Interactions of Matrix Shape, Coherence, and Rank on Matrix Completion

Alexander Osypov
Alexander.Osypov@utdallas.edu

David Terry
David.Terry@utdallas.edu

May 16, 2021

Abstract

In this project, we explore matrix completion for recommender systems. Using synthetic data and real datasets such as Amazon, Netflix, MovieLens, we draw the relations among matrix coherence, rank, and regularization models in order to understand matrix recovery. We observe that the most-squared matrix tends to give the highest accuracy for matrix completion via nuclear norm minimization, which can be justified by matrix coherence. Most matrix completion techniques rely on singular value decomposition (SVD). We find that an approximated scheme of SVD is not only faster but also better than the standard SVD. Lastly, we discover that the TL1 model achieves the best matrix recovery results among some existing approaches such as nuclear norm and L_p Schatten norm.

Keywords— Matrix coherence; SVD; rank; regularization models; nuclear norm minimization; Schatten norm

Contents

1 Introduction

Matrix completion refers to recovering a data matrix from partial samples of its entries. For example, in a recommender rating system such as Netflix, one seeks to predict the "rating" or "preference" a user would give to an item, based on a user's past behaviors and similar decisions made by other users. Since users often have similar preferences that produce linearly dependent ratings, the data matrix shall be low rank. Since directly minimizing the rank is NP-hard, one popular method minimizes the sum of singular values, referred to as the nuclear norm [1]. Some nonconvex models are listed as follows,

- L_p [2, 13, 14] for $0 < p \leq 1$: $\|\mathbf{x}\|_p^p = \sum_j |x_j|^p$;
- Transformed L_1 (TL1) [7, 18, 19]: $\sum_j \dfrac{(a+1)|x_j|}{a+|x_j|}$;
- Smoothly clipped absolute deviation (SCAD) [3]: $\sum_j \Phi_{\lambda,\gamma}^{\mathrm{SCAD}}(x_j)$ with

$$\Phi_{\lambda,\gamma}^{\mathrm{SCAD}}(x_j) = \begin{cases} \lambda|x_j|, & |x_j| \leq \lambda; \\ \dfrac{2\gamma\lambda|x_j| - |x_j|^2 - \lambda^2}{2(\gamma-1)}, & \lambda < |x_j| \leq \gamma\lambda; \\ \dfrac{(\gamma+1)\lambda^2}{2}, & |x_j| > \gamma\lambda; \end{cases}$$

- Minimax concave penalty (MCP) [17]: $\sum_j \Phi_{\lambda,\gamma}^{\mathrm{MCP}}(x_j)$ with

$$\Phi_{\lambda,\gamma}^{\mathrm{MCP}}(x_j) = \begin{cases} \lambda|x_j| - \dfrac{|x_j|^2}{2\gamma}, & |x_j| \leq \gamma\lambda; \\ \dfrac{1}{2}\gamma\lambda^2, & |x_j| > \gamma\lambda; \end{cases}$$

- L_1-L_2 [15, 6, 12]: $\|\mathbf{x}\|_1 - \|\mathbf{x}\|_2 = \sum_j |x_j| - \sqrt{\sum_j x_j^2}$.

We rely on a fixed-point continuation (FPC) algorithm [8] to examine the influence of matrix shape, coherence, rank, and various regularization functionals on the performance of matrix completion. As each iteration of FPC involves SVD, we explore an approximated SVD to combine with FPC, labelled as FPCA, to speed up the computational efficiency. As the matrix coherence [1] is often used to determine to what degree it is possible to complete a matrix, we will further investigate whether coherence has any influence over the FPC/A algorithms. In summary, we aim at the following three research questions:

1. Under the same sampling ratio, what type of matrix can be recovered with higher probability?
2. Which algorithm is better, FPC or FPCA?
3. Which model is the best for low-rank matrix completion?

2 Matrix Coherence

We present the definition of matrix coherence [1] in Definition 1.

Definition 1. *Let $U \in \mathbb{R}^{n\times r}$ contain orthonormal columns with $r < n$ and define $\mathbf{P}_U = UU^\top$ as its associated orthogonal projection matrix. Then the **coherence** of* $\mathbf{U}$ *is:*

$$\mu(U) = \frac{n}{r} \max_{1\le i\le n} \|\mathbf{P}_U \mathbf{e}_i\|^2 = \frac{n}{r} \max_{1\le i\le n} \|U_i\|^2. \tag{1}$$

The algorithm to compute the coherence of an orthonormal matrix is summarized in Algorithm 1

Algorithm 1 Matrix Coherence of an orthogonal matrix

Input: an $n \times r$ orthonormal matrix U
Set $M = 1$
for $i = 1 : n$ **do**
    $d = \mathrm{norm}(U(i,:))^2$
    if $d > M$ **then**
        $M = d$
return Coherence $= n(M/r)$;

The exact recovery of a low-rank matrix is characterized in Theorem 2 with the following two assumptions,

- A0 The coherence obeys $\max(\mu(U), \mu(V)) \le \mu_0$ for $\mu_0 > 0$;
- A1 The $m\times n$ matrix $\Sigma_{1\le k\le r} \mathbf{u}_k \mathbf{v}_k^*$ has a maximum magnitude bounded by $\mu_1\sqrt{r/(mn)}$ for $\mu_1 > 0$.

Theorem 2. *Let X be an $m \times n$ matrix of rank r obeying **A0** and **A1** and put $N = \max(m, n)$. Suppose we observe M entries of X with locations sampled uniformly at random. Then there exist constants C, c such that if*

$$M \le C \max(\mu_1^2, \sqrt{\mu_0}\mu_1, \mu_0 N^{1/4}) Nr(\beta \log N) \tag{2}$$

for some $\beta > 2$, then the solution to the nuclear minimization is unique and equal to X with probability at least $1 - cN^{-\beta}$.

This Theorem asserts that if the coherence is low, few samples are required to recover X, and a large value indicates that the recovery of the matrix is unlikely.

There are a couple of things that we have to consider. First is the computational time of coherence. Since our data matrix is quite large, computing the matrix coherence in the normal fashion might be quite inefficient, considering that we first need to find SVD and then get the max norm. We can use the approximation of matrix coherence, which after 50 simulations of 10,000 matrices gives an average of around 3.5%. Since the max is not given a bound, the complexity of the initial algorithm is $O(n)$, with n representing the columns in the sparse matrix. The main algorithm can be improved by using the bound of the coherence, which will be using the max of 1, since the largest possible value is n/r from [1]. Thus with the upper bound, we will have a best case performance of $O(1)$ and the worst case performance of $O(nl^2)$, due to the computation of SVD, giving us an average case performance of $O(\frac{nl^2}{2})$, which is a significant reduction from $O(4nl^2)$ using the initial calculation of matrix coherence.

3 Low-rank Recovery Algorithms

We consider an algorithm for minimizing the nuclear norm, called fixed point continuation (FPC) [8], which is summarized in Algorithm 2.

Algorithm 2 Fixed Point Continuation (FPC)

```
Input: a sampling operator A that contains ones (zeros) where data are recorded (missing) and recorded data b.
Select μ_1 > μ_2 > ... > μ_L = μ̄ > 0 and τ > 0.
Initialize X by zero filling.
for μ = μ_1, μ_2, ..., μ_L do
    while NOT converged, do
        Compute Y = X − τA ∗ (A(X) − b)
        Compute SVD of Y, i.e., Y = UΣV^⊤
        Compute X = U diag[shrink(σ, τμ)]V^⊤
```

Note that the shrink operator in the last step is applied to every singular value, which is defined by

$$\text{shrink}(\sigma, \rho) = \max(\sigma - \rho, 0). \tag{3}$$

Since there are significant computation costs in calculating the SVD in Algorithm 2, we consider an approximated SVD through Algorithm 3. The combination of FPC and the approximated SVD is referred to as FPCA.

Algorithm 3 Linear Time Approximate SVD [8]

```
Input: A ∈ R^{m×n}, c_s, k_s ∈ Z^+ s.t. 1 ≤ k_s ≤ c_s ≤ n, {p_i}_{i=1}^n s.t. p_i ≥ 0, Σ_{i=1}^n p_i = 1.
Output: H_k
for t = 1 : c_s, do
    Pick i_t ∈ 1, ..., n with P_r[i_t = α] = p_α, α = 1, ..., n.
    Set C^(t) = A^(i_t) / sqrt(c_s p_{i_t})
Compute C^⊤C and its SVD; say C^⊤C = Σ_{t=1}^{c_s} σ_t^2(C) y^t y^t⊤.
Compute h^t = C y^t / σ_t(C) for t = 1, ..., k_s.
return H_{k_s}, where H_{k_s}^(t) = h^t, and σ_t(C), t = 1, ..., k_s.
```

4 Data Summary

4.1 Synthetic data

We artificially generate a low-rank matrix by multiplying two randomly generated matrices together of dimensions $m \times r$ and $r \times n$, respectively, with $r \ll \min(m, n)$. The resulting ground-truth matrix X_g is $m \times n$, and has a rank that is at most r.

Following the works of [4, 6, 16], we consider an over-sampled discrete cosine transform (DCT), defined as $A = [\mathbf{a}_1, \ldots, \mathbf{a}_n] \in \mathbb{R}^{m\times n}$ with

$$\mathbf{a}_j := \frac{1}{\sqrt{m}} \cos\left(\frac{2\pi j \mathbf{w}}{F}\right), \quad j = 1, \cdots, n, \tag{4}$$

where $\mathbf{w}$ is a random vector uniformly distributed in $[0,1]^m$ and $F \in \mathbb{R}$ is a positive parameter. Again we multiply two over-sampled DCT matrices of size $m \times r$ and $r \times n$ together to generate the low-rank ground-truth matrix.

4.2 Real data

We use two types of amazon reviews (Books and Movies) [9] and the popular MovieLens data set [5] for testing. The Amazon Book data set has 22,507,115 ratings, the Amazon Movie data set has 4,607,047 ratings, while the MovieLens data set has 25,000,095 ratings. The format of all data sets consists of three columns including a user or rater ID, a product or movie ID, and the rating on a scale of one to five. We can form a matrix with the structure of unique users as rows, unique product as columns, and the corresponding ratings as entry values. All data sets (Books, Movies, and MovieLens) originally have 1000 rows and 1000 columns made from the 1000 user and product ID's associated with the most reviews.

Furthermore, we progressively eliminate rows and columns in order to make the data set more dense so that we can compute the performance quantitatively. For this purpose we run a loop eliminating columns (products) without a certain number of entries (reviews). Then we run a similar loop of the rows (users). This step is repeated, gradually increasing the number of required entries, until a matrix of acceptable density is returned, or until no matrix is returned in which case the previous iteration is used. The result is summarized in the below table.

In order to fully populate the ground-truth low-rank matrices to be completed, we first use the MATLAB's command `randi` to fill in all the zero entries with a random number from 1 to 5. Then we apply the Singular Value Decomposition (SVD), thus getting two unitary matrices U, V and a diagonal matrix S with singular values. We then choose to retain only the first 5, 10, or 20 singular values. By multiplying the truncated diagonal matrix with U and V to the left and the right, respectively, we obtain a ground-truth matrix with controllable rank for testing.

Table 1: Real World Data

Data Set	**Dimensions**	**Density in percentage**
MovieLens	1000 × 1000	34.80%
MovieLens	500 × 500	47.70%
MovieLens	308 × 366	80.14%
MovieLens	68 × 247	93.23%
Amazon Books	1000 × 1000	1.89%
Amazon Books	500 × 500	2.75%
Amazon Books	14 × 45	48.57%
Amazon Movie	1000 × 1000	3.71%
Amazon Movie	500 × 500	3.71%
Amazon Movie	39 × 91	41.90%

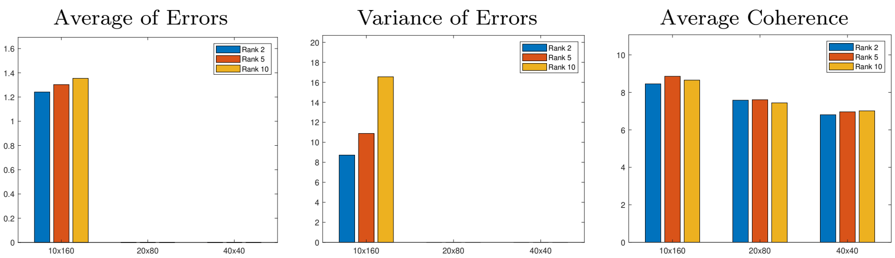

Figure 1: The influence of shape and rank on FPCA completion accuracy: the most-square ground-truth matrices have the highest recovery accuracy with the smallest coherence value.

5 Desirable Properties for Matrix Completion

We investigate three matrix shapes: $10 \times 160, 20 \times 80$, and 40×40 with three levels of rank by setting $r = 2, 5$, and 10. After randomly sampling 50% of the entries, we perform matrix completion by FPCA, thus getting a recovered matrix, denoted by X_{comp}. We define the *percentage error* of X_g and X_{comp} by,

$$E(X_g, X_{comp}) = \frac{\|X_{comp} - X_g\|_F}{\|X_g\|_F} \times 100\%.$$

We repeat each combination of matrix shape and rank over 100 trials, and plot the mean percentage errors, the variance of said errors, and the coherence of X_g in Figure 1.

For any given set of dimensions, the error is generally smaller for a smaller prescribed rank. Additionally, the error also decreases as the matrix's dimensions become more square, i.e., the ratio of m and n is closer to 1 (though this is less obvious for 20×80 and 40×40 matrices given how small the errors of these matrices already are). Therefore, we conclude that FPCA tends to prefer low-rank, square matrices. On one hand, a lower rank matrix means more linear dependencies and more correlation among the rows/columns, thus making it easier to fill in missing entries. On the other hand, a more square matrix has larger rank than those more rectangular matrices so that low-rank regularization takes effect. In other words, a rectangular matrix is of low-rank, and hence there is no need for low-rank regularization.

Additionally in Figure 1, we plot the average coherence of generated ground truth matrices, showing that lower coherence often corresponds to better results. A lower coherence (as defined in Section 2) conveys that a matrix is easier to recover (as opposed to a higher coherence), and indeed, one may observe, on average, a reduced coherence for desirable properties like low rank and dimension ratios close to or equal to 1. Therefore, we conclude that matrix completion via nuclear norm minimization tends to give best results when a matrix is of lowest rank with the most square shape. It is consistent in the tensor completion [11].

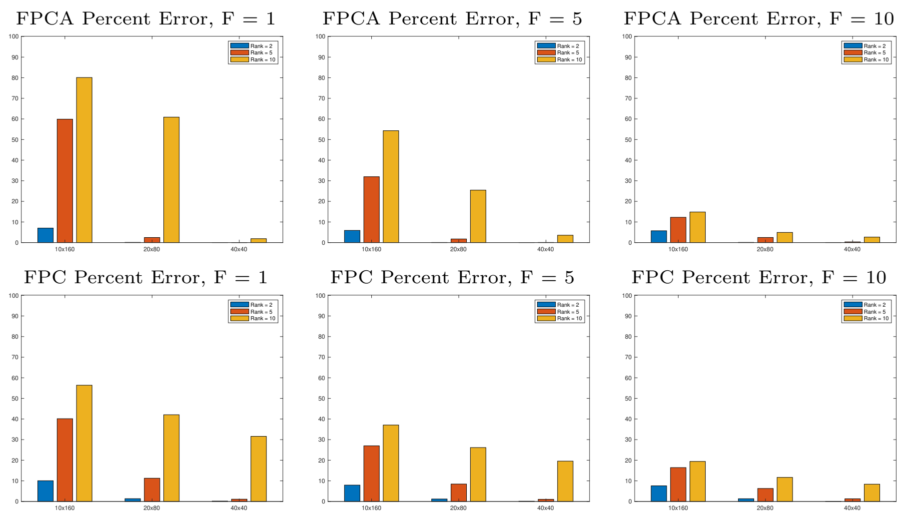

Figure 2: The comparison between FPC and FPCA of over-sampled DCT matrices across varying shapes, ranks, and F values.

6 Recovery Algorithms

We compare FPC and FPCA for a sampling ratio of 0.5, relying on these algorithms to recover the missing values of an over-sampled DCT matrix. We vary the shapes, ranks, and F values for such matrices and present the average percentage errors over 100 random realizations in Figure 2. Generally speaking, we observe that FPCA tends to better than FPC in terms of percentage error. Recall from Section 5 that FPCA matrix completion prefers low-rank, and square matrices. Figure 2 implies that this trend holds for FPC and DCT-type of matrices. However, we observe in Figure 2 that FPC with an exact calculation of SVD is more desirable than an approximated scheme (FPCA) for high-rank, non-square matrices.

In summary, we conclude that while FPCA completion is *generally* preferable to FPC completion, for sampled matrices of higher rank and more rectangular shape, FPC completion is preferable. We repeat this experiment across F values of 1, 5, and 10, and the average percentage errors are shown in Figure 2.

7 Model Selection

Given a function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$, the proximal operator [10] $\mathbf{prox}_\mu^f : \mathbb{R}^n \to \mathbb{R}^n$ of f with a parameter $\mu > 0$ is defined by

$$\mathbf{prox}_\mu^f(\mathbf{v}) = \arg\min_{\mathbf{x}} \left(\mu f(\mathbf{x}) + \frac{1}{2}\|\mathbf{x} - \mathbf{v}\|_2^2 \right). \tag{5}$$

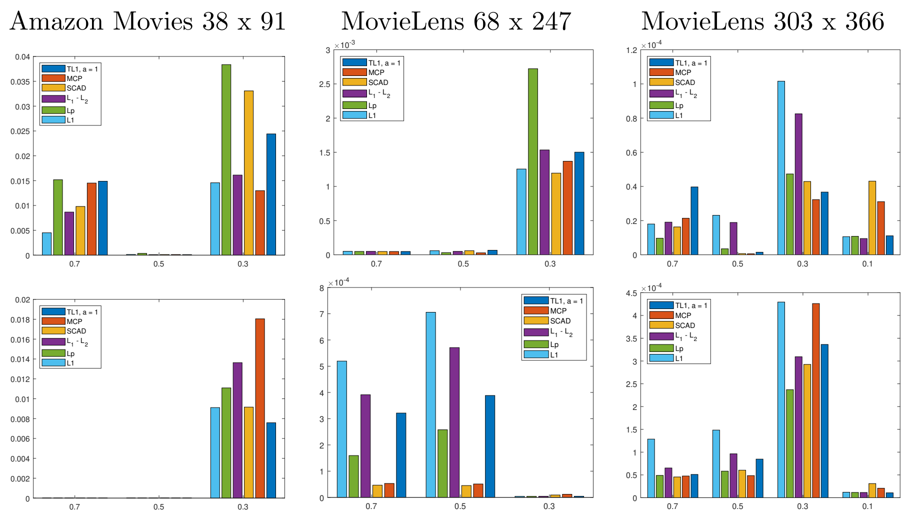

Figure 3: Comparison among various regularizations for matrix completion with ground-truth matrix rank of 5 (top) and 10 (bottom) in terms of relative errors.

If f is the L_1 norm, then the corresponding proximal operator is the *soft shrinkage* operator, defined by

$$\mathrm{shrink}_\mu(\mathbf{v}) = \begin{cases} \mathbf{v} - \mu, & \mathbf{v} > \mu, \\ \mathbf{0}, & |\mathbf{v}| \le \mu, \\ \mathbf{v} + \mu, & \mathbf{v} < -\mu, \end{cases}$$

which is equivalent to (3).

We can minimize any aforementioned non-convex regularizations discussed in the Introduction Section by replacing the shrinkage operator (3) by its own proximal operator. By doing so, we can examine these nonconvex regularizations for low-rank matrix completion. Take TL1 for an example. Its proximal operator [18] is expressed as

$$\mathbf{prox}_\mu^{\mathrm{TL1}}(\mathbf{v}) = \begin{cases} \mathrm{sign}(\mathbf{v})\left[\frac{2}{3}(a + |\mathbf{v}|)\cos(\frac{\varphi(\mathbf{v})}{3}) - \frac{2a}{3} + \frac{|\mathbf{v}|}{3}\right], & \mathbf{v} > \mu, \\ 0, & |\mathbf{v}| \le \mu, \end{cases}$$

where $\varphi(\mathbf{v}) = \arccos\left(1 - \frac{27\mu a(a+1)}{2(a+|\mathbf{v}|^3)}\right)$.

To guarantee the low-rankness of real data, we manually process the data matrix by keeping the largest r singular values for $r = 5, 10$, and 20.

We consider the following real data matrices: Amazon Movies of size 38×91 as well as MovieLens of sizes 68×247, 303×366, and 500×500. We generate pseudo ground-truth low-rank matr ices by fully populating each matrix and adjusting ranks to be 5, 10, and 20. We then compare the realtive error of the aforementioned sparse promoting regularizations, i.e., L_1, L_p, TL1, SCAD and MCP, within the FPCA framework for matrix completion under the sampling ratios of 0.7, 0.5, 0.3, and 0.1. As shown in Figure 3, all the methods perform very well (e.g., returning less than 2.5% relative error) when rank is 5 or 10 and sampling ratio is 30% or higher, with one exception

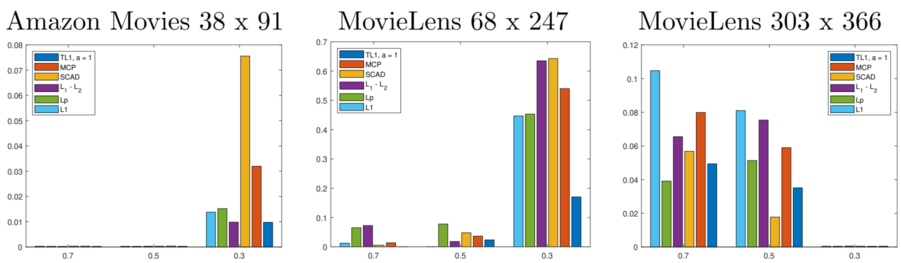

Figure 4: Comparison among various regularizations for matrix completion with ground-truth matrix rank of 20 in terms of relative errors.

in the 38×91 MovieLens. Even in this case (the top/left plot in Figure 3), we can see that the relative errors are all lower than 4%, especially with L_1, L_1-L_2, and MCP returning relative errors less than 2%. In addition, Figure 3 shows that all the regularization models yield relative errors less than 1% with sampling ratio over 50%, exception for 38×91 Amazon Movies with rank 5 and 500×500 MovieLens with rank 10.

At the higher rank of 20, we begin to see differences among all the competing methods. As shown in Figure 4, the TL1 model outperforms the other methods, when the underlying matrices are skinny or tall such as the 38×91 Amazon Movies and the 68×247 MovieLens. While TL1 also performs well in the 303×366 MovieLens, all the nonconvex regularizations perform better than the convex nuclear norm. Furthermore, we observe that SCAD returns the lowest relative error in this case when the selection ratio is 50% in the almost square, 303×366, MovieLens matrix, but SCAD also return the highest relative error in the skinny, 68×247 MovieLens and 38×91 Amazon Movies matrices (Figure 4).

Finally we examine the perfectly square MovieLens matrix with dimensions 500 by 500 in Figure 5. We see all the methods return relative errors less than .1% at rank 5. We also see inconsistency at rank 10 in the sense that the winner is different at different sampling ratios. By comparing Figure 3 and Figure 5, we observe that all the methods for the 500×500 matrix return much higher errors than for the 303×366 MovieLens. At rank 20 we see that TL1, SCAD, and L_p tend to give the best results with relative errors less than .1% for sampling ratios grater than or equal to 30%. L_1, L_1-L_2, and MCP all perform poorly at rank equal to 20.

8 Conclusions

Matrix completion prefers matrices that are low rank and square. This is justified by a decreased coherence for matrices with these desirable properties. FPCA *generally* achieves better recovery than FPC. But for matrices with undesirable properties, FPC with an exact SVD is better than FPCA. TL1 outperforms the other methods when the underlying matrices are skinny or tall, while TL1, SCAD, and L_p all perform good at a higher rank.

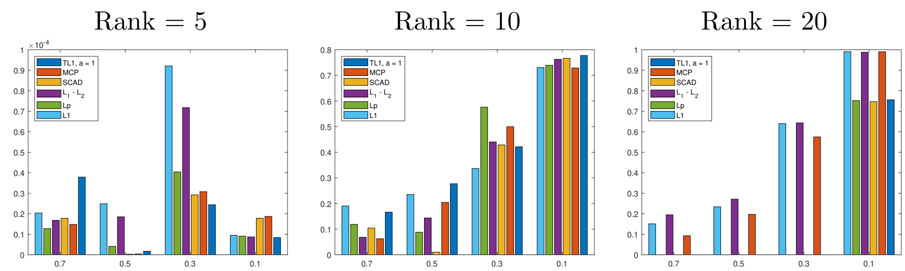

Figure 5: Comparison among various regularizations for matrix completion of 500×500 MovieLens with prescribed rank of 5, 10, and 20.

9 Individual Assessments, Issues, and Lessons Learnt

9.1 David

Early on focus was united in understanding the algorithms for SVD, Linear Time SVD, and FPCA and collecting or creating data to test our algorithms on. Once an academic basis was formed I took on the task of converting data sets into a usable format. I believe I could have sped up this process by casting a wider net for sources earlier on. I spent a fair amount of time at the start of the semester designing an effective way to read in tables and generate matrices of use. Once I had a working algorithms I should have abandoned my original data sets as they were unhelpfully sparse. I eventually moved from using the various Amazon review sets to the MovieLens data frames with much success.

Once there were real world data options to test out algorithms I focused on assessing the effectiveness of FPC in comparison to FPCA. Initial trials were counter intuitive, but once certain regularization steps were taken we confirmed our findings were in line with other sources in the academic community. Adam Shaker took on the challenge of defining under what matrix conditions FPCA was indeed a better algorithm, while I took on researching what regularization methods could help improve FPCA.

Another area where I could have adjusted my approach earlier is in utilizing the resource of the on campus remote desktops. As I began to test the various non convex regularization methods I realized the limitation of my person hardware and wasted many hours attempting to run code that my system could not handle. While our algorithms still took many hours to return useful data, once I switched to utilizing the on campus desktops the research process was greatly sped up.

While I also believe I could have aimed to optimize the code that I wrote, I believe I was successful in providing quality research to this project. From the beginning of the project I was able to utilize my prior knowledge of MatLab to be an aid to the rest of my team and begin processing our data. I have also learned the details of matrix completion in depth enough to study and improve aspects of our tested algorithms, and believe our entire team worked equally and efficiently in the collection of our research into this paper and our team poster.

9.2 Alex

My personal objective for this project was to increase my understanding of mathematics and data science through a real-world application. For myself, my belief is that while I am quite strong in data science, I still need a lot of work on mathematics, which is why I along with my team, chose to work on this project.

We were given several tasks to do, among them was to understand the norms of the matrix and work on minimization problems. This was quite challenging, but everyone on the team was able to complete this assessment. The next part was to read many research papers about our topic, which contained a lot of important information that we used.

After we split up into two teams, Adam Shaker and David Terry worked on the FPC/A algorithms (which I studied as well in order to understand them later on), and I worked on implementing matrix coherence in MATLAB. I never used MATLAB before, so certain syntax did not make sense, and I had to teach parts of it to myself. I eventually was able to implement matrix coherence, and also create an approximating algorithm for matrix coherence, but left it out of the report for clarity.

At one point, I went down the rabbit hole to understand a certain method called Nyström, which I thought was a faster way of calculating matrix coherence, but it was another matrix completion algorithm. For time's sake, we decided to continue with FPC/A algorithms since we were the most familiar with them and already had their implementations in code. Future work may include analysis of this algorithm against FPC/A.

I think at the end, I was able to achieve my original objectives by applying the skills I learned during my time at UTD in order to analyze and adjust algorithms, as well as learn to read highly technical papers to apply their concepts in our work.

I have to give a lot of credit to Adam Shaker and David Terry. Adam is a rising sophomore, but was able to put in a lot of effort in working on the trials for FPC/A, and did his best to understand advanced statistics and linear algebra concepts despite not taking those classes yet. I've never worked with David either, but I was impressed at how much knowledge he has acquired beyond the normal academic scope to achieve our results. I am also very thankful to our advisor and sponsor, Yifei Lou, who constantly met with us to check our progress, give us valuable advice, and encourage us to pursue our interests in this project.

Overall, I am very happy with the output created by my team, and even though we are leaving UTD, I know Adam will continue this work further, and I know that he will do an excellent job.

10 Future Work

10.1 Parameter Honing

When testing the non-convex regularization methods, assumption were made about the parameters. Namely, these parameters are the parameter $a = 1$ as used by the TL1 and L_1-L_2 algorithms, and the parameter $\gamma = 3$ in the MCP and SCAD algorithms. These parameters were chosen as a base line to demonstrate the efficacy of the regularization methods in comparison with the L_1 norm. Given the time to asses more data on the specific characteristics of these parameters we could possibly see better or at least more consistent results for our non-convex methods.

10.2 Efficiency

While not talked about in this paper some algorithms were found to run faster than other under certain conditions. However, regardless of if we used FPCA or FPC, when it came to larger matrices (500×500) our algorithms returned much slower results. While in a single run our results took only minutes, computing 100 tests runs for verification of results could sometimes take hours. This is due to either the complexity of our algorithms or a lack of efficiency in our code. If the algorithms are at fault for the time draw then it is on the user to decided if the accuracy of our matrix completion scenarios warrants the time spent running it. However, I would suspect efficiencies could be made design of our MatLab code for these algorithms. This could prove quite beneficial if the result was greater scalability.

References

[1] E. J. Candèsl and B. Recht. Exact matrix completion via convex optimization. *Found. Comput. Math*, 9:717–772, 2012.

[2] Rick Chartrand. Exact reconstruction of sparse signals via nonconvex minimization. *IEEE Signal Process Lett.*, 14(10):707–710, 2007.

[3] J. Fan and R. Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *J. Am. Stat. Assoc.*, 96(456):1348–1360, 2001.

[4] Albert Fannjiang and Wenjing Liao. Coherence pattern–guided compressive sensing with unresolved grids. *SIAM J. Imag. Sci.*, 5(1):179–202, 2012.

[5] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. 5(4), December 2015.

[6] Yifei Lou, Penghang Yin, Qi He, and Jack Xin. Computing sparse representation in a highly coherent dictionary based on difference of L_1 and L_2. *J. Sci. Comput.*, 64(1):178–196, 2015.

[7] J. Lv and Y. Fan. A unified approach to model selection and sparse recovery using regularized least squares. *Ann. Stat.*, 37(6A):3498–3528, 2009.

[8] S. Ma, D. Goldfarb, and L. Chen. Fixed point and bregman iterative methods for matrix rank minimization. *Math. Program*, 128(1-2):321–353, 2011.

[9] Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. pages 188–197, November 2019.

[10] Neal Parikh, Stephen Boyd, et al. Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3):127–239, 2014.

[11] Jonathan Popa, Susan E Minkoff, and Yifei Lou. An improved seismic data completion algorithm using low-rank tensor optimization: Cost reduction and optimal data orientation. *Geophysics*, 86(3):1–88, 2021.

[12] Jing Qin and Yifei Lou. l_{1-2} regularized logistic regression. In *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, pages 779–783. IEEE, 2019.

[13] Z. Xu, X. Chang, F. Xu, and H. Zhang. $l_{1/2}$ regularization: A thresholding representation theory and a fast solver. *IEEE Trans. Neural Netw. Learn. Syst.*, 23:1013–1027, 2012.

[14] Z. Xu, Guo H., W. Yao, and H. Zhang. Representative of $l_{1/2}$ regularization among l_q $(0 < q < 1)$ regularizations: an experimental study based on phase diagram. *Acta Automatica Sinica*, 38(7):1225–1228, 2012.

[15] Penghang Yin, Ernie Esser, and Jack Xin. Ratio and difference of l_1 and l_2 norms and sparse representation with coherent dictionaries. *Comm. Inf. Syst.*, 14(2):87–109, 2014.

[16] Penghang Yin, Yifei Lou, Qi He, and Jack Xin. Minimization of ℓ_{1-2} for compressed sensing. *SIAM J. Sci. Comput.*, 37(1):A536–A563, 2015.

[17] C. Zhang. Nearly unbiased variable selection under minimax concave penalty. *Ann. Stat.*, pages 894–942, 2010.

[18] Shuai Zhang and Jack Xin. Minimization of transformed L_1 penalty: Closed form representation and iterative thresholding algorithms. *Comm. Math. Sci.*, 15:511–537, 2017.

[19] Shuai Zhang and Jack Xin. Minimization of transformed L_1 penalty: theory, difference of convex function algorithm, and robust application in compressed sensing. *Math. Program.*, 169(1):307–336, 2018.

A Print Name, Signatures, Date