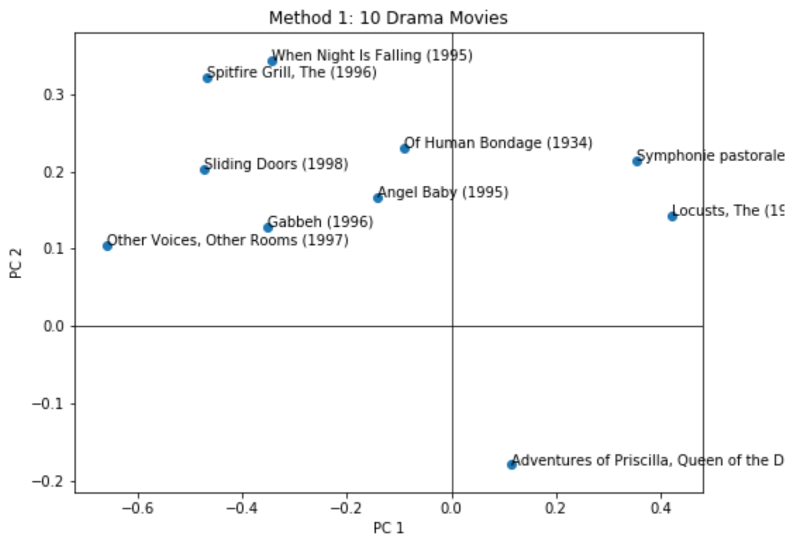
(b) The 10 most popular movies (movies which have received the most ratings).

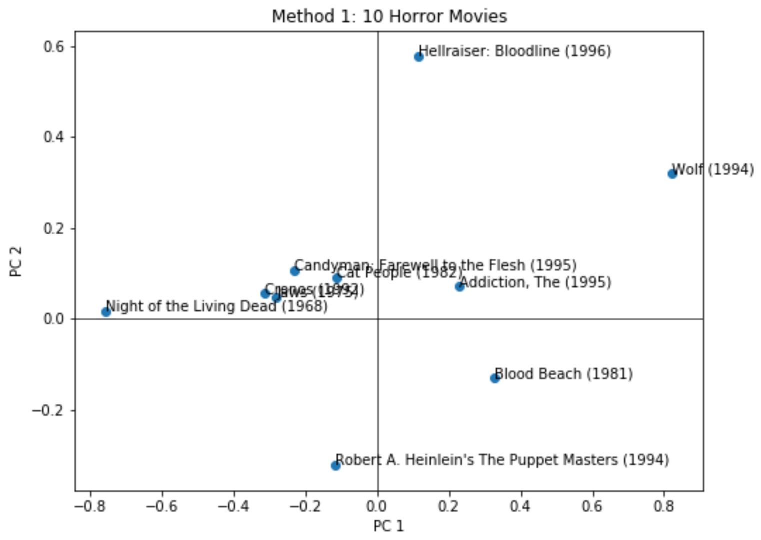


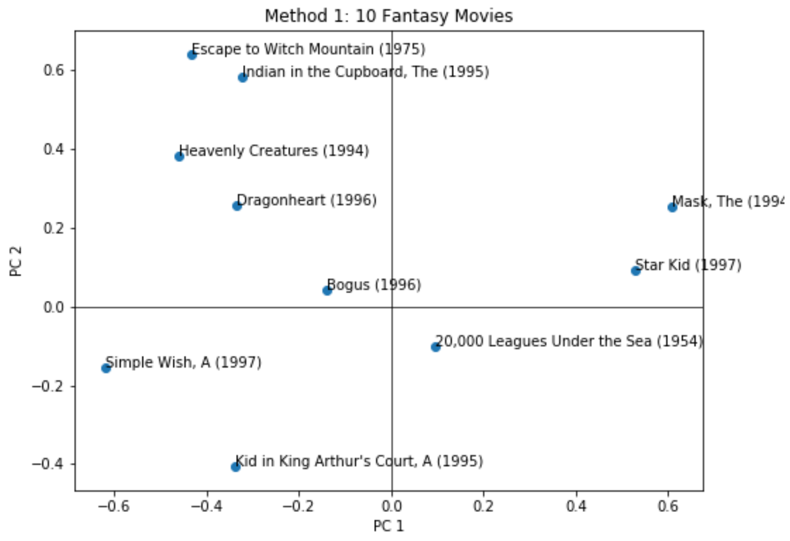
(c) The 10 best movies (movies with the highest average ratings).



(d) 10 Movies from [*′Drama′*, *′Horror′*, *′Fantasy′*].







## 2. Re-use code from HW5, but incorporate bias terms for each user and movie.

The second method is similar to method 1, but now we also model the global biases of movies and users. The loss function with regularization is then:

$$J = \frac{\lambda}{2}(||U||^2 + ||V||^2 + ||a||^2 + ||b||^2) + \frac{1}{2}\sum_{(i,j)\in S}((y_{ij} - \mu) - (u_i^T v_j + a_i + b_j))^2.$$

The gradients $\partial_{u_i}$, $\partial_{v_j}$, $\partial_{a_i}$, $\partial_{b_j}$ are:

$$\begin{aligned}
\partial_{u_i} &= \lambda u_i - \sum_j v_j((y_{ij} - \mu) - (u_i^T v_j + a_i + b_j)), \\
\partial_{v_j} &= \lambda v_j - \sum_i u_i((y_{ij} - \mu) - (u_i^T v_j + a_i + b_j)), \\
\partial_{a_i} &= \lambda a_i - \sum_i ((y_{ij} - \mu) - (u_i^T v_j + a_i + b_j)), \\
\partial_{b_j} &= \lambda b_j - \sum_i ((y_{ij} - \mu) - (u_i^T v_j + a_i + b_j)).
\end{aligned}$$

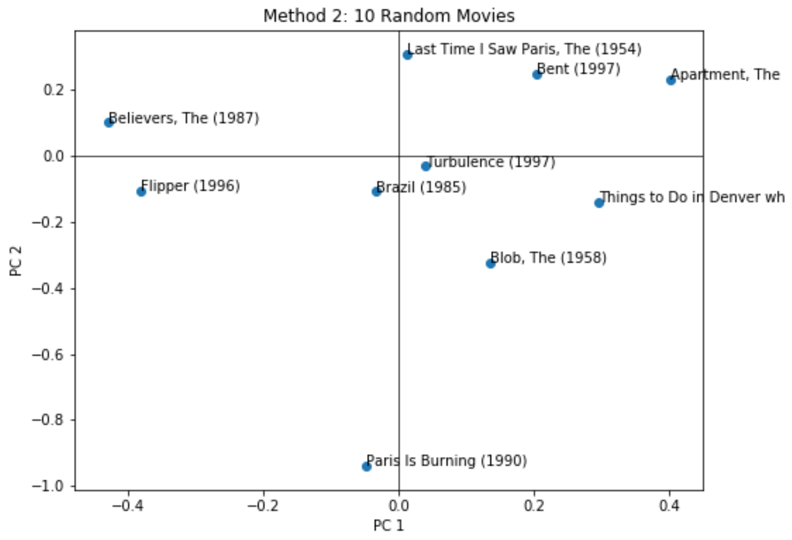
Then we can use these gradients to calculate $U$, $V$, $a$ and $b$ by stochastic gradient descent.

$$\begin{aligned}
u_i &= u_i - \eta \partial_{u_i}, \\
v_j &= v_j - \eta \partial_{v_j}, \\
a_i &= a_i - \eta \partial_{a_i}, \\
b_j &= b_j - \eta \partial_{b_j}.
\end{aligned}$$

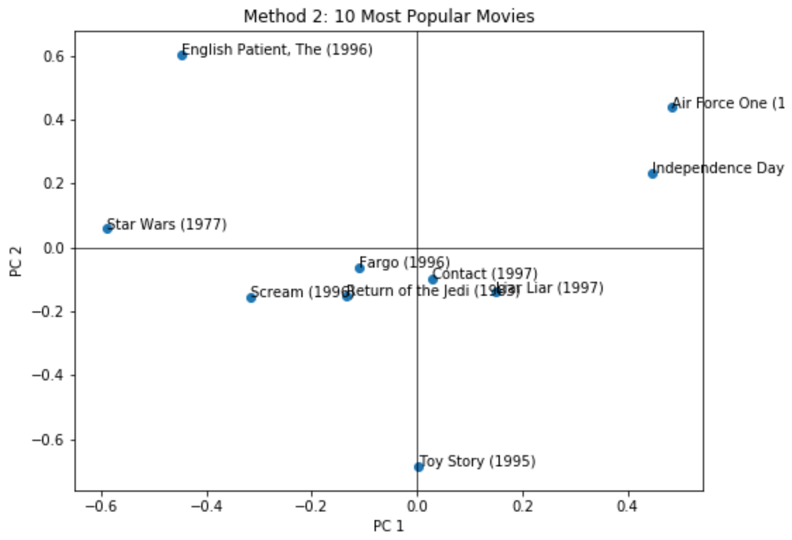
Again, for training, I initialize U, V, a and b to be uniform random variables in [-0.5, 0.5]. Other variables and details are the same. I modified functions in method 1 to take in additional arguments if 'bias' is set to 'True'. As for regularization and learning rate, since this method is quite similar with method 1, we could assume that the same set of optimal hyper-parameters apply here. So we use $\lambda = 0.1$ and $\eta = 0.01$.

Again, we apply SVD to V and project V into a 2D space. Now, we visualize the following:

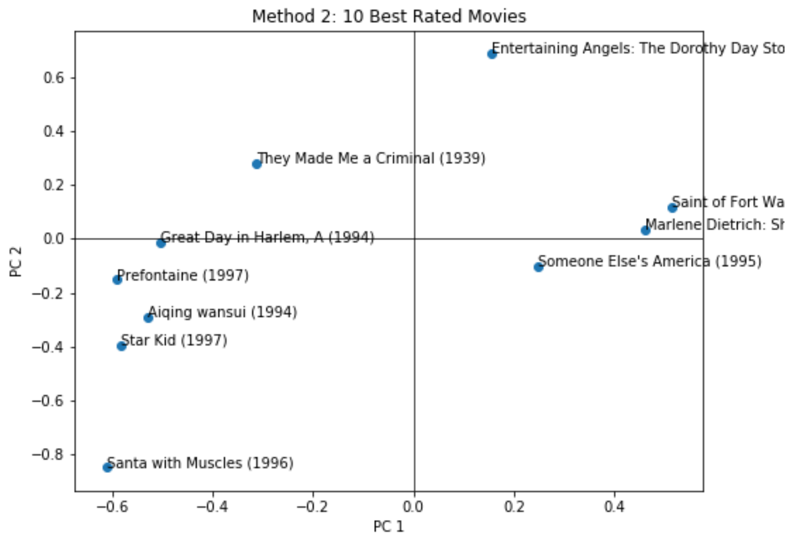
(a) 10 random movies from the MovieLens dataset.



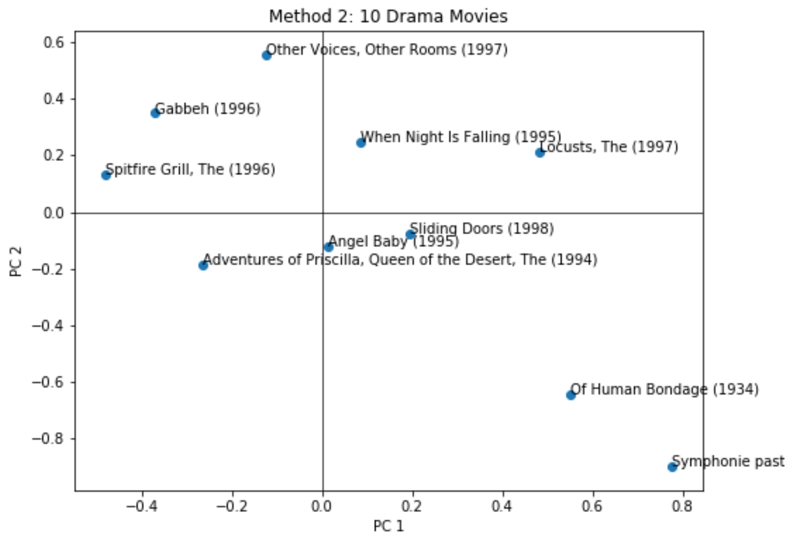
(b) The 10 most popular movies (movies which have received the most ratings).

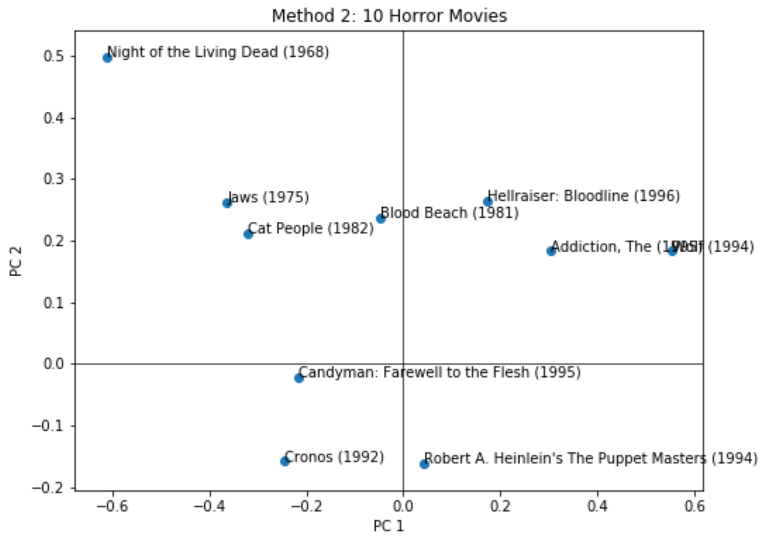


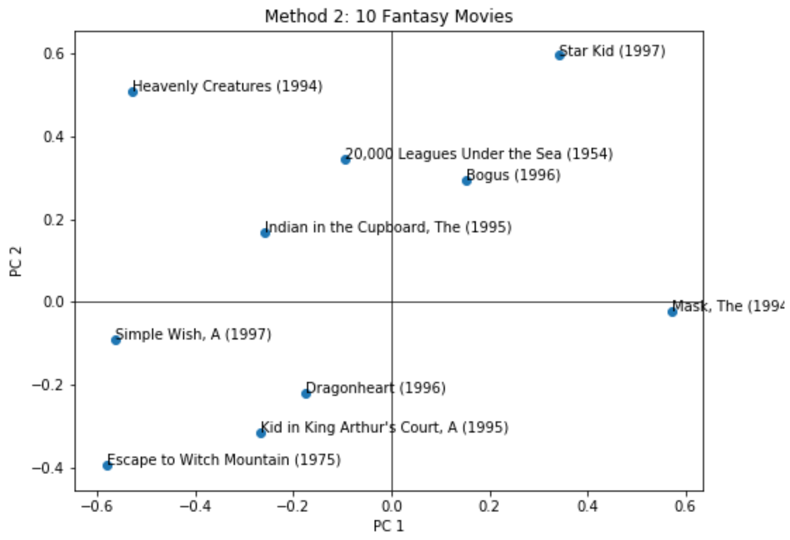
(c) The 10 best movies (movies with the highest average ratings).



(d) 10 Movies from [*′Drama′*, *′Horror′*, *′Fantasy′*].


Method 2: 10 Drama Movies
Other Voices, Other Rooms (1997)
Gabbeh (1996)
When Night Is Falling (1995)
Locusts, The (1997)
Spitfire Grill, The (1996)
Sliding Doors (1998)
Angel Baby (1995)
Adventures of Priscilla, Queen of the Desert, The (1994)
Of Human Bondage (1934)
Symphonie past
PC 1
PC 2


Method 2: 10 Horror Movies
Night of the Living Dead (1968)
Jaws (1975)
Hellraiser: Bloodline (1996)
Blood Beach (1981)
Cat People (1982)
Addiction, The (1995)
Wolf (1994)
Candyman: Farewell to the Flesh (1995)
Cronos (1992)
Robert A. Heinlein's The Puppet Masters (1994)
PC 1
PC 2



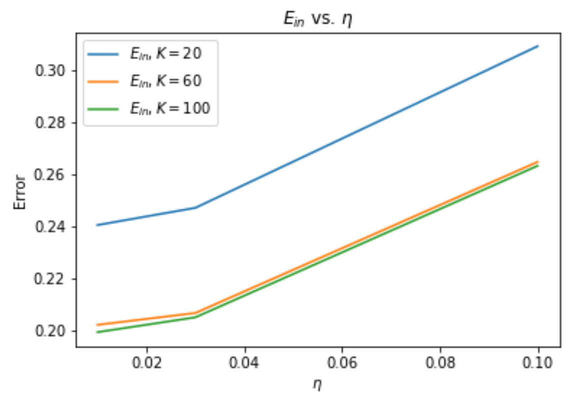
## 3. Off-the-shelf implementation using Surprise SVD.

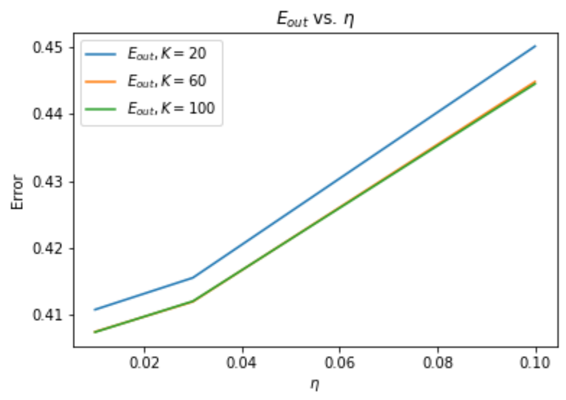
Lastly for the third method, I use an off-the-shelf implementation of matrix factorization, the *Surprise* module.

Before implementation of this method, we notice that some movies don't have ratings in the training set 'train.txt'. This will mess up with the factorization and indexing and the predictions won't be accurate because of these missing entries. So first, 'Movie Id' needs to be re-indexed to [1, len(movies in the train set)].

Then I load the train set, the test set, and the whole data set from *pandas* dataframe objects to *Surprise* Dataset objects.

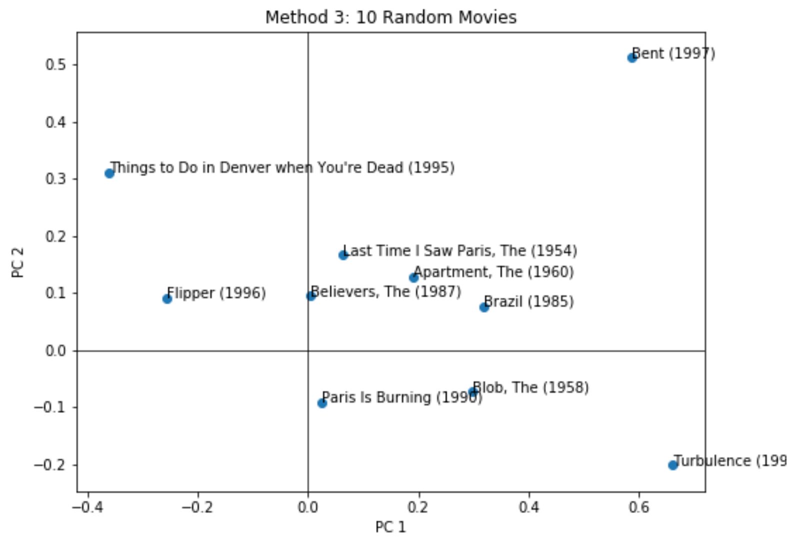
To choose hyper-parameters, we look at the number of latent factors and learning rate here. I decided not to look at regularization because it doesn't matter too much for us anyway and in the interest of time. So, I looped through $K = [20, 60, 100]$ and $\eta = [0.01, 0.03, 0.1]$ to see this method's performance. And the following 2 plots show the in-sample and out-of-sample performance. We can see that $K = 100$ and $\eta = 0.01$ work the best with the lowest $E_{\text{out}}$ and no overfitting or underfitting occurs.



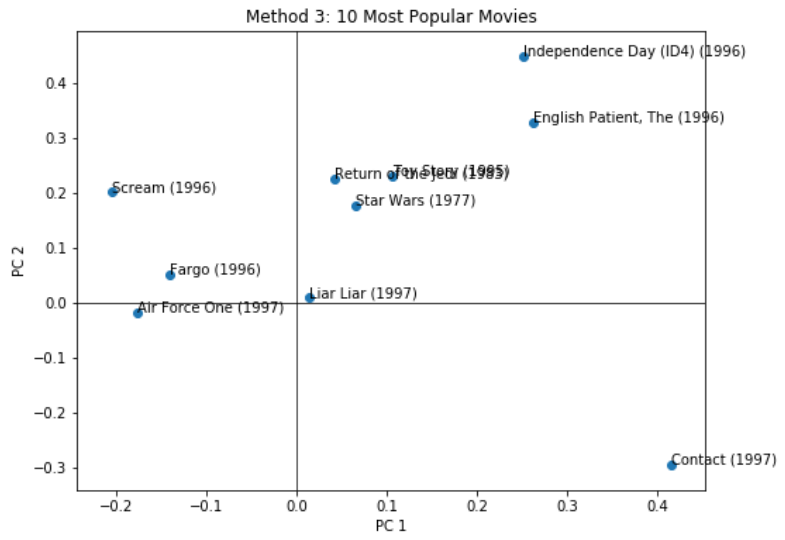


Again, we apply SVD to V and project V into a 2D space. Now, we visualize the following:

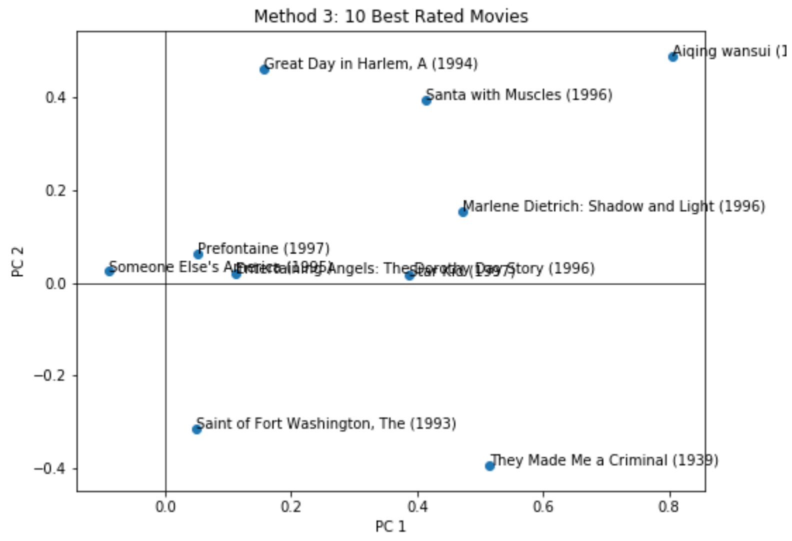
(a) 10 random movies from the MovieLens dataset.



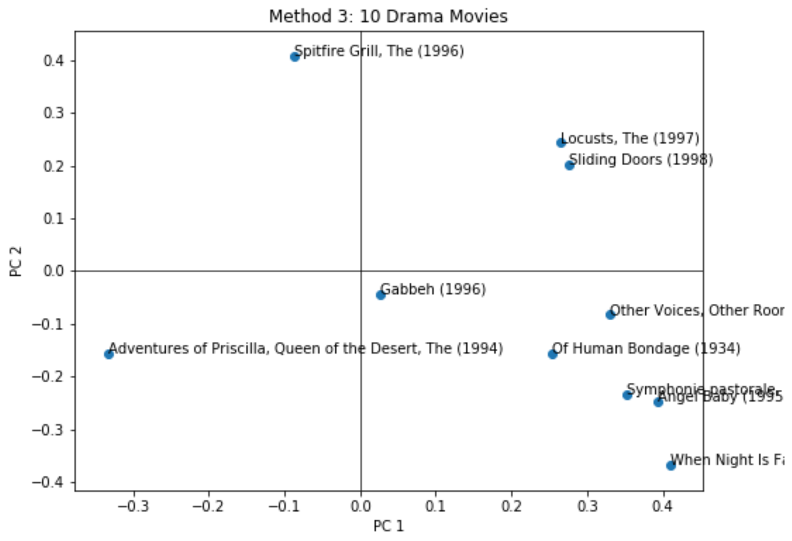
(b) The 10 most popular movies (movies which have received the most ratings).

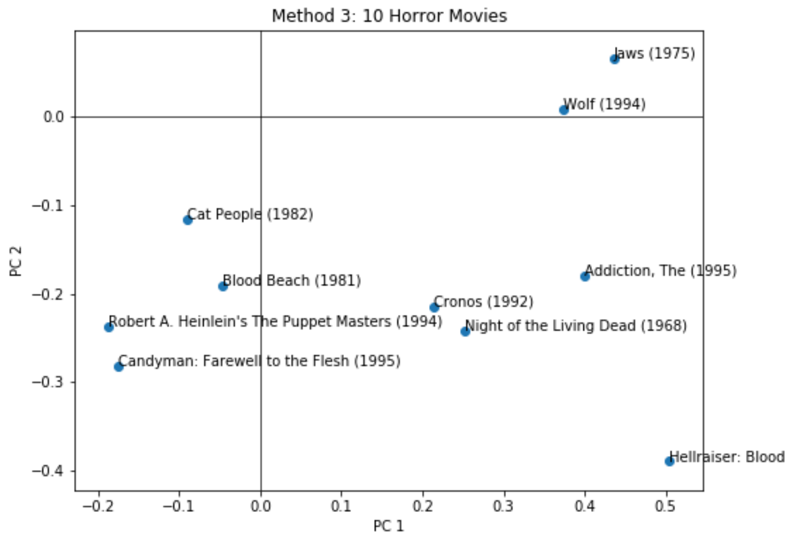


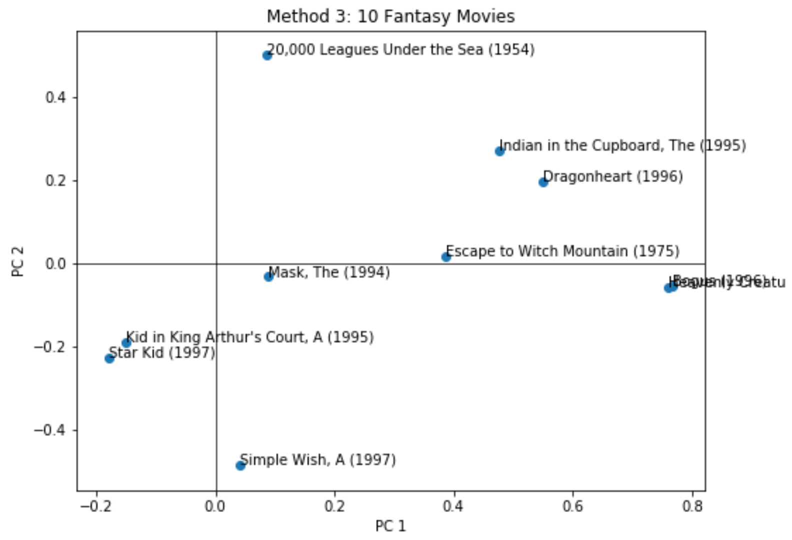
(c) The 10 best movies (movies with the highest average ratings).



(d) 10 Movies from [*'Drama'*, *'Horror'*, *'Fantasy'*].


Method 3: 10 Drama Movies
Spitfire Grill, The (1996)
Locusts, The (1997)
Sliding Doors (1998)
Gabbeh (1996)
Other Voices, Other Roor
Adventures of Priscilla, Queen of the Desert, The (1994)
Of Human Bondage (1934)
Symphonie pastorale
Angel Baby (1995
When Night Is Fa
PC 1
PC 2


Method 3: 10 Horror Movies
Jaws (1975)
Wolf (1994)
Cat People (1982)
Addiction, The (1995)
Blood Beach (1981)
Cronos (1992)
Robert A. Heinlein's The Puppet Masters (1994)
Night of the Living Dead (1968)
Candyman: Farewell to the Flesh (1995)
Hellraiser: Blood
PC 1
PC 2



## 4. Comparison of the three methods.

Method 1 and 2 are similar in the sense that I implemented these two myself using SGD while method 3 is using off-the-shelf implementation.

Method 2 and 3 are similar in the sense that these two use the same MSE loss function including global biases, while method 1 doesn't include biases.

The performance in terms of $E_{\text{in}}$ and $E_{\text{out}}$ are shown below for these three methods. Method 1 and 2 have almost the same performance $E_{\text{out}}$, but $E_{\text{in}}$ is lower for method 2 than for method 1, because method 2 has two more vectors of global biases to fit in the model. So we can expect method 2 to have better $E_{\text{in}}$ than method 1.

Method 1: E_in = 0.3172, E_out = 0.4287.

Method 2: E_in = 0.2550, E_out = 0.4274.

Method 3: E_in = 0.1993, E_out = 0.4075.

Method 3 works the best both in $E_{\text{in}}$ and $E_{\text{out}}$, because here we are fitting using more latent factors $K = 100$. So the better performance is expected as well.

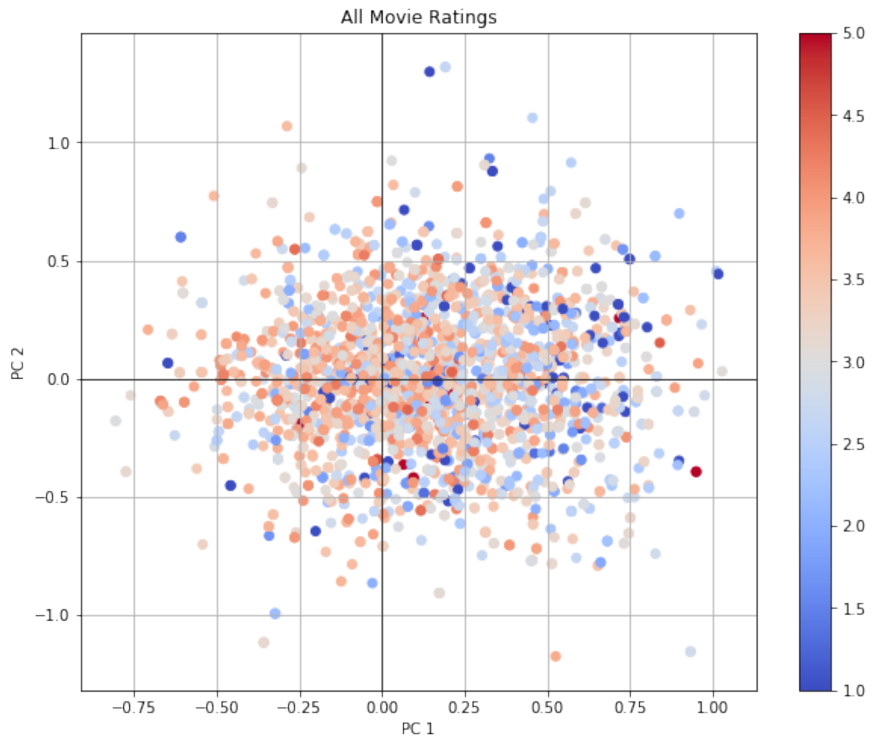
I found method 3 also worked much faster than the other two, even with a much larger number of latent factors. And method 3 has the best performance. So, for real-life implementation, I would go with the off-the-shelf modules.

## 4 Matrix Factorization Visualizations [30 points]

(I would leave most of the plots up in Section 3 to save space and mostly just summarize my findings here.)

**- General comments:**

In general, I think the 2D visualizations are still very randomly distributed in space. I would not say that they are very good visualizations. It's hard to discover many trends out of these 2D visualizations (to my eyes). Perhaps going up to 3D would help. Here, I show a plot of all the movies in this 2D space using method 2, with their ratings (mean of all ratings of a movie) as the color bar. I think method 2 shows my favorite characteristics: highly rated movies seems to cluster near the origin.



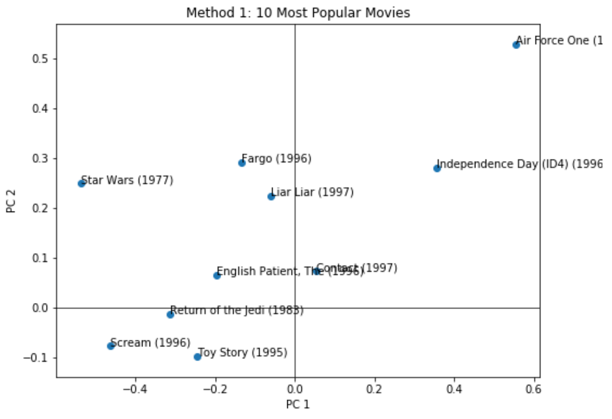
**- For the 10 random movies:**

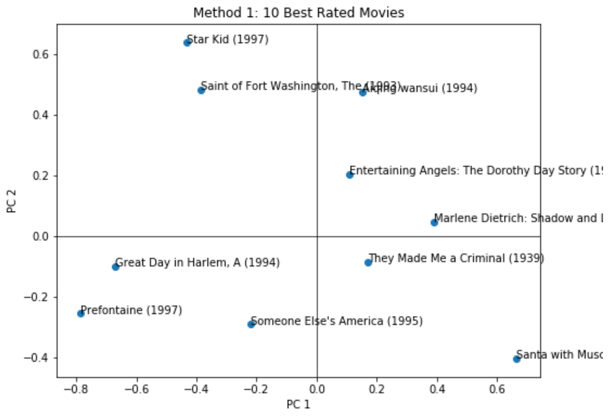
Just as expected, they were randomly distributed in space for all 3 methods. Nothing special here.

**- For the 10 most popular movies compared to the 10 best movies:**

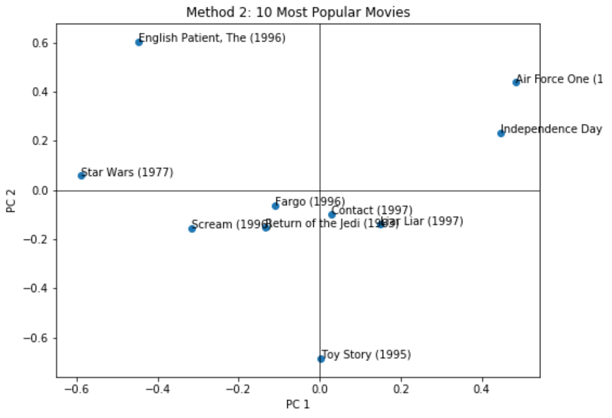
These 2 categories seem to show opposite traits here.

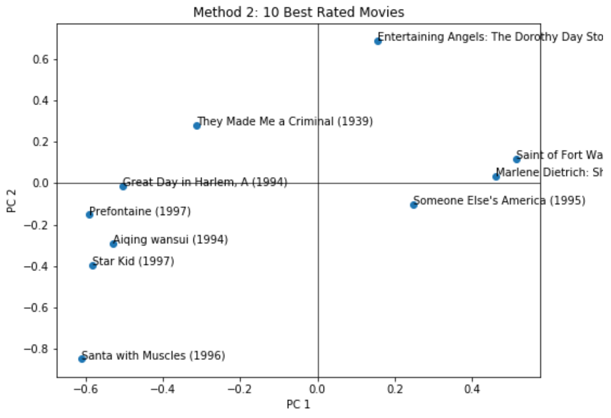
For method 1, the 10 most popular seem to cluster near $\mathrm{PC2} = 0$, while the 10 best seem to disperse from $\mathrm{PC2} = 0$. (Note that the scales of the plots are different.)



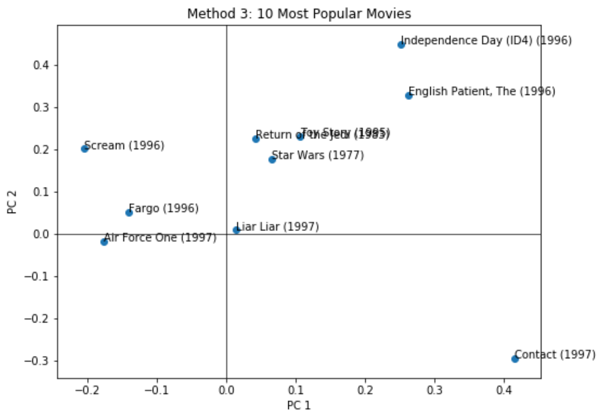


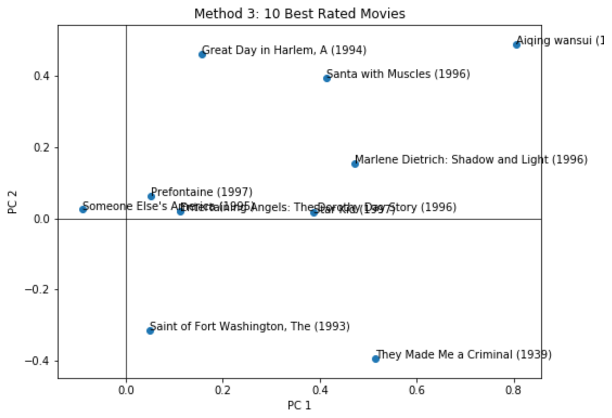
For method 2, the 10 most popular seem to cluster near the origin, while the 10 best seem to disperse from it.





For method 3, the 10 most popular seem to cluster near $\text{PC1} = 0$, while the 10 best seem to disperse from $\text{PC1} = 0$. (Note that the scales of the plots are different.)





This seems to be saying that the most popular movies are very different from the best rated movies, which is counterintuitive. And just as I explained and digged deeper in Section 2, the best movies here are not really "best rated" since they have only 1-3 ratings and they could not represent the whole distributions well.

**- For Drama, Horror, Fantasy genres:**

For method 1: Drama movies seem to cluster in the $\mathrm{PC2} > 0$ quadrants; Horror movies cluster near $\mathrm{PC2} = 0$; Fantasy movies are more away from the origin, perhaps indicating their inpopularity.

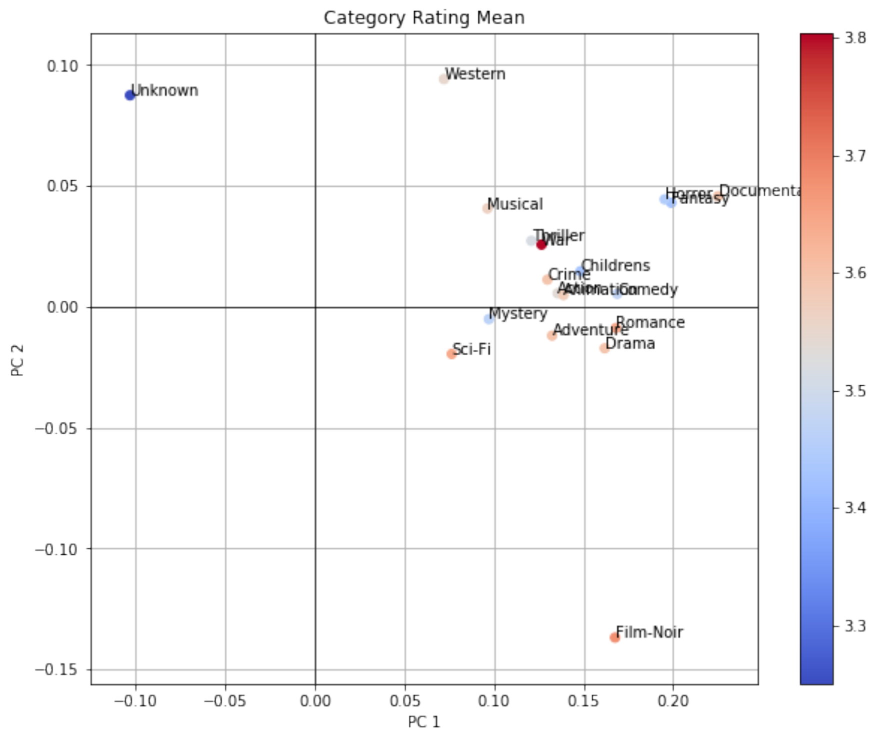
For method 2: Drama movies are near origin; Horror and Fantasy are more away from the origin.

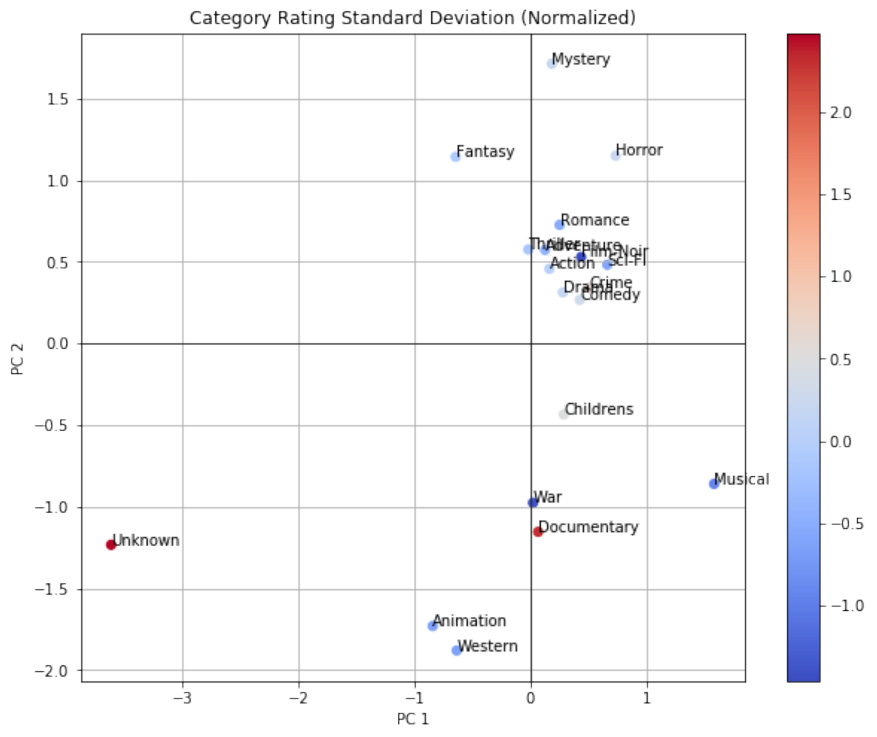
For method 3: Horror movies are mostly below $\mathrm{PC2} = 0$; Drama and Fantasy are more located $\mathrm{PC1} > 0$.

**- Comparing the 3 methods:**

In general, they produce very different visualizations for the same movies, but there seems to be some patterns lying underneath for each method, albeit differently. Still, I think the 2D representations are hard to spot many trends/patterns, at least to my eyes.

**- Comparing genres:** My piazza post @349 discusses the comparison of genres using mean and standard deviation of ratings of all movies in a genre. In short, by clustering movies into genres, we seem to be able to separate more popular genres from the less popular ones, especially using standard deviation of all ratings within a genre. We can train a classifier to determine whether a movie is in a popular genre, or a regressor on how likely it is for a movie to be popular. Please read my piazza post for more on this.


Category Rating Mean
Unknown
Western
Musical
Horror
Fantasy
Documentary
Thriller
War
Childrens
Crime
Action
Animation
Comedy
Mystery
Romance
Adventure
Drama
Sci-Fi
Film-Noir
PC 1
PC 2


Category Rating Standard Deviation (Normalized)
Mystery
Fantasy
Horror
Romance
Thriller
Adventure
Film-Noir
Action
Sci-Fi
Drama
Crime
Comedy
Childrens
Musical
War
Documentary
Unknown
Animation
Western
PC 1
PC 2