An Accelerated Proximal Gradient Algorithm for Nuclear Norm Regularized Least Squares Problem

Sangwoon Yun

Computational Sciences
Korea Institute for Advanced Study

November, 2009
(Joint work with Kim-Chuan Toh (NUS))

Outline

- Matrix Completion
- General Problem Model:
  Nuclear Norm Regularized Least Squares Problem
- Accelerated Proximal Gradient Algorithm
- Techniques for Acceleration/Reducing Computation
- Numerical Experience on Large-Scale Matrix Completion
- Conclusions & Future Work

Matrix Completion

Compressed Sensing: Is it possible to reconstruct a signal accurately from a few observed samples?

Impossible in general, but if the signal is known to be sparse in some basis, 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 $A \in \Re^{p \times n}$ ($p \ll n$) satisfies certain restricted isometry property.

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: Given a sparse matrix where M_{ij} is the rating given by user i on movie j, predict the rating a user would assign to a movie he has not seen, i.e., we would like to infer users preference for unrated movies. (impossible! in general)

The problem is ill-posed. Intuitively, users' preferences depend only on a few factors, i.e., $\mathrm{rank}(M)$ is small.

Thus can be formulated as the low-rank matrix completion problem:

$$\min_{X \in \Re^{m \times n}} \left\{ \mathrm{rank}(X) \,|\, X_{ij} = M_{ij},\ (i,j) \in \Omega \right\}, \quad \text{(NP hard!)}$$

where $\Omega =$ index set of p observed entries.

We assume $m \leq n$ w.l.o.g.

By (Candés & Recht 08, Candés & Tao 09), a random rank-r matrix can be recovered exactly with high probability from a uniform random sample of $p = O(rn\,\mathrm{polylog}(n))$ entries by solving the following convex relaxation:

$$(NNM) \quad \min_{X \in \Re^{m \times n}} \left\{ \|X\|_* := \sum_{i=1}^{m} \sigma_i(X) \mid X_{ij} = M_{ij},\ (i,j) \in \Omega \right\}.$$

where $\sigma_i(X)$'s are singular values of X.

Euclidean distance matrix completion: Given partial distance matrix M, find points $z_1, \ldots, z_n \in \Re^d$ such that $\|z_i - z_j\| = M_{ij}\ \forall\ (i,j) \in \Omega$. Typically want d to be small. A convex relaxation is:

$$\min_{X \in \mathcal{S}^n} \left\{ Tr(X) \mid Tr(A_{ij}X) = M_{ij}^2,\ (i,j) \in \Omega,\ X \succeq 0 \right\}$$

A_{ij} : outer product of $e_i - e_j$.

General Problem Model: Nuclear Norm Regularized Least Squares Problem

Here we first consider a more general NNM problem (Fazel '02):

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

Can also include $X \succeq 0$ if X is symmetric.

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:

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

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:

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

where $\mu > 0$.

This motivates us to consider the following nuclear norm regularized LS problem:

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

where $\mu > 0$ is a given parameter.

The problem (NNM) can be reformulated as an SDP:

$$\min \left\{ \left(\mathrm{Tr} W_1 + \mathrm{Tr} W_2\right)/2 \,|\, \mathcal{A}(X) = b, \quad \begin{pmatrix} W_1 & X \\ X^T & W_2 \end{pmatrix} \succeq 0 \right\}$$

matrix dimension = $m + n$; number of linear constraints = p.

The state-of-the-art interior-point method solvers like CSDP/SDPA/SeDuMi/SDPT3 are not suitable for large problems with $m + n \geq 3000$ or $p \geq 6000$ because the computational cost grows like $O(pn^3 + p^2n^2 + p^3)$ and the memory requirement grows like $O(n^2 + p^2)$.

Recent focus: first-order methods.

Accelerated Proximal Gradient Algorithm

Proximal point mapping of the nuclear norm function

Given G and $\tau > 0$, we consider the proximal point mapping of $\mu\|X\|_*$:

$$S_\tau(G) := \operatorname{argmin}_{X \in \Re^{m\times n}} \frac{\tau}{2}\|X - G\|_F^2 + \mu\|X\|_*.$$

$S_\tau(G)$ can be computed analytically! Let the SVD of G be

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

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

Let $x_+ = \max\{x, 0\}$. Then

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

Recent Algorithms (For brevity, we let $f(X) := \frac{1}{2}\|\mathcal{A}(X) - b\|^2$.)

Singular Value Thresholding (SVT) algorithm (Cai, et al. '08): designed to solve a Tikhonov regularized version of NNM:

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

The SVT algorithm is a gradient ascent method applied to the **dual** of the above problem:

$$X^k = S_{\tau_k}(G^k)$$
$$G^{k+1} = G^k - \delta_k \nabla f(X^k),$$

where $\tau_k = 1$ and δ_k is a positive step size ($\nabla f(X^k) = \mathcal{A}^*(\mathcal{A}(X^k) - b)$).

Advantage of SVT algorithm for matrix completion:

- Selecting a large μ gives a sequence of low-rank iterates X^k.
- The matrix G^k is sparse $\Rightarrow$ matrix-vector multiply is cheap.

Fixed Point Continuation (FPC) algorithm (Ma, et al. 08) for solving the NNRLS problem:

$$X^k = S_{\tau_k}(G^k)$$

$$G^{k+1} = X^k - \tau_k^{-1} \nabla f(X^k).$$

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

It is a matrix extension of the FPC algorithm for ℓ_1-regularized LLS problem proposed by (Hale et al. 07).

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. Hence main computational cost of both SVT and FPC algorithms lies in computing a partial SVD of G^k.

- SVT uses PROPACK (a variant of the Lanczos algorithm);
- FPC uses a fast Monte Carlo algorithm such as the Linear Time SVD algorithm.

Note that we only need singular values larger than μ/τ_k.

- The algorithms can choose the number of s.v. to compute, but cannot automatically compute only those we need.
- Both FPC and SVT have their own procedure to choose and update the predetermined number of largest s.v. to reduce the computation.

APG algorithm

Motivation: to develop an algorithm having the same advantage as the SVT algorithm but has better iteration complexity, efficiency and robustness.

(Beck and Teboulle 08) proposed a fast iterative shrinkage-thresholding algorithm (FISTA) to solve ℓ_1-regularized LLS problems. It belongs to the class of accelerated proximal gradient (APG) algorithms studied earlier by Nesterov, Nemirovski, and others. Recently, (Tseng 08) gave a unified treatment.

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

The APG algorithm we adapted has the following template:

$$
\begin{aligned}
X^k &= S_{\tau_k}(G^k) \\
t_k &= \tfrac{1}{2}(1 + \sqrt{1 + 4t_{k-1}^2}) \\
Y^k &= X^k + \tfrac{t_{k-1}-1}{t_k}(X^k - X^{k-1}) \\
G^{k+1} &= Y^k - \tau_k^{-1}\nabla f(Y^k).
\end{aligned}
$$

Iteration Complexity

Define $F(X) = f(X) + \mu\|X\|_*$ and let X^* be an optimal solution.

Let $\{X^k\}$ be the sequence generated by APG with $\tau_k = \|\mathcal{A}\|^2$. Then,

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

Hence

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

Note: $\|X^*\|_F \leq \|b\|^2/(2\mu)$.

Key inequality for the proof of iteration complexity

Given $\tau_k \geq \|\mathcal{A}\|^2$. At Y^k, consider the following upper quadratic approximation of $F(X)$:

$$\begin{aligned} Q_{\tau_k}(X, Y^k) &= f(Y^k) + \langle \nabla f(Y^k), X - X^k \rangle + \frac{\tau_k}{2}\|X - Y^k\|_F^2 + \mu\|X\|_* \\ &= \frac{\tau_k}{2}\|X - G^k\|_F^2 + \mu\|X\|_* + f(Y^k) - \frac{1}{2\tau_k}\|\nabla f(Y^k)\|_F^2 \end{aligned}$$

where $G^k = Y^k - \tau_k^{-1}\nabla f(Y^k)$. Then

$$F(X) \leq Q_{\tau_k}(X, Y^k) \quad \forall\, X$$

Let $X^k = S_{\tau_k}(G^k) = \operatorname{argmin}_X Q_{\tau_k}(X, Y^k)$. Then

$$F(X^k) \leq Q_{\tau_k}(X^k, Y^k) = \min_X Q_{\tau_k}(X, Y^k)$$

Comparison of APG and FPC, without using continuation strategy

Original rank-r matrix: $M = M_1 M_2^T$, where $M_1, M_2 \in \Re^{n \times r}$ are random matrices with i.i.d. standard Gaussian entries.

Then randomly select a subset of p entries to form the vector b.

Set $\mu = 10^{-4}\|\mathcal{A}^*(b)\|$.

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
500/10	49471	1.21e-02	10900	203	5.91e-01	1132	16	7.63e-04

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

Techniques for Acceleration/Reducing Computation

A. Linesearch-like technique. Setting $\tau_k = \|\mathcal{A}\|^2 \ \forall \ k$ is too conservative. The APG algorithm can be accelerated by taking smaller τ_k by performing a linesearch-like procedure (we call this version as APGL).

$\hat{\tau}_0 = \eta\tau_{k-1}$ with $\eta \in (0,1)$, say $\eta = 0.8$.

For $j = 0, 1, 2, \ldots,$
  Set $G = Y^k - (\hat{\tau}_j)^{-1}\nabla f(Y^k)$, 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; (accept a smaller τ if above inequality holds)
  else,
    $\hat{\tau}_{j+1} = \min\{\eta^{-1}\hat{\tau}_j, \|\mathcal{A}\|^2\}$
  end
end

This technique has another important advantage: τ_k is typically smaller than $\|\mathcal{A}\|^2$, and so the number of s.v. of G^k larger than μ/τ_k is fewer.

B. Continuation technique. If the parameter μ is larger, then the number of s.v. of G^k larger than μ/τ_k is fewer. But the target parameter μ is usually chosen to a moderately small number. This motivates us to use a continuation strategy:

- Solve a sequence of problems corresponding to a decreasing sequence of parameters: $\mu_0 > \mu_1 > \cdots > \mu_\ell = \mu_{\text{target}}$.

  Specifically, we set $\mu_0 = \|\mathcal{A}^*(b)\|$ and update $\mu_k = \max\{0.7\mu_{k-1}, \mu_{\text{target}}\}$ with $\mu_{\text{target}} = 10^{-4}\mu_0$.

- For the new problem associated with μ_{j+1}, the approximate solution for the current problem associated with μ_j is used as the starting point.

C. Truncation technique. For the pure APG algorithm, X^k is initially not low-rank. But the positive s.v. would typically separate into two clusters with the first cluster having much large mean value than the second cluster.

One may use the number of s.v. in the first cluster to estimate the rank of the optimal solution. The second cluster of smaller positive s.v. can be attributed to the presence of noise in the data b, or the fact that X^k has yet to converge to a low-rank optimal solution.

The second cluster of smaller s.v. can usually be discarded without affecting the convergence of the APG algorithm. This motivates us to set the second cluster of small positive s.v. to 0 when the new iterate is updated.

We take $X^{k+1} = U\mathrm{Diag}(e \circ (\sigma - \mu_k/\tau_k)_+)V^T$,
$e = [1, \ldots, 1, 0, \ldots, 0]^T$ is the indicator vector for the clusters.

SVT versus APGL for matrix completion

- (Empirical observation)
  SVT: selecting a large μ gives a sequence of low-rank iterates X^k.

  APGL: with truncation and continuation also gives a sequence of low-rank iterates X^k.

- SVT: G^k is sparse since the sparsity pattern of Ω is fixed.

  APGL: Y^k is low-rank since X^k's are low-rank. The matrix $\nabla f(Y^k)$ is sparse since the sparsity pattern of Ω is fixed. Thus
  $G^k = Y^k - \tau_k^{-1} \nabla f(Y^k)$ is the sum of a low-rank and a sparse matrix $\Rightarrow$ matrix-vector multiply is cheap & able to store very large matrix.

For computing $S_{\tau_k}(G^k)$, the above comparison shows that APGL is computationally as attractive as SVT.

Numerical Experience on Large-Scale Matrix Completion

Implemented APGL in MATLAB running on a 3.2GHz PC with 4G RAM. We use PROPACK to compute partial SVD.

Termination Criterion:

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

where $S^k = \tau^{k-1}(Y^{k-1} - X^k) + \nabla f(X^k) - \nabla f(Y^{k-1}) \in \partial F(X^k)$.

In addition, we also stop the APGL algorithm when

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

Test Results: Noiseless random matrix completion problem (average over 5 random instances)

Unknown M				Results			
n	p	r	μ	iter	#sv	time (secs)	error
10,000	1.2e+06	10	1.36e-02	48	10	2.36e+01	2.62e-04
	5.0e+06	50	5.96e-02	67	50	3.14e+02	1.96e-04
	8.0e+06	100	9.94e-02	67	100	8.61e+02	2.69e-04
50,000	6e+06	10	1.35e-02	63	10	1.76e+02	1.75e-04
100,000	12e+06	10	1.34e-02	69	10	4.63e+02	2.16e-04

The APGL algorithm solves the random matrix completion problems with $m = n = 10^5$ in less than 8 minutes on the average.

Test Results: Noisy random matrix completion problem (average over 5 random instances)

Original rank-r matrix M is generated as before. Then corrupt M by a noise matrix Ξ with i.i.d standard Gaussian entries. Specifically, set

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

In our experiments,

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

Unknown M				Results			
n	p	r	μ	iter	#sv	time (secs)	error
10,000	1.2e+06	10	1.37e-02	45	10	2.89e+01	5.01e-02
	5.0e+06	50	5.97e-02	62	50	2.89e+02	5.01e-02
	8.0e+06	100	9.95e-02	64	100	8.17e+02	5.81e-02
50,000	6.0e+06	10	1.35e-02	55	10	2.02e+02	4.51e-02
100,000	12e+06	10	1.34e-02	59	10	5.50e+02	4.52e-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

Jester joke data set contains 4.1 million ratings on 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 35jokes.
(4) `jester-all`: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:
(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.

We randomly select about 50% of the ratings given by each user to form the vector b.

Since some of the entries in M are missing, we cannot compute the relative error as before. Instead, we compute the Normalized Mean Absolute Error (NMAE):

$$\text{NMAE} = \frac{\text{MAE}}{\text{rating range}},$$

where

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

Γ is the index set of given ratings.

Test Results

	m/n	$\|\Gamma\|$	iter	time (secs)	NMAE	#sv
j-1	24983/ 100	1.81e+06	50	7.15e+01	1.89e-01	79
j-2	23500/ 100	1.71e+06	50	6.86e+01	1.88e-01	79
j-3	24938/ 100	6.17e+05	47	6.24e+01	1.94e-01	78
j-all	73421/ 100	4.14e+06	51	2.18e+02	1.91e-01	79
m-100K	943/ 1682	1.00e+05	100	7.39e+00	2.05e-01	5
m-1M	6040/ 3706	1.00e+06	89	2.45e+01	1.76e-01	5
m-10M	71567/ 10674	9.91e+06	100	2.02e+02	1.64e-01	5

Here, we set the maximum number of iterations allowed in the algorithm to 100.

- The 73421×100 problem is solved in 4 minutes: NMAE $= 1.91 \times 10^{-1}$.
- The 71567×10674 problem is solved in 4 minutes: NMAE $= 1.64 \times 10^{-1}$.

Conclusions & Future Work

- The APG algorithm with a fast method, such as PROPACK, for computing partial SVD is simple and suitable for solving large-scale matrix completion problems.
- Three techniques: linesearch-like, continuation, and truncation, have been incorporated to accelerate the convergence of the original APG algorithm.
- Numerical results shows the practical efficiency of the APGL algorithm for large-scale matrix completion problems. Able to solve random MC problems with dimension $10^5 \times 10^5$ in less than 10 minutes.
- We expect first-order methods to work well mainly for problems where $\|\mathcal{A}\|$ is small. Can other methods, such as interior point or semismooth Newton methods, be developed to solve the nuclear norm least squares problems for which $\|\mathcal{A}\|$ is large.

Thank You!

Toh K.-C. and Yun S., An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems.