Lecture 16: Frank-Wolfe (aka Conditional Gradient) Method

Yudong Chen

1 Setup

Consider the constrained problem
\[
\min_{x \in \mathcal{X}} f(x), \quad (P)
\]
We still assume that \( f \) is \( L \)-smooth and convex, and \( \mathcal{X} \) is closed, convex and non-empty.

In many settings, computing projection onto \( \mathcal{X} \) is expensive, but linear optimization \( \min_{x \in \mathcal{X}} c^\top x \) is easy. This is typical when \( \mathcal{X} \) is a polytope \( \{ x \in \mathbb{R}^d : a_i^\top x \leq b_i, i = 1, \ldots, m \} \).

Examples:
- Probability simplex and \( \ell_1 \) ball: Projection uses \( \Theta(d \log d) \) arithmetics operations (sorting).
  Linear optimization oracle only takes \( \Theta(d) \) (finding the smallest element of the gradient \( c \)).
  This is not a dramatic difference, but linear optimization has other benefits such as sparsity of solution. See Section 5.
- For some polytopes, projection (exactly) is computationally hard, but LP is poly-time. E.g.,
  matching polytope for a general graph with \( |V| \) vertices has \( \sim 2^{|V|} \) constraints, but LP is tractable (e.g., using Edmonds’ algorithm).

Frank-Wolfe (FW) method uses a linear optimization oracle instead of a projection oracle.

2 Frank-Wolfe method

**Algorithm 1** Frank-Wolfe

- Input: initial point \( x_0 \in \mathcal{X} \), algorithm parameters \( a_k > 0, k = 0, 1, \ldots \)
- For \( k = 0, 1, \ldots \)
  \[ v_k = \arg\min_{u \in \mathcal{X}} \langle \nabla f(x_k), u \rangle, \]
  \[ x_{k+1} = \frac{A_k - 1}{A_k} x_k + \frac{a_k}{A_k} v_k, \]
  where \( A_k = \sum_{i=0}^k a_i = A_{k-1} + a_k. \)

Observe that \( v_k \in \mathcal{X} \) by definition, hence
\[ x_{k+1} = \left( 1 - \frac{a_k}{A_k} \right) x_k + \frac{a_k}{A_k} v_k \in \mathcal{X}, \quad \forall k \]
by convexity of \( \mathcal{X} \) and induction.
3 Convergence rate of Frank-Wolfe

We introduce a new style of analysis.

1. We will maintain an upper bound $U_k \geq f(x_{k+1})$ and a lower bound $L_k \leq f(x^*)$. Consequently, the difference $G_k := U_k - L_k$ is an upper bound on the optimality gap $f(x_{k+1}) - f(x^*)$.

2. Recall that $A_k := \sum_{i=0}^{k} a_i$, which is strictly increasing in $k$. We will show that

$$A_k G_k \leq A_{k-1} G_{k-1} + E_k,$$

where $E_k$ is some “error” term. This implies that

$$G_k \leq \frac{A_0 G_0 + \sum_{i=1}^{k} E_i}{A_k}.$$

3. We will choose $\{a_k\}$ so that $A_0 G_0 + \sum_{i=1}^{k} E_i$ grows slowly with $k$ compared to $A_k$, hence $G_k$ converges to 0 quickly.

Let us apply the above strategy to FW.

**Upper bound:** Simply take $U_k = f(x_{k+1})$. Then

$$A_k U_k - A_{k-1} U_{k-1} = A_k f(x_{k+1}) - A_{k-1} f(x_k).$$

**Lower bound:** We have

$$f(x^*) \geq \frac{1}{A_k} \sum_{i=0}^{k} a_i \left( f(x_i) + \langle \nabla f(x_i), x^* - x_i \rangle \right) \quad \text{convexity of } f$$

$$\geq \frac{1}{A_k} \sum_{i=0}^{k} a_i f(x_i) + \frac{1}{A_k} \sum_{i=0}^{k} a_i \min_{u \in \mathcal{X}} \langle \nabla f(x_i), u - x_i \rangle$$

$$= \frac{1}{A_k} \sum_{i=0}^{k} a_i f(x_i) + \frac{1}{A_k} \sum_{i=0}^{k} a_i \langle \nabla f(x_i), v_i - x_i \rangle \quad \text{definition of } v_i$$

$$=: L_k.$$

Then

$$A_k L_k - A_{k-1} L_{k-1} = a_k f(x_k) + a_k \langle \nabla f(x_k), v_k - x_k \rangle.$$

**Evolution of $A_k G_k$:** Define $D := \max_{x, y \in \mathcal{X}} \|x - y\|_2$, which is the diameter of $\mathcal{X}$. Then for $k \geq 1$:

$$A_k G_k - A_{k-1} G_{k-1}$$
$$= (A_k U_k - A_{k-1} U_{k-1}) - (A_k L_k - A_{k-1} L_{k-1})$$
$$= A_k \left( f(x_{k+1}) - f(x_k) \right) - a_k \langle \nabla f(x_k), v_k - x_k \rangle$$

$$\leq A_k \langle \nabla f(x_k), x_{k+1} - x_k \rangle + \frac{A_k L}{2} \|x_{k+1} - x_k\|_2^2 - a_k \langle \nabla f(x_k), v_k - x_k \rangle \quad \text{smoothness of } f$$

$$\leq \frac{a_k^2 L}{2A_k} \|v_k - x_k\|_2^2$$

$$\leq \frac{a_k^2 L}{2A_k} D^2, \quad \text{this is } E_k$$

(1)
where (i) holds because

\[ x_{k+1} = \frac{A_{k-1}}{A_k} x_k + \frac{a_k}{A_k} v_k \iff A_k (x_{k+1} - x_k) = a_k (v_k - x_k) \iff x_{k+1} - x_k = \frac{a_k}{A_k} (v_k - x_k). \]

(Exercise) Using similar argument as above, verify yourself that

\[ A_0 G_0 \leq \frac{a_0^2 L}{2 A_0} D^2. \]  

**Final bound:** Summing (1) over \( k \) and (2), we get

\[ A_k G_k \leq \sum_{i=0}^{k} \frac{a_i^2 L}{2 A_i} D^2 \iff f(x_{k+1}) - f(x^*) \leq G_k \leq \frac{L D^2}{2} \cdot \frac{1}{A_k} \sum_{i=0}^{k} \frac{a_i^2}{A_i}. \]

We want to choose \( \{a_i\} \) to make RHS to decay fast with \( k \). Different choices work, but whenever you see something like \( \frac{a_i^2}{A_i} \), you should try \( a_i \propto i^2, \frac{a_i}{A_i} \approx 1 \). In particular, setting \( a_i = i + 1 \), we have \( A_i = \frac{(i+1)(i+2)}{2} \) and hence

\[ f(x_{k+1}) - f(x^*) \leq \frac{L D^2}{(k+1)(k+2)} \sum_{i=0}^{k} \frac{2(i+1)^2}{(i+1)(i+2)} \leq \frac{2L D^2}{k+2}. \]

Therefore, we get an \( O \left( \frac{LD^2}{\epsilon} \right) \) convergence rate. Equivalently, FW achieves \( f(x_k) - f(x^*) \leq \epsilon \) after at most \( O \left( \frac{LD^2}{\epsilon} \right) \) iterations.

### 4 Lower bound

Is it possible to beat FW? Not in the worst case, if we are only accessing \( \mathcal{X} \) via a linear optimization oracle.

**Theorem 1.** Consider any algorithm that accesses the feasible set \( \mathcal{X} \) only via a linear optimization oracle. There exists an \( L \)-smooth convex function function \( f : \mathbb{R}^d \to \mathbb{R} \) such that this algorithm requires at least

\[
\min \left\{ \frac{d}{2}, \frac{LD^2}{16\epsilon} \right\}
\]

iterations (i.e., calls to the linear optimization oracle) to construct a point \( \hat{x} \in \mathcal{X} \) with \( f(\hat{x}) - \min_{x \in \mathcal{X}} f(x) \leq \epsilon \). The lower bound applies even if \( f \) is strongly convex.

**Proof sketch.** Take \( f(x) = \frac{1}{d} \|x\|_2^2 \) and \( \mathcal{X} = \left\{ x \in \mathbb{R}^d : x \geq 0, \sum_{i=1}^d x_i = 1 \right\} \) (the probability simplex). Note that the smoothness parameter of \( f \) is \( L = 1 \), the diameter of \( \mathcal{X} \) is \( D = 2 \), and \( f \) is strongly convex. Moreover, the optimal solution and value are

\[ x^* = \frac{1}{d} \mathbf{1} = \frac{1}{d} \sum_{i=1}^d e_i, \quad f(x^*) = \frac{1}{2d}. \]
where \( e_i = (0, \ldots, 0, 1, 0, \ldots, 0)^T \) denotes the \( i \)-th standard basis vector.

Linear optimization over the polytope \( \mathcal{X} \) returns one of its vertex \( e_i \). After \( k \) iterations, one would only uncover \( k \) basis vectors \( e_{i_1}, e_{i_2}, \ldots, e_{i_k} \). The best solution one can construct from them is \( \hat{x} = \frac{1}{k} \sum_{j=1}^{k} e_{i_j} \), hence

\[
 f(\hat{x}) - f(x^*) \geq \frac{1}{2} \left( \frac{1}{\min\{k,d\}} - \frac{1}{d} \right).
\]

To make the RHS \( \leq \epsilon \), we need \( k \geq \min \left\{ \frac{d}{2}, \frac{1}{4\epsilon} \right\} = \min \left\{ \frac{d}{2}, \frac{LD^2}{16\epsilon} \right\} \).

See Lan ’13 for the complete proof.

5 Additional remarks

FW was out of favor for a long time, as it has sublinear convergence even when \( f \) is strongly convex. However, there has been a recent upsurge of activity on FW.

- A sublinear rate is acceptable in many machine learning and data science problems with large-scale and noisy data.

- The optimal solution \( v_k \) of linear optimization lies at a vertex of the feasible set \( \mathcal{X} \). Such a solution often has certain sparsity properties not possessed by projection onto \( \mathcal{X} \). Sparsity often leads to better computational and statistical efficiency. For example:

  - When \( \mathcal{X} \) is the probability simplex or \( \ell_1 \) ball, each \( v_i \) is 1-sparse (has only 1 nonzero entry). Consequently, the iterate \( x_k \) of FW is \( k \)-sparse since it is a convex combination of \( \{v_1, \ldots, v_k\} \).

  - The nuclear norm \( \|x\|_{\text{nuc}} \) of a matrix \( x \) is defined as the sum of its singular values. When \( \mathcal{X} = \{x \in \mathbb{R}^{d \times d} : \|x\|_{\text{nuc}} \leq R \} \) is the nuclear norm ball, each \( v_i \) is a rank-1 matrix, hence \( x_k \) has rank at most \( k \).

- Conservative Policy Iteration (CPI), a basic algorithm in Reinforcement Learning, is an incarnation of FW. See this short paper on the connection between several reinforcement learning and constrained optimization algorithms (including CPI and FW).