STAT 202B Final Project: The Matrix Completion Problem

Ryan Anderson
raanderson@g.ucla.edu, UID: 306076860

March 24, 2023

Abstract

Major progress on the matrix completion problem was achieved via incorporating approaches from statistics and convex optimization. We describe the surge of interest in this problem and how that led to rapid advances in both theory and implementation.

1 Introduction

Consider a situation in which we have a sparse matrix, usually broad and sparse so that $A \in \mathbb{R}^{mxn}$ has $n > m$, but not necessarily. Our goal is to estimate non-zero values to "fill" the matrix.

This problem arose naturally in the course of constructing "recommender engines" for products like Netflix, in which very many users are to be recommended movies based on a very limited history of movies watched and rated. For this reason, most discussions of the matrix competition problem make reference to the "Netflix problem," although it could also have just as easily been the "Yelp problem" or "Spotify problem".

The naming may be more apposite than seems at first glance – coincident with the growth of Netflix and similar services there appeared the first paper to stoke major interest in novel solutions to the problem, Candes and Recht (2008)

Theoretical and algorithmic progress on the matrix problem has progressed rapidly, as will be discussed further below.

2 Literature Review & Theoretical Notes

Candes and Recht (2008) kickstarted contemporary interest in finding solutions to the matrix completion problem. Much in that paper was developed by analogy with concepts from electrical engineering and signal processing, where Candes was active in the literature before. The initial situation in that paper is motivated by the idea of only having a sample of a larger matrix available, and poses the question of under which circumstances the full matrix can be recovered.

Central to the matrix completion problem is the idea that the matrices involved are low-rank – in Netflix terms, this is the idea that user ratings can be determined by variables like genre, cast, year of release. The low rank assumption reduces the degrees of freedom in the matrix from O(mn) to (m+n-r)r. With this in mind, the matrix completion problem is simple to state as an optimization problem: given an incomplete matrix $A \in \mathbb{R}^{mxn}$, we seek a matrix in $\mathbb{R}^{mxn}$ with minimal rank that agrees on non-zero values of A.

Candes and Recht (2008) note that rank minimization is NP-hard, and so instead cast the problem in terms of minimizing the sum of the singular values of a matrix which agrees on the non-zero values of A. These are equivalent problems since the sum of the singular values of a matrix is always equivalent to its norm; because the sum of the singular values is also known as the nuclear norm of a matrix, this is referred to as nuclear norm optimization. This problem, which is convex as it is merely optimizing a norm over a space of matrices, is given below:

$$\min_{X \in \mathbb{R}^{mxn}} \|X\|_* \text{ s.t. } X_{ij} = A_{ij}$$

However, it turns out that the low rank property is not sufficient, as there exists matrices like $\left(\begin{smallmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{smallmatrix}\right)$ that are rank 1, but very difficult to recover without seeing a large number of samples. To solve this, Candes and Recht (2008) introduced the notion of coherence, which was lifted directly from signal processing.

It is motivated by the idea that the sparse matrix we use as a starting off point can be seen as the output of a sampling operator on the full matrix we wish to recover. If we assume this sampling operator acted roughly uniformly, then sparse matrices with "concentrated" singular vectors will perform poorly at recovering the full matrices they were sampled from. A matrix whose column space is low coherence does not have concentrated singular vectors - instead it "spreads its energy" over the ambient space the column space lives in.

Formally, the coherence $\mu(U)$ of a subspace $U \in \mathbb{R}^n$ vis-a-vis the standard basis e_i is given by

$$\mu(U) = \frac{n_1}{r} \max_{i \in 1,\ldots,n_1} \|P_O(U)e_i\|_2^2$$

Low coherence matrices are exactly those which can be recovered via nuclear norm minimization. In particular, their main result establishes that for sufficiently low coherence and sufficiently low rank matrices, the amount of retained data required to recover the full matrix uniquely and exactly is $n^{1.2}log(n)$.

Developments in the theory after Candes and Recht (2008) have focused on improv-

ing that bound on the number of samples required for exactness. Candes and Tao (2009) showed the bound could be pushed to essentially $nlog(n)$ (where for $M \in \mathbb{R}^{n_1 x n_2}, n = max(n_1, n_2)$) and noted that "exact recovery by nuclear norm minimization occurs nearly as soon as it is information-theoretically possible". Recht (2009) uses a strictly probabilistic argument to put a very low upper bound on the amount of samples needed to recover exactly, pushing it all the way down to $32(n1 + n2)log^2(n2)$.

A remarkable recent step forward in this literature was taken by Bertsimas et al, who introduced the use of the matrix perspective trick in order to get past coherence. They start with a very general problem:

$$\min_{X \in S_+^n} \langle C, X \rangle + \Omega(X) + \mu * Rank(X) \text{ s.t. } \langle A_i, X \rangle = b_i, X \in K, Rank(X) \leq k$$

One instantiation of this is our familiar least squares regression, albeit with a Frobenius regularizer and low rank constraint on the beta vector.

$$\min_{\beta \in \mathbb{R}^{pxn}} \frac{1}{2m} \|Y - X\beta\|_F^2 + \frac{1}{2\gamma} \|\beta\|_F^2 + \mu * Rank(\beta)$$

This could be solved by replacing the $Rank(X)$ term with the nuclear norm as specified above, but Bertsimas points out that this requires low coherence, which is a strong constraint. Instead, they claim to be able to solve this problem via the matrix perspective trick without coherence.

3 Algorithmic Notes

Candes and Recht (2008) provided numerical results by setting up the nuclear norm minimization program as a semi-definite problem and then using a standard SDP solver, SDPT3. Tutuncu, Toh, and Todd (n.d.) describe the SDPT3 algorithm: the solver uses an interior point method, which involves converting a generally convex problem into one with equality constraints only, then applying gradient descent with a Newton step.

Keshavan, Montanari, and Oh (2009) describe what they call *spectral matrix completion*, a different algorithm for matrix completion, and prove that this method can obtain arbitrary mean square error. Their required number of samples for exact convergence is $\sim nlog(n)$.

Spectral matrix completion gets its power from taking an SVD of the sparse matrix and then proceeding to throw out even more information if the number of samples is too high for the algorithm to handle. This second sparsification is referred to by the authors

as *trimming*. What's surprising is that getting rid of data can help the algorithm converge - in fact, this effect can be explained by the same notion of coherence introduced by Candes and Recht (2008).

After taking the SVD and trimming, the resultant matrix is low-dimensional and thus admits easy optimization through standard methods: Keshavan recommends gradient descent with line search.

Xu, Yin, Wen, and Zhang (2012) describes an ADMM approach to solving the matrix completion problem. They pose and provide an algorithm for solving the below problem:

$$\min_{(U,V,W,X,Y,Z)} \frac{1}{2}\|XY - Z\|_F^2 \text{ s.t. } X = U, Y = V, U \geq 0, V \geq 0, Z_{ij} = M_{ij}$$

Because any rank q matrix $M \in \mathbb{R}^{mxn}$ can be decomposed into factors $X \in \mathbb{R}^{mxq}, Y \in \mathbb{R}^{qxn}$, we can recover the full matrix as just $M = XY$. The update steps are given as:

$$X_{k+1} = arg\min L_A(X, Y_k, Z_k, U_k, V_k, \Lambda_k, \Pi_k) = (Z_k Y_k^T + \alpha U_k - \Lambda_k)(Y_k Y_k^T + \alpha I)^{-1}$$

$$Y_{k+1} = arg\min L_A(X_{k+1}, Y, Z_k, U_k, V_k, \Lambda_k, \Pi_k) = (X_{k+1} X_{k+1}^T + \beta I)^{-1}(X_{k+1}^T Z_k + \beta V_k - \Pi_k)$$

$$Z_{k+1} = arg \min_{Z_{ij}=M_{ij}} L_A(X_{k+1}, Y_{k+1}, Z, U_k, V_k, \Lambda_k, \Pi_k) = X_{k+1}Y_{k+1} + (M - X_{k+1}Y_{k+1})_0$$

$$U_{k+1} = arg \min_{U \geq 0} L_A(X_{k+1}, Y_{k+1}, Z_{k+1}, U, V_k, \Lambda_k, \Pi_k) = S_0(X_{k+1} + \Lambda_k/a)$$

$$V_{k+1} = arg \min_{V \geq 0} L_A(X, Y_k, Z_k, U_k, V_k, \Lambda_k, \Pi_k) = S_0(Y_{k+1} + \Pi_k/\beta)$$

$$\Lambda_{k+1} = \Lambda_k + \gamma\alpha(X_{k+1} - U_{k+1})$$

$$\Pi_{k+1} = \Pi_k + \gamma\beta(Y_{k+1} - V_{k+1}),$$

where $(M - X_{k+1}Y_{k+1})_0$ censors the data in the updated matrices to have the same sparsity as the input matrix and S_0 thresholds elementwise at 0. We stop at $\|(X_k Y_k - A)\|_F / \|A\|_F \leq tol$.

4 Simulation and Discussion

We demonstrate the efficacy of Xu et al's ADMM approach. We generate a low-rank matrix via first taking the QR decomposition of an appropriately sized random matrix, then providing $0.2n$ singular values distributed Gaussian. Our low rank matrix is then $Q\Sigma R$.

We censor this matrix with a matrix taking binary values, 75% of which are 1s. We then implement the above algorithm, with stopping tolerance 10^{-4}.

Unfortunately, our simulation failed to converge in any reasonable amount of time. You can see an attempt and the convergence of the error in Figure 1. In lieu of a discussion of those results we review the simulation results in Xu et al.

Compared to other top of the line algorithms, the ADMM algorithm was better at recovering original image data when the input data was extremely sparse, only about 10% of samples available. This is demonstrated in Figure 2.

References

Bertsimas, D., Cory-Wright, R., & Pauphilet, J. (2022, March). *A new perspective on low-rank optimization.* arXiv. Retrieved 2023-03-23, from http://arxiv.org/abs/2105.05947 (arXiv:2105.05947 [cs, math, stat])

Boyd, S. P., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge, UK ; New York: Cambridge University Press.

Candes, E. J., & Recht, B. (2008, May). *Exact Matrix Completion via Convex Optimization.* arXiv. Retrieved 2023-03-23, from http://arxiv.org/abs/0805.4471 (arXiv:0805.4471 [cs, math])

Candes, E. J., & Tao, T. (2009, March). *The Power of Convex Relaxation: Near-Optimal Matrix Completion.* arXiv. Retrieved 2023-03-23, from http://arxiv.org/abs/0903.1476 (arXiv:0903.1476 [cs, math])

Chi, Y. (n.d.). Low-Rank Matrix Completion.

Keshavan, R. H., & Montanari, A. (2010, January). *Regularization for Matrix Completion.* arXiv. Retrieved 2023-03-23, from http://arxiv.org/abs/1001.0279 (arXiv:1001.0279 [stat])

Keshavan, R. H., Montanari, A., & Oh, S. (2009, September). *Matrix Completion from a Few Entries.* arXiv. Retrieved 2023-03-24, from http://arxiv.org/abs/0901.3150 (arXiv:0901.3150 [cs, stat])

Morgenshtern, V. (n.d.). Lecture 19–20: Matrix Completion.

Recht, B. (2009, October). *A Simpler Approach to Matrix Completion.* arXiv. Retrieved 2023-03-23, from http://arxiv.org/abs/0910.0651 (arXiv:0910.0651 [cs, math])

Tutuncu, R. H., Toh, K. C., & Todd, M. J. (n.d.). SDPT3 — a Matlab software package for semidefinite-quadratic-linear programming, version 3.0.

Xu, Y., Yin, W., Wen, Z., & Zhang, Y. (2012, April). An Alternating Direction Algorithm for Matrix Completion with Nonnegative Factors. *Frontiers of Mathematics in China*, *7*(2), 365–384. Retrieved 2023-03-23, from http://arxiv.org/abs/1103.1168 (arXiv:1103.1168 [cs, math]) doi: 10.1007/s11464-012-0194-5

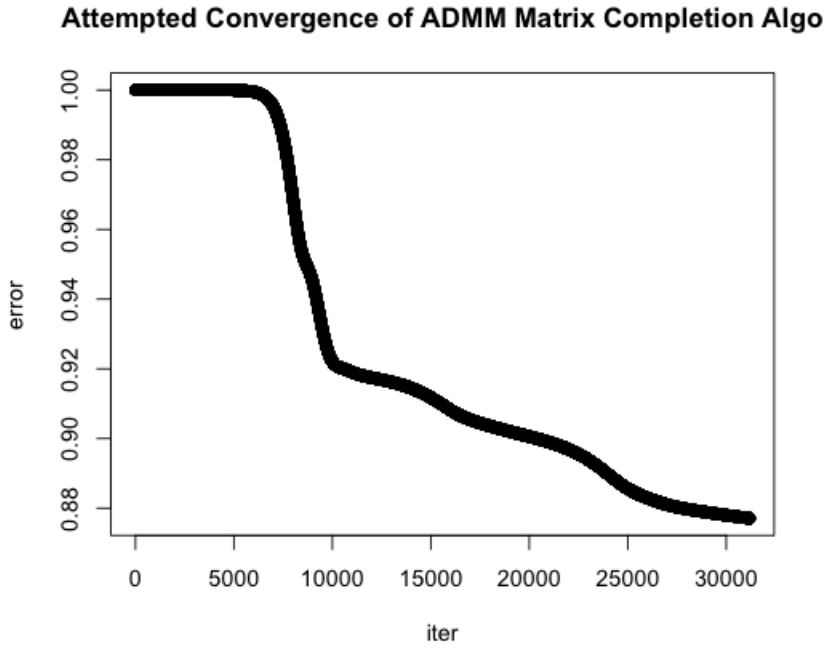

Figure 1: Attempt at simulation

Figure 5: Recovered 768×1024 Kittens with estimate rank $q = 40$ by Algorithm 1 (left), LMaFit (middle), and FPCA (right)

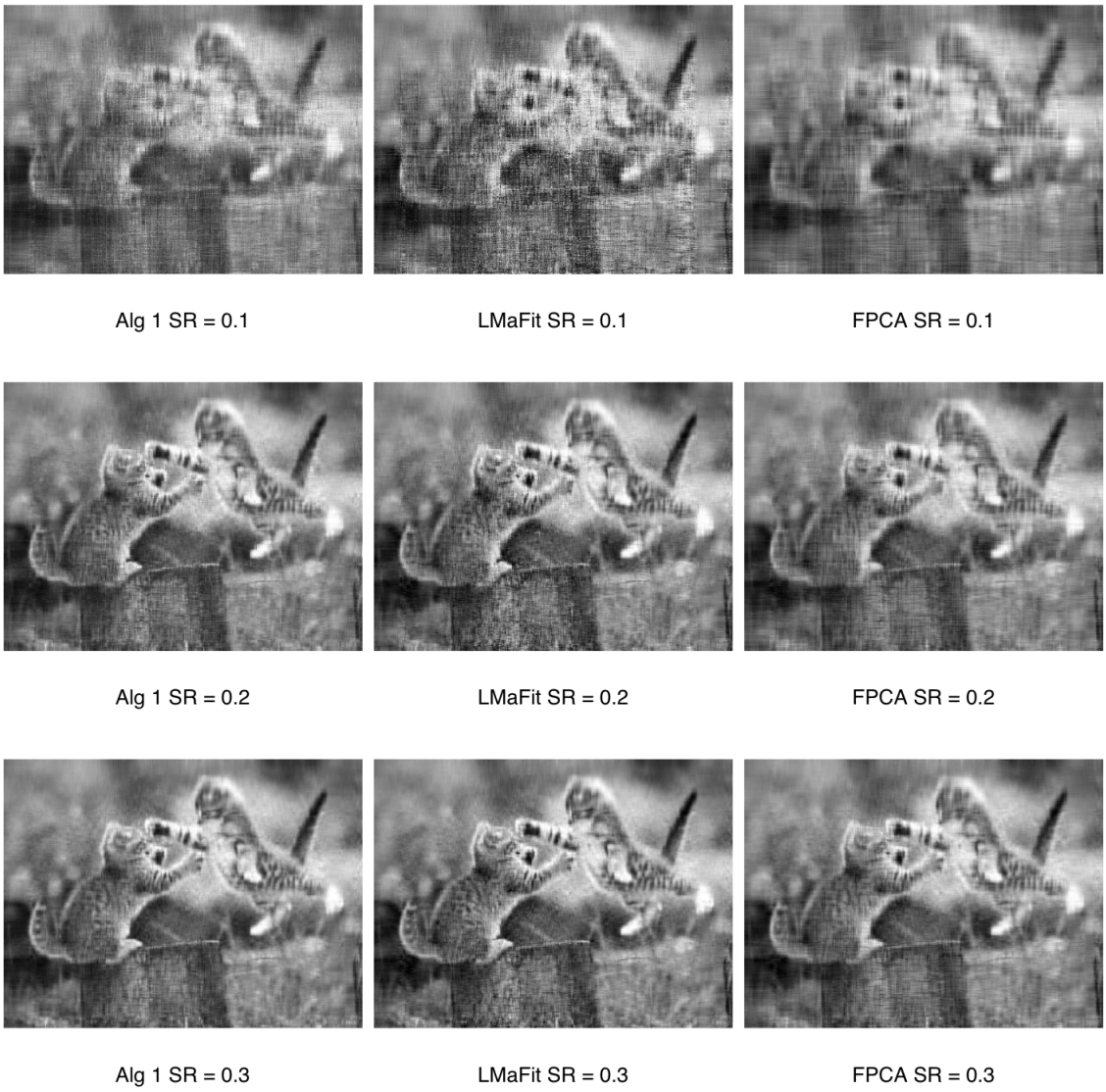

Figure 2: Image data recovery from Xu et al.

```
neg_threshold <- function(x){ max(x,0) }


a <- qr(matrix(rnorm(10000),ncol=100,nrow=100))
sing_values_vec <- sort(c(abs(rnorm(20,5,10)),rep(0,80)),decreasing=TRUE)
sing_values <- diag(sing_values_vec)
M <- qr.Q(a) %*% sing_values %*% qr.R(a)
B <- matrix(rbinom(10000,1,0.25),100,100)
```

```
A <- M * B
q <- 1
maxiter <- 1000
tol <- 10e-2
alpha <- 1e6
gamma <- 1.618
beta <- alpha/100
Y <- matrix(abs(rnorm(10000,0,1)),nrow=20,ncol=100)
Z <- A
U <- matrix(0,nrow=100,ncol=20)
V <- matrix(0,nrow=20,ncol=100)
LambMat <- matrix(0,nrow=100,ncol=20)
PiMat <- matrix(0,nrow=20,ncol=100)
IdMatY <- diag(20)
IdMatX <- diag(100)
results <- c()
iter <- 0
while(TRUE){
  X <- (Z %*% t(Y) + alpha*U - LambMat) %*% solve(Y %*% t(Y) + alpha*IdMatY)
  Y <- (t(X) %*% Z + beta*V - PiMat) %*% solve(X %*% t(X) + beta*IdMatX)
  Z <- X %*% Y +  B*(M - (X %*% Y))
  U <- apply(X + (1/alpha)*LambMat, c(1,2), neg_threshold)
  V <- apply(Y + (1/beta)*PiMat, c(1,2), neg_threshold)
  LambMat <- LambMat + gamma*alpha*(X - U)
  PiMat <- PiMat + gamma*beta*(Y - V)
  results <- c(results,norm(B*(X %*% Y - A),type='F')/norm(B*A,type='F'))
  if(norm(B*(X %*% Y - A),type='F')/norm(B*A,type='F') <= tol){ break }
  print(norm(B*(X %*% Y - A),type='F')/norm(B*A,type='F'))
  iter <- iter+1
}
plot(1:iter,results,main='Attempted Convergence of ADMM Matrix Completion Algo'
```