Smooth Optimization for Sparse Semidefinite Programs

Alexandre d’Aspremont
ORFE, Princeton University

Support from EUROCONTROL, NSF and Google.
Introduction

Smooth Optimization:

- Produces smooth (Lipschitz-continuous gradient) approximation of structured semidefinite optimization problems.


- Total complexity in $O(1/\epsilon)$ instead of $O(1/\epsilon^2)$.

Smooth semidefinite optimization:

- Difference with I.P. methods: large number of simpler iterations.

- Key step is a matrix exponential: can be computed efficiently.
Smoothing technique

Example: **maximum eigenvalue minimization** problem:

\[
\min f(x) := \lambda_{\text{max}}(Ax - b)
\]

in the variable \(x \in \mathbb{R}^n\) with parameters \(A \in \mathbb{R}^{m \times n}\) and \(b \in \mathbb{R}^m\).

Solve **smooth approximation** with:

\[
\min f_\mu(x) := \mu \log \left( \text{Tr} \exp \left( \frac{Ax - b}{\mu} \right) \right)
\]

where \(\log\) and \(\exp\) are the matrix (not componentwise) logarithm and exponential, respectively.
Smoothing technique

\( f_\mu(Ax - b) \) is a \( \mu \log n \)-uniform approximation of \( \lambda_{\text{max}}(Ax - b) \):

\[
\lambda_{\text{max}}(Ax - b) \leq f_\mu(x) \leq \lambda_{\text{max}}(Ax - b) + \mu \log n
\]

and the gradient of \( f_\mu(x) \), given by:

\[
\nabla f_\mu(x) := \left( \text{Tr} \exp \left( \frac{Ax - b}{\mu} \right) \right)^{-1} \exp \left( \frac{Ax - b}{\mu} \right)
\]

is Lipschitz continuous with constant given by:

\[
L = \frac{\|A\|^2}{\mu}
\]
Smoothing technique

- If we set:
  \[ \mu = \frac{\epsilon}{2 \log n}, \]

- solving
  \[ \min f_\mu(x) \]
  produces an \( \epsilon \)-approximation of the solution to the original problem.

- Because \( \nabla f_{mu} \) is Lipschitz continuous, Nesterov (1983) shows that the complexity of solving this problem is given by:
  \[ \frac{4 \|A\|}{\epsilon} \sqrt{\log n \|x^*\|^2} \]
Nesterov’s method

• Nesterov (2005) shows that this result holds for all problems with a min-max format:

\[ f(x) = \hat{f}(x) + \max_u \{ \langle Tx, u \rangle - \hat{\phi}(u) \mid u \in Q_2 \} \]

• assuming that:
  
  ◦ \( f \) is defined over a compact convex set \( Q_1 \subset \mathbb{R}^n \)
  
  ◦ \( \hat{f}(x) \) is convex, differentiable and has a Lipschitz continuous gradient with constant \( M \geq 0 \)
  
  ◦ \( T \) is a linear operator: \( T \in \mathbb{R}^{n \times n} \)
  
  ◦ \( \hat{\phi}(u) \) is a continuous convex function over some compact set \( Q_2 \subset \mathbb{R}^n \).
Nesterov’s method

To summarize: if a problem can be written according to this min-max model, the algorithm works as follows.

- **Regularization.** Add strongly convex penalty inside the min-max representation to produce an $\epsilon$-approximation of $f$ with Lipschitz continuous gradient (generalized Moreau-Yosida regularization step, see Lemaréchal & Sagastizábal (1997) for example).

- **Optimal first order minimization.** Use optimal first order scheme for Lipschitz continuous functions detailed in Nesterov (1983) to solve the regularized problem.

Caveat: Only efficient if the subproblems involved in these steps can be solved explicitly or very efficiently.
Nesterov’s method

- The min-max model makes this an ideal candidate for robust optimization.

- For fixed problem size, the number of iterations required to get an \( \epsilon \) solution is given by

\[
O\left(\frac{1}{\epsilon}\right)
\]

compared to \( O\left(\frac{1}{\epsilon^2}\right) \) for generic first-order methods.

- Each iteration has low memory requirements.

- Change in granularity of the solver: larger number of cheaper iterations.
• The key step at each iteration is computing the gradient:

\[ \nabla f_\mu(x) := \left( \text{Tr} \exp \left( \frac{Ax - b}{\mu} \right) \right)^{-1} \exp \left( \frac{Ax - b}{\mu} \right) \]

• This amounts to a \textbf{matrix exponential} computation.

• Classic problem. See “Nineteen dubious ways to compute the exponential of a matrix” by Moler & Van Loan (2003).

• In general, Padé approximation techniques are the classic solution. We can do better here because of the matrix structure.
Main Result

When minimizing a function with Lipschitz-continuous gradient using the method in Nesterov (1983), an approximate gradient is sufficient to get the \(O(1/\epsilon)\) convergence rate:

If the function and gradient approximations satisfy:

\[
|f(x) - \tilde{f}(x)| \leq \delta \quad \text{and} \quad |\langle \tilde{\nabla} f(x) - \nabla f(x), y \rangle| \leq \delta \quad x, y \in Q_1,
\]

we have:

\[
f(x_k) - f(x^*) \leq \frac{Ld(x^*)}{(k + 1)(k + 2)\sigma} + 10\delta
\]

where \(L, d(x^*)\) and \(\sigma\) are problem constants.
Benefits

• Because the eigenvalues of the gradient matrix decrease exponentially fast, only a few eigenvalues are necessary to compute the gradient with the required precision.

• How few? Pick $X \in S_n$ with coefs $\mathcal{N}(0, \sigma^2/n)$. Wigner’s semicircle law: eigenvalues of $X$ are asy. dist. according to:

$$p(x) = \frac{1}{2\pi\sigma^2} \sqrt{4\sigma^2 - x^2},$$

in the limit, the proportion of eigenvalues required is given by:

$$P_{\lambda} \triangleq P \left( e^{\frac{\lambda_i(X) - \lambda_{\text{max}}(X)}{\mu}} \leq \gamma \right) = \int_{-2\sigma}^{2\sigma + \frac{\epsilon\log n}{\log\gamma}} \frac{1}{2\pi\sigma^2} \sqrt{4\sigma^2 - x^2} dx.$$

• With $n = 5000$, $\delta = 10^{-6}$ and $\epsilon = 10^{-2}$, we get $nP_{\lambda} = 2.3$ eigs.
Figure 1: Percentage of eigenvalues required versus duality gap on random max. eigenvalue minimization problems.
Numerical performance

- Consider the following sparse PCA relaxation, from d’Aspremont, El Ghaoui, Jordan & Lanckriet (2005):

\[
\begin{align*}
\text{minimize} & \quad \lambda_{\text{max}}(C + U) \\
\text{subject to} & \quad |U_{ij}| \leq \rho, \quad i, j = 1, \ldots, n,
\end{align*}
\]

- Use ARPACK to compute eigenvalues (sparse eig. package).

- Generate a $100 \times 100$ matrix $U$ with uniformly distributed coefficients.

- Let $e \in \mathbb{R}^{100}$ be a sparse vector with:

\[
e = (1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, \ldots)
\]

and form a test matrix $A = U^T U + vee^T$, where $v$ is a signal-to-noise ratio.
Numerical performance

**Left:** Duality gap versus CPU time for various values of the signal to noise ratio $v$. **Right:** Percentage of eigenvalues required versus duality gap for various values of the signal to noise ratio $v$. 
Conclusion

- Smooth first-order minimization with approximate gradient.
- An order of magnitude faster on semidefinite optimization problems.
- Link between problem structure and number of eigs required hard to establish.
References


