An Accelerated Proximal Gradient Algorithm for Nuclear Norm Regularized Least Squares Problems Arising in Matrix Completion

Sangwoon Yun
Singapore-MIT Alliance, National University of Singapore

June 3, 2009

(Joint work with Kim-Chuan Toh (NUS))

Outline

- Matrix Completion
- General Problem Model:
  Nuclear Norm Regularized Linear Least Squares Problem
- Accelerated Proximal Gradient Algorithm
- Techniques for Further Acceleration
- Numerical Experience
- Conclusions & Future Work

Matrix Completion

Compressed Sensing & Matrix Completion

Is it possible to reconstruct signal exactly or at least accurately from a few observed samples of a signal? (impossible! in general)

However, if the signal is known to be sparse then accurate recovery is possible by ℓ_1 minimization (Candés et al. '06, Donoho '06):

$$\min_{x \in \Re^n} \left\{ \|x\|_1 \ : \ Ax = b \right\},$$

where $A \in \Re^{p \times n}$ ($p < n$ or $p \ll n$).

Now, imagine that we only observe a few entries of a data matrix. Then is it possible to accurately guess the entries that we have not seen?

Netflix problem : suppose we observe a few movie ratings from a large data matrix in which rows are users and columns are movies.
Can we predict the rating a user would hypothetically assign to a movie he/she has not seen? i.e., we would like to infer users preference for unrated movies. (impossible! in general)

However, if the unknown matrix is known to have low rank or approximately low rank, then accurate recovery is possible by nuclear norm minimization (Candés & Recht '08, Candés & Tao '09).

This is known as the matrix completion problem.

Can be formulated as the the following minimization problem:

$$\min_{X \in \Re^{m \times n}} \Big\{ \mathrm{rank}(X) \ : \ X_{ij} = M_{ij}, \ (i,j) \in \Omega \Big\},$$

where M is the unknown matrix with p available sampled entries and Ω is a set of pairs of indices of cardinality p.

By (Candés & Recht '08), a random low-rank matrix can be recovered exactly with high probability from a rather small random sample of its entries and it can be done by solving the following convex relaxation:

$$\min_{X \in \Re^{m \times n}} \Big\{ \|X\|_* \ : \ X_{ij} = M_{ij}, \ (i,j) \in \Omega \Big\}.$$

where $\|X\|_* = \sum_{i=1}^{q} \sigma_i(X)$ and $\sigma_i(X)$'s are the singular values of X and $q = \min\{m, n\}$.

The above problem can be reformulated as a semidefinite program as follows:

$$\begin{array}{ll} \min\limits_{X,W_1,W_2} & \frac{1}{2}\left(\mathrm{Tr}W_1 + \mathrm{Tr}W_2\right) \\ \text{subject to} & X_{ij} = M_{ij},\ (i,j) \in \Omega, \quad \begin{pmatrix} W_1 & X \\ X^T & W_2 \end{pmatrix} \succeq 0. \end{array}$$

But this semidefinite problem has one $(m+n) \times (m+n)$ semidefinite constraint and p affine constraints.

The state-of-art solver SDPT3 and others like SeDuMi are based on interior-point methods and they are not suitable for problems with large $m+n$ or p because the computational cost grows like $O(p(m+n)^3 + p^2(m+n)^2 + p^3)$ and the memory requirement grows like $O((m+n)^2 + p^2)$.

General Problem Model: Nuclear Norm Regularized Linear Least Squares Problem

A convex relaxation of an affine rank minimization problem (Fazel '02):

$$\min_{X \in \Re^{m \times n}} \left\{ \|X\|_* \ : \ \mathcal{A}(X) = b \right\}.$$

The matrix completion problem is a special case of the above problem with $\mathcal{A}(X) = X_\Omega$, where X_Ω is the vector in $\Re^{|\Omega|}$ obtained from X by selecting those elements whose indices are in Ω.

When the matrix variable is restricted to be diagonal, the above problem reduces to the following ℓ_1 minimization problem:

$$\min_{x \in \Re^n} \left\{ \|x\|_1 \ : \ Ax = b \right\}.$$

This problem has attracted much interest in compressed sensing.

Nuclear Norm Regularized Linear Least Squares Problem

If the observation b is contaminated with noise, then $Ax = b$ might not be feasible and so an appropriate norm of the residual $Ax - b$ should be minimized.

The appropriate model to consider can be the following ℓ_1-regularized linear least squares problem:

$$\min_{x \in \Re^n} \ \frac{1}{2}\|Ax - b\|_2^2 + \mu \|x\|_1,$$

where $\mu > 0$.

This motivates us to consider the following nuclear norm regularized linear least squares problem:

$$\min_{X \in \Re^{m \times n}} \ \frac{1}{2}\|\mathcal{A}(X) - b\|_2^2 + \mu \|X\|_*.$$

Accelerated Proximal Gradient Algorithm

We consider the proximal regularized subproblem:

$$\min_{X \in \Re^{m \times n}} \frac{\tau}{2}\|X - G\|_F^2 + \mu\|X\|_*,$$

where $G = Y - \tau^{-1}\mathcal{A}^*(\mathcal{A}(Y) - b)$.

Suppose the singular value decomposition (SVD) of G is given by:

$$G = U\Sigma V^T, \quad \Sigma = \mathrm{Diag}(\sigma),$$

where $U \in m \times q$ and $V \in n \times q$ with orthogonal columns, $\sigma \in \Re^q$ is the vector of positive singular values with $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_q > 0$ and $q \leq \min\{m, n\}$.

Let $x_+ = \max\{x, 0\}$. Then the solution $S_\tau(G)$ of the above problem is given by

$$S_\tau(G) = U\mathrm{Diag}((\sigma - \mu/\tau)_+)V^T.$$

Recent Algorithms

- Singular Value Thresholding (SVT) algorithm (Cai, et al. '08)

The Tikhonov regularized version of linear constrained nuclear norm minimization:

$$\min_{X \in \Re^{m \times n}} \left\{ \|X\|_* + \frac{1}{2\mu}\|X\|_F^2 \ : \ \mathcal{A}(X) = b \right\}.$$

The SVT algorithm can be expressed as follows:

$$\begin{cases} X^k = S_{\tau^k}(G^k) \\ G^{k+1} = G^k - \delta^k \mathcal{A}^*(\mathcal{A}(X^k) - b), \end{cases}$$

where $\tau^k = 1$ for all k and δ^k is a positive step size.

SVT algorithm is a gradient method applied to the dual problem of the above problem, where each step moves the current dual iterate in the direction of the gradient.

The advantage of SVT algorithm for matrix completion:

1. Selecting a large μ gives a sequence of low-rank iterates.
2. The matrix G^k is sparse for all k because the sparsity pattern of Ω is fixed throughout.

- Fixed Point Continuation (FPC) algorithm (Ma, et al. '08) for solving the nuclear norm regularized linear least squares problem.

FPC algorithm is a matrix extension of the fixed point continuation algorithm proposed by (Hale et al. '07) for an ℓ_1-regularized linear least squares problem.

The FPC algorithm can be expressed as follows:

$$\begin{cases} X^k = S_{\tau^k}(G^k) \\ G^{k+1} = X^k - (\tau^k)^{-1}\mathcal{A}^*(\mathcal{A}(X^k) - b). \end{cases}$$

This algorithm may terminate in $O(1/\epsilon)$ iterations with an ϵ-optimal solution.

Singular Value Decomposition

When the current point X^k of the FPC and SVT algorithms is updated at iteration k, $S_{\tau^k}(G^k)$ is obtained by the SVD of G^k.

Main computational cost of both algorithms lies in computing the SVD of G^k.

For the FPC, a fast Monte Carlo algorithm such as the Linear Time SVD algorithm is used.

For the SVT, the PROPACK (a variant of the Lanczos algorithm) is used.

But we only need singular values that are greater than μ/τ^k.

Even though the algorithms for SVD can choose the number of S.V. to compute, they can not automatically compute only those S.V. we need.

Both FPC and SVT have their own procedure to choose and update the predetermined number of largest S.V. to reduce the computational time.

APG algorithm

Question: Can an algorithm be developed to take an advantage of the SVT algorithm and to have better iteration complexity?

Recently, Beck and Teboulle proposed a fast iterative shrinkage-thresholding algorithm (FISTA) to solve ℓ_1-regularized linear least squares problems.

Accelerated proximal gradient algorithms have an attractive iteration complexity of $O(1/\sqrt{\epsilon})$ for achieving ϵ-optimality.

The APG algorithm can be expressed as follows:

$$\begin{cases} X^k = S_{\tau^k}(G^k) \\ t^{k+1} = \frac{1+\sqrt{1+4(t^k)^2}}{2} \\ Y^{k+1} = X^k + \frac{t^k-1}{t^{k+1}}(X^k - X^{k-1}) \\ G^{k+1} = Y^{k+1} - (\tau^k)^{-1}\mathcal{A}^*(\mathcal{A}(Y^{k+1}) - b). \end{cases}$$

Iteration Complexity

Let $\{X^k\}$, $\{Y^k\}$, $\{t^k\}$ be the sequences generated by APG with $\tau^k = \lambda_{\max}(\mathcal{A}^*\mathcal{A})$ for all k. Then, for any $k \geq 1$, we have

$$F(X^k) - F(X^*) \leq \frac{2\lambda_{\max}(\mathcal{A}^*\mathcal{A})\|X^* - X^0\|_F^2}{(k+1)^2},$$

where $F(X) = \frac{1}{2}\|\mathcal{A}(X) - b\|_2^2 + \mu\|X\|_*$ and X^* is an optimal solution. Hence

$$F(X^k) - F(X^*) \leq \epsilon \quad \text{whenever } k \geq \sqrt{\frac{2\lambda_{\max}(\mathcal{A}^*\mathcal{A})}{\epsilon}}\left(\|X^0\|_F + \chi\right) - 1,$$

where

$$\chi = \begin{cases} \min\{\|b\|_2^2/(2\mu), \|\mathcal{A}^*(\mathcal{A}\mathcal{A}^*)^{-1}b\|_*\} & \text{if } \mathcal{A} \text{ is surjective} \\ \|b\|_2^2/(2\mu) & \text{otherwise.} \end{cases}$$

We have $O(1/\sqrt{\epsilon})$ iteration complexity result.

Key inequality for the proof of the iteration complexity:

$$\begin{aligned} F(X^k) &\leq \frac{\tau^k}{2}\|X^k - G^k\|_F^2 + \mu\|X^k\|_* + \frac{1}{2}\|\mathcal{A}(Y^k) - b\|_2^2 - \frac{1}{2\tau^k}\|\mathcal{A}^*(\mathcal{A}(Y^k) - b)\|_F^2 \\ &= \frac{1}{2}\|\mathcal{A}(Y^k) - b\|_2^2 + \mathrm{Tr}\left((\mathcal{A}^*(\mathcal{A}(Y^k) - b))^T(X^k - Y^k)\right) \\ &\quad + \frac{\tau^k}{2}\|X^k - Y^k\|_F^2 + \mu\|X^k\|_* \left(=: Q_{\tau^k}(X^k, Y^k)\right), \end{aligned}$$

where $G^k = Y^k - (\tau^k)^{-1}\mathcal{A}^*(\mathcal{A}(Y^k) - b)$.

- Comparison of APG and FPC, without using continuation strategy.

The matrix M is randomly generated.

Unknown M			**FPC**			**APG**		
n/r	p	μ	iter	#sv	error	iter	#sv	error
100/10	5666	8.21e-03	7723	61	1.88e-01	655	13	1.06e-03
200/10	15665	1.05e-02	12180	96	2.45e-01	812	12	1.02e-03
500/10	49471	1.21e-02	10900	203	5.91e-01	1132	16	7.63e-04

$$\text{error} := \|X_{\text{sol}} - M\|_F / \|M\|_F.$$

Techniques for Further Acceleration

- Linesearch-like technique:

In practice, too conservative to set $\tau^k = \lambda_{\max}(\mathcal{A}^*\mathcal{A})$ for all k.
To accelerate the convergence, it's desirable to take a smaller value for τ^k by performing a linesearch-like procedure (we call this version as APGL).

$\hat{\tau}_0 = \eta\tau^{k-1}$ with $\eta \in (0, 1)$.

For $j = 0, 1, 2, \ldots,$
  Set $G = Y^k - (\hat{\tau}_j)^{-1}\mathcal{A}^*(\mathcal{A}(Y^k) - b)$, compute $S_{\hat{\tau}_j}(G)$.
  If $F(S_{\hat{\tau}_j}(G)) \leq Q_{\hat{\tau}_j}(S_{\hat{\tau}_j}(G), Y^k)$,
    set $\tau^k = \hat{\tau}_j$, stop;
  else,
    $\hat{\tau}_{j+1} = \min\{\eta^{-1}\hat{\tau}_j, \lambda_{\max}(\mathcal{A}^*\mathcal{A})\}$
  end
end

This technique has another important advantage: τ^k is typically smaller than $\lambda_{\max}(\mathcal{A}^*\mathcal{A})$, and so the number of S.V. of G^k that are greater than μ/τ^k is smaller.

- Continuation technique:

If the parameter μ is larger, then the number of S.V. to be evaluated is smaller. But the target parameter μ is usually chosen to a moderately small number.

This motivates us to use the continuation technique employed in the FPC.

We solve a sequence of the nuclear norm linear least squares problem defined by a decreasing sequence $\{\mu^0, \mu^1, \ldots, \mu^\ell = \mu\}$ with a given finite positive integer ℓ.

When a new problem, associated with μ^{j+1} is to be solved, the approximate solution for the current problem with μ^j is used as the starting point.

- Truncation technique:

For the APG algorithm without any acceleration techniques, X^k is generally not low-rank before the final phase of the algorithm.

But positive S.V. of X^k typically would separate into two clusters with the first cluster having much large mean value than that of the second cluster.

One may view the number of S.V. in the first cluster as a good estimate on the rank of the low-rank optimal solution.

The second cluster of smaller positive S.V. can be attributed to the presence of noise in the given data b.

The second cluster of smaller S.V. can generally be discarded without affecting the convergence of the APG algorithm.

This motivates us to set the second cluster of small positive S.V. to zero when the new iterate is updated.

1. For the SVT algorithm, selecting a large μ gives a sequence of low-rank iterates.

By using the acceleration techniques, the APGL algorithm can give a sequence of low-rank iterates.

2. For the SVT algorithm, The matrix G^k is sparse for all k because the sparsity pattern of Ω is fixed throughout.

By using the acceleration techniques, Y^k in the APG algorithm can keep the low-rank property for all k.

$\mathcal{A}^*(\mathcal{A}(Y^k) - b)$ are sparse because of the sparsity pattern of Ω.

The matrix G^k in the APG algorithm is typically the sum of a low-rank matrix and a sparse matrix.

Hence this property can also make the APG algorithm computationally as attractive as the SVT algorithm.

Numerical Experience on Matrix completion

- Implement APG(L) method in Matlab using PROPACK package to evaluate partial singular value decompositions.

- $\eta = 0.8$ for linesearch-like technique

- For continuation technique: Initially, we set $\mu^0 = \|\mathcal{A}^*(b)\|_2$ and update $\mu^k = \max\{0.7\mu^{k-1}, \mu\}$ with $\mu = 10^{-4}\mu^0$ at iteration k.

- For truncation technique: Choose r^k be the smallest integer such that $\chi_{r^k} \geq 5$ where $\chi_j := \text{mean}(\varrho^k(1:j))/\text{mean}(\varrho^k(j+1:q))$ with $\varrho^k = (\sigma^k - \mu^k/\tau^k)_+$, σ^k being the vector of positive S.V. of G^k, and q being the number of positive components of ϱ^k.
Set $X^{k+1} = U\text{Diag}(\bar{\varrho}^k)V^T$, where $\bar{\varrho}^k$ is obtained from ϱ^k by setting $\bar{\varrho}_i^k = \varrho_i^k$ for $i = 1, ..., r$, and $\bar{\varrho}_i^k = 0$ for $i = r+1, ..., q$.

- Termination Criterion:

$$\frac{\|S^k\|_F}{\tau^{k-1}\max\{1, \|X^k\|_F\}} \leq 10^{-4},$$

where
$S^k = \tau^{k-1}(Y^{k-1} - X^k) + \mathcal{A}^*(\mathcal{A}(X^k) - \mathcal{A}(Y^{k-1})) \in \partial(\frac{1}{2}\|\mathcal{A}(X^k) - b\|_2^2 + \mu\|X^k\|_*)$.

In addition, we also stop the APG(L) algorithm when

$$\frac{\big|\|\mathcal{A}(X^k) - b\|_2 - \|\mathcal{A}(X^{k-1}) - b\|_2\big|}{\max\{1, \|b\|_2\}} < 5 \times 10^{-4}.$$

Noiseless random matrix

• n (we set $m = n$): matrix dimension, r: predetermined rank, p : the number of entries to sample.

Generate the original matrix $M = M_L M_R^T$ (M_L, M_R are $n \times r$ matrices with i.i.d. standard Gaussian entries).

Then select a subset Ω of p elements uniformly at random from $\{(i, j) : i, j = 1, \ldots, n\}$.

$$b = \mathcal{A}(M), \quad \mathcal{A}(M) = M_\Omega.$$

• Measure the accuracy of the computed solution $X_{\rm sol}$ by the relative error defined by:

$$\text{error} \ := \ \|X_{\rm sol} - M\|_F / \|M\|_F.$$

Test Results

- run 5 random instances.

Unknown M				Results			
n	p	r	μ	iter	#sv	time	error
1000	119406	10	1.44e-02	40	10	3.00e+00	2.90e-04
	389852	50	5.39e-02	40	50	1.69e+01	3.08e-04
	569900	100	8.63e-02	48	100	5.02e+01	3.82e-04
5000	597973	10	1.38e-02	51	10	1.56e+01	2.52e-04
	2486747	50	6.10e-02	53	50	1.27e+02	6.21e-04
	3957533	100	1.03e-01	53	100	3.20e+02	3.86e-04
10000	1199532	10	1.36e-02	53	10	2.98e+01	2.98e-04
	4987078	50	5.96e-02	51	50	2.43e+02	2.40e-04
	7960222	100	9.94e-02	55	100	6.73e+02	3.60e-04
20000	2401370	10	1.35e-02	56	10	6.71e+01	3.29e-04
30000	3599920	10	1.35e-02	60	10	1.10e+02	2.01e-04
50000	5998352	10	1.35e-02	61	10	2.14e+02	2.15e-04
100000	12000182	10	1.34e-02	68	10	5.31e+02	1.80e-04

The APGL algorithm was able to solve random matrix completion problems with $m = n = 10^5$ each in less than 10 minutes.

Noise random matrix

• For the noisy random matrix completion problems, the matrix M created by the same way as we did for noiseless case

Corrupted by a noise matrix Ξ, and

$$b = \mathcal{A}(M + \sigma\Xi),$$

where the entries of Ξ are i.i.d. standard Gaussian random variables.

In our experiments, σ is chosen to be

$$\sigma \;=\; 0.1\,\frac{\|\mathcal{A}(M)\|_F}{\|\mathcal{A}(\Xi)\|_F}.$$

• Measure the accuracy of the computed solution $X_{\rm sol}$ by the relative error defined by:

$$\text{error} \;:=\; \|X_{\rm sol} - M\|_F / \|M\|_F.$$

Test Results

- run 5 random instances.

Unknown M				Results			
n	p	r	μ	iter	#sv	time	error
1000	119406	10	1.44e-02	35	10	2.75e+00	4.49e-02
	389852	50	5.39e-02	38	50	1.53e+01	5.51e-02
	569900	100	8.63e-02	44	100	4.23e+01	6.38e-02
5000	597973	10	1.38e-02	44	10	1.52e+01	4.52e-02
	2486747	50	6.10e-02	52	50	1.24e+02	4.97e-02
	3957533	100	1.03e-01	48	100	2.64e+02	5.86e-02
10000	1199532	10	1.37e-02	48	10	3.06e+01	4.52e-02
	4987078	50	5.97e-02	44	50	1.92e+02	4.99e-02
	7960222	100	9.95e-02	53	100	6.30e+02	5.73e-02
20000	2401370	10	1.36e-02	52	10	6.50e+01	4.53e-02
30000	3599920	10	1.35e-02	55	10	1.03e+02	4.53e-02
50000	5998352	10	1.35e-02	57	10	2.00e+02	4.53e-02
100000	12000182	10	1.34e-02	60	10	4.68e+02	4.53e-02

The errors are smaller than the noise level 0.1 and are consistent with (actually more accurate) the theoretical result (Candés and Plan '09).

Two real matrix completion problems

- The Jester joke data set contains 4.1 million ratings for 100 jokes from 73421 users.

(1) `jester-1`: 24983 users who have rated 36 or more jokes;
(2) `jester-2`: 23500 users who have rated 36 or more jokes;
(3) `jester-3`: 24938 users who have rated between 15 and 35 jokes.
We let `jester-all` be the data set obtained by combining all the above data sets.

For each data set, we let M be the original incomplete data matrix such that the i-th row of M corresponds to the ratings given by the i-th user on the jokes.

For each user, we randomly choose 10 ratings from the set of indices for which M_{ij} is given, to form the data vector b.

- The MovieLens data has 3 data sets with the following characteristics.

(1) `movie-100K`: 100,000 ratings for 1682 movies by 943 users;
(2) `movie-1M`: 1 million ratings for 3900 movies by 6040 users;
(3) `movie-10M`: 10 million ratings for 10681 movies by 71567 users.

For each data set, the matrix M is very sparse.

In our experiments, we randomly select about 50% of the ratings given by each user to form the data vector b.

- Since some of the entries in M are missing, we cannot compute the relative error of the estimated matrix X.

Instead, we computed the Normalized Mean Absolute Error (NMAE).

$$\mathrm{NMAE} = \frac{\mathrm{MAE}}{r_{\max} - r_{\min}},$$

where $r_{\min}$ and $r_{\max}$ are lower and upper bounds for the ratings, the Mean Absolute Error (MAE) is defined as

$$\mathrm{MAE} = \frac{1}{|\Gamma|} \sum_{(i,j)\in\Gamma} |M_{ij} - X_{ij}|,$$

Γ is the set of indices for which M_{ij} is given.

For jester joke, ratings $\in [-10, +10]$ ($r_{\min} = -10$, $r_{\max} = 10$).

For MovieLens, ratings $\in \{1, 2, 3, 4, 5\}$ ($r_{\min} = 1$, $r_{\max} = 5$).

Test Results

	m/n	$\|\Gamma\|$, $\|\Omega\|/\|\Gamma\|$	μ	iter	time	NMAE	#sv
jester-1	24983/ 100	1.81e+06, 13.80%	5.76e-01	26	2.74e+01	1.64e-01	88
jester-2	23500/ 100	1.71e+06, 13.75%	5.66e-01	26	2.58e+01	1.65e-01	88
jester-3	24938/ 100	6.17e+05, 40.42%	8.30e-01	32	4.21e+01	1.18e-01	80
jester-all	73421/ 100	4.14e+06, 17.75%	1.06e+00	27	8.62e+01	1.59e-01	89
moive-100K	943/ 1682	1.00e+05, 49.92%	3.21e-01	58	2.61e+00	1.84e-01	1
moive-1M	6040/ 3706	1.00e+06, 49.86%	9.47e-01	61	1.17e+01	1.81e-01	1
moive-10M	71567/ 10674	9.91e+06, 49.84%	2.66e+00	72	2.14e+02	1.55e-01	11

Here, we set 10^{-3} instead of 10^{-4} in the stopping condition.

We can solve the matrix completion problem with dimension 73421×100 within 2 minutes with an NMAE value of 1.59×10^{-1}.

We can also solve the matrix completion problem with dimension 71567×10674 in less than 4 minutes with an NMAE value of 1.55×10^{-1}.

Conclusions & Future Work

1. The accelerated proximal gradient algorithm with a fast method, such as PROPACK, for computing partial singular value decomposition is simple and suitable for solving large-scale matrix completion problems.

2. Three techniques, linesearch-like, continuation, and truncation techniques, have been developed to accelerate the convergence of the original APG algorithm.

3. Numerical results shows the practical efficiency of the APGL algorithm for large-scale matrix completion problems.

4. Can other methods, such as interior point methods, or semismooth Newton methods, developed for ℓ_1-regularized linear least squares problems be extended to solve the nuclear norm linear least squares problems?

5. Can other acceleration techniques be developed for solving large-scale matrix completion problems?

Thank you!

Toh K.-C. and Yun S., An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems.
(PDF file available at http://www.math.nus.edu.sg/ matys/index.html)