CMSC828V Project 2 Report

Yuelin Liu & Konstantinos Pantazis

November 5, 2020

Note: Code for either student can be found at:

Yuelin Liu: https://github.com/liuy0421/828v-project2

Konstantinos Pantazis: https://github.com/kpantazis/project-2AMSC808N

1 Movie Rankings: Non-Negative Matrix Factorization (NMF)

In this first section, we perform k-means clustering and variations of NMF methods on a complete submatrix of the `MovieRankings36` dataset and compare the performance and results of each method.

1.1 Preprocessing

The complete submatrix, $\tilde{M}$, of the `MovieRankings36` dataset used in this section is obtained by iteratively removing the row or column that has the highest proportion of missing data in a greedy fashion. (See `readdata_p1.m` by Yuelin for implementation detail.)

$\tilde{M}$ has size 7×20, meaning we have simply removed rows of missing data. We can visualize the 7 data points with their first three principle components (Figure 1).

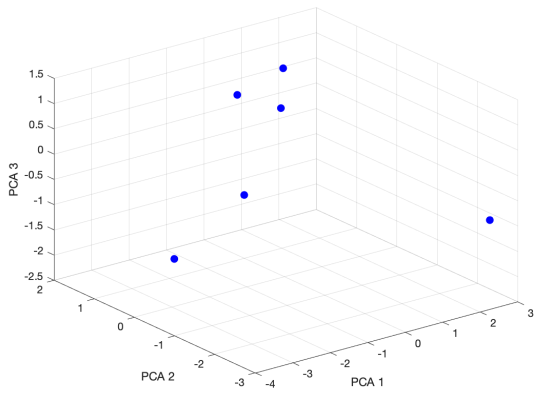

Figure 1: The first, second, and third principle component of the 7 data points in the 7×20 complete submatrix, $\tilde{M}$, from the `MovieRankings36` dataset.

1.2 K-means clustering

We first performed k-means clustering on $\tilde{M}$, with Euclidean distance being the distance metric between any pair of points. We heuristically initialize the k centers by first randomly picking a data point as the first center, then iteratively picking the point furthest from the average of the current centers as the next center. We run the k-means algorithm for a maximum of 100 iterations, and breaks when the cluster assignments cease to change for all the points between iterations. (See `kmeans.m` by Yuelin for implementation detail.)

After running k-means with $k = \{2, 3\}$, $k = 3$ seems to produce relatively stable cluster assignments for the 7 points in $\tilde{M}$. Given the small number of data points, 3 seems to be a reasonable number of clusters.

K-means produces user clusters $\{1, 2, 4, 16\}, \{8, 29\}, \{25\}$ quite consistently. Figure 2 shows the first three principle components of the labeled points.

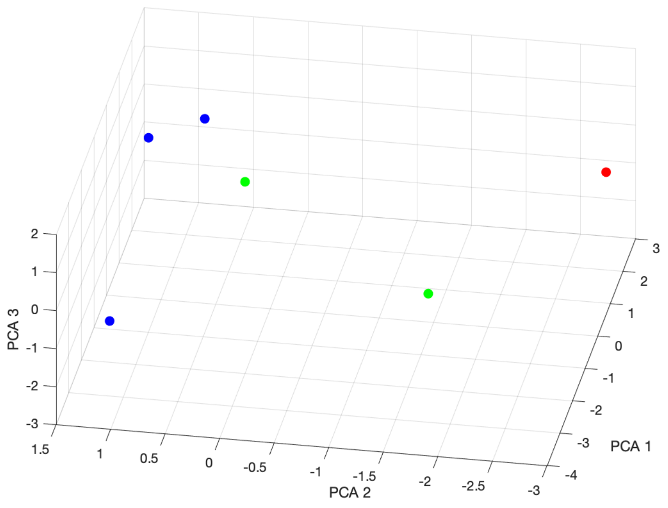

Figure 2: The first, second, and third principle component of the 7 data points in the 7×20 complete submatrix, $\tilde{M}$, from the `MovieRankings36` dataset, labeled by the cluster assignment achieved by Euclidean k-means.

The result from k-means clustering suggests that users $\{1, 2, 4, 16\}$ share similar tastes in movies, users $\{8, 29\}$ have similar tastes in movies, and user $\{25\}$ have their own unique taste in movies from others.

1.3 NMF: Projected Gradient Descent (PGD)

The first NMF method we implemented and experimented with is PGD. We use random initialization strategy for W_0 and H_0, and employ an exponential step decrease strategy $\alpha_k = 0.02/1.01^k$. (See `PGD.m` by Yuelin for implementation detail.)

Figure 3 shows the squared Frobenius error, $\frac{1}{2}\|A - WH\|_F^2$, across 20 iterations of PGD. The squared error decreased exponentially, and convergence was achieved after around 5 iterations, suggesting that PGD performs quite well on our small complete matrix $\tilde{M}_{7\times 20}$.

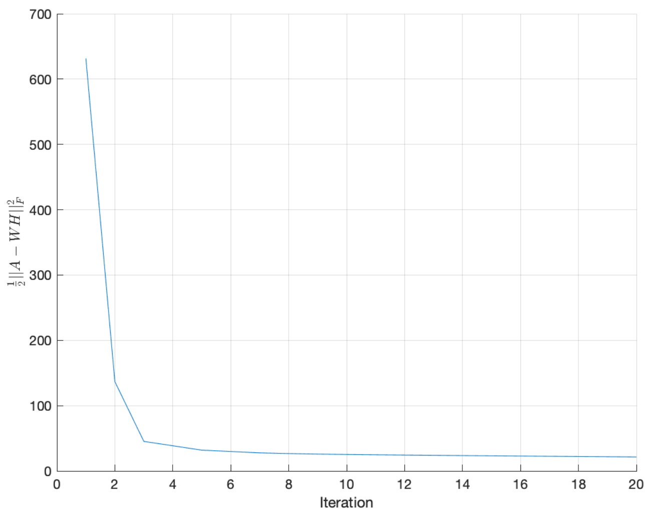

Figure 3: The squared Frobenius error ($\frac{1}{2}\|A - WH\|_F^2$) across iterations (`maxiter` $= 20$) for PGD on $\tilde{M}$. We use the exponential step decrease strategy $\alpha_k = 0.02/1.01^k$.

1.4 NMF: Multiplicative Update [Lee & Seung, 2001]

The second NMF method we implemented and experimented with is the multiplicative update method by [Lee & Seung, 2001]. To make the approaches comparable, we used the same randomly initialized W_0 and H_0 as Section 1.3. In each iteration, we update H then update W according to the updated H. (See `MU.m` by Yuelin for implementation detail.)

Figure 4 shows the squared Frobenius error, $\frac{1}{2}\|A - WH\|_F^2$, across 20 iterations of multiplicative update. The squared error decreased very quickly, and convergence was achieved. It is interesting to note that, multiplicative update achieved a lower Frobenius reconstruction error (17.18) compared to projected gradient descent (21.15).

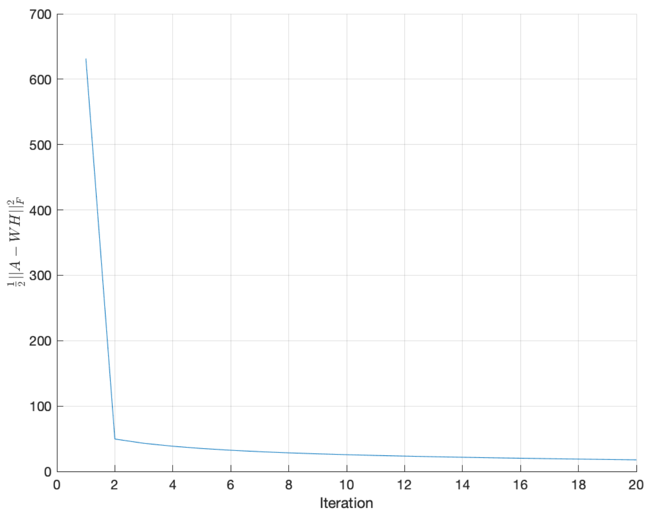

Figure 4: The squared Frobenius error ($\frac{1}{2}\|A - WH\|_F^2$) across iterations (`maxiter` $= 20$) for multiplicative update NMF method by [Lee & Seung, 2001] on $\tilde{M}$.

1.5 NMF: PGD Followed by Multiplicative Update

In class, we discussed that, since multiplicative update tends to steps conservatively, sometimes we achieve better results if we start with PGD, then complete the routine with iterations of multiplicative update. In this section, we use the same randomly initialized W_0 and H_0 as Section 1.3 and 1.4.

The result from Section 1.4 shows that, while multiplicative update made a big initial two steps, it made step 3-5 more conservatively. During iteration 3 to 5, we found that PGD produces lower squared Frobenius reconstruction error than the multiplicative update method, despite that multiplicative update made more aggressive initial two steps. After iteration 5, multiplicative update method consistently produces lower squared Frobenius error afterwards. By this observation, we ran PGD initialized with W_0 and H_0 for 5 iterations, then ran multiplicative update for the rest 15 iterations.

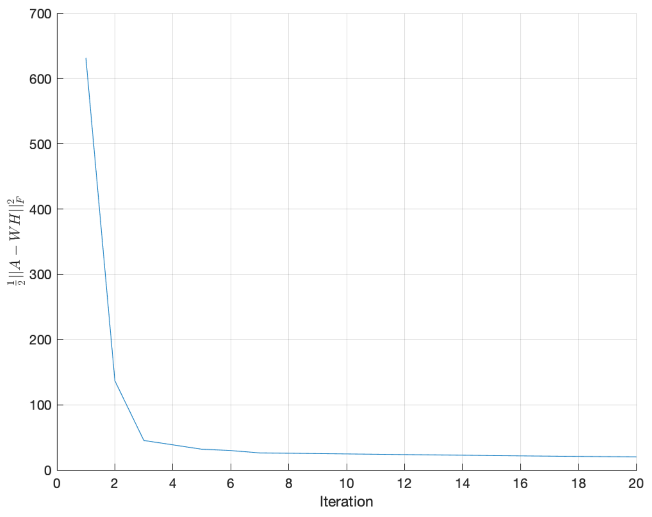

Figure 5: The squared Frobenius error ($\frac{1}{2}\|A - WH\|_F^2$) for PGD (5 iterations) followed by multiplicative update NMF method (15 iterations) on $\tilde{M}$.

Figure 5 shows overall good convergence results for the hybrid method. The final squared Frobenius norm achieved by the hybrid method (11.88) outperforms that of PGD (21.15) or multiplicative update alone (17.18).

Given all results, the the hybrid method with PGD at the start followed by the Lee-Seung multiplicative update scheme seems to be the most efficient.

2 Movie Rankings: Matrix Completion

In the second section, we tackle the matrix completion problem. With observed incomplete matrix $P_{Omega}(A)$, we seek a low rank approximation of A, such that reconstruction error on the observed entries, along with a regularization term on the norm of the factorization, is minimized.

We experiment with two approaches - computing a low rank factorization $A \approx XY$ via alternating directions regularizing Frobenius norms of X and Y (Section 2.1), and computing a low rank approximation $A \approx M$ regularizing with nuclear norm trick on M (Section 2.2) - on the original 36×20 `MovieRankings36` dataset and compare the performance and results of either approaches and with different values for the hyperpamateter $\lambda = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

2.1 Low Rank Factorization via Alternating Directions

The first approach to matrix completion is using alternating directions. We heuristically picked $k = 5$ as the inner dimension for the factorization, and experimented with $\lambda = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. For each experiment, we use the same randomly initialized X and Y matrix for the sake of comparison. (See `AD.m` by Yuelin for implementation details.)

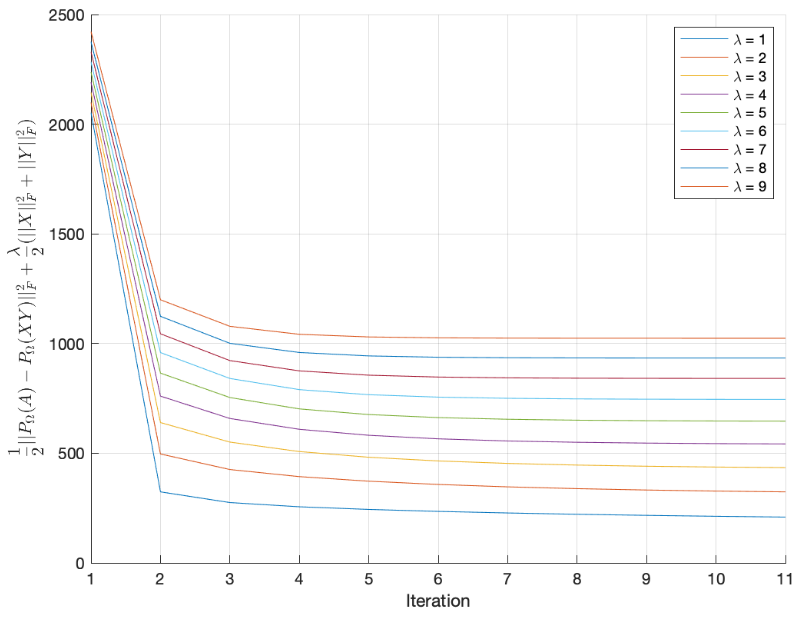

Figure 6: Loss function value $\frac{1}{2}||P_\Omega(A) - P_\Omega(XY)||_F^2 + \frac{\lambda}{2}(||X||_F^2 + ||Y||_F^2)$ across 10 iterations of alternating directions for $k = 5$ and $\lambda = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

Figure 6 shows the value of the loss function $\frac{1}{2}||P_\Omega(A) - P_\Omega(XY)||_F^2 + \frac{\lambda}{2}(||X||_F^2 + ||Y||_F^2)$ across 10 iterations of alternating directions for $k = 5$ and $\lambda = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. As we can see, in general, the objective function values are able to converge quickly across 10 iterations for all 9 λ values; however, the convergence value for the objective function increases with λ. This makes sense as by increasing λ, we are penalizing for not sparse X and Y at an increasing degree, and it's harder to fit the observed matrix $P_{\Omega}(A)$ with too sparse X and Y.

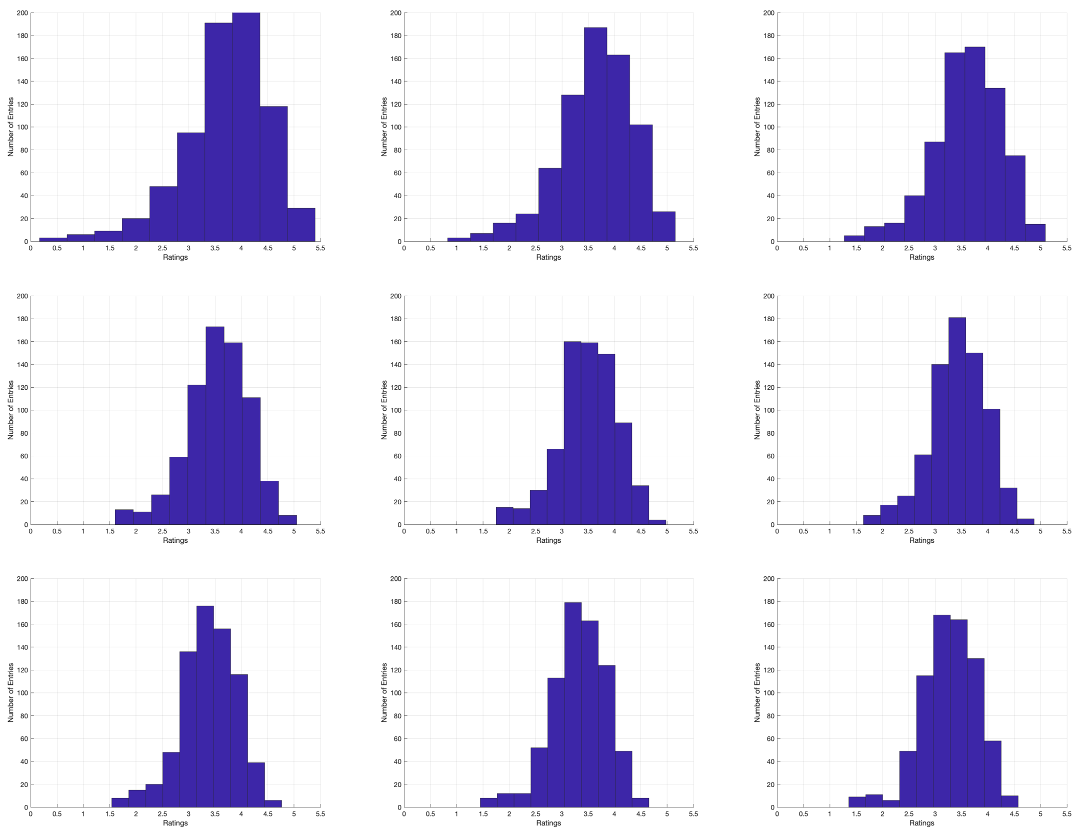

Figure 7: Distribution of ratings produced by XY across 10 iterations of alternating directions for $k = 5$ and $\lambda = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. The plots for the 9 λ values are ordered left to right, row by row. The y-axis range is fixed for all plots.

We also examined the ratings distribution produced by XY for the 9 different λ values (Figure 7). While all resemble normal distributions, the higher λ gets, the closer the mean to the center, and the smaller the variance. Compared to the distribution of ratings from the observed values in the original matrix (Figure 8), which has mean skewed towards the higher end, it seems $\lambda = 2$ produced more reasonable results, especially considering higher λ produced distributions of ratings without 5 star ratings, which is not realistic.

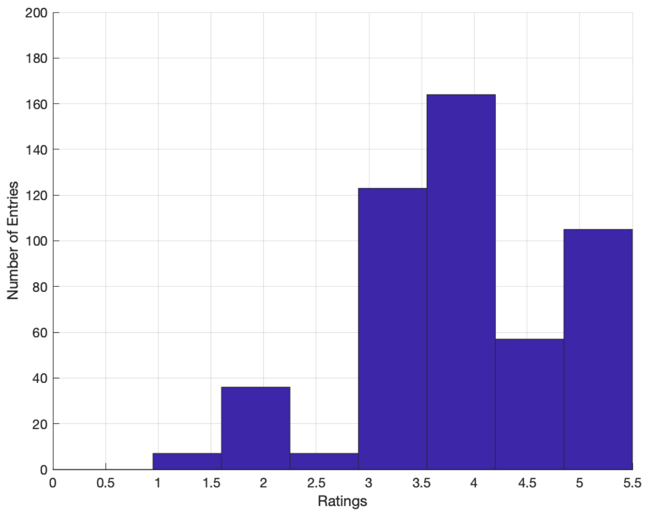

Figure 8: Distribution of ratings In the original `MovieRankings36` dataset.

2.2 Low Rank Approximation via Nuclear Norm Trick

In the second matrix completion approach, instead of explicitly factorizing the original matrix A into low rank matrix product XY, we derive a low rank approximation of A, M, directly. We do so by adding a regularizng term that penalizes the nuclear norm (the sum of singular values) of M. Similar to Section 2.1, we experimented with $\lambda = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, and used the same randomly initialized M_0 for all experiments for the sake of comparison. (See `NNT.m` by Yuelin for implementation detials.)

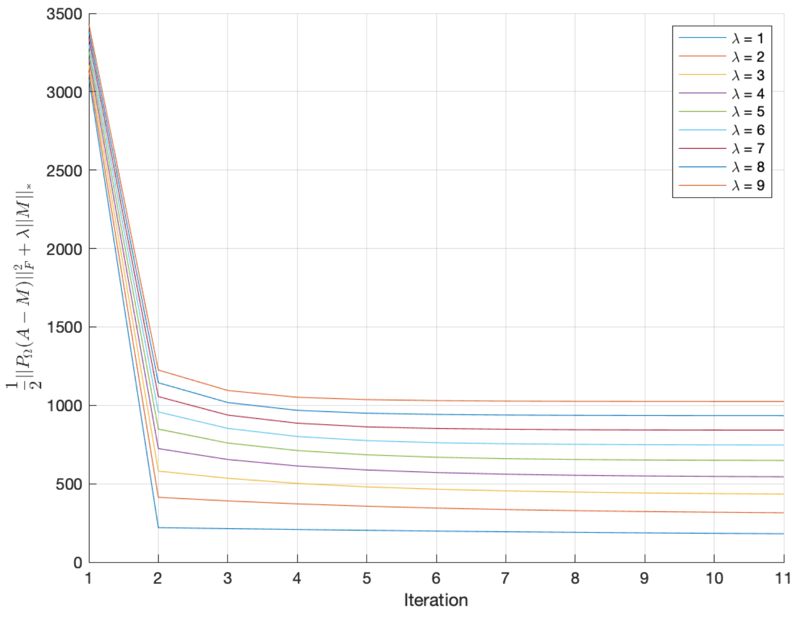

Figure 9: Loss function value $\frac{1}{2}||P_\Omega(A - M)||_F^2 + \lambda||M||_*$ across 10 iterations for $\lambda = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

Figure 9 shows the value of the loss function $\frac{1}{2}||P_\Omega(A - M)||_F^2 + \lambda||M||_*$ across 10 iterations for $\lambda = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. We were able to observe a trend similar to that in Figure 6: higher value of λ tend to increase the value the algorithm converges to in the end.

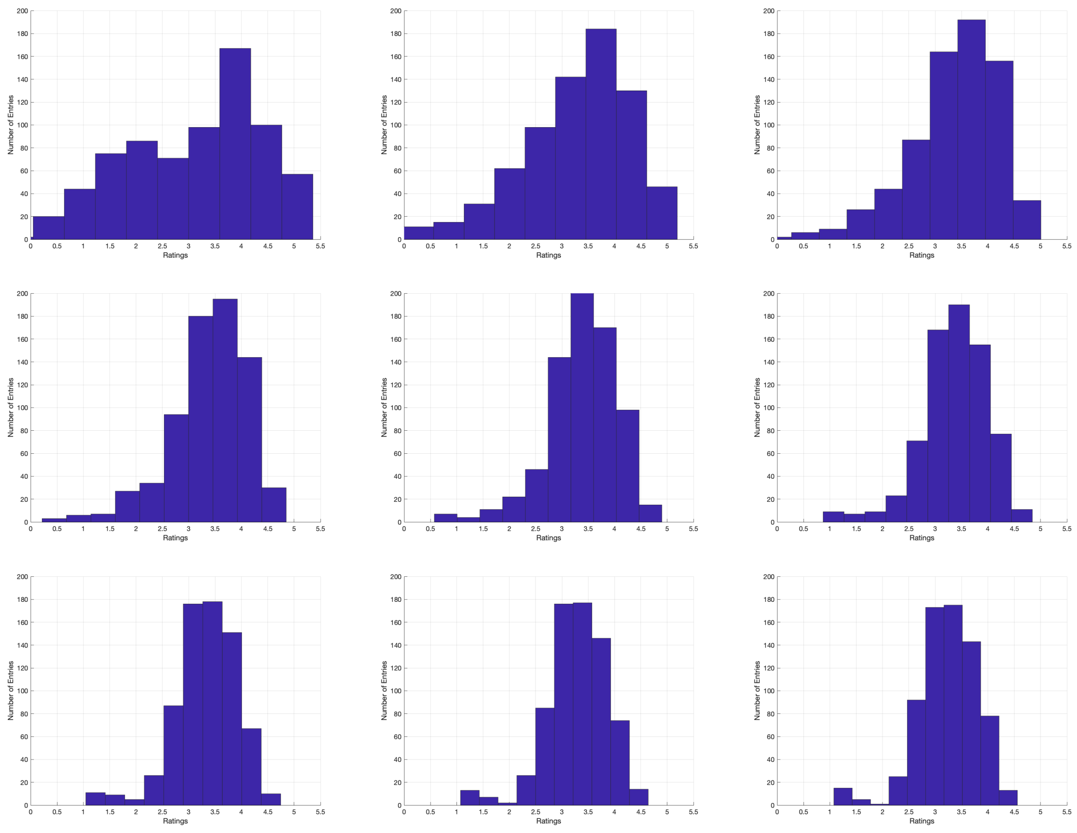

Figure 10: Distribution of ratings produced by M across 10 iterations for $\lambda = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. The plots for the 9 λ values are ordered left to right, row by row. The y-axis range is fixed for all plots.

We also examined the ratings distribution produced by M across 10 iterations for $\lambda = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ (Figure 10). Similar to Figure 7, all resemble a normal distribution, and we observe a similar trend of the peak shifting to the center while variance becomes lower as λ increases. $\lambda = 3$ seems to give the most reasonable result, as there are both 1 and 5 star ratings, with the peak skewed towards the high end, resembling the ratings distribution of the original matrix (Figure 8).

2.3 Comparison: Alternating Directions v.s. the Nuclear Norm Trick

Both low rank factorization via alternating directions (Section 2.1) and direct low rank approximation via the nuclear norm trick (Section 2.2) achieved desirable convergence behavior and similarly reasonable results on the `MovieRankings36` dataset. Using the nuclear norm trick tend to achieve better performance in terms of loss function value. I find the nuclear norm trick is easier to use, as one doesn't have to spend time tuning the best inner dimension k to use for approximating a low rank factorization; however, the need for computing SVD at each iteration seems to make the nuclear norm trick less efficient than alternating directions, which computes $n + m$ small least squares problem at each iteration.

3 Text Documents: CUR Factorization

In this section, we work with a collection of 139 documents on `US:Indiana:Evansville` and `US:Florida`. With `readdata.m` provided by Prof. Cameron, we obtain $M_{139\times 18446}$ bag-of-word matrix, where M_{ij} is the count of word j in the vocabulary in document i. For ground truth labels, we have $y_i = -1$ for Indiana, and $y_i = 1$ for Florida.

We implemented and experimented with CUR factorization. We varied the number of singular values $k = \{2, 3, 4, 5, 6, 7, 8, 9, 10\}$, and $c = r = ak$ for $a = \{1, 2, 3, 4, 5, 6, 7, 8\}$. (See `CUR.m` by Yuelin for implementation detail.)

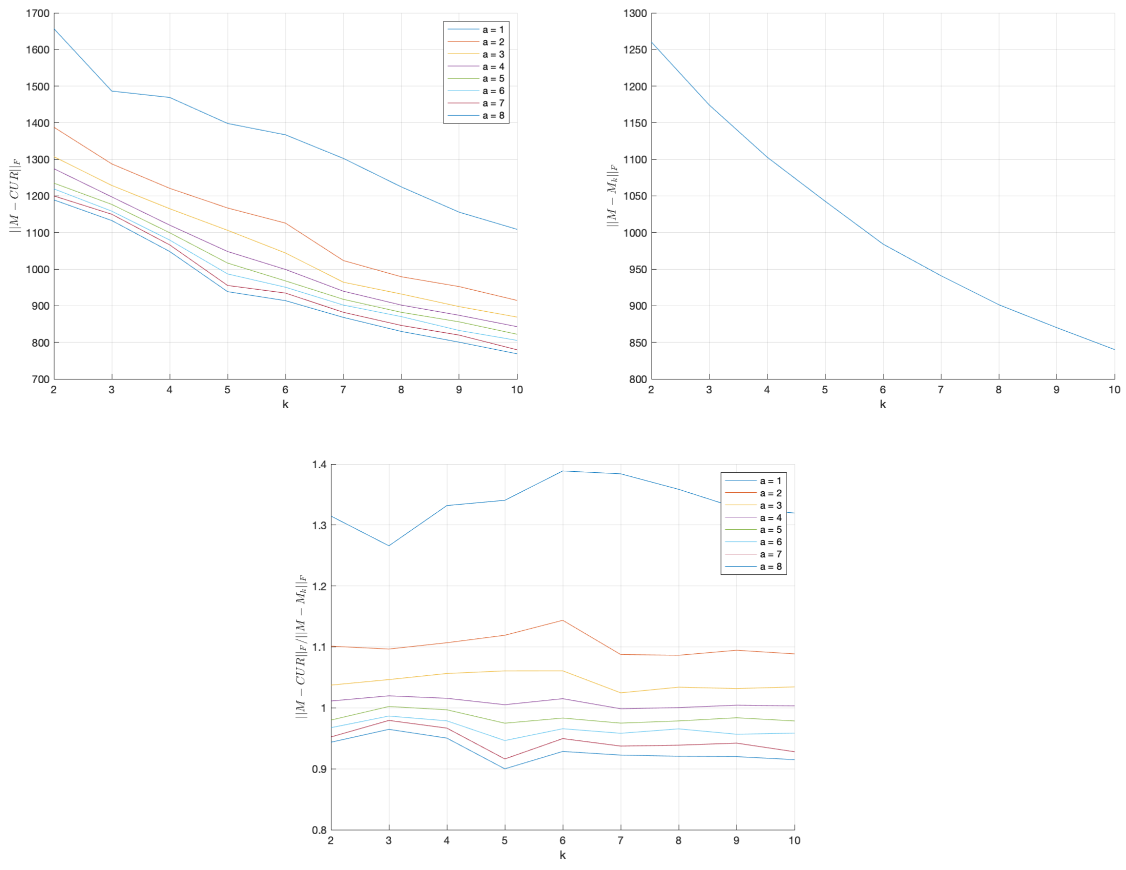

Figure 11: $\|M - CUR\|_F$ for $a = \{1, 2, 3, 4, 5, 6, 7, 8\}$ (top left), $\|M - M_k\|$ (top right), and $\|M - CUR\|_F / \|M - M_k\|_F$ for $a = \{1, 2, 3, 4, 5, 6, 7, 8\}$ (bottom), as we vary $k = \{2, 3, 4, 5, 6, 7, 8, 9, 10\}$.

Our results show that, Frobenius reconstruction errors decrease with higher value of k (Figure 11, top left, right), and larger value of a tend to lead to lower Frobenius error (Figure 11 top left). The ratio $\|M - CUR\|_F / \|M - M_k\|$, turns < 1 for $a \geq 5$ on this dataset, and is not constant for any a across values of k (Figure 11, bottom).

A reasonable choice for the parameters seem to be $a = 8, k = 10$.

4 Text Documents: Text Categorization

In this section, we aim to identify words that are effective in categorizing the documents into Indiana and Florida. We first perform preliminary categorization based on the counts of top leverage words in all the documents (Section 4.1), then develop a preprocessing scheme for the data based on common practices in natural language processing (Section 4.2), and lastly examine whether preprocessing improve separation in principle component projected space (Section 4.3).

4.1 Preliminary Text Categorization with Maximal Leverage Words

With feature labels provided in `features_idx.txt` and $k = 23$, we looked up the 23 words with maximum leverage scores in the dataset. They are: "gif", "shim", "the", "clearpixel", "bullet", "spacer", "limo", "and", "jpg", "florida", "del", "rentals", "812", "you", "country", "for", "radio", "church", "vanderburgh", "los", "miami", "evansville", and "uruguaysc".

We computed the SVD on matrix $M'_{139\times 23}$, a submatrix of $M_{139\times 18446}$, with the 23 columns being the top 23 words with maximal leverage. While these 23 words separate documents from Florida and Indiana reasonably well when projected onto their first three principle components (Figure 12, right), they still have a large degree of intermixing when projected onto the top two principle components (Figure 12, left). Can we achieve better separation through data preprocessing?

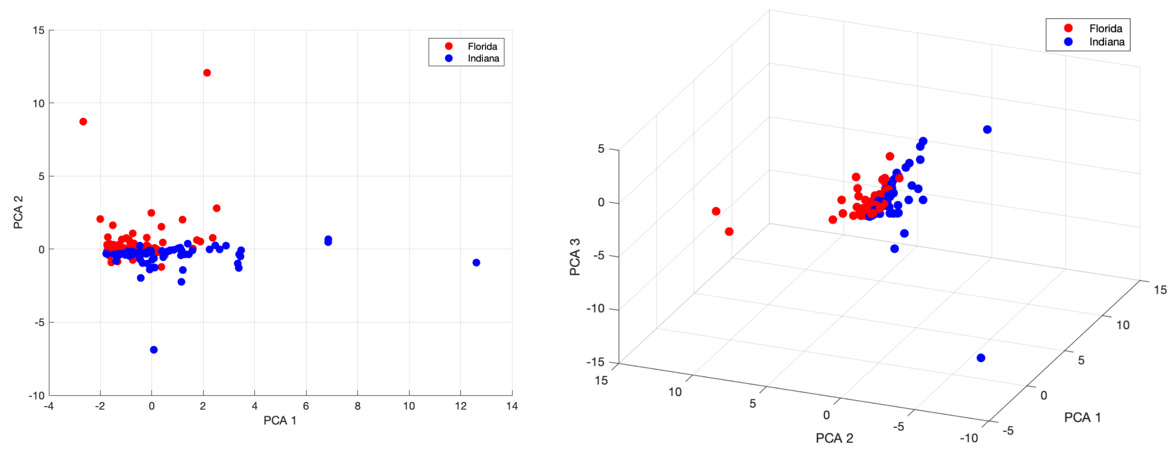

Figure 12: Projection of the document-word submatrix induced by the top $k = 23$ words of maximal leverage: "gif", "shim", "the", "clearpixel", "bullet", "spacer", "limo", "and", "jpg", "florida", "del", "rentals", "812", "you", "country", "for", "radio", "church", "vanderburgh", "los", "miami", "evansville", and "uruguaysc", onto the first and second (left), and first, second, and third (right) principle components.

4.2 Data Preprocessing for Document-Word Matrix

Looking at the top $k = 23$ by maximum leverage from Section 4.1, we see words that contain numerals (e.g. "812"), common stop words (e.g. "the", "and", "you","for"), and short words (e.g. "gif", "jpg", "shim"). Intuitively, common stop words will be as frequent in documents from both Florida and Indiana, and therefore will be unhelpful. Similarly, words containing numerals are likely noise from, for example, phone numbers and unicodes, and upon first glance there indeed were a few in the dataset. There has also been evidence that short words carry less information content [Piantadosi *et al.*, PNAS 2010].

Hence, with the above considerations, we create the following simple preprocessing scheme that heuristically filter words (columns) according to the following conditions:

1. Remove words containing numerals
2. Remove short words with length < 5
3. Remove common words occurring in $\geq 70\%$ of documents

Such filtering steps are quite common in natural language processing pipelines. It removed $\sim 5,000$ words from the dataset, resulting in a preprocessed 139×13544 document-word matrix, and we use this matrix in Section 4.3 for text categorization. (See `Preprocess.m` by Yuelin for implementation detail.)

4.3 Text Categorization Using Preprocessed Data

With the preprocessed data, we first examined its separation directly. With $k = 23$ and $c = 10,000$, we performed CUR factorization, which selected 1516 columns of the preprocessed matrix.

We projected the 139×1516 document-word submatrix onto its principle components (Figure 13). As we can see, with all 1516 words, there is not visible separation between documents from either states when projected onto its first two or three principle components.

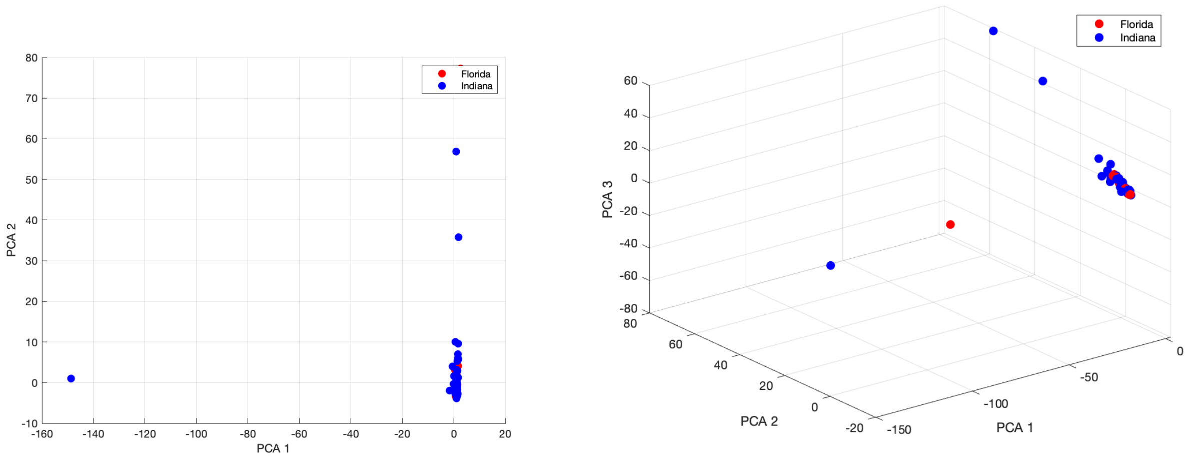

Figure 13: Projection of the 139×1516 document-word submatrix selected by CUR, with $k = 23, c = 10,000$, onto the first and second (left), and first, second, and third (right) principle components.

Then, with $k = 23$, we computed the leverage scores of the 1516 words, and selected 5 words with the maximal leverage scores. In this specific instance, the 5 words are "resources", "little", "unshaven", "notion", and "florida". With these 5 words, we form a 139×5 submatrix of the original, and projected it onto its first two and first three principle components (Figure 14).

Documents from Indiana and Florida are quite visually separable in the projections formed with the top 5 maximal leverage words, compared to their projections in Figure 13 using all 1516 words.

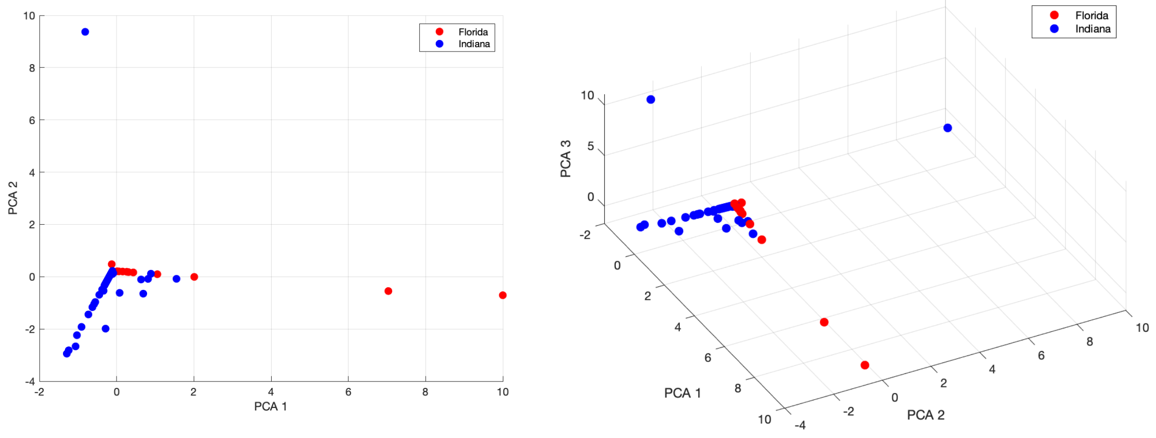

Figure 14: Projection of the 139×5 document-word submatrix induced by the top 5 maximal leverage words ($k = 23$) in the 139×1516 matrix resulted from CUR, onto the first and second (left), and first, second, and third (right) principle components. The top words are: "resources", "little", "unshaven", "notion", and "florida".

As CUR factorization is stochastic, we repeated the column selection process some more times with $k = 23, c = 10,000$. I noticed that, while the number of columns selected each time would vary, the word "florida" would always be one of the top 5 maximal leverage words computed from the submatrix formed by the selected columns, and the projection onto principle components using the top 5 words look very similar each time. This is most likely because the word "florida" is highly indicative of documents from Florida.

Out of curiosity, we wanted to see whether we can achieve better separation with more maximal leverage words. The lucky number seems to be 23. The separation seems to be cleaner in projections onto both the first two and first three principle components (Figure 15).

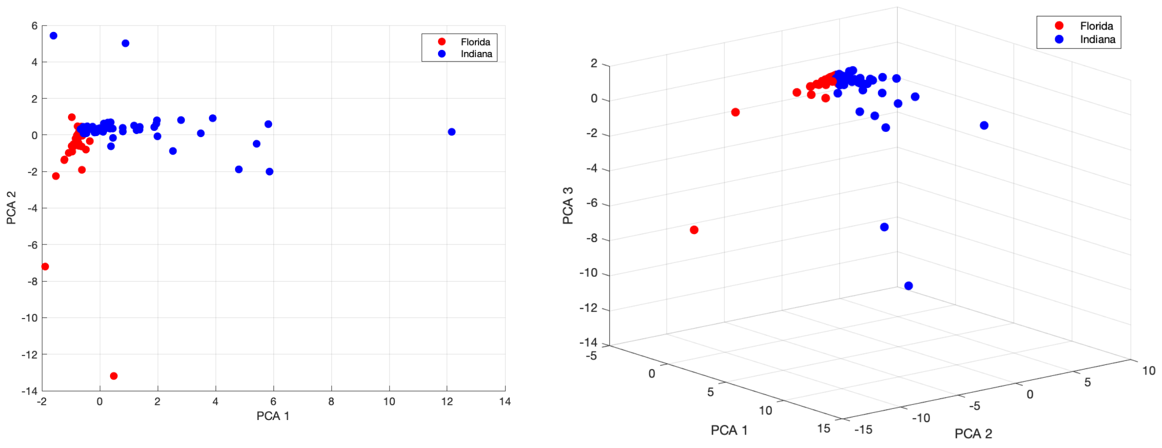

Figure 15: Projection of the 139×23 document-word submatrix induced by the top 23 maximal leverage words ($k = 23$) in the 139×1516 matrix resulted from CUR, onto the first and second (left), and first, second, and third (right) principle components. The top words are: "resources", "little", "unshaven", "notion", "florida", "organizations", "praise", "please", "course", "locations", "transactions", "friendships", "village", "composed", "ranch", "parkway", "confronta", "gables", "express", "twice", "school", "faith", and "concerns".