Preconditioning with Matrix Factorization

Mike Phulsuksombati

June 5, 2014

Abstract

Pietsch factorization and Grothendieck factorization are the two landmark theorems in modern functional analysis. They were first introduced to the numerical linear algebra community by the work of [Tro09] in the column subset selection problem, which seeks to extract from a matrix a column submatrix that has lower spectral norm. Despite their broad application in functional analysis, the application of both factorization techniques are still not well explored in numerical linear algebra literatures.

This project presents a novel method to preconditioning a matrix based on matrix factorizations—Pietsch factorization and Grothendieck factorization. Both methods factor the matrix in term of diagonal matrices and a new matrix of which the spectral norm is bounded above by some constant. These factorizations can be computed efficiently via solving convex optimization problems. Our result shows that the inverse of the diagonal matrices obtained can be used as a preconditioner to diagonally scale the original matrix so that it has a lower condition number.

1 Introduction

In this section, we introduce the Pietsch factorization and Grothendieck factorization and show how to turn them into convex programs. In section 2, we show how both factorizations can be used to construct a preconditioner. Section 3 presents the test result against other precondition techniques in preconditioning a conjugate gradient (CG) method. We conclude in section 4 with a few recommendations and directions for future research.

Before we continue, let us instate some notation. We use $D$ to denote a diagonal matrix and $\alpha$ to denote a positive constant. We write $\|\cdot\|$ for the spectral norm and $\kappa(\cdot)$ for the condition number.

1.1 Pietsch Factorization via Convex Optimization

We can produce a Pietsch factorization by solving a convex programming problem.
Theorem 1.1. (Pietsch Factorization) There exists a factorization $B = TD$ satisfies $\|T\| \leq \alpha$ for some $\alpha \in \mathbb{R}_+$ if and only if $D$ satisfies

$$\lambda_{\text{max}}(B^TB - \alpha^2D^2) \leq 0$$

where $B \in \mathbb{R}^{m \times s}$, $T \in \mathbb{R}^{m \times s}$ and $D \in \mathbb{R}_+^{s \times s}$ is diagonal with $\text{Tr}(D^2) = 1$.

Recall that the maximum eigenvalue is a convex function on the space of Hermitian matrices, so it can be minimized in polynomial time. We are led to consider the convex program with variable $F$

$$\begin{align*}
\text{minimize} & \quad \lambda_{\text{max}}(B^TB - \alpha^2F^2) \\
\text{subject to} & \quad \text{Tr}(F) = 1, \ F \text{ diagonal, and } F \succeq 0.
\end{align*}$$

Now, if $F$ is a feasible point with a nonnegative objective value, we can factorize

$$B = TD \quad \text{with} \quad D = F^{\frac{1}{2}}, \ T = BD^{-1}, \ \text{and } \|T\| \leq \alpha.$$ 

1.2 Grothendieck factorization via Convex Optimization

Similarly, we can produce a Grothendieck factorization by solving a convex program.

Theorem 1.2. (Grothendieck Factorization) For a Hermitian matrix $G \in \mathbb{R}^{s \times s}$, there exists a factorization $G = DTD$ satisfies $\|T\| \leq \alpha$ for some $\alpha \in \mathbb{R}_+$ if and only if $D$ satisfies

$$\lambda_{\text{max}}\begin{bmatrix}
-\alpha D^2 & G \\
G & -\alpha D^2
\end{bmatrix} \leq 0$$

where $T \in \mathbb{R}^{s \times s}$ and $D \in \mathbb{R}_+^{s \times s}$ is diagonal with $\text{Tr}(D^2) = 1$.

We are led to consider the convex program with variable $F$

$$\begin{align*}
\text{minimize} & \quad \lambda_{\text{max}}\begin{bmatrix}
-\alpha F & G \\
G & -\alpha F
\end{bmatrix} \\
\text{subject to} & \quad \text{Tr}(F) = 1, \ F \text{ diagonal, and } F \succeq 0.
\end{align*}$$

If $F$ is a feasible point with a nonnegative objective value, we can factorize

$$G = DTD \quad \text{with} \quad D = F^{\frac{1}{2}}, \ T = D^{-1}GD^{-1}, \ \text{and } \|T\| \leq \alpha.$$ 

1.3 Entropic Mirror Descent

Programs (1) and (2) can be framed as nonsmooth convex problems over the probability simplex. We can efficiently solve these problems by Entropic Mirror Descent (EMD) [BT03], which is designed specifically for this class of problems. The EMD algorithm requires $O(\alpha^4)$ iterations to deliver a solution with constant precision. In practice, far fewer iterations suffice. The algorithm is given in the appendix.
2 Preconditioning with Matrix Factorization

2.1 Precondition effect

From experiments on matrices of various kinds, we observe that the condition number of matrices $T$ from both factorizations are always less than or equal to the condition numbers of the original matrix. This is not an obvious result from the statement of the theorems because there is nothing that implies such a thing about condition number.

![Effect of $\alpha$ on condition number](image1)

(a) Pietsch factorization  
(b) Grothendieck factorization

Figure 1: Condition number of $T$ from the factorization compared with original matrix $B \in \mathbb{R}^{50 \times 50}$

Figure 1 shows that there exists a range of $\alpha$ such that the condition number of the original matrix $B$ is less than the condition number of $T$ from the factorization. However, if $\alpha$ is too small we see that such factorization does not exist and if $\alpha$ is large enough the condition number of $T$ will be the same as the condition number of $B$, and $D$ will be a scaling of identity matrix. Hence, we propose a conjecture that this holds true for any matrix.

**Conjecture 1.** For the factorization $B = T_1D$ where program (1) is feasible and the factorization $G = DT_2D$ where program (2) is feasible, we have

\[ \kappa(T_1) \leq \kappa(B) \]
\[ \kappa(T_2) \leq \kappa(G) \]

From our simulation, we have not find any counterexample for this conjecture yet.

2.2 Finding Preconditioner

We can use Pietsch factorization and Grothendieck factorization to find the preconditioner of the matrix. Consider the simple CG method example of solving $Ax = b$. Using Pietsch
factorization for some value of $\alpha$ such that program (1) is feasible, we can factorize $A$ as $A = TD$. If the condition number of $T$ is small, we can precondition $A$ with $D^{-1}$ and solve the equation indirectly as

$$AD^{-1}Dx = T(Dx) = b.$$  

Note that both factorizations also preserve the positive definiteness of the matrix as well as its sparsity. This is because the factor $T$ is the original matrix diagonally scaled by $D^{-1}$ where all the diagonal entries is positive.

### 2.3 Finding optimal $\alpha$

In a general setting, a target value for $\alpha$ is not available. Observe that the optimal $\kappa(T)$ happens at the point before the factorization is not feasible. Thus, to get the optimal value of $\kappa(T)$ we need to find a minimum $\alpha$, denoted by $\alpha^*$, such that the factorization is feasible. This can be done easily using a bisection method on $\alpha$. However, this is computationally expensive since it involves solving a number of factorizations. Hence, we propose that for square matrix $B \in \mathbb{R}^{n \times n}$ the $\alpha^*$ can be estimated with the following proposition:

**Proposition 2.1.** For Pietsch factorization and Grothendieck factorization on square matrix $B \in \mathbb{R}^{n \times n}$, $\alpha^*$ such that $\kappa(T)$ is minimized can be estimated as

$$\alpha_{\text{pietsch}}^* \approx n \sqrt{\pi \|B\|}$$

$$\alpha_{\text{grothendieck}}^* \approx n \|B\|$$

![Figure 2: Prediction vs. true optimal $\alpha$ on randomly generated matrices](image)

(a) Pietsch factorization  
(b) Grothendieck factorization

The result from this estimate is plotted in figure 2 and figure 3. We can see that our estimate is slightly greater than the actual optimal $\alpha$. This is to prevent the programs from becoming infeasible.

With this estimate, the factorization such that $\kappa(T)$ is as close to the optimum as possible can be directly computed. In other word, we can compute the preconditioner of the matrix with these factorizations without having to worry about finding $\alpha^*$. 

4
3 Testing Result on Conjugate Gradient Method

The performance of the Pietsch factorization (PIE) and Grothendieck factorization (GRO) is tested against other diagonal scaling techniques—asymmetric scaling (ASY), symmetric scaling (SYM), and Jacobian scaling (JCB), in preconditioning a CG method with $Bx = c$. The original matrix is denoted by (ORG).

We use the standard minimizing condition number program described in [Boy+94], which can be converted to solving generalize eigenvalue problem (GEVP), to benchmark the factorizations performance in reducing the condition number.

For each $n = 50, 60, 70$, we consider 10 sample of symmetric positive definite matrix $B \in \mathbb{R}^{n \times n}$ and vector $x \in \mathbb{R}^n$ generated with i.i.d Gaussian entries. Vector $c$ is produced by setting $c = Bx$.

We compare condition number reduced ratio defined as $\frac{\kappa(\text{method}) - \kappa(B)}{\kappa(B)}$, the precondition time, solving time, and total running time. Here, we present the result from case $n = 50$. The full result is presented in table 2 in the appendix.

<table>
<thead>
<tr>
<th></th>
<th>ORG</th>
<th>ASY</th>
<th>SYM</th>
<th>JCB</th>
<th>PIE</th>
<th>GRO</th>
<th>GEVP</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\kappa$ reduced ratio</td>
<td>0</td>
<td>0.1076</td>
<td>0.1238</td>
<td>0.1367</td>
<td>0.1818</td>
<td>0.2286</td>
<td>0.4245</td>
</tr>
<tr>
<td>Precon Time</td>
<td>0</td>
<td>0.00026</td>
<td>0.00052</td>
<td>0.00017</td>
<td>0.32929</td>
<td>1.32571</td>
<td>2.95389</td>
</tr>
<tr>
<td>Solve Time</td>
<td>0.02541</td>
<td>0.01308</td>
<td>0.02080</td>
<td>0.01288</td>
<td>0.01223</td>
<td>0.01239</td>
<td>0.01213</td>
</tr>
<tr>
<td>Total Time</td>
<td>0.02541</td>
<td>0.01334</td>
<td>0.02132</td>
<td>0.01305</td>
<td>0.34150</td>
<td>1.3381</td>
<td>2.96602</td>
</tr>
</tbody>
</table>

Table 1: Average performance on 10 samples of symmetric positive definite matrix $B \in \mathbb{R}^{50 \times 50}$

As expected, simple equilibrium techniques (ASY, SYM, and JCB) perform better than optimization based preconditioners (PIE, GRO, and GEVP) in term of precondition speed, but spend more time solving the CG method. However, the equilibrium techniques win on the total time. Thus, for solving the CG method, preconditioning with the two factorizations is not recommended.
However, in some application that the condition number reduction must be guaranteed such as in minimizing the round-off error in matrix process such as QR factorization or LU factorization, Pietsch factorization and Grothendieck factorization provide the new alternatives. Despite their speed, the equilibrium techniques are not guaranteed to reduce the condition number and possibly increase in unusual case such as in Vandemonde matrix. The two factorizations provide the faster way of reducing condition number compared to GEVP while the reduction is guaranteed by conjecture 1 (if it holds).

4 Conclusion

The results of this project indicate that Pietsch factorization and Grothendieck factorization can serve as an alternative way to precondition a matrix in the case that the reduction of condition number must be gauranteed. The primary novelty in this work are

- We prove by experiment that the condition number of $T$ from the factorization is less than or equals to the original matrix.
- How to estimate $\alpha^*$? This allows to use both factorizations to precondition matrix.

4.1 Future work

These preliminary results raise a number of interesting questions and suggest a number of lines for future work

- Mathematical proof for conjecture 1.
- Can we find better upper bound for $\alpha^*$? Can we use $\alpha^*$ to characterize matrix (ie. like $\lambda_2/\lambda_1$)?
- Relationship of $\alpha^*$ to the $(\infty, 2)$ operator norm of a matrix defined as follows $\|B\|_{\infty \rightarrow 2} = \max\{\|Bx\|_2 : \|x\|_\infty = 1\}$. To find $\|B\|_{\infty \rightarrow 2}$ is computationally difficult. The formal statement of Pietsch factorization states that the factorization exists when $\|B\|_{\infty \rightarrow 2} \leq \|T\| \leq \sqrt{\pi/2} \|B\|_{\infty \rightarrow 2}$. From our preliminary test, $\alpha^*$ seems to give the upper bound for $\|B\|_{\infty \rightarrow 2}$. 
5 Appendix

The diagonal of $F$ in programs (1) and (2) is represented as $f$ in the EMD algorithm given as follow:

---

given Objective function $J$, dimension $s$, and number $T$ of iterations

Let $f^{(1)} = s^{-1}e$.

for $t = 1$ to $T$

1. Find $g \in \partial J(f^{(t)})$.
2. $\beta = \left(\frac{2\log{s}}{T\|g\|_\infty}\right)^{1/2}$.
3. $h = f^{(t)} \cdot \exp(-\beta g)$
4. Update $f^{(t+1)} = h / \text{Tr}(h)$.

end for

return $f \in \text{argmin}_t J(f^{(t)})$
Table 2: Performance of each techniques in preconditioning a CG method

<table>
<thead>
<tr>
<th></th>
<th>ORG</th>
<th>ASY</th>
<th>SYM</th>
<th>JCB</th>
<th>PIE</th>
<th>GRO</th>
<th>GEVP</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \kappa ) reduced ratio</td>
<td>0</td>
<td>0.1076</td>
<td>0.1238</td>
<td>0.1367</td>
<td>0.1818</td>
<td>0.2286</td>
<td>0.4245</td>
</tr>
<tr>
<td>Precon Time</td>
<td>0</td>
<td>0.00026</td>
<td>0.00052</td>
<td>0.00017</td>
<td>0.32929</td>
<td>1.32571</td>
<td>2.95389</td>
</tr>
<tr>
<td>Solve Time</td>
<td>0.02541</td>
<td>0.01308</td>
<td>0.02080</td>
<td>0.01288</td>
<td>0.01223</td>
<td>0.01239</td>
<td>0.01213</td>
</tr>
<tr>
<td>Total Time</td>
<td>0.02541</td>
<td>0.01334</td>
<td>0.02132</td>
<td>0.01305</td>
<td>0.34150</td>
<td>1.3381</td>
<td>2.96602</td>
</tr>
</tbody>
</table>

\( a) \) Average performance on 10 samples of symmetric positive definite matrix \( B \in \mathbb{R}^{50 \times 50} \)

<table>
<thead>
<tr>
<th></th>
<th>ORG</th>
<th>ASY</th>
<th>SYM</th>
<th>JCB</th>
<th>PIE</th>
<th>GRO</th>
<th>GEVP</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \kappa ) reduced ratio</td>
<td>0</td>
<td>0.0330</td>
<td>0.0527</td>
<td>0.0339</td>
<td>0.1122</td>
<td>0.1433</td>
<td>0.3020</td>
</tr>
<tr>
<td>Precon Time</td>
<td>0</td>
<td>0.00030</td>
<td>0.00072</td>
<td>0.00021</td>
<td>0.58115</td>
<td>1.66362</td>
<td>3.11834</td>
</tr>
<tr>
<td>Solve Time</td>
<td>0.02021</td>
<td>0.01783</td>
<td>0.01779</td>
<td>0.01745</td>
<td>0.01479</td>
<td>0.01491</td>
<td>0.01437</td>
</tr>
<tr>
<td>Total Time</td>
<td>0.02021</td>
<td>0.01813</td>
<td>0.01851</td>
<td>0.01766</td>
<td>0.59594</td>
<td>1.67853</td>
<td>3.13271</td>
</tr>
</tbody>
</table>

\( b) \) Average performance on 10 samples of symmetric positive definite matrix \( B \in \mathbb{R}^{60 \times 60} \)

<table>
<thead>
<tr>
<th></th>
<th>ORG</th>
<th>ASY</th>
<th>SYM</th>
<th>JCB</th>
<th>PIE</th>
<th>GRO</th>
<th>GEVP</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \kappa ) reduced ratio</td>
<td>0</td>
<td>0.0567</td>
<td>0.0685</td>
<td>0.0506</td>
<td>0.1445</td>
<td>0.1820</td>
<td>0.3571</td>
</tr>
<tr>
<td>Precon Time</td>
<td>0</td>
<td>0.00030</td>
<td>0.00077</td>
<td>0.00027</td>
<td>0.83424</td>
<td>2.10230</td>
<td>3.89984</td>
</tr>
<tr>
<td>Solve Time</td>
<td>0.02233</td>
<td>0.01858</td>
<td>0.01952</td>
<td>0.02015</td>
<td>0.01794</td>
<td>0.01718</td>
<td>0.01718</td>
</tr>
<tr>
<td>Total Time</td>
<td>0.02233</td>
<td>0.01888</td>
<td>0.02029</td>
<td>0.02042</td>
<td>0.85218</td>
<td>2.11948</td>
<td>3.91702</td>
</tr>
</tbody>
</table>

\( c) \) Average performance on 10 samples of symmetric positive definite matrix \( B \in \mathbb{R}^{70 \times 70} \)