A More Efficient Approach to Large Scale Matrix Completion Problems

Matthew Olson

August 25, 2014

Abstract

This paper investigates a scalable optimization procedure to the low-rank matrix completion problem posed by Candes and Recht [2]. We identify the singular value decomposition as a computational bottleneck for large problem instances, and propose utilizing an approximately computed SVD borne out of recent advances in random linear algebra. We then use this approximately computed SVD to implement some popular first order methods for solving the matrix completion problem. Our results indicate that these modified routines can easily handle very large data sets. In particular, we are able to recover 35 million entries from a $10^4 \times 10^4$ matrix in a matter of minutes.

1 Introduction

1.1 Motivation

A great deal of current research in statistics has focused around exploiting structured sparsity in large data sets. One important example of this consists of the recovery of a low-rank matrix from its partially observed entries. As a motivating example, suppose one asks a group of m people to rate a large collection of n movies. One can then collect this data in a m $\times$ n matrix $\mathbf{M}$, where the rows are indexed by people, and the columns are indexed by movies. Of course, not every person will have seen every movie, so this matrix will in general have many missing entries. The goal is to fill in these missing entries so that one can infer how a person would rate a movie he has not seen yet. This type of problem has found a variety of applications ranging from movie recommender systems to computer vision.

Generally speaking, the problem of inferring missing entries from a partially observed matrix is an ill-posed problem: there are many different ways one could "complete" the matrix. However, Candes and Recht showed that if one assumes the underlying matrix is low-rank, one can recover the matrix exactly with high probability by solving a convex optimization problem [2]. In order to formulate the optimization problem, let Ω consist of the indices

of a m × n matrix $\mathbf{M}$ that are observed, and let $\mathcal{P}_\Omega(\mathbf{X})$ be the orthogonal projection of a matrix $\mathbf{X}$ on the set of all matrices that are zero except possibly at the indices in Ω. To recover the underlying matrix, Candes and Recht propose solving [1]

$$\begin{aligned} \underset{\mathbf{X}\in\mathbb{R}^{\mathrm{m}\times\mathrm{n}}}{\text{minimize}} \quad & ||\mathbf{X}||_* \\ \text{subject to} \quad & \mathcal{P}_\Omega(\mathbf{X}) = \mathcal{P}_\Omega(\mathbf{M}) \end{aligned} \tag{1}$$

where $||X||_*$ is the nuclear norm, i.e. the sum of the singular values of $\mathbf{X}$. The nuclear norm is often used as a computationally tractable relaxation for the rank of a matrix.

1.2 Computational Challenges

When the dimensions of the underlying matrix m and n are at most a few hundred, problem (1) can be efficiently solved as a semidefinite program using interior point methods. However, this is no longer the case when one considers data sets with thousands, if not hundreds of thousands of rows and columns (for example, movie rating data sets can easily consist of tens of thousands of users rating thousands of movies). As a result, it is imperative to find optimization routines that scale well to huge data sets to use low-rank matrix recovery in practical settings.

The most widely investigated optimization methods for this problem are first order methods that involve evaluating the subdifferential or proximal operator (to be defined later) of the nuclear norm. Computing these quantities boils down to a computing a partial or full singular value decomposition of a very large matrix. It is widely known that computing singular value decompositions of very large matrices can become prohibitively expensive, and so a recent line of research has developed randomized methods that scale well for approximately computing singular values and vectors: the algorithm we consider in this paper is the ThinSVD from [7].

This aim of this paper is to utilize the scalable SVD from [7] to solve large scale instances of (1). In particular, we will implement two popular optimization procedures, the Singular Value Thresholding Algorithm and Douglas-Rachford splitting, using the ThinSVD. Computational experiments show that these modified procedures scale very well with the size of the data, and we are able to recover matrices as large as $10^4 \times 10^4$ in a few minutes on a desktop computer.

[1] This procedure solves the so called matrix completion problem in the "noiseless" setting. One can also consider the related problem of observing the matrix entries with, say, Gaussian error.

2 Algorithms

This section will describe the optimization procedures that will be used in subsequent parts of the paper, namely, the Singular Value Thresholding algorithm (SVT) [5] and Douglas-Rachford splitting. The first algorithm was designed specifically to solve (1), while the second is a general purpose algorithm for minimizing the sum of two closed, convex functions. Both methods are based on the proximal mapping of a convex function, which we will describe below.

2.1 Proximal Operator

We will begin by defining the proximal mapping of a closed, convex function, and then will derive some useful properties specific to the nuclear norm.

Definition 2.1. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a closed convex function. The proximal mapping of f evaluated at $x \in \mathbb{R}^n$ is defined as*

$$\boldsymbol{prox}_f(x) = \arg\min_u f(u) + 1/2||x-u||^2.$$

The first thing to note is that the proximal operator is well-defined: if f is closed and convex, then $f(u) + 1/2||x-u||^2$ is strongly convex and so it has a unique minimizer. In practice, one can often think of the proximal operator as a projection. In fact, if $\delta_C(x)$ is the indicator of a closed, convex set C, that is

$$\delta_C(x) = \begin{cases} 0 & x \in C \\ \infty & x \notin C \end{cases}$$

then it is easy to see that $\mathbf{prox}_{\delta_C}(x)$ is simply the projection of x onto C. As it will be an essential building block in the optimization procedures described in following sections, our first result will be to calculate the proximity operator of the nuclear norm. This calculation will be based on a well-known theorem about the subgradients of certain symmetric matrix norms due to Lewis [6].

Theorem 2.2 (Lewis, 1995)**.** *Let $U\Sigma V^T$ be a singular value decomposition for $X \in \mathbb{R}^{m\times n}$, and let $\sigma : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\wedge n}$ be the function that maps a matrix to its spectrum in non-increasing order, i.e. $\sigma(X) = diag(\Sigma)$. Suppose $F : \mathbb{R}^{m\times n} \to \mathbb{R}$ is a convex function that can be written $F(X) = g(\sigma(X))$ where $g : \mathbb{R}^{m\wedge n} \to \mathbb{R}$ is a convex function that is permutation invariant in its arguments. Then*

$$\partial F(X) = \left\{U\, diag(g) V^T \mid g \in \partial g(\sigma(X))\right\}.$$

The content of the above theorem is that the subgradient of a lot of matrix norms, in partcular the nuclear norm and spectral norm, can be

computed easily once one has a singular value decomposition in hand. We will use this result to derive the proximal operator $\mathbf{prox}_{\lambda||.||_*}(X)$, which we will denote in the future as S_λ.

Result 2.3. *If $X = U\Sigma V^T$ is a singular value decomposition, then*

$$\boldsymbol{prox}_{\lambda||.||_*}(X) = U\,diag((\sigma(X) - \lambda)_+)\; V^T \doteq S_\lambda(X).$$

Proof. First, using the subdifferential characterization of the minimum of a convex function, $Y = S_\lambda(X) = \arg\min_U \lambda||U||_* + 1/2||X - U||^2$ if and only if $0 \in \partial\big(\lambda||Y||_* + 1/2||Y - X||_F^2\big) = \lambda\partial||Y||_* + (Y - X)$. Let us try to solve this inclusion by assuming $Y = U\Delta V^T$ where Δ is a diagonal matrix with the same dimensions as Σ. We can then reduce the inclusion to $0 \in \lambda\partial||\Delta||_* + (\Delta - \Sigma)$, and this holds precisely when $\Delta_{ii} = (\Sigma_{ii} - \lambda)_+$. □

In other words, the proximal operator of the scaled nuclear norm is a simple soft-thresholding operation on the singular values of a matrix. As this operation can only reduce the rank of a matrix, it can heuristically be thought of as the reason nuclear norm minimzation produces low rank matrices. It is also worth noting at this point that the vector analogue of the nuclear norm is the ℓ^1 norm, and the corresponding proximal operator is the analogous vector soft-thresholding operator $s_\lambda(v)$ for $v \in \mathbb{R}^n$

$$(s_\lambda(v))_i = \text{sign}(v_i)(|v_i| - \lambda)_+.$$

This operation plays a central role in FISTA, a popular solution method for LASSO regression[1]. We will conclude with a final useful fact about proximal operators that reinforces their similarity to projection operators.

Result 2.4. *The proximal mapping is nonexpansive:*

$$||\boldsymbol{prox}_f(x) - \boldsymbol{prox}_f(y)|| \leq ||x - y||.$$

Proof. Let $u = \mathbf{prox}_f(x)$ and $v = \mathbf{prox}_f(y)$. Then $(x - u) \in \partial f(u)$ and $(y - v) \in \partial f(v)$. The characterization of the subgradient implies the two inequalities

$$f(v) \geq f(u) + (x - u)^T(v - u)$$
$$f(u) \geq f(v) + (y - v)^T(u - v).$$

If we add the first inequality to the second, we get $0 \geq (x - y)^T(v - u) + ||v - u||^2$. An application of the Cauchy-Schwartz inequality then allows us to conclude. □

For us, the practical value of Result 2.3 and Result 2.4 is as follows. Result 2.3 says that in order to compute first order quantities associated

with a scaled nuclear norm, we need only be able to compute the singular values above a certain threshhold. Result 2.4 says that the "error" we get from computing the proximal operator a a nearby point is no greater than the error we get from the first approximation. These facts suggest that a fast, approximate SVD calculation could be a useful tool in computations involved with solving (1). This is explored more in Section 3. For much more on the proximal mapping, see [8].

2.2 Singular Value Thresholding Algorithm

We will now consider a widely cited algorithm for solving the matrix completion problem, namely the Singular Value Thresholding algorithm, or SVT for short [5]. The authors of [5] propose solving the following related problem to (1), namely

$$\begin{aligned} \underset{\mathbf{X}}{\text{minimize}} \quad & \tau||\mathbf{X}||_* + \frac{1}{2}||\mathbf{X}||_F^2 \\ \text{subject to} \quad & \mathcal{P}_\Omega(\mathbf{X}) = \mathcal{P}_\Omega(\mathbf{M}). \end{aligned} \tag{2}$$

They show that as $\tau \to \infty$ the solution to this problem approaches that of (1), and they propose the following iterative scheme:

$$\begin{aligned} &\mathbf{Y}_0 = 0 \\ &\mathbf{X}_k = S_\tau(\mathbf{Y}_{k-1}) \\ &\mathbf{Y}_k = \mathbf{Y}_k + \delta_k\left(\mathcal{P}_\Omega(\mathbf{M}) - \mathcal{P}_\Omega(\mathbf{X}_k)\right). \end{aligned}$$

These iterations can be obtained as gradient ascent applied to the dual formulation of (2). To see this, denote the Lagrangian of the problem (2) by $\mathcal{L}(\mathbf{X}, \mathbf{Y}) = \tau||\mathbf{X}||_* + 1/2||\mathbf{X}||_F^2 + \langle \mathbf{Y}, (\mathcal{P}_\Omega(\mathbf{X}_k) - \mathcal{P}_\Omega(\mathbf{M}))\rangle$ and the dual function $q(\mathbf{Y}) = \inf_{\mathbf{X}} \mathcal{L}(\mathbf{X}, \mathbf{Y})$. Since strong duality holds, it is enough to maximize $q(Y)$ and we can do this via gradient ascent. Note that $q(\mathbf{Y})$ is differentiable since the objective function is strongly convex, and so the gradient can be found simply as

$$\bigtriangledown_{\mathbf{Y}} g(\mathbf{Y}) = \bigtriangledown_{\mathbf{Y}} \mathcal{L}(\tilde{\mathbf{X}}, \mathbf{Y}) = (\mathcal{P}_\Omega(\mathbf{M}) - \mathcal{P}_\Omega(\mathbf{X}))$$

where

$$\begin{aligned} \tilde{\mathbf{X}} &= \arg\min_{\mathbf{X}} \mathcal{L}(\mathbf{X}, \mathbf{Y}) \\ &= \arg\min_{\mathbf{X}} \tau||\mathbf{X}||_* + 1/2||\mathbf{X} - \mathcal{P}_\Omega(Y)||_F^2 \\ &= S\tau(\mathcal{P}_\Omega(\mathbf{Y})). \end{aligned}$$

Cai, et al. [5] make the crucial observation that, at least empirically, the iterates $\{\mathbf{X}_k\}$ tend to increase in rank monotonically to the solution of

(2). This fact is key to computation for large problems, as one only needs to compute a truncated SVD at intermediate steps of the procedure. In the results section, we will take advantage of this fact when implementing the SVT by making use of the ThinSVD.

2.3 Douglas-Rachford Splitting

The second optimization procedure we consider to solve (1) is Douglas-Rachford splitting. This is a general purpose method that can be applied to minimize the sum of two closed, convex functions without assuming differentiability. When applied to the dual, it is also know as the Alternating Direction Method of Multipliers (ADMM). ADMM has seen a recent surge in interest as a routine for solving problems from large scale optimization problems in machine learning and compressed sensing [9]. For this reason, and others to be described, it is a reasonable candidate to try out on (1). See Combettes et al. [3] for a thorough discussion of Douglas-Rachford splitting in signal processing problems.

Suppose we wish to minimize $h(x) = f(x) + g(x)$. Such problems are known as "composite" since the objective function can be decomposed into the sum of functions which are hopefully easier to handle. For $\lambda > 0$ define the following operator based on the proximal mappings of f and g:

$$F_\lambda(z) = z + \mathbf{prox}_{\lambda f}\left(2\mathbf{prox}_{\lambda g}(z) - z\right) - \mathbf{prox}_{\lambda g}(z).$$

Using the result below, it turns out that to minimize h we simply need to find a fixed point of $F_\lambda(z)$.

Result 2.5. *If $z = F_\lambda(z)$ then $x = \boldsymbol{prox}_{\lambda g}(z)$ is a minimizer of $f + g$.*

Proof. If $z = F_\lambda(z)$ and $x = \mathbf{prox}_{\lambda g}(z)$, then $\mathbf{prox}_{\lambda f}(2x - z) = x$. The former fact ($x = \mathbf{prox}_{\lambda g}(z)$) implies that $z - x \in \lambda\partial g(x)$ while the latter implies $x - z \in \lambda\partial f(x)$. Adding this two inclusions, we get that $0 \in \lambda\partial f(x) + \lambda\partial g(x)$. Hence, x minimizes $f + g$. □

It turns out that we can find fixed points of $F_\lambda(z)$ by iterating $z_k = F_\lambda(z_{k-1})$. Specializing this to the problem (1) and introducing some auxiliary variables into the fixed point iteration, we get the the following iterative optimziation scheme:

$$\begin{aligned}
\overline{X}_k &= S_\lambda(Z_{k-1}) \\
\overline{Z}_k &= 2\overline{X}_k - Z_{k-1} \\
X_{k+1} &= \mathcal{P}_\Omega\left(\overline{Z}_k\right) \\
Z_{k+1} &= Z_k + X_{k+1} - \overline{X}_k.
\end{aligned}$$

Note in particular that the first and third steps involve the evaluation of the proximal operator. This procedure also has a worst case run-time of

$O(1/k)$, which is much better than a naive projected subgradient method which would involve the same computational cost. Importantly for us, it turns out that this procedure still converges even if the proximal operators are evaluated with error (as long as the errors are summable) [4]. As the bulk of the computational cost is in computing a SVD, this result allows us to consider faster approximate SVD calculations via the ThinSVD. This possibility is explored in a later section.

2.4 Randomized SVD

As described briefly in the previous sections, ThinSVD is a randomized algorithm in linear algebra that provides a fast and scalable truncated singular value decomposition [7]. Suppose we have a m $\times$ n matrix $\mathbf{A}$ and we wish to find a good rank-k approximation to $\mathbf{A}$ in the spectral or frobenius norm sense. It happens that (with some slight tweaks) $\mathbf{Y} = \mathbf{A\Omega}$ is a very good candidate, where $\mathbf{\Omega}$ is a n $\times$ p ($p > k$) matrix consisting of random Gaussian entries. The intuition behind this result is that as ker($\mathbf{A}$) is a proper subspace of $\mathbb{R}^m$, then the columns of $\mathbf{\Omega}$ almost surely miss ker($\mathbf{A}$). We can then perform an SVD on $\mathbf{Y}$ to obtain a good rank-k approximation to $\mathbf{A}$. [2]. The ThinSVD is outlined below

1) Draw a random Gaussian m $\times$ (k+p) matrix $\mathbf{\Omega}$.

2) Set $\mathbf{Y} = \mathbf{A\Omega}$.

3) Use a QR decomposition to find $\mathbf{Y} = \mathbf{QR}$.

4) Perform SVD to find $\mathbf{Q}^T\mathbf{A} = \hat{\mathbf{U}}\Sigma\mathbf{V}^T$.

5) Set $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$

When looking at the ThinSVD algorithm, it is essential to note that the QR and SVD decompositions in steps (3) and (4) are done on matrices with far less column than $\mathbf{A}$, and are thus relatively cheap to compute. The majority of the work, therefore, is spent on randomly generating Gaussian numbers and matrix multiplication. These operations readily lend themselves to parallel, large scale computations, and for this reason randomized methods in linear algebra are becoming an intensely researched area [7]. The remaining sections will explore how this tool can be used to evaluate the proximal operator of the nuclear norm, providing a scalable first order method for nuclear norm minimization.

[2] A technical detail that matters in practice, the rate of decay of the singular values of $\mathbf{A}$ greatly affects the quality of this approximation. One can employ oversampling and power-iteration "tweaks" to mitigate problems like this. See [7] for more details

3 Numerical Results

In this section we will provide the results of some computational experiments that will illustrate the usefulness of using the ThinSVD in the SVT and Douglas-Rachford splitting approaches to solving (1). We will begin by discussing the implementation details, and then discuss results from low rank matrix recovery, euclidean distance matrix completion, and a real data matrix completion problem. It is worth noting that in all of these settings, the problem sizes are much too large for cone solvers based on interior point methods to handle (for instance, CVX on our desktop chokes once the problem dimensions exceed a few hundred).

3.1 Implementation Details

The major novelty in our application of the SVT and Douglas-Rachford splitting procedures to solve (1) is the use of a randomized SVD to approximate the proximal operator of the nuclear norm. As described in Section 2, the proximal operator simply thresholds singular values below a certain level. Therefore, to approximate this operation, we used the random SVD to find the first 100 singular values and vectors of a matrix, and then thresholded these quantities accordingly: if the smallest singular value was above the threshold level, we computed an additional "chunk" of singular values and vectors [3]. Previous literature uses exact partial SVD calculations in this step (such as those provided via the Lanczos algorithm), but we find that the random SVD scales better and is thus appropriate for large data problems.

Both algorithms also required chosing some additional tuning parameters. For the SVT, we followed the suggestions in [5] and chose $\tau = 5\sqrt{nm}$ and $\delta = 1.2p$, where n and m are the number of rows and columns of the matrix we are trying to recover, and p is the proportion of entries we observe. For the Douglas-Rachford method, we chose a thresholding level of $\lambda = 200$. This choice required some tinkering, and had a suprising affect on the rate of convergence.

Finally, we declared convergence for each algorithm once the relative reconstruction error

$$\frac{||X - X^*||_F}{||X^*||_F}$$

was less than $\epsilon = 10^{-4}$. In our experiments we knew the solution ahead of time, and so we used this convergence criterion so that we could compare the speed of each algorithm. In practice, one would clearly use different convergence criteria.

[3] The number 100 is a somewhat arbitrary number, but as a tuning parameter it worked well for our experiments. There also exist easily implementable incremental methods that wouldn't require choosing a predetermined number of singular values to compute.

3.2 Gaussian Matrix Recovery

			SVT		Douglas-Rachford	
	N	Rank	Iterations	Time	Iterations	Time
1	5000	10	42	90	52	106
2	5000	20	48	101	54	112
3	5000	50	50	128	66	136
4	10000	10	38	275	52	343
5	10000	20	42	305	54	380
6	10000	50	48	351	60	420

Table 1: Gaussian Matrix Recovery Results

In the first experiment we recover large randomly generated matrices of varying ranks from partially observed entries. As an illustration, to generate a problem with rank 50 and dimensions 5×10^3 by 5×10^3, we generate a 5×10^3 by 50 matrix $\mathbf{M}_1$ of Gaussian entries, a 50 by 5×10^3 matrix $\mathbf{M}_2$ with Gaussian entries, and then we form the product $\mathbf{M} = \mathbf{M}_1\mathbf{M}_2$. We then "reveal" 30% of these entries uniformly at random to our algorithm.

Table 3.2 provides the number of iterations and time for the SVT and Douglas-Rachford algorithm to solve (1) for problems with varying size and rank. Even with the approximately evaluated singular value thresholding operator, both procedures "recover" the original matrix in a reasonably short amount of time. For instance, the SVT recovered a rank 10 matrix of size $10^4 \times 10^4$ in under five minutes. It appears that the time until convergence for both procedures increases with the rank of the problem, but not dramatically.

3.3 Euclidean Distance Matrix Recovery

		SVT		Douglas-Rachford	
	N	Iterations	Time	Iterations	Time
1	1000	50	9	58	10
2	5000	32	69	54	113
3	10000	28	201	74	523

Table 2: Euclidean Distance Matrix Results

In the second computational experiment, we solve problem (1) when the underlying matrix is a Eucidean distance matrix of vectors in $\mathbb{R}^{10}$. In general, Euclidean distance matrices of vectors in $\mathbb{R}^m$ are known to have rank at most $m + 2$, and thus they are inherently low rank objects. In order to generate examples from this problem class, we first generate N five dimensional vectors $x_1, \ldots, x_N$, and then form the matrix $\mathbf{M}$, where

$(\mathbf{M})_{i,j} = ||x_i - x_j||_2^2$. As in the previous section, we (somewhat arbitrarily) reveal 30% of the entries of this matrix, and then we solve (1) using the SVT and Douglas-Rachford algorithms. Table 3.2 provides the time to convergence for each of these algorithms under a variety of problem sizes. Again, in these large problems each procedure converges quite quickly. It is worth noting that for very small problem sizes ($N < 500$) some further tweaks were needed to get both the SVT and Douglas-Rachford procedures to converge. In particular, when the spectrum of a matrix does not decay quickly, the approximation of the ThinSVD suffers. One then needs to perform power iterations (see [7] for more details). From our experience, despite this additional effort, the ThinSVD still appears to outperform other truncated SVD procedures. In particular, we found that Matlab's partial SVD command `svds` could be at least ten times slower than the random partial SVD we used, and the scaling only got worse with problem size.

3.4 Image Recovery

The final example we consider is one of the canonical illustrations of matrix completion, recovering the MIT logo from a small set of observations. We consider a grayscale 1500×2800 image of the MIT logo and sample at random 30% of its entries. Using the the same procedure as in the previous sections, we were able to recover a good reconstruction of the true image. Figure 1 shows the sampled image (where missing entries are filled in with noise for illustration purposed), and Figure 2 shows the recovered image. In this case, the SVT converged after 230 iterations in 120 seconds, while Douglas-Rachford splitting converged after 56 iterations in 30 seconds. This illustrates that neither procedure dominates the other in terms of convergence speed.

4 Discussion

The goal of this paper was to demonstrate that optimization algorithms for the matrix compeltion problem (1) can be modified to recover large matrices in a scalable way. The key computation tool was the randomized singular value decompostion, which we used to approximately evaluate the proximal operator of the nuclear norm, the soft singular value thresholding operator. We then illustrated this in a series of computational experiments by implementing the SVT and Douglas-Rachford splitting methods to solve large scale instances of (1). We were able to recover $10^4 \times 10^4$ matrices in a matter of minutes. Furthermore, we observed in practice that the randomized SVD was anywhere from three to ten or higher times faster than determinstic partial SVD calculations.

In Section 2 we pointed out that the subdifferential and proximal mapping of many matrix norms depend on efficiently computing a singular value

Figure 1: Partially observed logo.

Figure 2: Image recovered by SVT algorithm.

decomposition. These include the spectral norm, nuclear norm, frobenius norm, and more generally any Shatten-p norms. This suggests that the methodology covered in this paper could also be applied to solve large scale instances of other problems in statisics involving these matrix norms.

References

[1] M. Teboulle A. Beck. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. IMAGING SCIENCES*, 2009.

[2] E.J. Candes and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 2009.

[3] P. Combettes and J.-C. Pesquet. Proximal splitting methods in signal processing. *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, 2011.

[4] J. Eckstein and D.P. Bertsekas. On the douglas-rachford splitting method and the proximal point algorithm for maximal monotone mappings. *Mathematical Programming*, 1992.

[5] Z. Shen J.-F. Cai, E.J. Candes. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 2010.

[6] A.S. Lewis. The convex analysis of unitarily invariant matrix functions. *Journal of Convex Analysis*, 1995.

[7] J.A. Tropp N. Halko P.-G Martinsson. Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions. *SIAM Rev.*, 2011.

[8] N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 2014.

[9] E. Chue B. Peleato S. Boyd, N. Parikh and J. Eckstein. Distributed optimization in statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 2011.