Movie Recommendation Grand Challenge

Ben Strober, Yuan He, Zhishuai Zhang
Johns Hopkins University
3400 N. Charles Street, Baltimore, MD 21218
{bstrobe1, yhe23, zzhang99}@jhu.edu

Abstract

In this report, we investigate an important and popular problem, which tries to provide movie recommendations by predicting user ratings, as well as clustering groups of movies according to known user ratings. This problem has strong connection to matrix completion and high dimensional clustering naturally. To address this problem, three matrix completion algorithms and three clustering algorithms have been investigated and evaluated. We verify that both aspects of this problem can be solved reasonably well by our simple methods.

1. Introduction

We are motivated by the problem of predicting a user's movie preference based on ratings that user has previously given to other movies, as well as ratings provided by other users. As every user does not rate every available movie, this data set is quite sparse. Luckily, it is known to have a low-dimensional structure so we will utilize various low rank matrix completion methods. Additionally, we wish to discern additional structure/information provided by performing an unsupervised clustering of the movies. Nearly all of the algorithms provided come from the class text book book [5].

The remainder of this report is organized as follows. Section 2 introduces the problems our project is trying to solve. The proposed methods for the problems are illustrated in Section 3, and comprehensive experimental results are shown in Section 4. Conclusions are drawn in Section 5. Section 6 is the miscellanea.

2. Problem Description

2.1. Dataset

A popular movie rating dataset *MovieLens* [1] is used in our project. We select the 100K dataset published in Apr 1998, which contains 100,000 ratings range in 1-5 from 943 users on 1682 movies. In this dataset, each user has rated at least 20 movies to make sure reasonable information of each user has been contained.

We select two genres in this 100K dataset, *Children* and *War* whose styles and audiences may be quite distinct, in this way, four subsets are composed – *Children* subset (subset 1), *War* subset (subset 2), union of *Children* and *War* subset (subset 3), and the full 100K dataset (subset 4).

2.2. Matrix Completion

One important task of a movie recommendation system is to predict the rating of a movie for a user who has not provided a rating for that movie. This task is essentially a matrix completion task which will be addressed in our project.

For each subset of movies , the data set can be represented as a matrix whose rows and columns are users and movies respectively. The observed values of this matrix range in 1-5; the unobserved ratings are considered missing entries, and are assigned a value of zero. The goal is to predict the unknown entries from the known entries. Formally, given rating matrix $X \in \mathbb{R}^{D\times N}$, in which D is number of users and the N is the number of movies, and the known entry indexes $\Omega \subseteq \{(i,j) : i \in \{1,2,\cdots D\}, j \in \{1,2,\cdots,N\}\}$, and $X_{(i,j)\in\Omega} \in \{1,2,3,4,5\}$, our goal is to predict $X_{(i,j)\notin\Omega}$.

2.3. Clustering

Learning movie categories (ideally genres) directly from the users' ratings in an unsupervised fashion can provide information about the movies and new users' preference. Thus it is another important task in the movie recommendation system. Given the large number of users, unsupervised learning on the high dimensional data is required.

Based on matrix completed by the best matrix completion method, the movies are clustered based on users' rating. For example, if two movies share similar ratings from the users, then these two are clustered together. Formally, given a rating matrix $X \in \mathbb{R}^{D\times N}$, the goal is to cluster columns in X: $x_j, j \in \{1,2,\cdots,N\}$, into n distinct clusters based on the features in the D-dimensional space.

3. Proposed Solution

3.1. Matrix Completion

For the matrix completion task, three methods are proposed to address it. The detailed description of them are in the following three subsections.

3.1.1 Matrix factorization

The first method for matrix completion is the matrix factorization method whose success has been shown in the Netflix Prize in 2009 [3]. The basic idea of this method is to map both users and items to a joint latent factor d-dimensional space $U \in \mathbb{R}^{d\times D}$ and $V \in \mathbb{R}^{d\times N}$, and then model the user-item interaction as the inner product in that space, i.e. $X = U^T V$, $X_{(i,j)} = \boldsymbol{u}_i^T \boldsymbol{v}_j$. Then the loss function will be $||P_\Omega(X - U^T V)||_F^2$, where P_Ω is a mask operator which set all entries outside Ω to be zero. In addition, to avoid overfitting to the training data, a regularization term $\lambda_u||U||_F^2 + \lambda_v||V||_F^2$ is added to the loss function. Finally our goal is to minimize $||P_\Omega(X - U^T V)||_F^2 + \lambda_u||U||_F^2 + \lambda_v||V||_F^2$ over U, V and d.

Two methods have been introduced to solve the above optimization problem [3]. One is stochastic gradient descent (SGD) and the other one is alternating least squares (ALS).

In SGD, the optimizer runs over the training dataset, and for each given training entry $X_{(i,j)}$, the optimizer computes the current prediction according to the current model, and calculate the prediction error $E_{(i,j)} = X_{(i,j)} - \boldsymbol{u}_i^T \boldsymbol{v}_j$. Then the parameters in the model are tuned according to the gradient: $\boldsymbol{u}_i \leftarrow \boldsymbol{u}_i + \gamma(E_{(i,j)}\boldsymbol{v}_j - \lambda_u \boldsymbol{u}_i)$ and $\boldsymbol{v}_j \leftarrow \boldsymbol{v}_j + \gamma(E_{(i,j)}\boldsymbol{u}_i - \lambda_v \boldsymbol{v}_j)$.

In the ALS method, the optimizer alternates between U and V. In each iteration, the U is fixed, and the V is solved as a least-squares problem, and then the U is optimized while the V is fixed. Considering there are only part of the entries known, by solving the least-squares problem, we get the following updating equations: $\boldsymbol{v}_j \leftarrow (\lambda_v I + \sum_{i:(i,j)\in\Omega} \boldsymbol{u}_i \boldsymbol{u}_i^T)^{-1} \sum_{i:(i,j)\in\Omega} \boldsymbol{u}_i X_{(i,j)}$, and $\boldsymbol{u}_i \leftarrow (\lambda_u I + \sum_{j:(i,j)\in\Omega} \boldsymbol{v}_j \boldsymbol{v}_j^T)^{-1} \sum_{j:(i,j)\in\Omega} \boldsymbol{v}_j X_{(i,j)}$.

In our project, the latter one (ALS) is selected as the optimizer, since it does not require to loop over each single training entry a time, which may be inefficient where the training data is huge and non-sparse.

3.1.2 Robust PCA with missing data

The second method is robust-PCA assuming outliers and noise, and with missing data. The objective function is: $min_{L,E}||L||_* + \lambda||E||_{2,1}$ s.t. $P_\Sigma(X) = P_\Sigma(L + E)$.

To solve this problem, we used augmented Lagrange multiplier algorithm, where the Lagrangian function can be written out as:

$$\mathcal{L} = ||L||_* + \lambda||E||_{2,1} + \langle Z, \mathcal{P}_\Omega(X) - \mathcal{P}_\Omega(L + E)\rangle + \frac{\beta}{2}||\mathcal{P}_\Omega(X - L - E)||_F^2$$

The objective function is updated iteratively for L, E and Z.

1. To optimize over L:

$$\begin{aligned}\mathcal{L} &= ||L||_* + \langle Z, \mathcal{P}_\Omega(X) - \mathcal{P}_\Omega(L + E)\rangle \\ &+ \frac{\beta}{2}||\mathcal{P}_\Omega(X - L - E)||_F^2 + \textit{constant} \\ &= ||L||_* + \langle Z, \mathcal{P}_\Omega(X - L + E)\rangle \\ &+ \frac{\beta}{2}||\mathcal{P}_\Omega(X - L - E)||_F^2 + \textit{constant} \\ &= ||L||_* + \frac{\beta}{2}||\mathcal{P}_\Omega(X - L - E) + \frac{Z}{\beta}||_F^2 + \textit{constant} \\ &= ||L||_* + \frac{\beta}{2}||(X - L - E) + \mathcal{P}_\Omega(\frac{Z}{\beta})||_F^2 + \textit{constant}\end{aligned}$$

Setting $\partial L = 0$, the solution is:

$$L_{k+1} = \mathcal{D}_{1/\beta}(X - E_k + \frac{1}{\beta}\mathcal{P}_\Omega(Z_k))$$

2. To optimize over E:

$$\begin{aligned}\mathcal{L} &= ||E||_{2,1} + \langle Z, \mathcal{P}_\Omega(X) - \mathcal{P}_\Omega(L + E)\rangle \\ &+ \frac{\beta}{2}||\mathcal{P}_\Omega(X - L - E)||_F^2 + \textit{constant} \\ &= ||E||_{2,1} + \langle Z, \mathcal{P}_\Omega(X - L + E)\rangle \\ &+ \frac{\beta}{2}||\mathcal{P}_\Omega(X - L - E)||_F^2 + \textit{constant} \\ &= ||E||_{2,1} + \frac{\beta}{2}||\mathcal{P}_\Omega(X - L - E) + \frac{Z}{\beta}||_F^2 + \textit{constant} \\ &= ||E||_{2,1} + \frac{\beta}{2}||(X - L - E) + \mathcal{P}_\Omega(\frac{Z}{\beta})||_F^2 + \textit{constant}\end{aligned}$$

Setting $\partial L = 0$, the solution is:

$$E_{k+1} = A\mathcal{S}_{\lambda/\beta}(\textit{diag}(a))\textit{diag}(a)^{-1}$$

Where

$$A = X - L_k + \mathcal{P}_\Sigma(\frac{Z_k}{\beta})$$

And a is a vector where

$$a_j = ||A_{,j}||_2$$

3. To optimize over Z:

$$Z_{k+1} = Z_k + \beta \frac{\partial \mathcal{L}}{\partial Z} = \beta \mathcal{P}_\Sigma(X - L_k - E_k)$$

3.1.3 Low Rank Matrix Completion

The goal of low rank matrix completion is to minimize the following objective function:

$min_A\ \tau||A||_* + \frac{1}{2}||A||_F^2$ s.t. $P_\Omega(A) = P_\Omega(X)$
Using the method of Lagrange multipliers, we can write this minimization problem with the Lagrangian:

$L(A, Z) = \tau||A||_* + \frac{1}{2}||A||_F^2 + \langle Z, P_\Omega(X) - P_\Omega(A)\rangle$
The optimal solution can be found by iteratively solving the following two steps:

$A_k = argmin_A L(A, Z_{k-1})$
$Z_k = Z_{k-1} + \beta \frac{\partial L}{\partial Z}(A_k, Z_{k-1})$

Following some routine algebra (performed in lecture and in the textbook) we can find the following, simplified update equations:
$A \leftarrow D_\tau(P_\Omega(Z))$
$Z \leftarrow Z + \beta(P_\Omega(X) - P_\Omega(A))$

3.2. Clustering

For the clustering task, three methods are proposed to address it. The detailed description of them are in the following three subsections.

3.2.1 Normalized spectral clustering

The first method for clustering is the normalized spectral clustering introduced in 2001 [4] which is helpful when the natural clusters do not correspond to convex regions. In this method, Laplacian matrix $\mathcal{L}$ is computed to map the original data points $\{\boldsymbol{x}_i\} \subset \mathbb{R}^D$ into a new set of points $\{\boldsymbol{y}_i\} \subset \mathbb{R}^n$ embedded in a low-dimensional space through its null space. The structures of the low-dimensional embedded data become much simpler than the original data, and they can be grouped by a simple clustering method such as k-means. The algorithm is drawn in Algorithm 1. This algorithm is also a infrastructure of the following two algorithms, and as shown in Section 4, this method can already yield very good results when the input genres are distinct.

3.2.2 Locally Linear Manifold Clustering

This method to cluster data points is based on the assumption that the affine subspace spanned by data point x and its K nearest neighbors can be approximated by a same affine subspace but in low-dimensional embedding. The goal is to find such affine subspace, and a low-dimensional embedding to minimize the reconstruction error. With the coefficient matrix $C \in \mathbb{R}^{N\times N}$, the weight matrix $\mathcal{W}$ can be constructed by $\mathcal{W} = |C| + |C^T|$. Then normalized spectral clustering is applied as shown in Algorithm 1. The algorithm is drawn in Algorithm 2.

Algorithm 1: Normalized spectral clustering

Input : Number of clusters n and high dimensional coordinates $\{\boldsymbol{x}_i \in \mathbb{R}^D\}_{i=1}^N$.
1 Calculate affinity matrix $W \in \mathbb{R}^{N\times N}$, where $W_{i,j} = \exp(-||\boldsymbol{x}_i - \boldsymbol{x}_j||^2/2\sigma^2)$
2 Compute the degree matrix $\mathcal{D} = diag(W\mathbf{1})$ and the Laplacian $\mathcal{L} = \mathcal{D} - W$
3 Compute the n eigenvectors of $\mathcal{D}^{-1/2}\mathcal{L}\mathcal{D}^{-1/2}$ associated with its n smallest eigenvalues and normalize so that each rows of $[\boldsymbol{u}_1, \cdots, \boldsymbol{u}_n]$ has unit norm.
4 Let $\{\boldsymbol{y}_i\}$ be the columns of $Y = [\boldsymbol{u}_1, \cdots, \boldsymbol{u}_n]^T$.
5 Cluster the points $\{\boldsymbol{y}_i\}$ into n groups using the k-means algorithm.
Output: The segmentation of the data into n groups.

Algorithm 2: Locally Linear Manifold Clustering

Input : Number of clusters n, number of K-NN k and high dimensional coordinates $\{\boldsymbol{x}_j \in \mathbb{R}^D\}_{j=1}^N$.
1 Find the K-NN of each data points x_j
2 Compute the affine combination of K-NN for each data point such that $x_j = \sum c_{ij}x_i$
3 Calculate affinity matrix $W \in \mathbb{R}^{N\times N}$, where $W = |C| + |C^T|$
4 Perform step 2-5 in Algorithm 1.
Output: The segmentation of the data into n groups.

3.2.3 Matrix LASSO Minimization by ADMM

This algorithm seeks to model each point as a sparse representation of all of the other points. In doing so, we will represent each points using as few other points as possible (this will be achieved by using points in the same subspace). More rigorously, we are looking to optimize the following problem:
$min_{C,Z}\ ||C||_1 + \frac{\tau}{2}||X - XZ||_F^2$ s.t. $Z = C - diag(C)$

The augmented Lagrangian to this problem is given by:
$L(C, Z, \Delta_2) = ||C||_1 + \frac{\tau}{2}||X - XZ||_F^2 + \frac{\mu_2}{2}||Z - (C -$

$diag(C)) + \frac{\Delta_2}{\mu_2}||_F^2 + \eta$

After solving this algorithm, we construct affinity matrix W, $W = |C| + |C|^T$. We use than use normalized spectral clustering (Algorithm 1) to cluster the data.

4. Experiments

As discussed in Section 2, we use the *MovieLens* 100K dataset for our project, we select two genres – *Children* and *War*, and then compose four subsets discussed in Section 2. We will run and evaluate our methods on the above four subsets.

In addition, to make a more comprehensive experiment, we randomly select 3%, 6%, 9%, 12% and 15% entries to be test entries and remove them from the training matrix. In this way, we create several train-test datasets with different sparsity levels (range from 3% to 15%). We perform the sparsity selection 20 different times with different random seeds, to obtain 20 different train-test datasets for each sparsity level. Finally, we generate $5 \times 20 = 100$ train-test datasets, containing 5 different sparsity levels. Our methods will be ran and evaluated on those 100 datasets and statistical results will be shown in this section.

4.1. Matrix Completion

4.1.1 Detail

To evaluate the matrix completion performance, a popular metric named root mean square error (RMSE) which has been discussed in [2] is used in our project. The error is defined to be error $= \sqrt{\frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{n}}$, in which $\hat{y}_i$ is the prediction of the i-th entry, and the y_i is the ground truth of the i-th entry.

For the matrix factorization method, the alternating least squares optimizer is implemented and evaluated. The hyper-parameters are set to be $\lambda_u = \lambda_v = 1$, and our experiment shows they will not affect the RMSE result dramatically but will affect the converging speed. The d is initially brute-force searched and we finally find that when d goes up, the loss function will reduce, but the RMSE result will oppositely go up. So finally the d is set to be 1 to report performance. The convergence of U and V are determined by the percentage of loss function drops, and if it drops less than 5%, the training is considered to converge. Quantitative results will be reported in subsection 4.1.2.

For the second matrix completion method, the penalty parameters λ and β are selected to minimize RSME in the testing data.

For the third matrix completion method, τ was selected based on a univariate grid-search of values of tau that minimized the RMSE in the medium data-set composed of movies from the Romance and Drama categories.

4.1.2 Result

Follow the implementations and hyper-parameters discussed above, the quantitative results are shown in Figure 1, for all five sparsity levels, all four subsets and all three matrix completion methods. In each boxplot, the red line means the median over 20 trials, and top and bottom of the box correspond to the 75th percentile and 25 percentile respectively. It can be easily found that the matrix factorization method achieves the best performance with RMSE around 1. Another finding is with more data (subset 3 and 4), the performance will be better.

Another important experiment is the optimization over d for the matrix factorization method. d is initially searched by brute-force search. As we can see in Figure 2, when d goes up, the loss function will reduce, however, the RMSE will increase, which may mean a more complicated model with a higher d can easily get overfitted to the training data, thus perform well on the training entries but bad on the testing entries.

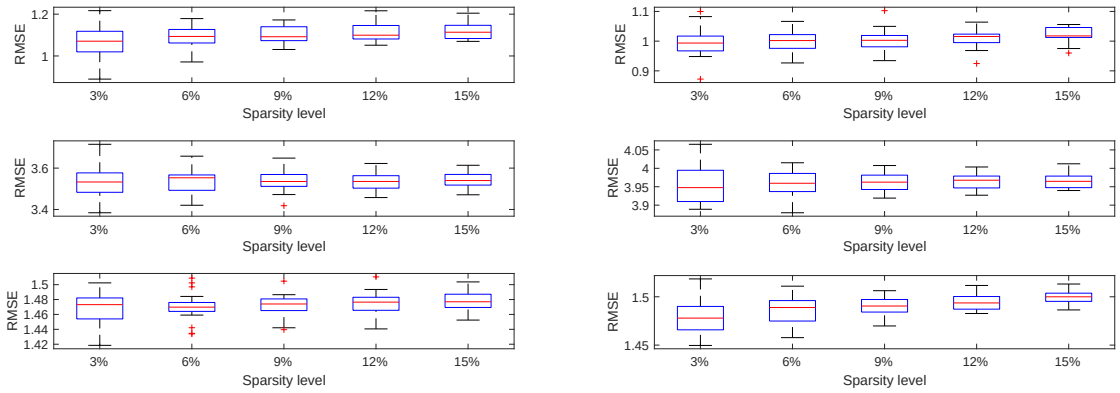
4.2. Clustering

4.2.1 Detail

For the first method, the Gaussian distance (radial basis function kernel) is used to compute affinity matrix, and the hyperparameters are selected to be $2\sigma^2 = 10$, and k-means provided by MATLAB is used to do clustering in Step 5. For the second method, the number of neighbors was chosen to optimize the clustering performance. For the third method, both hyperparameters were selected using a grid search that minimized the pvalue of the fisher exact test's while using the 'drama' and 'comedy' data.

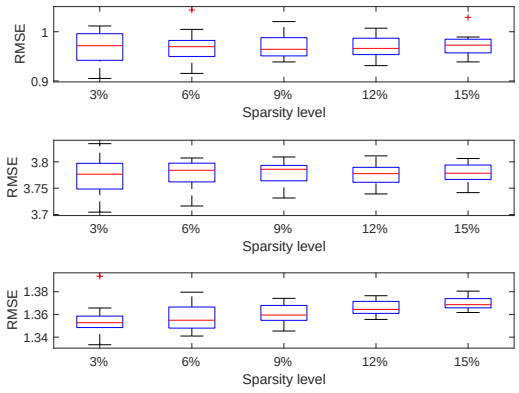
To evaluate the clustering quality in the median dataset consisting of movies from two genres (war and children), both visualization and quantitative measurement are computed. First, the data points in the 2-dimensional embedding are plotted, with different colors representing different genres. The more separable the two colors are, the better clustering is. Then a contingency table was constructed with each row representing a cluster, first column representing Genre 1 and the second column representing Genre 2. Fisher exact test was done for this table to compute odds ratio and the p-value.

To evaluate the clustering quality in the large dataset containing all movies from 19 genres, first the fraction of overlap divided by the union between a cluster and a genre is computed, and then hierarchical clustering is applied to the 19×19 matrix to show how similar the clusters are to the genres.

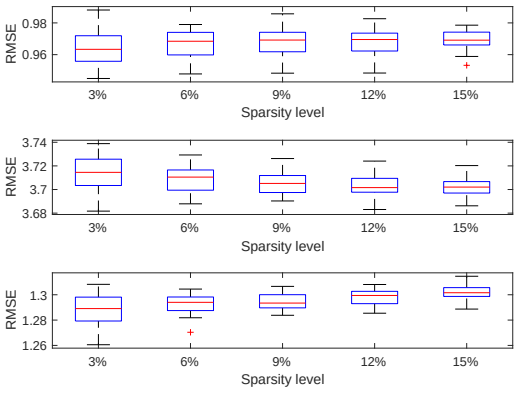


(a) subset 1

(b) subset 2



(c) subset 3



(d) subset 4

Figure 1. Results for matrix completion, the top, middle and bottom plots correspond to matrix factorization, LRMC and robust PCA respectively

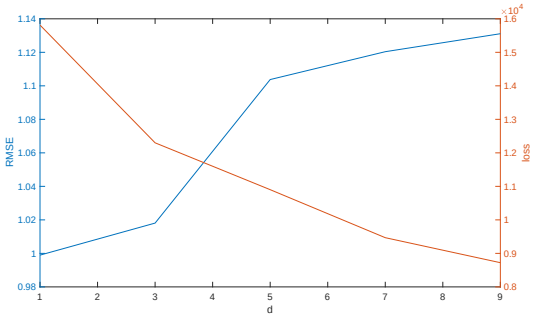


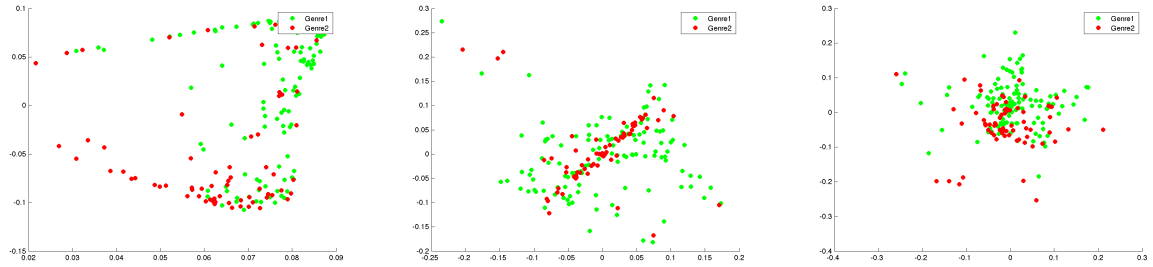
Figure 2. Results for matrix factorization with different d

4.2.2 Result

The clustering in the 2-dimensional space is shown in Figure 3. The two colors represent movies' true genres, and the x-axis and y-axis represent coordinates in the low-dimensional space. It shows that all of the three clustering methods performed well in separating the movies from different genres.

Figure 4 shows the distribution of p-value from Fisher exact test comparing the odds ratio of movies in cluster 1 to be included in Genre 1 verses movies in cluster 2 to be included in Genre 1. More extreme p-values represent that the two clusters are separated better. We can see that spectral clustering works the best among these three methods.

Figure 5 shows the clustering of overlapping fraction among clusters and genres. The x-axis represents unsupervised clusters, and y-axis represents genre labels. By looking at the clustering of the rows, we can see that the 19 genres actually form 2 large clusters, with the upper half being War, Mystery etc. and the bottom half including Childing, Comedy, Romance, Drama etc. This reveals that the currently available data cannot provide the resolution to distinguish 19 different genres, but instead can provide insights



(a) Spectral Clustering (b) Local Linear Manifolding Clustering (c) Sparse Subspace Clustering

Figure 3. Visualize the low-dimensional embedding for the median dataset

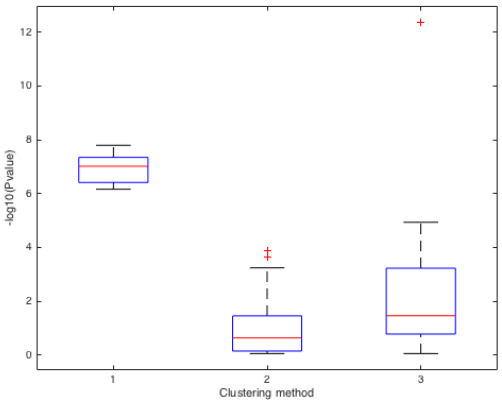


Figure 4. Distribution of p-values in fisher exact test

into 2 far more general categories (perhaps genres). Then the right plot shows the result of clustering all the movies using 2 clusters, and again showed the incapability to pick up 19 genres from the row clustering.

4.3. Matrix Completion after Clustering

Following the methods discussed above, we apply the best matrix completion method to the two clusters yielded by the clustering algorithm.

The results are shown in Figure 6, the rows correspond to cluster 1 ('Children') and cluster 2 ('War'), and the columns correspond to after clustering and using original labels. As we can see, after clustering, the matrix completion will perform better on 'War', but oppositely, worse on 'Children'.

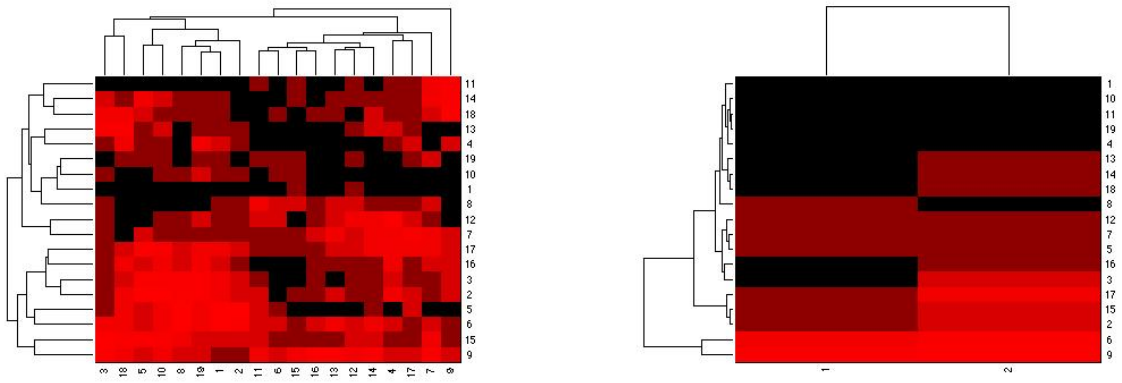
5. Conclusions

We are able to recover (with reasonable accuracy) many missing entries in the ratings (individuals X movies) from an originally very sparse matrix by making the valid assumption that the data comes from an underlying low dimensional matrix. To exploit the underlying low dimensional structure, we performed three different matrix completion methods. All of which are driven by the assumption that the data is low rank. The method that performs best was Matrix Factorization (method 1). This method is unique from the other two methods we tried in that it penalizes both the complexity of the low dimensional representation and the low dimensional mapping of data points onto the low dimensional space. These differences likely contribute to the increased prediction power.

We next try to determine if we can uncover groups of movies that have similar ratings to one another (across samples). And if using these groups, we can perform a more accurate matrix completion. All of the methods we used were based on spectral clustering, but with different affinity matrices. We first discovered that we have more power to cluster movies when the movies we are clustering truly (using observed genre labels) come from distinct/unique genres (ie 'war' and 'children'). We secondly learned we obtained the most accurate clustering (according to known genre labels) when using simple Normalized spectral clustering, which uses a Gaussian affinity matrix. This is likely because the data is quite noisy, and it is near impossible to learn a more complex (nonlinear) affinity. Finally, we show that we successfully uncover approximately 2 groups with clustering, even though the data is said to come from 19 genres. This is likely because this data set does not contain nearly enough information to distinguish artistic genres. Further, we do not observe significant increased power to perform matrix completion using the unsupervised clusters (compared to known genre labels).

6. Miscellanea

Zhishuai implemented 'Matrix Factorization' and 'Normalized Spectral Clustering'. Yuan implemented 'Robust PCA with missing data' and 'Locally Linear Manifold Clustering'. Ben implemented 'Low rank matrix completion' and Matrix LASSO minimization by ADMM'. Experimental analysis was performed as a team.



(a) Fit 19 clusters (b) Fit 2 clusters

Figure 5. Clustering of overlap fraction between clusters and genres in the large dataset

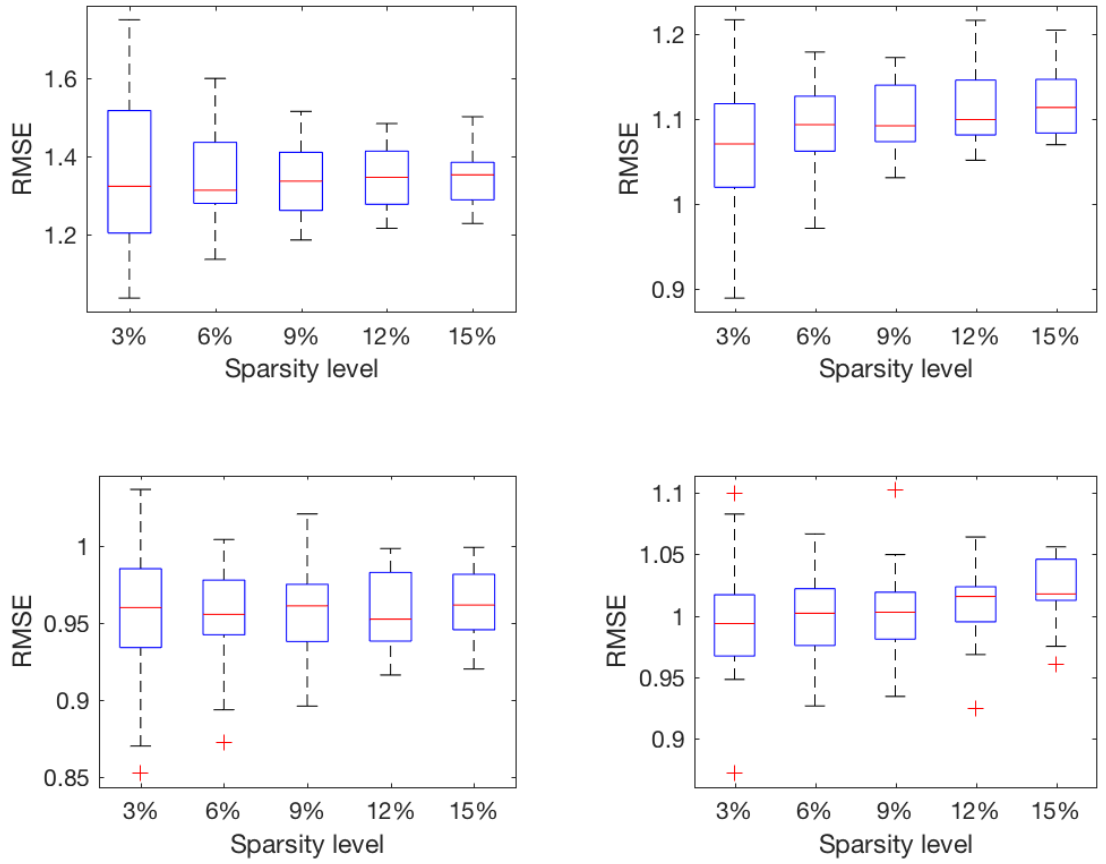


Figure 6. Matrix completion performance after clustering, top row corresponding to cluster 1 (‘Children’), bottom row corresponding to cluster 2 (‘War’); left column corresponding to result from clustering, right column corresponding to result from original genre split

References

[1] F. M. Harper and J. A. Konstan. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(4):19, 2016.

[2] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, Jan. 2004.

[3] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, Aug. 2009.

[4] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 849–856. MIT Press, 2002.

[5] R. Vidal, Y. Ma, and S. Sastry. *Generalized Principal Component Analysis*. Interdisciplinary Applied Mathematics. Springer New York, 2016.