Large Scale Optimization
Outline

- First-order methods: introduction
- Exploiting structure
- First order algorithms
  - Subgradient methods
  - Gradient methods
  - Accelerated gradient methods
- Other algorithms
  - Coordinate descent methods
  - Localization methods
  - Franke-Wolfe
  - Dykstra, alternating projection
  - Stochastic optimization
Coordinate Descent
Coordinate Descent

We seek to solve

\[
\begin{align*}
& \text{minimize} & & f(x) \\
& \text{subject to} & & x \in C
\end{align*}
\]

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

\[
\begin{align*}
& \text{minimize} & & f(x_1, \ldots, x_p) \\
& \text{subject to} & & x_i \in C_i, \quad i = 1, \ldots, p
\end{align*}
\]

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

$$
x_i^{(k+1)} = \arg\min_{x_i \in C_i} f(x_1^{(k)}, \ldots, x_i^{(k)}, \ldots, x_p^{(k)})
$$

$$
x_j^{(k+1)} = x_j^{(k)}, \quad j \neq i
$$

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 $\ell_1$ norm).
- Need $f(x)$ strongly convex to get linear complexity bound.
- Few clean results outside of this setting.
Example.

- Consider the box constrained minimization problem

\[
\begin{align*}
\text{minimize} & \quad x^T A x + b^T x \\
\text{subject to} & \quad \|x\|_\infty \leq 1
\end{align*}
\]

in the variable \( x \in \mathbb{R}^n \). We assume \( A \succ 0 \).

- The set \( \|x\|_\infty \leq 1 \) is a box, i.e. a product of intervals.

- Each minimization subproblem means solving a second order equation.

- The dual is

\[
\min_{y \in \mathbb{R}^n} (b + y)^T A^{-1} (b + y) - 4\|y\|_1
\]

which can be interpreted as a penalized regression problem in the variable \( y \in \mathbb{R}^n \).
Localization methods
Localization methods

- Function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ 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.

From $f(x) \geq f(x_0) + \nabla f(x_0)^T(x - x_0)$ we conclude

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

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

- by evaluating $\nabla f$ we rule out a halfspace in our search for $x^*$:

$$x^* \in \{x \mid \nabla f(x_0)^T(x-x_0) \leq 0\}$$

- idea: get one bit of info (on location of $x^*$) by evaluating $\nabla f$

- for nondifferentiable $f$, can replace $\nabla f(x_0)$ with any subgradient $g \in \partial f(x_0)$
Localization methods

suppose we have evaluated $\nabla f(x_1), \ldots, \nabla f(x_k)$ then we know

$x^* \in \{x \mid \nabla f(x_i)^T (x - x_i) \leq 0\}$

on the basis of $\nabla f(x_1), \ldots, \nabla f(x_k)$, we have localized $x^*$ to a polyhedron

question: what is a ‘good’ point $x_{k+1}$ at which to evaluate $\nabla f$?
Localization methods

Basic localization (or cutting-plane) algorithm:

1. after iteration \( k - 1 \) we know \( x^* \in \mathcal{P}_{k-1} \):

\[
\mathcal{P}_{k-1} = \{x \mid \nabla f(x^{(i)})^T (x - x^{(i)}) \leq 0, \ i = 1, \ldots, k - 1\}
\]

2. evaluate \( \nabla f(x^{(k)}) \) (or \( g \in \partial f(x^{(k)}) \)) for some \( x^{(k)} \in \mathcal{P}_{k-1} \)

3. \( \mathcal{P}_k := \mathcal{P}_{k-1} \cap \{x \mid \nabla f(x^{(k)})^T (x - x^{(k)}) \leq 0\} \)
Localization methods

- $P_k$ gives our uncertainty of $x^*$ at iteration $k$
- want to pick $x^{(k)}$ so that $P_{k+1}$ is as small as possible
- clearly want $x^{(k)}$ near center of $C^{(k)}$
Example: bisection on $\mathbb{R}$

- $f : \mathbb{R} \rightarrow \mathbb{R}$
- $\mathcal{P}_k$ is interval
- obvious choice: $x^{(k+1)} := \text{midpoint}(\mathcal{P}_k)$

\[
\text{bisection algorithm}
\]

given interval $C = [l, u]$ containing $x^*$

repeat
1. $x := (l + u)/2$
2. evaluate $f'(x)$
3. if $f'(x) < 0$, $l := x$; else $u := x$
Example: bisection on $\mathbb{R}$

\[ \text{length}(\mathcal{P}_{k+1}) = u_{k+1} - l_{k+1} = \frac{u_k - l_k}{2} = \frac{1}{2} \text{length}(\mathcal{P}_k) \]

and so \[ \text{length}(\mathcal{P}_k) = 2^{-k} \text{length}(\mathcal{P}_0) \]
Example: bisection on $\mathbb{R}$

interpretation:

- $\text{length}(\mathcal{P}_k)$ measures our uncertainty in $x^*$
- Uncertainty is halved at each iteration; get exactly one bit of info about $x^*$ per iteration
- # steps required for uncertainty (in $x^*$) $\leq \epsilon$:
  \[
  \log_2 \frac{\text{length}(\mathcal{P}_0)}{\epsilon} = \log_2 \frac{\text{initial uncertainty}}{\text{final uncertainty}}
  \]

question:

- can bisection be extended to $\mathbb{R}^n$?
- or is it special since $\mathbb{R}$ is linear ordering?
Center of gravity algorithm

Take $x^{(k+1)} = CG(\mathcal{P}_k)$ (center of gravity)

$$CG(\mathcal{P}_k) = \frac{\int_{\mathcal{P}_k} x \, dx}{\int_{\mathcal{P}_k} \, dx}$$

**Theorem.** If $C \subseteq \mathbb{R}^n$ convex, $x_{cg} = CG(C')$, $g \neq 0$,

$$\text{vol} \left( C \cap \{ x \mid g^T(x - x_{cg}) \leq 0 \} \right) \leq (1 - 1/e) \text{vol}(C) \approx 0.63 \text{ vol}(C')$$

(independent of dimension $n$)

hence in CG algorithm, $\text{vol}(\mathcal{P}_k) \leq 0.63^k \text{ vol}(\mathcal{P}_0)$
Center of gravity algorithm

- $\text{vol}(\mathcal{P}_k)^{1/n}$ measures uncertainty (in $x^*$) at iteration $k$
- uncertainty reduced at least by $0.63^{1/n}$ each iteration
- from this can prove $f(x^{(k)}) \to f(x^*)$ (later)
- max. # steps required for uncertainty $\leq \epsilon$:

$$1.51n \log_2 \frac{\text{initial uncertainty}}{\text{final uncertainty}}$$

(cf. bisection on $\mathbb{R}$)
Center of gravity algorithm

advantages of CG-method

- guaranteed convergence
- number of steps proportional to dimension $n$, log of uncertainty reduction

disadvantages

- finding $x^{(k+1)} = \text{CG}(\mathcal{P}_k)$ is harder than original problem
- $\mathcal{P}_k$ becomes more complex as $k$ increases
  (removing redundant constraints is harder than solving original problem)

(but, can modify CG-method to work)
analytic center of polyhedron $\mathcal{P} = \{ z \mid a_i^T z \leq b_i, \ i = 1, \ldots, m \}$ is

$$AC(\mathcal{P}) = \arg\min_z - \sum_{i=1}^{m} \log(b_i - a_i^T z)$$

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

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

then, $\mathcal{P} \subseteq \mathcal{E} = \{ z \mid (z - x^*)^T H^* (z - x^*) \leq m^2 \}$ (not hard to show)
Lower bound in ACCPM

- let $\mathcal{E}(k)$ be outer ellipsoid associated with $x^{(k)}$

- a lower bound on optimal value $p^*$ is

$$p^* \geq \inf_{z \in \mathcal{E}(k)} \left( f(x^{(k)}) + g^{(k)}T(z - x^{(k)}) \right)$$

$$= f(x^{(k)}) - m_k \sqrt{g^{(k)}T H^{(k)} - 1} g^{(k)}$$

($m_k$ is number of inequalities in $\mathcal{P}_k$)

- gives simple stopping criterion $\sqrt{g^{(k)}T H^{(k)} - 1} g^{(k)} \leq \epsilon / m_k$
Best objective and lower bound

since ACCPM isn’t a descent a method, we keep track of best point found, and best lower bound

best function value so far: \( u_k = \min_{i=1,...,k} f(x^{(k)}) \)

best lower bound so far: \( l_k = \max_{i=1,...,k} \left[ f(x^{(k)}) - m_k \sqrt{g^{(k)\top}H^{(k)}-1} g^{(k)} \right] \)

can stop when \( u_k - l_k \leq \epsilon \)
Basic ACCPM

given polyhedron $\mathcal{P}$ containing $x^*$

repeat

1. compute $x^*$, the analytic center of $\mathcal{P}$, and $H^*$
2. compute $f(x^*)$ and $g \in \partial f(x^*)$
3. $u := \min\{u, f(x^*)\}$
   
   $l := \max\{l, f(x^*) - m \sqrt{g^T H^* - 1} g\}$
4. add inequality $g^T (z - x^*) \leq 0$ to $\mathcal{P}$

until $u - l < \epsilon$

here $m$ is number of inequalities in $\mathcal{P}$
Dropping constraints

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
Dropping constraints in ACCPM

\( x^* \) is AC of \( \mathcal{P} = \{ x \mid a_i^T x \leq b_i, \; i = 1, \ldots, m \} \), \( H^* \) is barrier Hessian at \( x^* \)

define (ir)relevance measure \( \eta_i = \frac{b_i - a_i^T x^*}{\sqrt{a_i^T H^* a_i}} \)

- \( \eta_i / m \) is normalized distance from hyperplane \( a_i^T x = b_i \) to outer ellipsoid
- if \( \eta_i \geq m \), then constraint \( a_i^T x \leq b_i \) is redundant

common ACCPM constraint dropping schemes:

- drop all constraints with \( \eta_i \geq m \) (guaranteed to not change \( \mathcal{P} \))
- drop constraints in order of irrelevance, keeping constant number, usually \( 3n - 5n \)
Example

PWL objective, $n = 10$ variables, $m = 100$ terms

simple ACCPM: $f(x^{(k)})$ and lower bound $f(x^{(k)}) - m\sqrt{g^{(k)}H^{(k)}-1g^{(k)}}$
ACCPM with constraint dropping

\[ u_k - l_k \]

\[ u_k - p^* \]

- no dropping
- drop \( \eta_i > m \)
- keep \( 3n \)
ACCPM with constraint dropping

number of inequalities in $\mathcal{P}$:

\[ \begin{align*}
  &\text{no dropping} \\
  &\text{drop } \eta_i > m \\
  &\text{keep } 3n
\end{align*} \]

\[ k \]

\[ \begin{align*}
  &\text{no dropping} \\
  &\text{drop } \eta_i > m \\
  &\text{keep } 3n
\end{align*} \]

\[ k \]

... constraint dropping actually improves convergence (!)
Challenges in cutting-plane methods:

- can be difficult to compute appropriate next query point
- localization polyhedron grows in complexity as algorithm progresses

can get around these challenges . . .

**ellipsoid method** is another approach

- developed in 70s by Shor and Yudin
- used in 1979 by Khachian to give polynomial time algorithm for LP
Ellipsoid algorithm

idea: localize $x^*$ in an ellipsoid instead of a polyhedron

1. at iteration $k$ we know $x^* \in \mathcal{E}^{(k)}$

2. set $x^{(k+1)} := \text{center}(\mathcal{E}^{(k)})$; evaluate $\nabla f(x^{(k+1)})$ (or $g^{(k)} \in \partial f(x^{(k+1)})$)

3. hence we know

$$x^* \in \mathcal{E}^{(k)} \cap \{z \mid \nabla f(x^{(k+1)})^T(z - x^{(k+1)}) \leq 0\}$$

(a half-ellipsoid)

4. set $\mathcal{E}^{(k+1)} := \text{minimum volume ellipsoid covering}$

$$\mathcal{E}^{(k)} \cap \{z \mid \nabla f(x^{(k+1)})^T(z - x^{(k+1)}) \leq 0\}$$
compared to cutting-plane method:

- localization set doesn’t grow more complicated
- easy to compute query point
- but, we add unnecessary points in step 4
Properties of ellipsoid method

- reduces to bisection for $n = 1$
- simple formula for $E^{(k+1)}$ given $E^{(k)}$, $\nabla f(x^{(k+1)})$
- $E^{(k+1)}$ can be larger than $E^{(k)}$ in diameter (max semi-axis length), but is always smaller in volume
- $\text{vol}(E^{(k+1)}) < e^{-\frac{1}{2n}} \text{vol}(E^{(k)})$
  (note that volume reduction factor depends on $n$)
Example
Updating the ellipsoid

\[ \mathcal{E}(x, A) = \{ z \mid (z - x)^T A^{-1} (z - x) \leq 1 \} \]
Updating the ellipsoid

(for $n > 1$) minimum volume ellipsoid containing

$$\mathcal{E} \cap \{ z \mid g^T(z - x) \leq 0 \}$$

is given by

$$x^+ = x - \frac{1}{n + 1} A\tilde{g}$$

$$A^+ = \frac{n^2}{n^2 - 1} \left( A - \frac{2}{n + 1} A\tilde{g}\tilde{g}^T A \right)$$

where $\tilde{g} \triangleq g / \sqrt{g^T A g}$
As in the ACCPM case, we can get **error bounds** on the current iterate.

\[ x^* \in \mathcal{E}_k, \text{ so} \]

\[
f(x^*) \geq f(x^{(k)}) + \nabla f(x^{(k)})^T(x^* - x^{(k)}) \geq f(x^{(k)}) + \inf_{x \in \mathcal{E}(k)} \nabla f(x^{(k)})^T(x - x^{(k)}) = f(x^{(k)}) - \sqrt{\nabla f(x^{(k)})^T A^{(k)} \nabla f(x^{(k)})} \]

simple stopping criterion:

\[
\sqrt{\nabla f(x^{(k)})^T A^{(k)} \nabla f(x^{(k)})} \leq \epsilon
\]
Stopping criterion

\[ f(x^{(k)}) - \sqrt{\nabla f(x^{(k)})^T A^{(k)} \nabla f(x^{(k)})} \]
Basic ellipsoid algorithm

ellipsoid described as \( \mathcal{E}(x, A) = \{ z \mid (z - x)^T A^{-1} (z - x) \leq 1 \} \)

given ellipsoid \( \mathcal{E}(x, A) \) containing \( x^* \), accuracy \( \epsilon > 0 \)
repeat
1. evaluate \( \nabla f(x) \) (or \( g \in \partial f(x) \))
2. if \( \sqrt{\nabla f(x)^T A \nabla f(x)} \leq \epsilon \), return \( x \)
3. update ellipsoid
   3a. \( \tilde{g} := \nabla f(x) / \sqrt{\nabla f(x)^T A \nabla f(x)} \)
   3b. \( x := x - \frac{1}{n+1} A \tilde{g} \)
   3c. \( A := \frac{n^2}{n^2 - 1} \left( A - \frac{2}{n+1} A \tilde{g} \tilde{g}^T A \right) \)

properties:

- can propagate Cholesky factor of \( A \); get \( O(n^2) \) update
- not a descent method
- often slow but robust in practice
Franke-Wolfe
Classical first order methods for solving

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad x \in C,
\end{align*}
\]

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

\[
\begin{align*}
\text{minimize} & \quad y^T x + d(x) \\
\text{subject to} & \quad x \in C,
\end{align*}
\]

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

The method detailed here assumes instead that the \textbf{affine minimization subproblem}

\[
\begin{align*}
\text{minimize} & \quad d^T x \\
\text{subject to} & \quad x \in C
\end{align*}
\]

can be solved efficiently for any \( y \in \mathbb{R}^n \).
Franke-Wolfe

Algorithm.

- Choose $x_0 \in C$.
- For $k = 1, \ldots, k^{\text{max}}$ iterate
  1. Compute $d \in \partial f(y_k)$
  2. Solve
     \[
     \begin{align*}
     \text{minimize} & \quad d^T x \\
     \text{subject to} & \quad x \in C
     \end{align*}
     \]
     in $x \in \mathbb{R}^n$, call the solution $x_d$.
  3. Update the current point
     \[x_{k+1} = x_k + \frac{2}{k+2}(d - x_k)\]

Note that all iterates are feasible.
Franke-Wolfe

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

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

The Franke-Wolfe algorithm will then produce an $\epsilon$ solution after

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

iterations.
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 the 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.
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

\[
\begin{align*}
\text{minimize} & \quad \|x - y\|_2 \\
\text{subject to} & \quad x \in C_i
\end{align*}
\]

in \(x \in \mathbb{R}^n\).
Algorithm (alternating projection)

- Choose $x_0 \in \mathbb{R}^n$.
- For $k = 1, \ldots, k_{max}$ iterate
  1. Project on $C_1$
     \[ x_{k+1/2} = \arg\min_{x \in C_1} \| x - x_k \|_2 \]
  2. Project on $C_2$
     \[ x_{k+1} = \arg\min_{x \in C_2} \| x - x_{k+1/2} \|_2 \]

Convergence. We can show $\text{dist}(x_k, C_1 \cap C_2) \to 0$. Linear convergence provided some additional regularity assumptions.
Algorithm (Dykstra)

Choose $x_0, z_0 \in \mathbb{R}^n$.

For $k = 1, \ldots, k_{\text{max}}$ iterate

1. Project on $C_1$

$$x_{k+1/2} = \arg\min_{x \in C_1} \|x - z_k\|_2$$

2. Update

$$z_{k+1/2} = 2x_{k+1/2} - z_k$$

3. Project on $C_2$

$$x_{k+1} = \arg\min_{x \in C_2} \|x - z_{k+1/2}\|_2$$

4. Update

$$z_{k+1} = z_k + x_{k+1} - x_{k+1/2}$$

Convergence. Usually faster than simple alternating projection.
Stochastic Optimization
Stochastic Optimization

Solve

\begin{align*}
\text{minimize} & \quad \mathbb{E}[f(x, \xi)] \\
\text{subject to} & \quad x \in C,
\end{align*}

in $x \in \mathbb{R}^n$, where $C$ is a simple convex set. The key difference here is that the function we are minimizing is \textit{stochastic}.

\textbf{Batch method.} A simple option is to approximate the problem by

\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{m} f(x, \xi_i) \\
\text{subject to} & \quad x \in C,
\end{align*}

where $\xi_i$ are sampled from the distribution of $\xi$.

Sampling is costly, we can do better. . .
Let $p_C(\cdot)$ be the Euclidean projection operator on $C$.

Algorithm (Robust stochastic averaging)

- Choose $x_0 \in C$ and a step sequence $\gamma_j > 0$.
- For $k = 1, \ldots, k_{\text{max}}$ iterate
  1. Compute a subgradient
     \[ g \in \partial f(x_k, \xi_k) \]
  2. Update the current point
     \[ x_{k+1} = p_C(x_k - \gamma_k g) \]
Call $\tilde{x}_k = \sum_{i=1}^{k} \gamma_i x_i$ and assume
\[
\max_{x \in C} \mathbb{E}[\|g\|^2_2] \leq M^2, \quad \text{and} \quad D_C = \max_{x,y \in C} \|x - y\|_2
\]

If we set $\gamma_i = D_C/(M\sqrt{k})$, we have
\[
\mathbb{E}[f(\tilde{x}_k) - f^*] \leq \frac{D_CM}{\sqrt{k}}
\]

Furthermore, if we assume
\[
\mathbb{E} \left[ \exp \left( \frac{\|g\|^2_2}{M^2} \right) \right] \leq e, \quad \text{for all } g \in \partial f(x_k, \xi) \text{ and } x \in C
\]
we get
\[
\text{Prob} \left[ f(\tilde{x}_k) - f^* \geq \frac{D_CM}{\sqrt{k}} (12 + 2t) \right] \leq 2 \exp(-t)
\]