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
<table>
<thead>
<tr>
<th>Method</th>
<th>Oracle</th>
<th>Assumptions</th>
<th># accesses</th>
<th>Adtl. runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center of Mass</td>
<td>1(^{st}) /</td>
<td>(</td>
<td>f_0</td>
<td>,</td>
</tr>
<tr>
<td></td>
<td>separation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ellipsoid</td>
<td></td>
<td></td>
<td>(O\left(n^2 \log \frac{B}{\epsilon}\right))</td>
<td>(O\left(n^4 \log \frac{B}{\epsilon}\right))</td>
</tr>
<tr>
<td>Vaidya++</td>
<td></td>
<td></td>
<td>(\tilde{O}\left(n \log \frac{B}{\epsilon}\right)) [Lee et al 2015]</td>
<td>(\tilde{O}\left(n^3 \log \frac{B}{\epsilon}\right))</td>
</tr>
<tr>
<td>Central Path</td>
<td>2(^{nd}) (and log like</td>
<td>(f_0) smooth, self-conc. (f_i) quadratic (</td>
<td>f_0</td>
<td>,</td>
</tr>
<tr>
<td></td>
<td>barrier for generalized inequalities)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Can we do better?
- Can we optimize with less than \(\omega(n)\) 1\(^{st}\) order accesses?
- Without assuming smoothness and self-concordance?
- Can we perform iterations faster?
<table>
<thead>
<tr>
<th>Method</th>
<th>Oracle</th>
<th>Assumptions</th>
<th># accesses</th>
<th>Adtl. runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center of Mass</td>
<td>1st</td>
<td>$</td>
<td>f</td>
<td>\leq B$</td>
</tr>
<tr>
<td>Ellipsoid</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vaidya++</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Grad Descent</td>
<td></td>
<td>$\mu \leq \nabla^2 f \leq M$</td>
<td>$O\left(\kappa \log \frac{B}{\epsilon}\right)$</td>
<td>$O\left(n\kappa \log \frac{B}{\epsilon}\right)$</td>
</tr>
<tr>
<td>Accelerated GD</td>
<td></td>
<td>$\kappa = M/\mu$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Grad Descent</td>
<td></td>
<td>$\nabla^2 f \leq M$</td>
<td>$O\left(\frac{MR^2}{\epsilon}\right)$</td>
<td></td>
</tr>
<tr>
<td>Accelerated GD</td>
<td></td>
<td>$|x^*| \leq R$</td>
<td>$O\left(\sqrt{\frac{MR^2}{\epsilon}}\right)$</td>
<td></td>
</tr>
<tr>
<td>Grad Descent</td>
<td></td>
<td>$|\nabla f| \leq L$</td>
<td>$O\left(\frac{L^2 R^2}{\epsilon^2}\right)$</td>
<td></td>
</tr>
<tr>
<td>??</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Newton</td>
<td>2nd</td>
<td>Smooth self-conc</td>
<td>$O\left(B \log \log \frac{1}{\epsilon}\right)$</td>
<td>$O\left(n^3 B \log \log \frac{1}{\epsilon}\right)$</td>
</tr>
</tbody>
</table>
Projected Gradient Descent

\[
\begin{align*}
\min_{x \in \mathbb{R}^n} & f(x) \\
\text{s.t.} & x \in K
\end{align*}
\]

Init \quad x^{(0)} \in K

Iterate \quad x^{(k+1)} \leftarrow \Pi_K \left( x^{(k)} - t^{(k)} \nabla f(x^{(k)}) \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

\[
\min_{x \in \mathbb{R}^n} f(x) \\
\text{s.t. } x \in K
\]

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

\[
\begin{align*}
\text{Init} & \quad x^{(0)} \in K \\
\text{Iterate} & \quad x^{(k+1)} \leftarrow \Pi_K \left( x^{(k)} - t^{(k)} \nabla f(x^{(k)}) \right)
\end{align*}
\]

\[
\begin{align*}
\mu \leq \nabla^2 \leq M \\
\nabla^2 \leq M, \|x^*\| \leq R \\
\|\nabla\| \leq L, \|x^*\| \leq R \\
\|\nabla\| \leq L, \mu \leq \nabla^2
\end{align*}
\]

<table>
<thead>
<tr>
<th>Method</th>
<th>(u \leq \nabla^2 \leq M)</th>
<th>(\nabla^2 \leq M, |x^*| \leq R)</th>
<th>(|\nabla| \leq L, |x^*| \leq R)</th>
<th>(|\nabla| \leq L, \mu \leq \nabla^2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD</td>
<td>(\kappa \log 1/\epsilon)</td>
<td>(M |x^*|^2/\epsilon)</td>
<td>(L^2 |x^*|^2/\epsilon^2)</td>
<td>(L^2/\mu\epsilon)</td>
</tr>
<tr>
<td>A-GD</td>
<td>(\sqrt{\kappa} \log 1/\epsilon)</td>
<td>(\sqrt{M |x^*|^2}/\epsilon)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Projection Oracles

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

- \( K = \{ x \in \mathbb{R}^n | \| x \|_2 \leq R \} \)
  \[ \Pi_K(x) = \frac{x}{\max(\|x\|_2, 1)} \quad O(n) \text{ time} \]

- \( K = \{ x \in \mathbb{R}^n | Ax = b \} \)
  \[ \text{projection onto the null-space} \quad O(np) \text{ (after pre-processing } A) \]

- \( K = \{ x \in \mathbb{R}^n | x \geq 0 \} \)
  \[ \Pi_K(x) = [x]_+ \quad O(n) \text{ time} \]

- \( K = \{ X \in S^n | X \succeq 0 \} \)
  \[ \text{positive eigen-components} \quad O(n^3) \text{ time} \]
  \[ \Pi_K(X) = \sum_i [\lambda_i]_+ v_i v_i^\top \text{ where } X = \sum_i \lambda_i v_i v_i^\top \]

- \( K = \{ x \in \mathbb{R}^n | Ax \leq b \} \)
  solve a QP (as hard as a generic QP)

- \( K = K_1 \cap K_2 \), e.g. \( K = \{ x | Ax = b, x \geq 0 \} \)
  in this case: solve a QP
Conditional Gradient Descent  
(The Frank Wolfe Method)

- Gradient Descent motivated by optimizing 1\textsuperscript{st} order approximation:
  \[ f(x) \approx f(x^{(k)}) + \langle \nabla f(x^{(k)}), x - x^{(k)} \rangle \]

- Optimize only over \( K \):
  \[ y^{(k)} = \arg\min_{y \in K} \langle \nabla f(x^{(k)}), y \rangle \]

- Then take a step toward \( y^{(k)} \):
  \[ x^{(k+1)} = x^{(k)} + t^{(k)} (y^{(k)} - x^{(k)}) \]

\begin{align*}
\text{Init} & \quad x^{(0)} \in K \\
\text{Iterate} & \quad y^{(k)} = \arg\min_{y \in K} \langle \nabla f(x^{(k)}), y \rangle \\
& \quad x^{(k+1)} \leftarrow x^{(k)} + t^{(k)} (y^{(k)} - x^{(k)})
\end{align*}

Requires 1\textsuperscript{st} order oracle for \( f \), and linear optimization oracle for \( K \):
\[ c \mapsto \arg\min_{y \in K} c^\top y \]
Conditional Gradient Descent

<table>
<thead>
<tr>
<th>Init</th>
<th>$x^{(0)} \in K$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterate</td>
<td>$y^{(k)} = \arg\min_{y \in K} \langle \nabla f(x^{(k)}), y \rangle$</td>
</tr>
<tr>
<td></td>
<td>$x^{(k+1)} \leftarrow x^{(k)} + t^{(k)}(y^{(k)} - x^{(k)})$</td>
</tr>
</tbody>
</table>

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

$c \mapsto \arg\min_{y \in K} c^T y$

- $K = \{x | Ax = b, Gx \leq h\}$  \(\Rightarrow\) solve an LP
  Reduces QP to a series of LPs

- $K = \{X \in S^n | 0 \preceq X, \text{tr}(X) \leq 1\}$

  $\arg\min_{X \in K} \langle X, C \rangle = \text{eigenvector of } -C \text{ with max positive eigenvalue}$

$\nabla f(x^{(k)})$
Conditional Gradient Descent

Init \[ x^{(0)} \in K \]
Iterate \[ y^{(k)} = \arg\min_{y \in K} \langle \nabla f(x^{(k)}), y \rangle \]
\[ x^{(k+1)} \leftarrow x^{(k)} + t^{(k)}(y^{(k)} - x^{(k)}) \]

Requires 1\textsuperscript{st} order oracle for \( f \), and linear optimization oracle for \( K \):
\[ c \mapsto \arg\min_{y \in K} c^T y \]

Assumptions: \( \forall x \in K \|x\| \leq K \) and \( \nabla^2 f(x) \leq M \)

Then, then with \( t^{(k)} = \frac{2}{k+1} \), find \( \varepsilon \)-suboptimal after at most
\[ k = O\left(\frac{MR}{\varepsilon}\right) \text{ iterations} \]

Is strong convexity helpful? Can we get \( \log 1/\varepsilon \)?

Non-smooth objectives?

Acceleration?
<table>
<thead>
<tr>
<th>Method</th>
<th>Oracle</th>
<th>Assumptions</th>
<th># accesses</th>
<th>Adtl. runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center of Mass</td>
<td>1&lt;sup&gt;st&lt;/sup&gt;</td>
<td>$</td>
<td>f</td>
<td>\leq B$</td>
</tr>
<tr>
<td>Ellipsoid</td>
<td>+Separation if needed</td>
<td>$\gamma_{1}, \gamma_{2}, \gamma_{3}$, $\gamma_{4}$</td>
<td>$O\left(n^2 \log \frac{B}{\epsilon}\right)$</td>
<td>$O\left(n^4 \log \frac{B}{\epsilon}\right)$</td>
</tr>
<tr>
<td>Vaidya++</td>
<td>+Projection if needed</td>
<td>$\gamma_{1}, \gamma_{2}, \gamma_{3}, \gamma_{4}$</td>
<td>$\tilde{O}\left(n \log \frac{B}{\epsilon}\right)$</td>
<td>$\tilde{O}\left(n^3 \log \frac{B}{\epsilon}\right)$</td>
</tr>
<tr>
<td>Grad Descent</td>
<td>+Projection or Linear Opt if needed</td>
<td>$\mu \leq \nabla^2 f \leq M$ $\kappa = M / \mu$ $</td>
<td>f</td>
<td>\leq B$</td>
</tr>
<tr>
<td>Accelerated GD</td>
<td>+Projection or Linear Opt if needed</td>
<td>$\nabla^2 f \leq M$ $|x^*| \leq R$</td>
<td>$O\left(\sqrt{\kappa} \log \frac{B}{\epsilon}\right)$</td>
<td>$O(n \sqrt{\kappa} \log \frac{B}{\epsilon})$</td>
</tr>
<tr>
<td>Grad Descent</td>
<td>+Projection or Linear Opt if needed</td>
<td>$\nabla^2 f \leq M$ $|x^*| \leq R$</td>
<td>$O\left(\frac{MR^2}{\epsilon}\right)$</td>
<td>$O\left(n \frac{MR^2}{\epsilon}\right)$</td>
</tr>
<tr>
<td>Accelerated GD</td>
<td>+Projection or Linear Opt if needed</td>
<td>$\nabla^2 f \leq M$ $|x^*| \leq R$</td>
<td>$O\left(\sqrt{\frac{MR^2}{\epsilon}}\right)$</td>
<td>$O\left(n \sqrt{\frac{MR^2}{\epsilon}}\right)$</td>
</tr>
<tr>
<td>Grad Descent</td>
<td>+Projection if needed</td>
<td>$|\nabla f| \leq L$ $|x^*| \leq R$</td>
<td>$O\left(\frac{L^2 R^2}{\epsilon^2}\right)$</td>
<td>$O\left(n \frac{L^2 R^2}{\epsilon^2}\right)$</td>
</tr>
<tr>
<td>??</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Newton</td>
<td>2&lt;sup&gt;nd&lt;/sup&gt;</td>
<td>Smooth self-conc</td>
<td>$O\left(B \log \log \frac{1}{\epsilon}\right)$</td>
<td>$O\left(n^3 B \log \log \frac{1}{\epsilon}\right)$</td>
</tr>
</tbody>
</table>
• 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 $\implies$ 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 $\epsilon$ 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.:

  $\mathcal{F} = \{f : \mathbb{R}^n \to \mathbb{R} | \forall x \mu \leq \nabla^2 f(x) \leq M \}$

• 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?
Bear Hunt

\[ \mathcal{F} = \left\{ f_b(x) = \begin{cases} 0 & \text{if } b = x \\ 1 & \text{otherwise} \end{cases} \right\} \quad b \in Bears \]

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

Claim: 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, $|B| \geq 2^{-T} \cdot |Bears|$
  $\Rightarrow$ if $A$ makes $< [\log_2 |Bears|]$ queries, then $|B| > 1$

Conclusion: If the $A$ always makes $< [\log_2 |Bears|]$ queries, it will be wrong on $f_b$
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 $f$ such that it will take $A$ many queries before it queries at an $\epsilon$-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 $\epsilon$-suboptimal points are in $B_k$
  - Answer query $x^{(k)}$ with $\nabla f_k(x^{(k)})$
• 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 \(f_k\)

\[\text{Claim: } f_k \text{ is convex, } |f_k(x)| \leq 1, \text{ and answer } 1 \ldots k \text{ are consistent with } f_k\]

\[\text{Claim: } \forall x \notin [l_k, r_k], f_k(x) \geq f_k(x^*_k) + 2^{-3k} \left( \frac{5}{14} \right)^k > f_k(x^*_k) + 2^{-5k}\]
Conclusion: for any algorithm $A$ that uses a 1st order oracle and any $\epsilon$, there exists a convex $f: [-1,1] \to \mathbb{R}$, $|f(x)| \leq 1$, such that on input $f$, $A$ calls the oracle at least $\frac{1}{5} \log_2 \frac{1}{\epsilon} - 1$ times before returning an $\epsilon$-suboptimal point.

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

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

Would 2nd order oracle help?

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

$$\min_{x \in \mathbb{R}} f(x)$$
$$\text{s.t. } -R \leq x \leq R$$