Matrix Optimization: Searching between the First and Second Order Methods

Defeng Sun

Department of Mathematics and Risk Management Institute
National University of Singapore

May 17, 2011

The Matrix Optimization Problem

We consider the "standard" matrix optimization problem (MOP) and its dual:

$$(\mathrm{P}) \qquad \begin{array}{ll} \min & \langle c, x\rangle + f(x) \\ \text{s.t.} & \mathcal{A}x = b \end{array}$$

and

$$(\mathrm{D}) \qquad \begin{array}{ll} \max & \langle b, y\rangle - f^*(z) \\ \text{s.t.} & \mathcal{A}^*y - c = z\,, \end{array}$$

where $\mathcal{X}$ is the Cartesian product of several finite dimensional real matrix spaces, symmetric or non-symmetric,

$\mathcal{A}^*$ is the adjoint of the linear operator $\mathcal{A} : \mathcal{X} \to \Re^m$, $c \in \mathcal{X}$, $b \in \Re^m$,

$f : \mathcal{X} \to (-\infty, \infty]$ is a closed proper convex function with its Fenchel conjugate f^*.

The Fenchel conjugate of f is defined by

$$f^*(z) := \sup_{x \in \mathcal{X}} \{\langle z, x \rangle - f(x)\}.$$

In standard linear programming, $f(x) = \delta_{\Re^n_+}(x)$, the indicator function over $\Re^n_+$ and $f^*(x) = \delta_{(-\Re^n_+)}(x)$.

In semidefinite programming (SDP), $f(x) = \delta_{\mathcal{S}^n_+}$, the indicator function over $\mathcal{S}^n_+$ and $f^*(x) = \delta_{(-\mathcal{S}^n_+)}(x)$.

Desirable Properties of f

We need conditions on f. Specifically, we require

- The Moreau-Yosida regularization of f

$$\psi_f(x) := \min_{z \in \mathcal{X}} \left\{ f(z) + \frac{1}{2}\|z - x\|^2 \right\}$$

has a closed form solution, denoted by $P_f(x)$.

- We can easily compute the directional derivative of

$$\nabla \psi_f(x) = x - P_f(x).$$

- The function $\nabla \psi_f$ is (strongly) semismooth.

Let us first look at one simple example with nonsymmetric matrices:

$$\min_{y \in \Re^k} \quad \|A_0 - \sum_{i=1}^k y_i A_i\|_2 \,, \tag{1}$$

where A_i are m by n matrices, $\|\cdot\|_2$ is the spectral (operator) norm of matrices (the largest singular value).

Use $\|\cdot\|_*$ to denote the nuclear norm (the sum of all singular values) and B_*^1 to denote the unit nuclear norm ball.

We can equivalently write (1) in the form of (D):

$$\begin{aligned} \max \quad & \langle 0, y \rangle - \|Z\|_2 \\ \text{s.t.} \quad & \mathcal{A}y - A_0 = Z \end{aligned}$$

and the corresponding form of (P):

$$\begin{aligned} \min \quad & \langle A_0, X \rangle + \delta_{B_*^1}(X) \\ \text{s.t.} \quad & \mathcal{A}^* X = 0\,. \end{aligned}$$

Why bother?

Note that we can write $t \geq \|X\|_2$ (here, $X \in \Re^{m\times n}$) equivalently as

$$\mathcal{S}^{m+n} \ni \begin{bmatrix} tI_m & X \\ X^T & tI_n \end{bmatrix} \succeq 0\,.$$

Thus, (1) is equivalent to an SDP problem:

$$\begin{array}{ll} \min & t \\ \text{s.t.} & X + \sum_{i=1}^{k} y_i A_i = A_0\,, \\ & \begin{bmatrix} tI_m & X \\ X^T & tI_n \end{bmatrix} \succeq 0. \end{array} \tag{2}$$

Actually, most of the MOPs we are considering are "SDP representable".

However, there are two issues to use the SDP representation (2):

- Can we solve these SDPs when m or n is not small?

- Is it necessary to increase the matrix dimension from mn to $\frac{1}{2}(m+n)^2$?

—— No one is likely to do so if $m = 1$ or $n = 1$ because in this case we can solve a second order cone programming (SOC) problem instead of an SDP problem?

—— How about $m \ll n$ or $n \ll m$?

—— Shall we do so if $m = n$?

An application in finance

Let us consider the widely used optimization model in the finance industry and many others:

$$\begin{aligned} \min \quad & \|D^{-1/2}(X-G)D^{-1/2}\|_F \\ \text{s.t.} \quad & \mathrm{diag}(X) = e\,, \\ & X \succeq 0\,, \end{aligned} \tag{3}$$

where G is an estimated matrix which often fails to be positive semi-definite, D is a symmetric and positive definite matrix (weight matrix), and e is the vector of all ones.

This problem is known as the nearest correlation matrix (NCM) problem, a terminology coined by Nick Higham in 2002. It is used in many situations: stress testing, VaR computation, asset pricing ...

SDP cone + SOC reformulation

One may write the NCM as a symmetric cone programming with both SDP cone and SOC cone constraints (assuming $D = I$, the identity matrix for notational convenience):

$$\begin{aligned} \min \quad & t \\ \text{s.t.} \quad & \operatorname{diag}(X) = e\,, \\ & y + \operatorname{svec}(X) = \operatorname{svec}(G)\,, \\ & X \succeq 0\,, \quad t \geq \|y\|_2\,. \end{aligned}$$

This is a perfect formula for employing modern interior point methods (IPMs).

n	Time (secs)	Iters
30	1.4	15
40	3.2	15
50	6.0	15
60	13.2	16
70	24.4	15
80	44.3	15
90	102.0	19
100	142.6	16

Table 1: Numerical results for SDPT3

For $n = 110$, it shows Out of Memory [Dell Laptop: 2.99 GB RAM].

The reason is simple: each step we need to store an m by m matrix at least. Here m is the number of equations

$$m = n + 1 + n(n + 1)/2.$$

For $n = 110$, we have $m = 6216$.

One may buy a better Laptop or PC. But even so in each step, the computational cost is

$$O(m^3) = O(n^6).$$

For large n, we will just feed wrong problems to IPMs.

First order methods

In optimization, we always look at the dual when we find a problem difficult to solve.
Rewrite the NCM as

$$\begin{aligned} \min \quad & \frac{1}{2}\|X - G\|_F^2 \\ \text{s.t.} \quad & \mathrm{diag}(X) = e\,, \\ & X \succeq 0\,, \end{aligned} \tag{4}$$

Then the dual of the NCM turns to be an unconstrained problem:

$$\begin{aligned} \max \quad & -\theta(y) := -\left[\frac{1}{2}\|\Pi_{\mathcal{S}_+^n}(G + \mathrm{Diag}(y))\|^2 - \langle e, y\rangle - \frac{1}{2}\|G\|^2\right] \\ \text{s.t.} \quad & y \in \Re^n\,, \end{aligned}$$

where $\Pi_{\mathcal{S}^n_+}(X)$ is the unique optimal solution (projection) to

$$\begin{array}{ll} \min & \frac{1}{2}\|Y - X\|_F^2 \\ \text{s.t.} & Y \in \mathcal{S}^n_+ \, . \end{array}$$

The convex function θ is continuously differentiable with

$$\nabla\theta(y) = \mathrm{diag}(\Pi_{\mathcal{S}^n_+}(G + \mathrm{Diag}(y))) - e, \quad y \in \Re^n.$$

Moreover, $\nabla\theta(\cdot)$ is globally Lispchitz continuous with modulus one, i.e.,

$$\|\nabla\theta(y) - \nabla\theta(z)\| \leq \|y - z\| \quad \forall\, y, z \in \Re^n\,.$$

To compute $\theta(y)$ and $\nabla\theta(y)$, one only needs to know how to compute $\Pi_{\mathcal{S}^n_+}(X)$.

Let $X \in \mathcal{S}^n$ have the following spectral decomposition[1]

$$X = P\Lambda P^T,$$

where Λ is the diagonal matrix of eigenvalues of X and P is a corresponding orthogonal matrix of orthonormal eigenvectors.

Then

$$X_+ := \Pi_{\mathcal{S}^n_+}(X) = P\Lambda_+ P^T.$$

[1] Use the divide and conquer algorithm, which is much faster than the shifted QR decomposition based algorithm.

Immediately, one will try the following projected gradient (PG) method:

$$y^{k+1} := y^k - \nabla\theta(y^k) = y^k - [\mathrm{diag}(\Pi_{\mathcal{S}^n_+}(G + \mathrm{Diag}(y^k))) - e].$$

In 2007, Marc Teboulle suggested to us the accelerated projected gradient (APG) method ($x^0 = z^0 = y^0$):

$$\begin{cases} z^{k+1} = z^k - \nabla\theta(y^k); \\ x^{k+1} = (1 - 2/(k+2))x^k + 2/(k+2)z^k; \\ y^{k+1} = [1 - 2/((k+1)+2)]x^{k+1} + 2/((k+1)+2)z^{k+1}. \end{cases}$$

In 2011, He et al considered the augmented Lagrangian alternating direction method (ADM) by writing the NCM as:

$$\begin{aligned} \min \quad & \frac{1}{2}\|X-G\|_F^2 + \frac{1}{2}\|Y-G\|_F^2 \\ \text{s.t.} \quad & X - Y = 0, \\ & \mathrm{diag}(Y) = e\,, \\ & X \succeq 0\,. \end{aligned}$$

Testing Case 1): Three first order algorithms are tested for a perturbed true correlation matrix: G_{true} is a 1000 by 1000 true correlation matrix and E is a symmetric random matrix with elements in $[-1, 1]$:

$$E = \texttt{randn(1000); E = triu(E) + triu(E,1)'}$$

and set

$$G := 90\% \times G_{\text{true}} + 10\% \times E$$

with its all diagonal being ones. For PG and APG methods, the residue represents the primal feasibility. So, the residue should be at least below 10^{-4}.

Algorithm	Time (secs)	Iters	Residue
PG	124.0	95	9.5×10^{-5}
APG	125.0	93	9.2×10^{-5}
ADM	51.0	36	9.3×10^{-5}

Table 2: Results for PG, APG, and ADM with $\varepsilon = 10^{-4}$

Algorithm	Time (secs)	Iters	Residue
PG	190	145	9.5×10^{-7}
APG	225	168	9.7×10^{-7}
ADM	82	58	9.5×10^{-7}

Table 3: Results for PG, APG, and ADM with $\varepsilon = 10^{-6}$

Tables 2 and 3 show that all the tested first order methods work well, in particular ADM [By introducing line searches to PG and APG methods, one can improve the performance of these two algorithms].

Testing Case 2): To see the robustness of the first order methods, set

$$G := \texttt{rand(1000,1000)}, \quad \texttt{G = G + G'}$$

with its diagonal matrices to be ones.

Algorithm	Time (secs)	Iters	Residue
PG	1130.0	1000	3.5×10^{-2}
APG	1120.0	1000	3.6×10^{-4}
ADM	305.0	257	1.0×10^{-4}

Table 4: Results for PG, APG, and ADM with $\varepsilon = 10^{-4}$

Algorithm	Time (secs)	Iters	Residue
PG	1130.0	1000	3.5×10^{-2}
APG	1120.0	1000	3.6×10^{-4}
ADM	515.0	434	9.7×10^{-7}

Table 5: Results for PG, APG, and ADM with $\varepsilon = 10^{-6}$

Tables 4 and 5 show that the performance of PG and APG worsens a lot while ADM does okay.

Testing Case 3): The weighted case: G is the same as in Case 1) but this time we set the weight matrix D to be:

$$D := \quad \texttt{diag(rand(1000,1))} \ .$$

Algorithm	Time (secs)	Iters	Residue
PG	>1000	1000	$2.9 \times 10^{+0}$
APG	>1000	1000	5.6×10^{-2}
ADM	>1000	1000	1.8×10^{-1}

Table 6: Results for PG, APG, and ADM with $\varepsilon = 10^{-4}$

We have seen for the NCM: IPMs can be pretty robust for small n while the first order methods can only deal with easy cases.

Any other possibility other than the IPMs and first order methods?

Note that the dual of the NCM is:

$$F(y) := \nabla\theta(y) = \mathrm{diag}(\Pi_{\mathcal{S}^n_+}(G + \mathrm{Diag}(y))) - e, \quad y \in \Re^n.$$

The functions F is strongly semismooth as $\Pi_{\mathcal{S}^n_+}$ is [Sun and Sun, 02]. That is, F is directionally diff. at y and

$$F(y+h) - F(y) - \partial F(y+h)h = O(\|h\|^2)\,.$$

Semismooth Newton-CG method

Qi and Sun [06] considered the following Semismooth Newton-CG method:

$$F(y^k) + W_k(y^{k+1} - y^k) \approx 0,$$

where W_k is any element from Clarke's generalized Jacobian $\partial F(y^k)$.

To get W_k computed would require $O(n^4)$ flops. So the exact semismooth Newton method will not be efficient.

However, Qi and Sun shows that $\partial F(y^*)$ are symmetric and positive definite as the NCM is primal non-degenerate (LICQ holds). That's the reason to apply a number of conjugate gradient (CG) steps to the semismooth Newton system.

Algorithm	Time	Iters	CGs	Residue
ADM (case 1)	82.0	58		9.5×10^{-7}
Newton-CG	11.0	6	12/6	6.0×10^{-8}
ADM (case 2)	515.0	434		9.7×10^{-7}
Newton-CG	14.0	9	29/9	6.5×10^{-7}
ADM (case 3)	>1000.0	1000		1.8×10^{-1}
Newton-CG	30.0	21	94/21	6.7×10^{-8}

Table 7: Results for ADM and semismooth Newton-CG method with $\varepsilon = 10^{-6}$

As one can see the semismooth Newton-CG method[2] for solving the NCM is robust and fast and the number of CGs used in each iteration of the semismooth Newton-CG method is really small ranging from 2 to 5.

Even a rough approximation to Newton's direction can be extremely helpful.

What can we say about general matrix optimization problems?

[2] NAG `http://www.nag.co.uk/` has both the C and Fortran versions.

Let us start with

$$\begin{aligned} \text{(P)} \quad & \max \quad \langle C, X \rangle \\ & \text{s.t.} \quad \mathcal{A}(X) = b, \quad X \succeq 0, \end{aligned}$$

where $\mathcal{A} : \mathcal{S}^n \to \Re^m$ is a linear map.
The dual problem of (P) is

$$\text{(D)} \quad \min \left\{ b^T y \mid \mathcal{A}^* y - C \succeq 0 \right\},$$

where $\mathcal{A}^* : \Re^m \to \mathcal{S}^n$ is the adjoint of $\mathcal{A}$.

Given a penalty parameter $\sigma > 0$, the augmented Lagrangian function for problem (D) is defined as

$$L_\sigma(y, X) = b^T y + \frac{1}{2\sigma}\big(\|\Pi_{\mathcal{S}^n_+}(X - \sigma(\mathcal{A}^* y - C))\|^2 - \|X\|^2\big),$$

where $(y, X) \in \Re^m \times \mathcal{S}^n$ and for any $X \in \mathcal{S}^n$.

The augmented Lagrangian function is continuously differentiable. For any given $X \in \mathcal{S}^n_+$, we have

$$\nabla_y L_\sigma(y, X) = b - \mathcal{A}\Pi_{\mathcal{S}^n_+}(X - \sigma(\mathcal{A}^* y - C)).$$

For given $X^0 \in \mathcal{S}^n$, $\sigma_0 > 0$, and $\rho > 1$, the augmented Lagrangian method for solving problem (D) and its dual (P) generates sequences $\{y^k\} \subset \Re^m$ and $\{X^k\} \subset \mathcal{S}^n$ as follows:

$$\begin{cases} y^{k+1} \approx \arg\min\limits_{y \in \Re^m} L_{\sigma_k}(y, X^k), \\ X^{k+1} = \Pi_{\mathcal{S}^n_+}(X^k - \sigma_k(\mathcal{A}^* y^{k+1} - C)), \quad k = 0, 1, 2, \ldots \\ \sigma_{k+1} = \rho\sigma_k \text{ or } \sigma_{k+1} = \sigma_k, \end{cases}$$

The augmented Lagrangian method for convex problems is a gradient ascent method applied to the corresponding augmented Lagrangian dual problems

$$\max_{X \in \mathcal{S}^n} \psi_\sigma(X) := \inf_{y \in \Re^m} L_\sigma(y, X) = L_\sigma(y(X), X)\,.$$

But, recent studies [Sun et al, 07] show that under the constraint nondegenerate conditions for (P) and (D) [LICQs], the augmented Lagrangian method for solving SDPs is actually an approximate semismooth Newton method.

Inner subproblems solving

Use the semismooth Newton-CG method for solving inner subproblem
we need to solve

$$\nabla_y L_{\sigma_k}(y, X^k) = b - \mathcal{A}\Pi_{\mathcal{S}^n_+}(U^k(y)) = 0\,.$$

where $U^k(y) := X^k - \sigma_k(\mathcal{A}^* y - C)$.
At a current iterate y, we solve a semismooth Newton equation by a CG method:

$$\mathcal{H}_y := \sigma_k \mathcal{A}\Pi'_{\mathcal{S}^n_+}(U^k(y))\mathcal{A}^*, \quad \mathcal{H}_y \Delta y = -\nabla_y L(y, X^k).$$

Practical Newton-CG augmented Lagrangian method [SDPNAL]

- Solve $\mathcal{H}_y \Delta y = \mathrm{rhs}$ by CG with a diagonal preconditioner.

  Stop when `relative-residual` ≤ 0.01.

- Stop the inner iteration when $\|\nabla_y L_{\sigma_k}(y^k, X^k)\| \leq 0.2\|X^{k+1} - X^k\|$.

  [Zhao, Sun, Toh, 10].

Comments on numerical results for SDPNAL:

want: rel-err $= \max\left\{\frac{\|R_p\|}{1+\|b\|}, \frac{\|R_d\|}{1+\|C\|}, \frac{\langle X,Z\rangle}{1+|\langle C,X\rangle|+|b^T y|}\right\} \leq 10^{-6}$.

PC: Intel Xeon 3.2GHz with 4G RAM, MATLAB

SDPNAL can be efficient as the theory predicted when the primal and dual non-degeneracies hold at the solutions. For example: for θ : theta162 $(m = 127600,\ n = 800)$, SDPNAL needs 17 outer iterations with total computing time of 173 seconds.

Another example: for 1zc.2048 $(m = 39425,\ n = 2048)$, SDPNAL needs 13 outer iterations with total computing time of 45 minutes and 16 seconds.

On the other hand, when the primal and dual non-degeneracies fail to hold, SDPNAL can perform poorly. For example, `2dc.512` $(n = 512)$, SDPNAL spends 2 hours 25 minutes and 15 seconds to only get a relative error 1.1×10^{-4}.

As a general solver, SDPNAL currently does not give up the search for a direction better than a gradient direction even the primal and dual degeneracies are detected. This can be costly and unnecessary if one knows that Newton's direction is not a good choice or difficult to approximate. Future work on these degenerate problems needs to be done.

SDPNAL can be downloaded from
`http://www.math.nus.edu.sg/ mattohkc/SDPNAL.html`.

Final remarks

• Nonsymmetric matrix problems need to be treated in their own formats.

• To exploit Newton's direction can be beneficial when non-degeneracies hold. $1+\varepsilon$ order methods can perform very well when the first and second order ones do no work efficiently.

• Variational analysis, in particular non-smooth analysis, can guide us in designing efficient algorithms.

• Degenerate programs call for new theory.