Tutorial: Algorithms for Large-Scale Semidefinite Programming

Alexandre d'Aspremont, *CNRS & Ecole Polytechnique*.

Support from NSF, ERC (project SIPA) and Google.

Introduction

A **semidefinite program** (SDP) is written

$$\begin{array}{ll} \text{minimize} & \mathbf{Tr}(CX) \\ \text{subject to} & \mathbf{Tr}(A_i X) = b_i, \quad i = 1, \ldots, m \\ & X \succeq 0, \end{array}$$

where $X \succeq 0$ means that the matrix variable $X \in \mathbf{S}_n$ is positive semidefinite.

Its **dual** can be written

$$\begin{array}{ll} \text{maximize} & b^T y \\ \text{subject to} & C - \sum_{i=1}^m y_i A_i \succeq 0, \end{array}$$

which is another semidefinite program in the variables y.

Introduction

Classical algorithms for semidefinite programming

- Following [Nesterov and Nemirovskii, 1994], most of the attention was focused on interior point methods.
- Basic idea: Newton's method, with efficient linear algebra to compute the Newton step (or solve the KKT system).
- Fast, and robust on small problems ($n \sim 500$).
- Computing the Hessian is too hard on larger problems. Exploiting structure (sparsity, etc.) is hard too.

Solvers

- Open source solvers: SDPT3, SEDUMI, SDPA, CSDP, . . .
- Very powerful modeling systems: CVX

Introduction

Solving a `MaxCut` relaxation using CVX

$$\begin{array}{ll} \text{max.} & \mathbf{Tr}(XC) \\ \text{s.t.} & \mathbf{diag}(X) = \mathbf{1} \\ & X \succeq 0, \end{array}$$

is written as follows in CVX/MATLAB

```
cvx_begin
.  variable X(n,n) symmetric
.  maximize trace(C*X)
.  subject to
.    diag(X)==1
.    X==semidefinite(n)
cvx_end
```

Introduction

Algorithms for **large-scale** semidefinite programming.

Structure $\Rightarrow$ algorithmic choices

Examples:

- SDPs with constant trace cast as max. eigenvalue minimization problems.
- Fast projection steps.
- Fast prox or affine minimization subproblems.
- Closed-form or efficiently solvable block minimization subproblems.
- Etc. . .

Introduction

Example. In many semidefinite relaxations of combinatorial problems, we can impose $\mathbf{Tr}(X) = 1$ and solve

$$\begin{array}{ll} \text{maximize} & \mathbf{Tr}(CX) \\ \text{subject to} & \mathbf{Tr}(A_i X) = b_i, \quad i = 1, \ldots, m \\ & \mathbf{Tr}(X) = 1,\ X \succeq 0, \end{array}$$

The dual can be written as a **maximum eigenvalue minimization** problem

$$\min_x \lambda_{\max}\left(C + \sum_{i=1}^m x_i A_i\right) - b^T x$$

in the variable $x \in \mathbb{R}^m$.

Outline

- Introduction
- **First-order methods**
  - Subgradient methods
  - Smoothing & accelerated algorithms
  - Improving iteration complexity
- Exploiting structure
  - Frank-Wolfe
  - Block coordinate descent
  - Dykstra, alternating projection
  - Localization, cutting-plane methods

Subgradient methods

Solve

$$\min_{x \in Q} \lambda_{\max}\left(A(x)\right) + c^T x$$

where $A(x) = C + \sum_{i=1}^m x_i A_i$, using the **projected subgradient** method.

Input: A starting point $x_0 \in \mathbb{R}^m$.

1: **for** $t = 0$ to $N - 1$ **do**

2: Set

$$x_{t+1} = P_Q(x_t - \gamma \partial \lambda_{\max}(A(x))).$$

3: **end for**

Output: A point $x = (1/N)\sum_{t=1}^N x_t$.

Here, $\gamma > 0$ and $P_Q(\cdot)$ is the Euclidean projection on Q.

Subgradient methods

- The **number of iterations** required to reach a target precision ϵ is

$$N = \frac{D_Q^2 M^2}{\epsilon^2}$$

  where D_Q is the diameter of Q and $\|\partial\lambda_{\max}(A(x))\| \leq M$ on Q.
- The **cost per iteration** is the sum of
  - The cost p_Q of computing the Euclidean projection on Q.
  - The cost of computing $\partial\lambda_{\max}(A(x))$ which is e.g. $v_1 v_1^T$ where v_1 is a leading eigenvector of $A(x)$.

Computing one leading eigenvector of a dense matrix X with relative precision ϵ, using a randomly started Lanczos method, with probability of failure $1-\delta$, costs

$$O\left(\frac{n^2 \log(n/\delta^2)}{\sqrt{\epsilon}}\right)$$

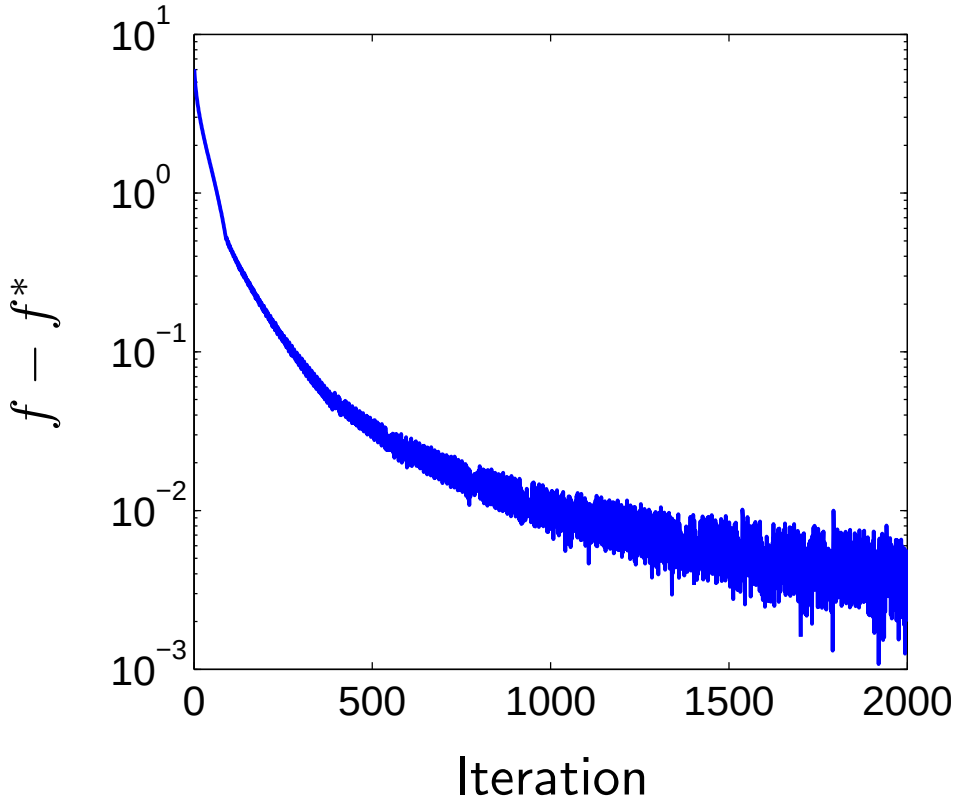
flops [Kuczynski and Wozniakowski, 1992, Th.4.2].

Subgradient methods

Solving $\min_{X \in Q} \lambda_{\max}(A(x))$ using projected subgradient.

- Easy to implement.
- Very poor performance in practice. The $1/\epsilon^2$ dependence is somewhat punishing. . .

Example below on MAXCUT.



Smoothing & accelerated algorithms

Smoothing & accelerated algorithms

[Nesterov, 2007] We can **regularize** the objective and solve

$$\min_{x \in Q} \; f_\mu(x) \triangleq \mu \log \mathbf{Tr} \left(\exp \left(\frac{A(x)}{\mu} \right) \right)$$

for some regularization parameter $\mu > 0$ ($\exp(\cdot)$ is the **matrix** exponential here).

- If we set $\mu = \epsilon / \log n$ we get

$$\lambda_{\max}(A(x)) \leq f_\mu(x) \leq \lambda_{\max}(A(x)) + \epsilon$$

- The gradient $\nabla f_\mu(x)$ is Lipschitz continuous with constant

$$\frac{\|A\|^2 \log n}{\epsilon}$$

where $\|A\| = \sup_{\|h\| \leq 1} \|A(h)\|_2$.

Smoothing & accelerated algorithms

- The number of iterations required to get an ϵ solution using the **smooth** minimization algorithm in Nesterov [1983] grows as

$$\frac{\|A\|\sqrt{\log n}}{\epsilon}\sqrt{\frac{d(x^*)}{\sigma}}$$

  where $d(\cdot)$ is strongly convex with parameter $\sigma > 0$.

- The **cost per iteration** is (usually) dominated by the cost of forming the matrix exponential

$$\exp\left(\frac{A(x)}{\mu}\right)$$

  which is $O(n^3)$ flops [Moler and Van Loan, 2003].

- Much better empirical performance.

Smoothing & accelerated algorithms

This means that the two classical complexity options for solving

$$\min_{X \in Q} \lambda_{\max}(A(x))$$

(assuming $A(x)$ cheap)

- **Subgradient methods**

$$O\left(\frac{D_Q^2(n^2 \log n + p_Q)}{\epsilon^2}\right)$$

- **Smooth optimization**

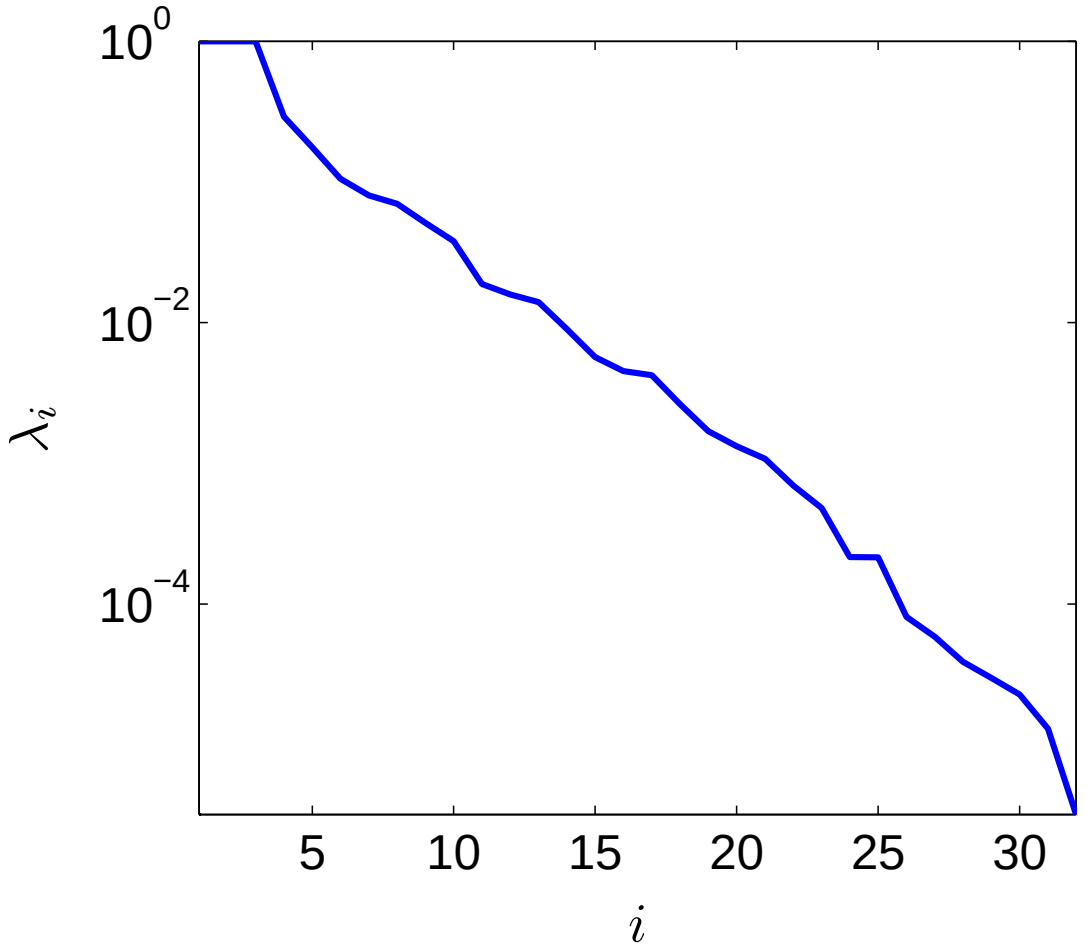
$$O\left(\frac{D_Q\sqrt{\log n}(n^3 + p_Q)}{\epsilon}\right)$$

  if we pick $\|\cdot\|_2^2$ in the prox term.

Improving iteration complexity

Approximate gradients

Approximate gradient is often enough. This means computing only a few leading eigenvectors.



Spectrum of $\exp((X - \lambda_{\max}(X)\mathbf{I})/0.1)$ at the MAXCUT solution.

Approximate gradients

Convergence guarantees using **approximate gradients**: if $\tilde{\nabla} f(x)$ is the approximate gradient oracle, we require

$$|\langle \tilde{\nabla} f(x) - \nabla f(x), y - z \rangle| \leq \delta \quad x, y, z \in Q,$$

(the condition depends on the diameter of Q). For example, to solve

$$\begin{array}{ll} \text{minimize} & \lambda_{\max}(A + X) \\ \text{subject to} & |X_{ij}| \leq \rho \end{array}$$

we only need to compute the j largest eigenvalues of $A + X$, with j such that

$$\frac{(n-j)e^{\lambda_j}\sqrt{\sum_{i=1}^{j} e^{2\lambda_i}}}{(\sum_{i=1}^{j} e^{\lambda_i})^2} + \frac{\sqrt{n-j}\; e^{\lambda_j}}{\sum_{i=1}^{j} e^{\lambda_i}} \leq \frac{\delta}{\rho n}.$$

The impact of the **diameter** makes these conditions quite conservative.

Approximate gradients

Other possible conditions (often less stringent), when solving

$$\min_{x \in Q} \max_{u \in U} \Psi(x, u)$$

If u_x is an approximate solution to $\max_{u \in U} \Psi(x, u)$, we can check $V_i(u_x) \leq \delta$

$$V_1(u_x) = \max_{u \in U} \nabla_2 \Psi(x, u_x)^T (u - u_x)$$

$$V_2(u_x) = \max_{u \in U} \left\{ \Psi(x, u) - \Psi(x, u_x) + \kappa \|u - u_x\|^2 / 2 \right\}$$

$$V_3(u_x) = \max_{u \in U} \Psi(x, u) - \Psi(x, u_x)$$

where

$$V_1(u_x) \leq V_2(u_x) \leq V_3(u_x) \leq \delta$$

The target accuracy δ on the oracle is a function of the target accuracy ϵ.

See [d'Aspremont, 2008a], [Devolder, Glineur, and Nesterov, 2011] for further details.

Stochastic Smoothing

Max-rank one Gaussian smoothing. Suppose we pick $u_i \in \mathbb{R}^n$ with i.i.d. $u_{ij} \sim \mathcal{N}(0,1)$ and define

$$f(X) = \mathbf{E}\left[\max_{i=1,\ldots,k} \lambda_{\max}(X + (\epsilon/n)u_i u_i^T)\right]$$

- Approximation results are preserved up to a constant $c_k > 0$

$$\lambda_{\max}(X) \leq \mathbf{E}[\lambda_{\max}(X + (\epsilon/n)uu^T)] \leq \lambda_{\max}(X) + c_k\epsilon$$

- The function $f(X)$ is smooth and the Lipschitz constant of its gradient is bounded by

$$L_f \leq \mathbf{E}\left[\frac{n}{2\epsilon}\left(\min_{i=1,\ldots,k} \frac{1}{u_{i,1}^2}\right)\right] \leq C_k\frac{n}{\epsilon}$$

  where $C_k = \frac{1}{\sqrt{2}}\frac{k}{k-2}$, is finite when $k \geq 3$.
- Computing $\max_{i=1,\ldots,k} \lambda_{\max}(X + (\epsilon/n)u_i u_i^T)$ costs $O(kn^2 \log n)$.

Stochastic Smoothing

Optimal Stochastic Composite Optimization. The algorithm in Lan [2009] solves

$$\min_{x \in Q} \Psi(x) \triangleq f(x) + h(x)$$

with the following assumptions

- $f(x)$ has Lipschitz gradient with constant L and $h(x)$ is Lipschitz with constant M,
- we have a **stochastic oracle** $G(x, \xi_t)$ for the gradient, which satisfies

$$\mathbf{E}[G(x, \xi_t)] = g(x) \in \partial \Psi(x) \quad \text{and} \quad \mathbf{E}[\|G(x, \xi_t) - g(x)\|_*^2] \leq \sigma^2$$

After N iterations, the iterate x_{N+1} satisfies

$$\mathbf{E}\left[\Psi(x_{N+1}^{ag}) - \Psi^*\right] \leq \frac{8LD_{\omega,Q}^2}{N^2} + \frac{4D_{\omega,Q}\sqrt{4\mathcal{M}^2 + \sigma^2}}{\sqrt{N}}$$

which is optimal. Additional assumptions guarantee convergence w.h.p.

Maximum Eigenvalue Minimization

For maximum eigenvalue minimization

- We have $\sigma \leq 1$, but we can reduce this by averaging q gradients, to control the tradeoff between smooth and non-smooth terms.
- If we set $q = \max\{1, D_Q/(\epsilon\sqrt{n})\}$ and $N = 2D_Q\sqrt{n}/\epsilon$ we get the following complexity picture

Complexity	Num. of Iterations	Cost per Iteration
Nonsmooth alg.	$O\left(\frac{D_Q^2}{\epsilon^2}\right)$	$O(p_Q + n^2 \log n)$
Smooth stochastic alg.	$O\left(\frac{D_Q\sqrt{n}}{\epsilon}\right)$	$O\left(p_Q + \max\left\{1, \frac{D_Q}{\epsilon\sqrt{n}}\right\} n^2 \log n\right)$
Smoothing alg.	$O\left(\frac{D_Q\sqrt{\log n}}{\epsilon}\right)$	$O(p_Q + n^3)$

Stochastic Smoothing

- Approximate gradients reduce empirical complexity. No **a priori** bounds on iteration cost.
- More efficient to run **a lot of cheaper iterations**, everything else being equal.

Many open questions. . .

- Not clear if rank one perturbations achieve the optimal complexity/smoothness tradeoff. Can we replicate the exponential smoothing stochastically?
- Non monotonic line search for stochastic optimization?
- **Bundle methods** also improve the performance of subgradient techniques [Lemaréchal et al., 1995, Kiwiel, 1995, Helmberg and Rendl, 2000, Oustry, 2000, Ben-Tal and Nemirovski, 2005, Lan, 2010]...

Outline

- Introduction
- First-order methods
  - Subgradient methods
  - Smoothing & accelerated algorithms
  - Improving iteration complexity
- **Exploiting structure**
  - Frank-Wolfe
  - Block coordinate descent
  - Dykstra, alternating projection
  - Localization, cutting-plane methods

Frank-Wolfe

Frank-Wolfe

- Classical first order methods for solving

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in C, \end{array}$$

in $x \in \mathbb{R}^n$, with $C \subset \mathbb{R}^n$ convex, relied on the assumption that the following **prox subproblem** could be solved efficiently

$$\begin{array}{ll} \text{minimize} & y^T x + d(x) \\ \text{subject to} & x \in C, \end{array}$$

in the variable $x \in \mathbb{R}^n$, where $d(x)$ is a strongly convex function.

- The Franke-Wolfe alg. assumes that the **affine minimization subproblem**

$$\begin{array}{ll} \text{minimize} & d^T x \\ \text{subject to} & x \in C \end{array}$$

can be solved efficiently for any $y \in \mathbb{R}^n$.

Frank-Wolfe

Frank and Wolfe [1956] algorithm. See also [Jaggi, 2011].

Input: A starting point $x_0 \in C$.

1: **for** $t = 0$ to $N - 1$ **do**

2: Compute $\nabla f(y_k)$

3: Solve the affine minimization subproblem

$$\begin{array}{ll} \text{minimize} & x^T \nabla f(x_k) \\ \text{subject to} & x \in C \end{array}$$

in $x \in \mathbb{R}^n$, call the solution x_d.

4: Update the current point

$$x_{k+1} = x_k + \frac{2}{k+2}(x_d - x_k)$$

5: **end for**

Output: A point x_N.

Note that all iterates are feasible.

Frank-Wolfe

- **Complexity.** Assume that f is differentiable. Define the curvature C_f of the function $f(x)$ as

$$C_f \triangleq \sup_{\substack{s,x \in \mathcal{M},\ \alpha \in [0,1], \\ y = x + \alpha(s-x)}} \frac{1}{\alpha^2}(f(y) - f(x) - \langle y - x, \nabla f(x) \rangle).$$

  The Franke-Wolfe algorithm will then produce an ϵ solution after

$$N_{\max} = \frac{4C_f}{\epsilon}$$

  iterations.

- Can use line search at each iteration to improve convergence.

Frank-Wolfe

- **Stopping criterion.** At each iteration, we get a lower bound on the optimum as a byproduct of the affine minimization step. By convexity,

$$f(x_k) + \nabla f(x_k)^T (x_d - x_k) \leq f(x), \quad \text{for all } x \in C$$

and finally, calling f^* the optimal value of problem, we obtain

$$f(x_k) - f^* \leq \nabla f(x_k)^T (x_k - x_d).$$

This allows us to bound the suboptimality of iterate at no additional cost.

Frank-Wolfe

Example. Semidefinite optimization with bounded trace.

$$\begin{array}{ll} \text{minimize} & f(X) \\ \text{subject to} & \mathbf{Tr}(X) = 1,\ X \succeq 0, \end{array}$$

in the variable $X \in \mathbf{S}_n$.

The affine minimization subproblem is written

$$\begin{array}{ll} \text{minimize} & \mathbf{Tr}(\nabla f(X) Y) \\ \text{subject to} & \mathbf{Tr}(Y) = 1,\ Y \succeq 0, \end{array}$$

in the variable $Y \in \mathbf{S}_n$, and can be solved by a partial eigenvalue decomposition, with the optimum value equal to $\lambda_{\min}(\nabla f(X))$ [cf. Jaggi, 2011]. Each iteration is a **rank one** update.

Block coordinate descent methods

Coordinate Descent

We seek to solve

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in C \end{array}$$

in the variable $x \in \mathbb{R}^n$, with $C \subset \mathbb{R}^n$ convex.

- Our main assumption here is that C **is a product of simpler sets**. We rewrite the problem

$$\begin{array}{ll} \text{minimize} & f(x_1, \ldots, x_p) \\ \text{subject to} & x_i \in C_i, \quad i = 1, \ldots, p \end{array}$$

  where $C = C_1 \times \ldots \times C_p$.

- This helps if the minimization subproblems

$$\min_{x_i \in C_i} f(x_1, \ldots, x_i, \ldots, x_p)$$

  can be solved very efficiently (or in closed-form).

Coordinate Descent

Algorithm. The algorithm simply computes the iterates $x^{(k+1)}$ as

$$\begin{aligned} x_i^{(k+1)} &= \operatorname*{argmin}_{x_i \in C_i} f(x_1^{(k)}, \ldots, x_i^{(k)}, \ldots, x_p^{(k)}) \\ x_j^{(k+1)} &= x_j^{(k)}, \quad j \neq i \end{aligned}$$

for a certain $i \in [1, p]$, cycling over all indices in $[1, p]$.

Convergence.

- Complexity analysis similar to coordinate-wise gradient descent (or steepest descent in ℓ_1 norm).
- Need $f(x)$ strongly convex to get explicit complexity bound [Nesterov, 2010].
- Generalization of block methods for SDP in "row-by-row" method of [Wen, Goldfarb, Ma, and Scheinberg, 2009].

Coordinate Descent

Example. **Covariance selection** [d'Aspremont et al., 2006]. The dual of the covariance selection problem is written

$$\begin{array}{ll} \text{maximize} & \log\det(S+U) \\ \text{subject to} & \|U\|_\infty \leq \rho \\ & S+U \succ 0 \end{array}$$

Let $C = S + U$ be the current iterate, after permutation we can always assume that we optimize over the last column

$$\begin{array}{ll} \text{maximize} & \log\det \begin{pmatrix} C^{11} & C^{12}+u \\ C^{21}+u^T & C^{22} \end{pmatrix} \\ \text{subject to} & \|u\|_\infty \leq \rho \end{array}$$

where C^{12} is the last column of C (off-diag.).

Coordinate Descent

We can use the block determinant formula

$$\det \begin{pmatrix} A & B \\ C & D \end{pmatrix} = \det(A) \det(D - CA^{-1}B)$$

to show that each row/column iteration reduces to a simple **box-constrained QP**

$$\begin{array}{ll} \text{minimize} & u^T (C^{11})^{-1} u \\ \text{subject to} & \|u\|_\infty \leq \rho \end{array}$$

the dual of this last problem is a LASSO optimization problem.

Dykstra, alternating projection

Dykstra, alternating projection

We focus on a simple **feasibility problem**

$$\text{find } x \in C_1 \cap C_2$$

in the variable $x \in \mathbb{R}^n$ with C_1, $C_2 \subset \mathbb{R}^n$ two convex sets.

We assume now that the **projection problems on C_i** are easier to solve

$$\begin{array}{ll} \text{minimize} & \|x - y\|_2 \\ \text{subject to} & x \in C_i \end{array}$$

in $x \in \mathbb{R}^n$.

Dykstra, alternating projection

Algorithm (alternating projection)

- Choose $x_0 \in \mathbb{R}^n$.
- **For** $k = 1, \ldots, k^{max}$ **iterate**

  1. Project on C_1
  $$x_{k+1/2} = \operatorname*{argmin}_{x \in C_1} \|x - x_k\|_2$$
  2. Project on C_2
  $$x_{k+1} = \operatorname*{argmin}_{x \in C_2} \|x - x_{k+1/2}\|_2$$

Convergence. We can show $\mathbf{dist}(x_k, C_1 \cap C_2) \to 0$. Linear convergence provided some additional regularity assumptions. See e.g. [Lewis, Malick, et al., 2008]

Dykstra, alternating projection

Algorithm (Dykstra)

- Choose $x_0, z_0 \in \mathbb{R}^n$.
- **For** $k = 1, \ldots, k^{max}$ **iterate**

  1. Project on C_1
  $$x_{k+1/2} = \operatorname*{argmin}_{x \in C_1} \|x - z_k\|_2$$
  2. Update
  $$z_{k+1/2} = 2x_{k+1/2} - z_k$$
  3. Project on C_2
  $$x_{k+1} = \operatorname*{argmin}_{x \in C_2} \|x - z_{k+1/2}\|_2$$
  4. Update
  $$z_{k+1} = z_k + x_{k+1} - x_{k+1/2}$$

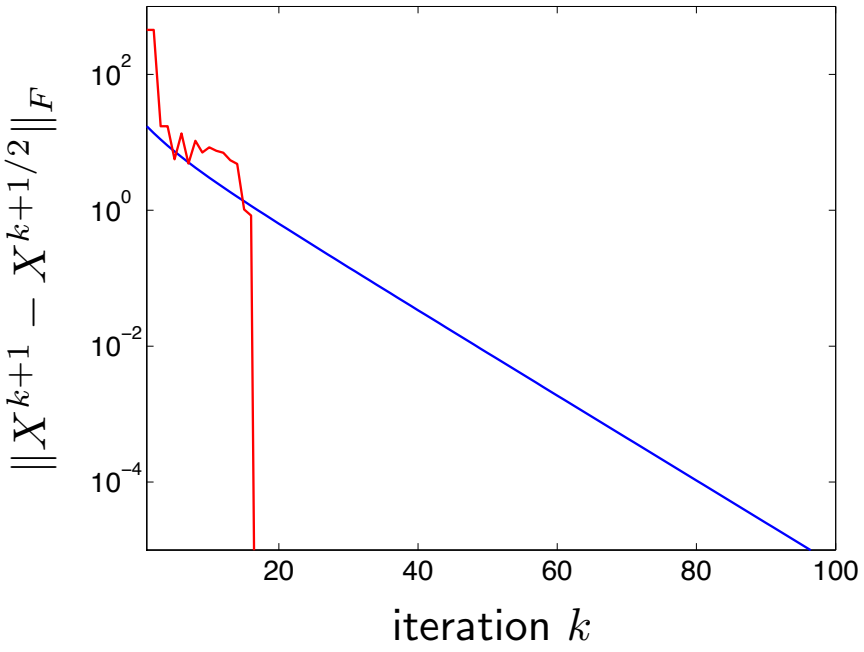
Convergence. Usually faster than simple alternating projection.

Dykstra, alternating projection

Example. **Matrix completion** problem, given coefficients b_{ij} for $(i,j) \in S$

$$\begin{array}{ll} \text{Find} & X \\ \text{such that} & X_{ij} = b_{ij}, \quad (i,j) \in S \\ & X \succeq 0, \end{array}$$

in the variable $X \in \mathbf{S}_n$.



Blue: alternating projection. Red: Dykstra. (from EE364B)

Dykstra, alternating projection

Countless variations. . .

- Proximal point algorithm
- Douglas-Rachford splitting
- Operator splitting methods
- Bregman iterative methods
- . . .

Localization methods

Localization methods

From EE364B course at Stanford. . .

- Function $f : \mathbb{R}^n \to \mathbb{R}$ convex (and for now, differentiable)
- **problem:** minimize f
- **oracle model:** for any x we can evaluate f and $\nabla f(x)$ (at some cost)

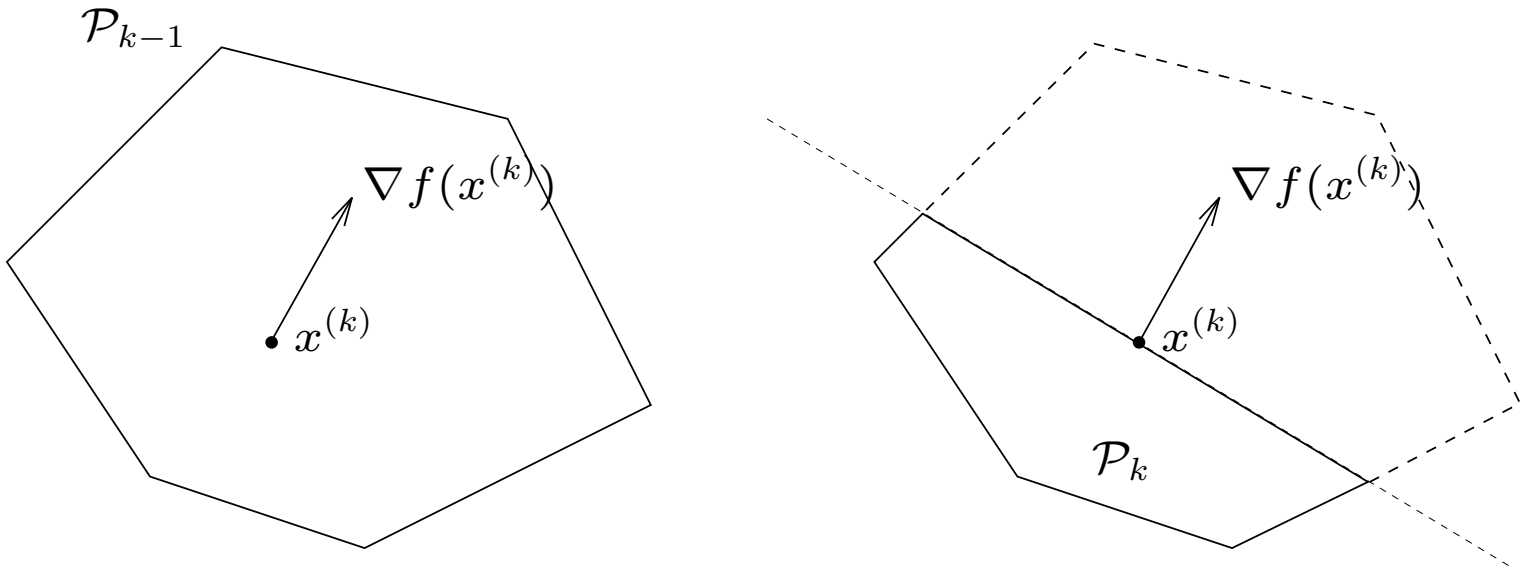
Main assumption: **evaluating the gradient is very expensive**.

Convexity means $f(x) \geq f(x_0) + \nabla f(x_0)^T (x - x_0)$, so

$$\nabla f(x_0)^T (x - x_0) \geq 0 \quad \Longrightarrow \quad f(x) \geq f(x_0)$$

i.e., all points in halfspace $\nabla f(x_0)^T (x - x_0) \geq 0$ are **worse** than x_0

Localization methods



- $\mathcal{P}_k$ gives our uncertainty of $x^\star$ at iteration k
- want to pick $x^{(k)}$ so that $\mathcal{P}_{k+1}$ is as small as possible
- clearly want $x^{(k)}$ near center of $C^{(k)}$

Localization methods

analytic center of polyhedron $\mathcal{P} = \{z \mid a_i^T z \preceq b_i,\ i = 1, \ldots, m\}$ is

$$\mathsf{AC}(\mathcal{P}) = \operatorname*{argmin}_z - \sum_{i=1}^{m} \log(b_i - a_i^T z)$$

ACCPM is localization method with next query point $x^{(k+1)} = \mathsf{AC}(\mathcal{P}_k)$ (found by Newton's method)

Localization methods

- let x^* be analytic center of $\mathcal{P} = \{z \mid a_i^T z \preceq b_i,\ i = 1, \ldots, m\}$
- let H^* be Hessian of barrier at x^*,

$$H^* = -\nabla^2 \sum_{i=1}^m \log(b_i - a_i^T z)\Big|_{z=x^*} = \sum_{i=1}^m \frac{a_i a_i^T}{(b_i - a_i^T x^*)^2}$$

- then, $\mathcal{P} \subseteq \mathcal{E} = \{z \mid (z - x^*)^T H^* (z - x^*) \leq m^2\}$ (not hard to show)
- let $\mathcal{E}^{(k)}$ be outer ellipsoid associated with $x^{(k)}$
- a lower bound on optimal value $p^\star$ is

$$\begin{aligned} p^\star &\geq \inf_{z \in \mathcal{E}^{(k)}} \left(f(x^{(k)}) + g^{(k)T}(z - x^{(k)}) \right) \\ &= f(x^{(k)}) - m_k \sqrt{g^{(k)T} H^{(k)-1} g^{(k)}} \end{aligned}$$

  (m_k is number of inequalities in $\mathcal{P}_k$)
- gives simple stopping criterion $\sqrt{g^{(k)T} H^{(k)-1} g^{(k)}} \leq \epsilon / m_k$

Localization methods

ACCPM algorithm.

Input: Polyhedron $\mathcal{P}$ containing $x^\star$.

1: **for** $t = 0$ to $N - 1$ **do**
2: Compute x^*, the analytic center of $\mathcal{P}$, and the Hessian H^*.
3: Compute $f(x^*)$ and $g \in \partial f(x^*)$.
4: Set $u := \min\{u, f(x^*)\}$ and $l := \max\{l, f(x^*) - m\sqrt{g^T H^{*-1} g}\}$.
5: Add inequality $g^T(z - x^*) \leq 0$ to $\mathcal{P}$.
6: **end for**

Output: A localization set $\mathcal{P}$.

Localization methods

ACCPM adds an inequality to $\mathcal{P}$ each iteration, so centering gets harder, more storage as algorithm progresses

Schemes for **dropping constraints** from $\mathcal{P}^{(k)}$:

- remove all redundant constraints (expensive)
- remove some constraints known to be redundant
- remove constraints based on some relevance ranking

Localization methods

Example. **Classification with indefinite kernels.** [Luss and d'Aspremont, 2008] Solve

$$\min_{\{K\succeq 0,\ \|K-K_0\|_F^2\leq\beta\}} \ \max_{\{\alpha^T y=0,\ 0\leq\alpha\leq C\}} \ \alpha^T e - \frac{1}{2}\,\mathbf{Tr}(K(Y\alpha)(Y\alpha)^T)$$

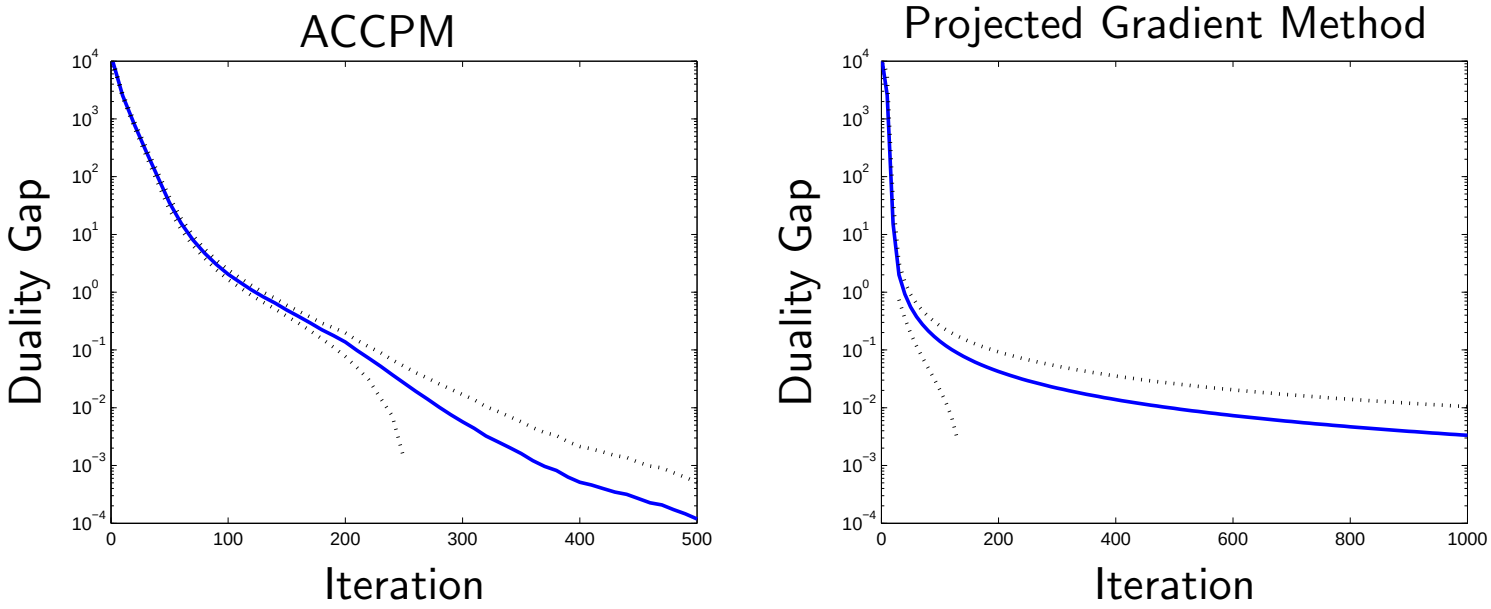
in the variables $K \in \mathbf{S}^n$ and $\alpha \in \mathbb{R}^n$. This can be written

$$\begin{array}{ll} \text{maximize} & \alpha^T e - \frac{1}{2}\sum_i \max(0, \lambda_i(K_0 + (Y\alpha)(Y\alpha)^T/4\rho))(\alpha^T Y v_i)^2 \\ & +\rho \sum_i \left(\max(0, \lambda_i(K_0 + (Y\alpha)(Y\alpha)^T/4\rho))\right)^2 + \rho\,\mathbf{Tr}(K_0 K_0) \\ & -2\rho \sum_i \mathbf{Tr}((v_i v_i^T)K_0)\max(0, \lambda_i(K_0 + (Y\alpha)(Y\alpha)^T/4\rho)) \\ \text{subject to} & \alpha^T y = 0, 0 \leq \alpha \leq C \end{array}$$

in the variable $\alpha \in \mathbb{R}^n$.

Computing the gradient at each iteration is expensive, but the feasible set is a Polyhedron.

Localization methods



Convergence plots for ACCPM (left) and projected gradient method (right) on random subsets of the USPS-SS-3-5 data set (average gap versus iteration number, dashed lines at plus and minus one standard deviation).

Conclusion

Countless other methods not discussed here. Some with no convergence guarantees.

- Low Rank semidefinite programming. (Choose a factorization $X = VV^T$ and solve in V). [Burer and Monteiro, 2003, Journée et al., 2008]
- Row by row methods. Some solver variants for MAXCUT require only matrix vector products. [Wen et al., 2009]
- Multiplicative update methods [Arora and Kale, 2007]. No implementation or performance details.

Some recent activity on subsampling.

- Variational inequality formulation [Juditsky et al., 2008, Baes et al., 2011].
- Columnwise or elementwise matrix subsampling [d'Aspremont, 2008b].

Conclusion

Large-scale semidefinite programs.

- First-order algorithms for solving (mostly) generic problems.
- For more specialized problems

Structure $\Rightarrow$ algorithmic choices

What subproblem can you solve easily? Which algorithm exploits it best?

*

References

S. Arora and S. Kale. A combinatorial, primal-dual approach to semidefinite programs. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 227–236, 2007.

M. Baes, M. Bürgisser, and A. Nemirovski. A randomized mirror-prox method for solving structured large-scale matrix saddle-point problems. *Arxiv preprint arXiv:1112.1274*, 2011.

A. Ben-Tal and A. Nemirovski. Non-Euclidean restricted memory level method for large-scale convex optimization. *Mathematical Programming*, 102(3):407–456, 2005.

S. Burer and R.D.C. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming*, 95(2):329–357, 2003.

A. d'Aspremont. Smooth optimization with approximate gradient. *SIAM Journal on Optimization*, 19(3):1171–1183, 2008a.

A. d'Aspremont. Subsampling algorithms for semidefinite programming. *arXiv:0803.1990*, 2008b.

A. d'Aspremont, O. Banerjee, and L. El Ghaoui. First-order methods for sparse covariance selection. *SIAM Journal on Matrix Analysis and Applications*, 30(1):56–66, 2006.

O. Devolder, F. Glineur, and Y. Nesterov. First-order methods of smooth convex optimization with inexact oracle. *CORE Discussion Papers,(2011/02)*, 2011.

M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

C. Helmberg and F. Rendl. A spectral bundle method for semidefinite programming. *SIAM Journal on Optimization*, 10(3):673–696, 2000.

M. Jaggi. Convex optimization without projection steps. *Arxiv preprint arXiv:1108.1170*, 2011.

M. Journée, F. Bach, P.A. Absil, and R. Sepulchre. Low-rank optimization for semidefinite convex problems. *Arxiv preprint arXiv:0807.4423*, 2008.

A. Juditsky, A.S. Nemirovskii, and C. Tauvel. Solving variational inequalities with Stochastic Mirror-Prox algorithm. *Arxiv preprint arXiv:0809.0815*, 2008.

K.C. Kiwiel. Proximal level bundle methods for convex nondifferentiable optimization, saddle-point problems and variational inequalities. *Mathematical Programming*, 69(1):89–109, 1995.

J. Kuczynski and H. Wozniakowski. Estimating the largest eigenvalue by the power and Lanczos algorithms with a random start. *SIAM J. Matrix Anal. Appl*, 13(4):1094–1122, 1992.

G. Lan. An optimal method for stochastic composite optimization. *Technical report, School of Industrial and Systems Engineering, Georgia Institute of Technology, 2009*, 2009.

G. Lan. Bundle-type methods uniformly optimal for smooth and non-smooth convex optimization. *Manuscript, Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL*, 32611, 2010.

C. Lemaréchal, A. Nemirovskii, and Y. Nesterov. New variants of bundle methods. *Mathematical programming*, 69(1):111–147, 1995.

A. Lewis, J. Malick, et al. Alternating projections on manifolds. *Mathematics of Operations Research*, 33(1):216–234, 2008.

R. Luss and A. d'Aspremont. Support vector machine classification with indefinite kernels. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 953–960. MIT Press, Cambridge, MA, 2008.

C. Moler and C. Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review*, 45(1):3–49, 2003.

Y. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27(2): 372–376, 1983.

Y. Nesterov. Smoothing technique and its applications in semidefinite optimization. *Mathematical Programming*, 110(2):245–259, 2007.

Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *CORE Discussion Papers*, 2010.

Y. Nesterov and A. Nemirovskii. *Interior-point polynomial algorithms in convex programming*. Society for Industrial and Applied Mathematics, Philadelphia, 1994.

F. Oustry. A second-order bundle method to minimize the maximum eigenvalue function. *Mathematical Programming*, 89(1):1–33, 2000.

Z. Wen, D. Goldfarb, S. Ma, and K. Scheinberg. Row by row methods for semidefinite programming. Technical report, Technical report, Department of IEOR, Columbia University, 2009.