Q1:
(i). Get the singular value vector of SVD of this dataset, sorting them in descending order.
(ii). What is the likely value of k (degree of the data).
(iii). Show a computational efficient way to compute the SVD decomposition of XX^T.
A:

(i). We first construct the matrix X and assign values from the data sheet for each $X[i, j]$. The SVD (singular value decomposition) of matrix is denoted as:

$$X = U\Sigma V^T$$

(note V^T is actually the conjugate transpose). The dimension of matrix X is 1000×40 (which is much smaller compared to the matrix XX^T, which we will talk about in part (iii) of the problem). The singular values vector is documented as in Table 1.

```
[ 1089.67547396  1027.89906723   909.56529034   885.82803953   759.97260629
   683.9141783    663.09093336   565.85700372   509.46586879   461.29400539
     9.15827494     9.00480151     8.90682002     8.80650525     8.66727228
     8.58844142     8.51717892     8.41727672     8.28530571     8.26504108
     8.17096672     8.13002631     8.06764902     7.98587734     7.91909172
     7.8586083      7.80691957     7.68862159     7.62347627     7.54280159
     7.43411144     7.33798541     7.26026044     7.1937195      6.97768789
     6.93967099     6.8187177      6.72712347     6.68866903     6.59213939]
```

Table 1. The Singular Value Vector

(ii). The value k of the degrees of the freedom can be examined by examining the singular values. There is a steep drop after 10th singular value according to Table 1., hence the degree of freedom k = 10 for our dataset.
Alternative approach (which eventually give the same result, as shown in Figure 1) is to examine the components using PCA algorithm. Our X is a 1000 by 40 matrix. Obtain $A = XX^T$ and using eigen value decomposition to obtain

$$A = P\Lambda P^T$$

where Λ is the eigenvalue diagonal matrix and P is the eigenvector matrix, then the standard deviation of component j is:

$$\sigma_j = \sqrt{\frac{1}{p}(PAP^T)_j}$$

We can easily verify that the steep drop occur after the 10th component. Hence the degree of freeom is 10.

(iii). XX^T is a 1000×1000 matrix. The computation for singular values is more expansive given the larger dimension of the matrix. We know from previous parts:

$$X = U\Sigma V^T$$

$$X^T = (U\Sigma V^T)^T = V\Sigma U^T$$

$$\rightarrow XX^T = U\Sigma^2 U^T$$

Luckily, we already obtained U and Σ from previous problem, hence we do not need to to re-do SVD for the matrix of dimension 1000×1000. We simply need to use U and Σ obtained from the matrix X of dimension of 1000×40 to construct the SVD of XX^T.

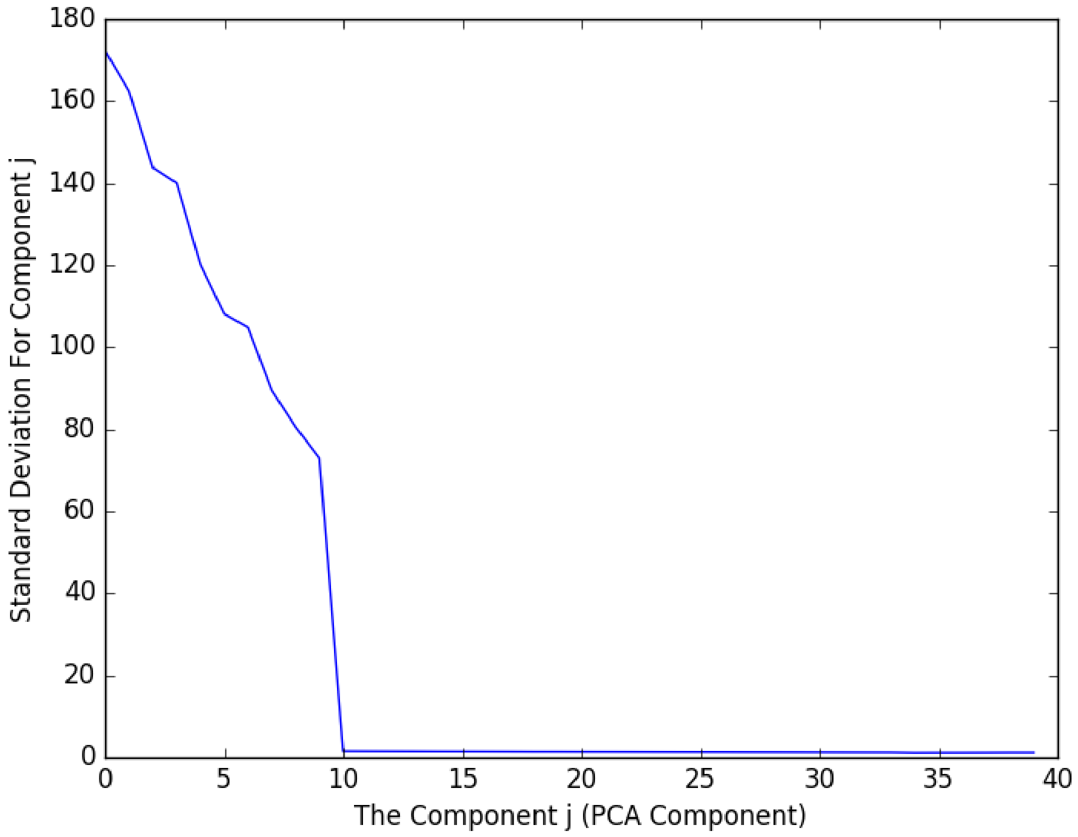
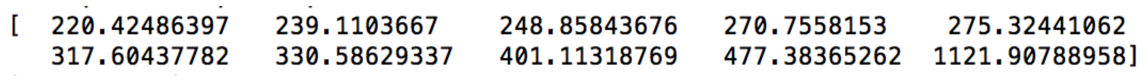


Figure 1: The standard deviation of component j (the first component is j = 0)

Q2:
(i). Using a sparse matrix SVD implementation to find the SVD decomposition of the *user* $\times$ *movie* rating matrix, where $X_{i,j}$ is the rating of the user i of movie j. Assume missing entries are zero. Write the equations and report the top 10 singular values in decreasing order.
(ii). Explain why sparse matrix SVD decompostion for predicting the missing ratings is undesirable when there are missing entries (i.e. why we should not assume the missing entries are zero)
(iii). For all non-zero entries $X_{i,j}$, subtract $X_{i,j}$ by the average α_j rating of movie j of the observed entries, $\alpha_j = \frac{\sum_i \mathbf{1}\{X_{i,j}>0\}X_{i,j}}{\sum_i \mathbf{1}\{X_{i,j}>0\}}$. Call this new sparse matrix Y. How can you use an SVD decomposition to predict whether or not user i will like movie j more than the average user that has seen j.
(iv). Use ICA decomposition of X to project all movies into two dimensions using scatter plot. Highlight the 2D positions of movies id 111, 288, 102, 291.
(v). Use the SVD result to do a PCA projection of all movies over the two most relevant dimensions for romance and action.

A:

(i). As we will be working with a sparse matrix, instead of using a regular numpy type of matrix, we use the `sparse` module from `scipy` library. The matrix is constructed similar as to those in previous part, where the entry $X[i,j]$ has been assigned user i's rating on movie j, if such a rating exists. We will leave the rest of the matrix entries '0' (however this is not quite desired). Using the sparse SVD implementation we will only extract the first 10 largest singular values. The first 10 largest singular values are documented as in Table 2. For the reminder of the task we will always tend to use up to a very limited amount of singular values to decomposite/reconstruct, as it is very computational expensive to perform SVD on sparse matrices.



```
[  220.42486397   239.1103667    248.85843676   270.7558153    275.32441062
   317.60437782   330.58629337   401.11318769   477.38365262  1121.90788958]
```

Table 2. The largest 10 singular values from sparse SVD

(ii). We have assigned missing entries zero. This is undesirable because a value of '0' from machine's perspective is not a missing value, but a score lower than 1. For movie ratings, a score 1 means user hate the movie and a score 5 means user love the movie. A score '0' means user hate the movie even worse compared to assigning it a score 1. The User-Movie-Rating matrix are mostly filled with '0' and computer can perceive it as most users hates most of the movies, which is not the case. Hence, using unscaled ratings and assign missing values as 0 **misinterpret** the data.

(iii). We build matrix Y by subtracting α_j from non-zero $X_{i,j}$ in matrix X. by doing this we neutralize the meaning of '0' (so that '0' does not mean hate the movie, but instead a neutral attitude on the movie).We can conduct the SVD on matrix Y.

$$Y = U\Sigma V^T$$

After we have conducted SVD decomposition and obtain the 10 largest singular values and the associated eigenvectors, we reconstruct Y using the singular values and vectors obtained from decomposition. Assume m is the dimension of users and n is the dimension of movies.

$$Y \approx U'\Sigma' V'^T$$

where U' is $m \times 10$, Σ' is 10×10 and V'^T is $10 \times n$. (And of course, we have lost information by disregarding the rest of singular values). Reconstructed Y would then be:

$$U'\Sigma' V'^T = Y'$$

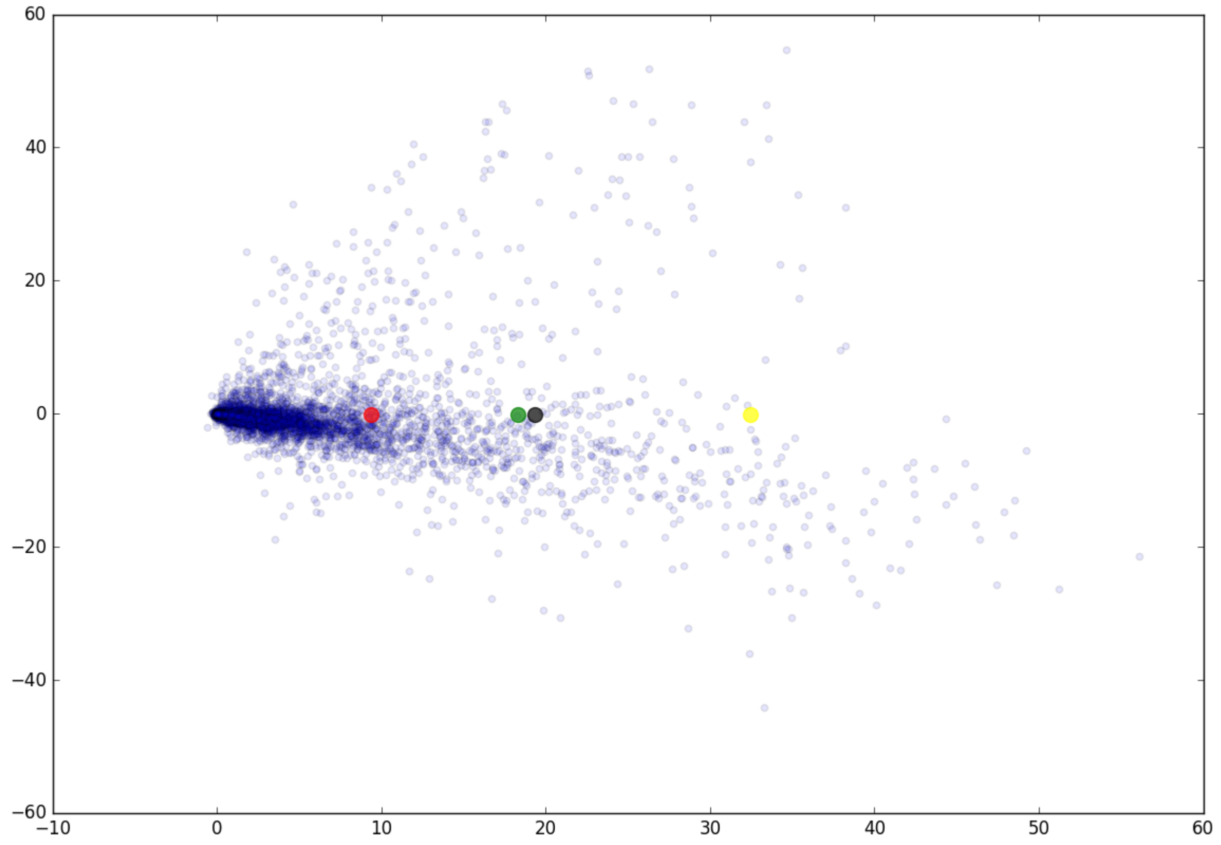


Figure 2: The ICA Scatter Plot of Movie's Points Cloud The specifically marked points are: Red - '111', Green - '288', Yellow - '102' and Black - '291'

From the reconstructed matrix Y'. For entry location (i, j), if $Y'[i, j] > 0$ and $X[i, j] = 0$ (originally missing values), then it would suggest that user i likes the movie j more than the average user that has seen movie j.

(iv). Independent component analysis (ICA): Given p-dimensional observed r.v. $\mathbf{x}$, find mixing matrix $\mathbf{H}$ and source $\mathbf{s}$ such that:

$$\mathbf{x} = \mathbf{Hs}$$

Now the problem is matrix X is very sparse, and if we attempt to use `FastICA` implementation in `sklearn` library it will not work as it would only work on dense matrix. To solve the issue we convert the sparse matrix built in previous part into dense matrix. (Can be solved in one line `Dense_Matrix = Sparse_Matrix.toarray()`) We set the number of latent components to be '2' (so that we could obtain 2D projection). Movies are decomposite into two components using ICA, Each component is a vector of $1 \times n$. More specifically, following ICA the decomposite components would be:

$$x_i = h_{i,1}s_1 + h_{i,2}s_2$$

We scatter plot the two components and highlight the 4 specific movies, as shown in the Figure 2 below. Note that each run may have varied results due to randomization in ICA. The ICA is unique up to a trivial transformations (i.e. a scale-permutation).

(v). Similar as in part (i) of Problem 2 we perform singular value decomposition of matrix Y but only

using the first 10 singular values with their associated vectors.
As there are no definitive correct way to determine which 2 dimensions are associated with action and romantic movie tags, thus this is an open problem. The two dimensions associated with action and romance movies are selected from the 10 latent vectors obtained using SVD. The dimension for each of the vector would be $1 \times n$ where n is the number of the movies.
We first create two binary vectors associated with action and romance categories. If a movie ID is within a category, then the flag in the binary vector is set to be 1 and 0 otherwise. We attempt to pick the action/romance dimension vector using cross-correlation. Intuitively the singular vector that has the highest cross-correlation should be associated with action/romance category. The cross correlation is defined as:

$$\rho_{xy} = \frac{E[(\vec{X} - \mu_{\vec{X}})(\vec{Y} - \mu_{\vec{Y}})]}{\rho_{\vec{X}}\rho_{\vec{Y}}}$$

After we pick the corresponding singular vector associated with action and romance movie, we construct a matrix X with dimension of $n \times 2$ (n is the number of movie). Following PCA procedure:

$$A = XX^T$$

Conduct eigen value decomposition

$$A = P\Lambda P^T$$

The PCA projection would then be:

$$PCA\ projection = PX$$

The PCA projection of all movies over the two most relevant dimensions for romance and action would then be as shown in Figure 3.

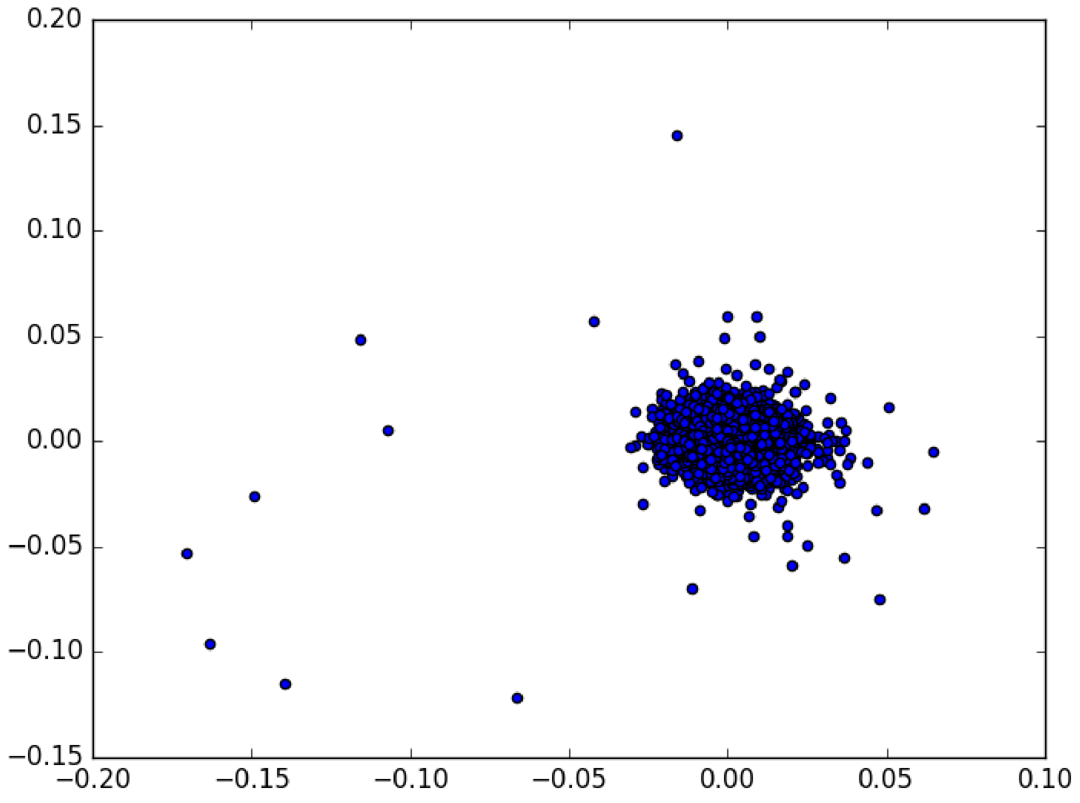


Figure 3: The PCA projection of all movies over the two most relevant dimension