Convex Optimization

Prof. Nati Srebro

Lecture 15:

Gradient Descent with Constraints

Reading: Bubeck Sections 3.1,3.3

Lower Bounds

Reading: Nemirovski "Information Based Complexity" Section 1.1

Further extended reading on n-dimensional lower bound: Section 3.1

Method	Oracle	Assumptions	# accesses	Adtl. runtime
Center of Mass	1st / separation	$\lvert f_0\rvert, \lvert f_i\rvert \leq B$ Find ϵ-feasible x s.t. $f_0(x) \leq p^* + \epsilon$ OR: find feasible x s.t. $f_0(x) \leq \inf_{f_i(\tilde{x})<\epsilon} f_0(\tilde{x}) + \epsilon$	$O\left(n \log \frac{B}{\epsilon}\right)$	NA
Ellipsoid			$O\left(n^2 \log \frac{B}{\epsilon}\right)$	$O\left(n^4 \log \frac{B}{\epsilon}\right)$
Vaidya++			$\tilde{O}\left(n \log \frac{B}{\epsilon}\right)$	$\tilde{O}\left(n^3 \log \frac{B}{\epsilon}\right)$ [Lee et al 2015]
Central Path	2nd (and log like barrier for generalized inequalities)	f_0 smooth, self-conc. f_i quadratic $\lvert f_0\rvert, \lvert f_i\rvert \leq B$, existence of ϵ-strictly feasible $\tilde{x}$ $m\left\lVert \nabla f_i\left(x^{(0)}\right)\right\rVert \lVert\tilde{x}\rVert \leq B$	$\tilde{O}\left(\sqrt{m} \log \frac{B}{\epsilon}\right)$	$\tilde{O}\left(\sqrt{m} n^3 \log \frac{B}{\epsilon}\right)$

- Can we do better?
- Can we optimize with less than $\omega(n)$ 1st order accesses?
- Without assuming smoothness and self-concordance?
- Can we perform iterations faster?

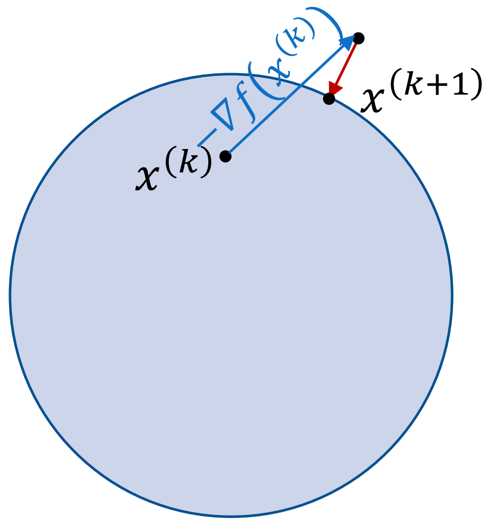
Method	Oracle	Assumptions	# accesses	Adtl. runtime
Center of Mass	1st	$\lvert f\rvert \le B$	$O\left(n \log \frac{B}{\epsilon}\right)$	NA
Ellipsoid			$O\left(n^2 \log \frac{B}{\epsilon}\right)$	$O\left(n^4 \log \frac{B}{\epsilon}\right)$
Vaidya++			$\tilde{O}\left(n \log \frac{B}{\epsilon}\right)$	$\tilde{O}\left(n^3 \log \frac{B}{\epsilon}\right)$
Grad Descent		$\mu \preccurlyeq \nabla^2 f \preccurlyeq M$ $\kappa = M/\mu$ $\lvert f\rvert \le B$	$O(\kappa \log {}^B/_\epsilon)$	$O(n\kappa \log {}^B/_\epsilon)$
Accelerated GD			$O(\sqrt{\kappa} \log {}^B/_\epsilon)$	$O(n\sqrt{\kappa} \log {}^B/_\epsilon)$
Grad Descent		$\nabla^2 f \preccurlyeq M$ $\lVert x^*\rVert \le R$	$O\left(\frac{MR^2}{\epsilon}\right)$	$O\left(n\frac{MR^2}{\epsilon}\right)$
Accelerated GD			$O\left(\sqrt{\frac{MR^2}{\epsilon}}\right)$	$O\left(n\sqrt{\frac{MR^2}{\epsilon}}\right)$
Grad Descent		$\lVert \nabla f\rVert \le L$ $\lVert x^*\rVert \le R$	$O\left(\frac{L^2R^2}{\epsilon^2}\right)$	$O\left(n\frac{L^2R^2}{\epsilon^2}\right)$
???			???	???
Newton	2nd	Smooth self-conc	$O(B \log\log {}^1/_\epsilon)$	$O(n^3 B \log\log {}^1/_\epsilon)$

Projected Gradient Descent

$$\min_{x \in \mathbb{R}^n} \quad f(x)$$
$$s.t. \quad x \in K$$

Init	$x^{(0)} \in K$
Iterate	$x^{(k+1)} \leftarrow \Pi_K\left(x^{(k)} - t^{(k)} \nabla f\left(x^{(k)}\right)\right)$

- Requires access to 1st order oracle
$$x \to f(x), \nabla f(x)$$
and "projection oracle" for K:
$$\Pi_K(x) = \arg\min_{y \in K} \|x - y\|_2$$



Projected Gradient Descent

$$\min_{x\in\mathbb{R}^n} \quad f(x)$$
$$s.t. \quad x \in K$$

Smooth: $t^{(k)} = 1/M$

Non-smooth: $t^{(k)} = \frac{R}{L\sqrt{k+1}}$

or $t^{(k)} = \frac{1}{\mu(k+1)}$

Init	$x^{(0)} \in K$
Iterate	$x^{(k+1)} \leftarrow \Pi_K\left(x^{(k)} - t^{(k)}\nabla f\left(x^{(k)}\right)\right)$

- Requires access to 1st order oracle

$$x \to f(x), \nabla f(x)$$

and "projection oracle" for K:

$$\Pi_K(x) = \arg\min_{y\in K} \|x-y\|_2$$

	$\mu \preccurlyeq \nabla^2 \preccurlyeq M$	$\nabla^2 \preccurlyeq M, \|x^*\| \le R$	$\|\nabla\| \le L, \|x^*\| \le R$	$\|\nabla\| \le L, \mu \preccurlyeq \nabla^2$
GD	$\kappa \log 1/\epsilon$	$\frac{M\|x^*\|^2}{\epsilon}$	$\frac{L^2\|x^*\|^2}{\epsilon^2}$	$\frac{L^2}{\mu\epsilon}$
A-GD	$\sqrt{\kappa}\log 1/\epsilon$	$\sqrt{\frac{M\|x^*\|^2}{\epsilon}}$		

Projection Oracles

$$\Pi_K(x) = \arg\min_{y \in K} \|x - y\|_2$$

- $K = \{x \in \mathbb{R}^n \,|\, \|x\|_2 \le R\}$

  $\Pi_K(x) = \frac{x}{\max\left(\frac{\|x\|_2}{R}, 1\right)}$ $O(n)$ time

- $K = \{x \in \mathbb{R}^n \,|\, Ax = b\}$

  ➔ projection onto the null-space $O(np)$ (after pre-processing A)

- $K = \{x \in \mathbb{R}^n \,|\, x \ge 0\}$

  $\Pi_K(x) = [x]_+$ $O(n)$ time

- $K = \{X \in S^n \,|\, X \succcurlyeq 0\}$

  positive eigen-components $O(n^3)$ time

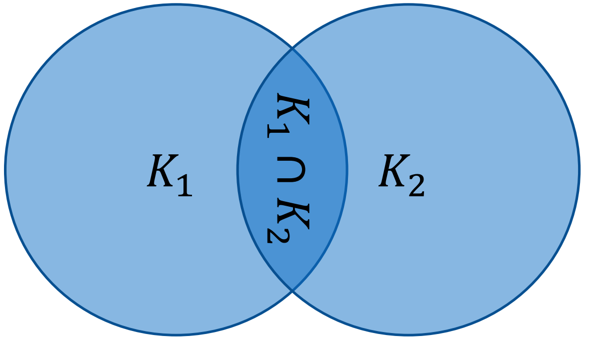
  $\Pi_K(X) = \sum_i [\lambda_i]_+ v_i v_i^\top$ where $X = \sum_i \lambda_i v_i v_i^\top$

- $K = \{x \in \mathbb{R}^n \,|\, Ax \le b\}$

  solve a QP (as hard as a generic QP)

- $K = K_1 \cap K_2$, e.g. $K = \{x \,|\, Ax = b, x \ge 0\}$

  in this case: solve a QP

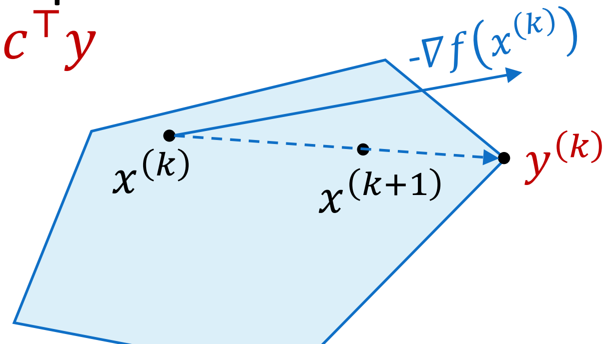


Conditional Gradient Descent (The Frank Wolfe Method)

- Gradient Descent motivated by optimizing 1st order approximation:
$$f(x) \approx f\left(x^{(k)}\right) + \left\langle \nabla f\left(x^{(k)}\right), x - x^{(k)} \right\rangle$$
- Optimize only over K: $y^{(k)} = \underset{y \in K}{\operatorname{argmin}} \left\langle \nabla f\left(x^{(k)}\right), y \right\rangle$
- Then take a step toward $y^{(k)}$: $x^{(k+1)} = x^{(k)} + t^{(k)}\left(y^{(k)} - x^{(k)}\right)$

Init	$x^{(0)} \in K$
Iterate	$y^{(k)} = \underset{y \in K}{\operatorname{argmin}} \left\langle \nabla f\left(x^{(k)}\right), y \right\rangle$
	$x^{(k+1)} \leftarrow x^{(k)} + t^{(k)}\left(y^{(k)} - x^{(k)}\right)$

Requires 1st order oracle for f, and linear optimization oracle for K:
$$c \mapsto \underset{y \in K}{\operatorname{argmin}}\, c^\top y$$



Conditional Gradient Descent

$$\boxed{\begin{array}{ll} \text{Init} & x^{(0)} \in K \\ \text{Iterate} & y^{(k)} = \underset{y \in K}{\operatorname{argmin}} \langle \nabla f(x^{(k)}), y \rangle \\ & x^{(k+1)} \leftarrow x^{(k)} + t^{(k)}\left(y^{(k)} - x^{(k)}\right) \end{array}}$$

Requires 1st order oracle for f, and linear optimization oracle for K:

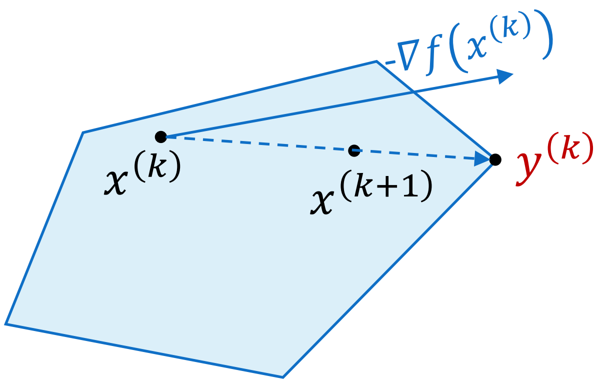
$$c \mapsto \underset{y \in K}{\operatorname{argmin}}\, c^\top y$$

- $K = \{x | Ax = b, Gx \le h\}$ ➔ solve an LP

  Reduces QP to a series of LPs

- $K = \{X \in S^n | 0 \preccurlyeq X, tr(X) \le 1\}$

  $\underset{X \in K}{\operatorname{argmin}} \langle X, C \rangle$ = eigenvector of $-C$ with max positive eigenvalue



Conditional Gradient Descent

Init	$x^{(0)} \in K$
Iterate	$y^{(k)} = \underset{y \in K}{\operatorname{argmin}} \langle \nabla f(x^{(k)}), y \rangle$
	$x^{(k+1)} \leftarrow x^{(k)} + t^{(k)}\left(y^{(k)} - x^{(k)}\right)$

Requires 1st order oracle for f, and linear optimization oracle for K:

$$c \mapsto \underset{y \in K}{\operatorname{argmin}}\, c^{\top} y$$

<u>Assumptions</u>: $\forall_{x \in K} \|x\| \le K$ and $\nabla^2 f(x) \preccurlyeq M$

Then, then with $t^{(k)} = \frac{2}{k+1}$, find ϵ-suboptimal after at most

$$k = O\left(\frac{MR}{\epsilon}\right) \text{ iterations}$$

Is strong convexity helpful? Can we get $\log 1/\epsilon$?

Non-smooth objectives?

Acceleration?

Method	Oracle	Assumptions	# accesses	Adtl. runtime
Center of Mass	1^{st} +Separation if needed	$\lvert f \rvert \le B$	$O\left(n \log \frac{B}{\epsilon}\right)$	NA
Ellipsoid			$O\left(n^2 \log \frac{B}{\epsilon}\right)$	$O\left(n^4 \log \frac{B}{\epsilon}\right)$
Vaidya++			$\tilde{O}\left(n \log \frac{B}{\epsilon}\right)$	$\tilde{O}\left(n^3 \log \frac{B}{\epsilon}\right)$
Grad Descent	+Projection if needed	$\mu \preccurlyeq \nabla^2 f \preccurlyeq M$ $\kappa = M/\mu$ $\lvert f \rvert \le B$	$O(\kappa \log {}^{B}/_{\epsilon})$	$O(n\kappa \log {}^{B}/_{\epsilon})$
Accelerated GD			$O(\sqrt{\kappa} \log {}^{B}/_{\epsilon})$	$O(n\sqrt{\kappa} \log {}^{B}/_{\epsilon})$
Grad Descent	+Projection or Linear Opt if needed	$\nabla^2 f \preccurlyeq M$ $\lVert x^* \rVert \le R$	$O\left(\frac{MR^2}{\epsilon}\right)$	$O\left(n \frac{MR^2}{\epsilon}\right)$
Accelerated GD			$O\left(\sqrt{\frac{MR^2}{\epsilon}}\right)$	$O\left(n \sqrt{\frac{MR^2}{\epsilon}}\right)$
Grad Descent	+Projection if needed	$\lVert \nabla f \rVert \le L$ $\lVert x^* \rVert \le R$	$O\left(\frac{L^2R^2}{\epsilon^2}\right)$	$O\left(n \frac{L^2R^2}{\epsilon^2}\right)$
???			???	???
Newton	2^{nd}	Smooth self-conc	$O(B \log\log {}^{1}/_{\epsilon})$	$O(n^3 B \log\log {}^{1}/_{\epsilon})$

- Computational Lower Bounds: "any Turing machine (or computer program) that solves the problem for every input, must make at least T computational steps for some inputs"
  - For any natural problem (in particular, any search problem in NP), can only get conditional lower bounds: "if (complexity assumption) then no efficient alg for X"
  - Optimization is in NP (Poly Verifiable): "Find x s.t. x is feasible and $f_0(x) \leq c$"
  - Very difficult to obtain (even conditional) polynomial lower bounds: NP-hard ➔ likely no poly-time. Much harder to prove "there exists n^3 alg but no n^2 alg".
- What's the input to optimization?
  - The objective function f? Code for the function?

  Uncomputable to even decides if it does something, let alone optimize.
- Oracle Lower Bounds:
  - "Any method that solves the problem (finds an ϵ suboptimal solution) for every f satisfying our assumptions, must call the oracle provided at least T times for some inputs"

- Can we get a lower bound for a specific optimization problem, e.g. specific objective $f(\cdot)$?
- **No**. For any specific f, there is always a very simple algorithm: "return x^*"
- Can maybe give lower bound on #access/runtime of a specific alg. A:
$$T(A, f) \geq T$$
but not lower bound for any algorithm:
$$\min_A T(A, f)$$
- Instead, need to discuss class of problems/objective:
$$\min_A \max_{f \in \mathcal{F}} T(A, f)$$
"Any algorithm must make at least T queries for some f satisfying the assumptions".
- Crucial to define class $\mathcal{F}$ of functions we are considering, e.g.:
$$\mathcal{F} = \{f: \mathbb{R}^n \to \mathbb{R} | \forall_x \mu \preccurlyeq \nabla^2 f(x) \preccurlyeq M\}$$
- To get such a lower bound, we need to show that for any possible method A, we can construct a "hard" $f \in \mathcal{F}$.
- How can we do this?


Jack
Mary
Walter
Christina
Melissa
Brett
Robert
Doug
Kelly
Timmy
Virgil
Sally
Lily
Pam
Steve
Zack
Melissa & Doug

Bear Hunt

$$\mathcal{F} = \left\{ f_b(x) = \begin{cases} 0 \; if \; b = x \\ 1 \; otherwise \end{cases} \middle| \; b \in Bears \right\}$$

Membership Oracle: $Q \subseteq Bears \rightarrow \delta_{b \in Q}$

<u>Claim</u>: for any (deterministic) algorithm A with access only to a membership oracle, there exists $f_b \in \mathcal{F}$ such that the algorithm must make $T \geq \lceil \log_2 |Bears| \rceil$ membership oracle queries before returning correct answer (0.5-suboptimal solution)

- To construct f_b based on A, we describe an adversary "playing" against A.
- Instead of picking bear in advance, adversary maintains set of plausible bears B consistent with all answers so far.
- For each query Q, provide answer and remove from B anything inconsistent.
- If algorithm outputs answer while $|B| > 1$, pick a different $b \in B$. f_b is the "hard" function for algorithm A.

Bear Hunt

- Initialize $B = Bears$ and simulate A
- On each query Q:

  If $|B \cap Q| > \frac{|B|}{2}$, answer “$b \in Q$”, $B \leftarrow B \cap Q$

  otherwise, answer “$b \notin Q$”, $B \leftarrow B \cap \overline{Q}$
- If A stops and outputs $\tilde{b}$ while $|B| > 1$, pick f_b s.t. $b \in B, b \neq \tilde{b}$.

Claim: after the simulation, for all $b \in B$, all answers are valid for input f_b

Claim: after T queries, $|B| \geq 2^{-T} \cdot |Bears|$

➔ if A makes $< \lceil \log_2 |Bears| \rceil$ queries, then $|B| > 1$

Conclusion: If the A always makes $< \lceil \log_2 |Bears| \rceil$ queries, it will be wrong on f_b

$$\boxed{\begin{array}{ll} \min\limits_{x \in \mathbb{R}} & f(x) \\ s.t. & -1 \leq x \leq 1 \end{array}}$$

Assumptions: f is convex and bounded, $|f(x)| \leq 1$

- Convenience trick: consider what A returns as the final query (now we just have to show all queries are at "bad" points)
- Goal: for any A, construct f that such that it will take A many queries before it queries at an ϵ-suboptimal point.

- Initialize "unexplored segment" $B_0 = [-1,1]$ and $f_0 = |x|$
- Simulate the algorithm, and for each query $x^{(k)}$,$k = 1..T$:
  - Update $B_k \subset B_{k-1}$ such that $x^{(k)} \notin B_k$
  - Update f_k by changing f_{k-1} only inside B_{k-1}

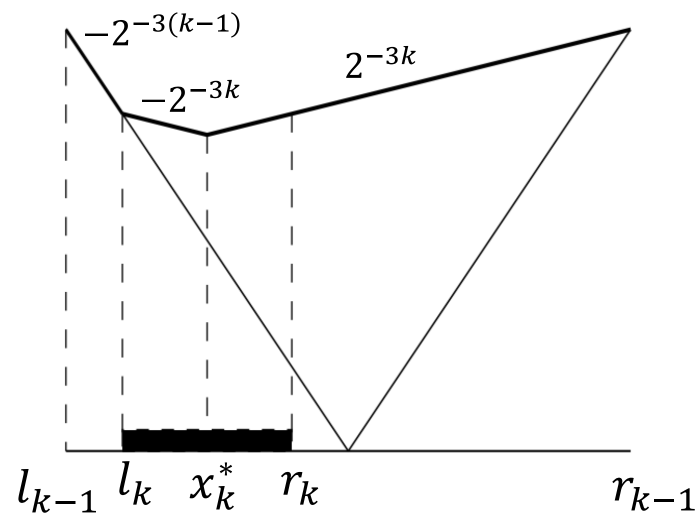
    This ensures all previous answers are still valid

    Also ensure: all ϵ-suboptimal points are in B_k
  - Answer query $x^{(k)}$ with $\nabla f_k\left(x^{(k)}\right)$

- Initialize "unexplored segment" $[l_0, r_0] = [-1,1]$ and $f_0 = |x|$

  We will always have $f_k(x) = 2^{-3k}\left|x - \frac{l_k+r_k}{2}\right| + a_k$ inside $[l_k, r_k]$

- Simulate the algorithm, and for each query $x^{(k)}$:
  - Set $[l_k, r_k] \leftarrow \left[l_{k-1} + \frac{1}{14}(r_{k-1} - l_{k-1}), l_{k-1} + \frac{6}{14}(r_{k-1} - l_{k-1})\right]$
    or $[l_k, r_k] \leftarrow \left[l_{k-1} + \frac{8}{14}(r_{k-1} - l_{k-1}), l_{k-1} + \frac{13}{14}(r_{k-1} - l_{k-1})\right]$
    s.t. $x^{(k)} \notin [l_{k+1}, r_{k+1}]$
  - Set $f_k(x) = f_{k-1}(x)$ for $x \notin [l_{k-1}, r_{k-1}]$ and as follows inside $[l_{k-1}, r_{k-1}]$:
  - Answer according to f_k



- <u>Claim</u>: f_k is convex, $|f_k(x)| \leq 1$, and answer $1..k$ are consistent with f_k

- <u>Claim</u>: $\forall x \notin [l_k, r_k]$, $f_k(x) \geq f_k(x_k^*) + 2^{-3k}\left(\frac{5}{14}\right)^k > f_k(x_k^*) + 2^{-5k}$

$$\boxed{\begin{array}{ll} \min_{x\in\mathbb{R}} & f(x) \\ s.t. & -R \le x \le R \end{array}}$$

Conclusion: for any algorithm A that uses a 1st order oracle and any ϵ, there exists a convex $f: [-1,1] \rightarrow \mathbb{R}$, $|f(x)| \le 1$, such that on input f, A calls the oracle at least $\frac{1}{5}\log_2\frac{1}{\epsilon} - 1$ times before returning an ϵ-suboptimal point.

By scaling $\tilde{f}(x) = B \cdot f(x/R)$:

for any algorithm A that uses a 1st order oracle and any B, R, ϵ, there exists a convex $f: [-R, R] \rightarrow \mathbb{R}$, $|f(x)| \le B$, such that on input f, A calls the oracle at least $\frac{1}{5}\log_2\frac{B}{\epsilon} - 1$ times before returning an ϵ-suboptimal point.

Would 2nd order oracle help?

Lower bound holds for 2nd, even 3rd, or any “local” oracle.