Optimization for Data Science

F. Rinaldi[1]

1 Università degli Studi di Padova | Dipartimento Matematica

Padova
2020

Outline

Optimization for Data Science

Frank-Wolfe and Friends

- The Frank-Wolfe method (aka conditional gradient method or reduced gradient method) is an iterative first-order optimization algorithm.
- Originally proposed by Marguerite Frank and Philip Wolfe in 1956 to solve quadratic programming problems with linear constraints.
- It has seen an impressive revival recently due to its nice properties compared to projected gradient methods, in particular for machine learning applications.
- We describe in depth the method and its theoretical properties.
- We explain why people in data science use this method in practice.

Our Problem

Problem

$$\begin{aligned} &\min f(x) \\ &x \in C \end{aligned} \tag{1}$$

- $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function with Lipschitz continuous gradient having constant $L > 0$;
- $C \subseteq \mathbb{R}^n$ is a convex compact (i.e., closed and bounded) set.

- *Diameter* of the set C is

$$D = \max_{x,y \in C} \|x - y\|_2.$$

- From compactness of C, we have that D is a finite value.

A Useful Result

Weierstrass Theorem

If we minimize a continuous function over a compact set, we can always be sure there exists a global minimum for the problem.

Description of the Algorithm

- We start with a feasible solution.
- At each iteration, we define a descent direction in the current iterate x_k by solving the problem:

$$\min_{x \in C} \nabla f(x_k)^\top (x - x_k).$$

- This is equivalent to minimize the linear approximation of f in x_k:

$$\min_{x \in C} f(x_k) + \nabla f(x_k)^\top (x - x_k).$$

- From compactness of C, we have that there exists a solution $\hat{x}_k \in C$ for the linearized problem.

Search Direction Analysis

- **CASE 1)** $\nabla f(x_k)^\top(\hat{x}_k - x_k) = 0.$

  Then we have

  $$0 = \nabla f(x_k)^\top(\hat{x}_k - x_k) \leq \nabla f(x_k)^\top(x - x_k) \quad \forall\, x \in C$$

  and x_k satisfies first order optimality conditions.

- **CASE 2)** $\nabla f(x_k)^\top(\hat{x}_k - x_k) < 0.$

  we have a new descent direction in x_k:

  $$d_k = \hat{x}_k - x_k.$$

  Thus we can have a new iterate

  $$x_{k+1} = x_k + \alpha_k d_k$$

  with $\alpha_k \in (0, 1]$ calculated by means of a line search.

Scheme of the Method

Algorithm 1 `Frank-Wolfe method`

Choose a point $x_1 \in C$
For $k = 1, \ldots$
    Set $\hat{x}_k = \operatorname*{Argmin}_{x \in C} \nabla f(x_k)^\top (x - x_k)$
    If $\hat{x}_k$ satisfies some specific condition, then STOP
    Set $x_{k+1} = x_k + \alpha_k(\hat{x}_k - x_k)$, with $\alpha_k \in (0, 1]$ suitably chosen stepsize
End for

Convergence Result

Theorem [Frank-Wolfe Convergence]

Let f be a convex function with Lipschitz continuous gradient having constant $L > 0$. The Frank-Wolfe method with stepsize $\alpha_k = \frac{2}{k+1}$ satisfies the following inequality:

$$f(x_{k+1}) - f(x^*) \leq \frac{2LD^2}{k+1}, \tag{2}$$

for all $k \geq 1$.

- We use diminishing stepsize $\alpha = \dfrac{2}{k+1}$ in the theorem.

Proof.

By using first order convexity properties, we have

$$f(x) \geq f(x_k) + \nabla f(x_k)^\top (x - x_k), \quad \forall\, x \in C.$$

Minimizing on both sides of the inequality over C, we get

$$f(x^*) \geq f(x_k) + \nabla f(x_k)^\top (\hat{x}_k - x_k),$$

that can be rewritten as

$$f(x^*) - f(x_k) \geq \nabla f(x_k)^\top (\hat{x}_k - x_k),$$

or equivalently as follows

$$-\left(f(x_k) - f(x^*)\right) \geq \nabla f(x_k)^\top (\hat{x}_k - x_k). \tag{3}$$

Proof II

We consider the point

$$x_{k+1} = x_k + \alpha_k d_k = x_k + \alpha_k(\hat{x}_k - x_k).$$

Using Lipschitz continuity of the gradient

$$f(x_{k+1}) \leq f(x_k) + \alpha_k \nabla f(x_k)^\top (\hat{x}_k - x_k) + \frac{\alpha_k^2 L}{2} \|x_k - \hat{x}_k\|^2$$

an inequality

$$-(f(x_k) - f(x^*)) \geq \nabla f(x_k)^\top (\hat{x}_k - x_k)$$

we write the chain of inequalities (start by using first inequality above):

$$\begin{aligned}
f(x_{k+1}) - f(x^*) \quad &\leq \quad f(x_k) + \alpha_k \nabla f(x_k)^\top (\hat{x}_k - x_k) + \frac{\alpha_k^2 L}{2} \|x_k - \hat{x}_k\|^2 - f(x^*) \\
&\quad\quad \text{(Use second inequality)} \\
&\leq \quad f(x_k) - f(x^*) - \alpha_k (f(x_k) - f(x^*)) + \frac{\alpha_k^2 L}{2} \|x_k - \hat{x}_k\|^2 \\
&\leq \quad (1 - \alpha_k)(f(x_k) - f(x^*)) + \frac{\alpha_k^2 L}{2} \|x_k - \hat{x}_k\|^2 \\
&\quad\quad \text{(Use Diameter)} \\
&\leq \quad (1 - \alpha_k)(f(x_k) - f(x^*)) + \frac{\alpha_k^2 L D^2}{2}.
\end{aligned}$$

Proof III

We set $r_{k+1} = f(x_{k+1}) - f(x^*)$ and rewrite

$$r_{k+1} \leq (1 - \alpha_k) r_k + \frac{\alpha_k^2 L D^2}{2}. \tag{4}$$

By induction we can show

$$r_{k+1} \leq \frac{2LD^2}{k+1}.$$

PART 1) We first prove that the inequality holds for $k = 1$. By means of inequality (4), we get

$$r_2 \leq (1 - \alpha_1) r_1 + \frac{\alpha_1^2 L D^2}{2}.$$

Since $\alpha_1 = 1$, we have

$$r_2 \leq \frac{LD^2}{2} \leq LD^2.$$

Proof IV

PART 2) Now we assume that inequality $r_{k+1} \leq \frac{2LD^2}{k+1}$ holds for any $k \geq 1$, we want to show that it holds also for $k+1$.
By using inequality

$$r_{k+1} \leq (1-\alpha_k)r_k + \frac{\alpha_k^2 LD^2}{2},$$

again, we get

$$\begin{aligned}
r_{k+2} &\leq (1-\alpha_{k+1})r_{k+1} + \frac{\alpha_{k+1}^2 LD^2}{2} \\
&\leq \left(1 - \frac{2}{k+2}\right)\frac{2LD^2}{k+1} + \frac{LD^2}{2}\left(\frac{2}{k+2}\right)^2 \\
&= 2LD^2\left(\frac{k}{(k+1)(k+2)} + \frac{1}{(k+2)^2}\right) \\
&\quad \text{(using the fact that } k+1 \leq k+2\text{)} \\
&\leq 2LD^2\left(\frac{k}{(k+1)(k+2)} + \frac{1}{(k+1)(k+2)}\right) \\
&= \frac{2LD^2}{k+2}.
\end{aligned}$$

Thus concluding the proof. □

Comments

- This result implies that the convergence rate is $\mathcal{O}(\frac{1}{k})$.
- It is possible to improve the rate (i.e., getting a linear rate) if we make stronger assumptions on the problem, that is:
  - feasible sets with special structure (like, e.g., polytopes);
  - function is σ-strongly convex;
  - optimal solution in the interior.

Iteration of Frank-Wolfe Method

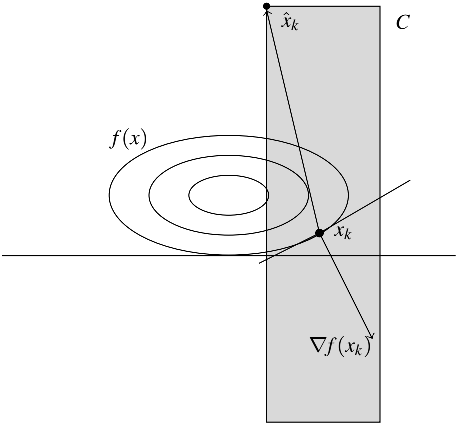

Figure: Iteration of the Frank-Wolfe method.

Duality for Beginners

Simple Dual Function

For a given point $x \in C$ we can define a simple dual function:

$$w(x) = \min_{z \in C} f(x) + \nabla f(x)^\top (z - x).$$

This minimum is always attained since C is compact and the linear function is continuous in z.

Weak Duality Result

For all pairs $x, y \in C$ it holds that

$$w(x) \leq f(y).$$

Proof. We have (by using definition of minimum and first order convexity):

$$\begin{aligned} w(x) &= \min_{z \in C} f(x) + \nabla f(x)^\top (z - x) \\ &\leq f(x) + \nabla f(x)^\top (y - x) \\ &\leq f(y). \end{aligned}$$

□

Duality Gap and Stopping Condition

Duality Gap

For a given point $x \in C$ we can define the duality gap:

$$g(x) = f(x) - w(x) = \max_{z \in C} \nabla f(x)^\top (x - z) = -\min_{z \in C} \nabla f(x)^\top (z - x).$$

- By the weak duality result we have

$$g(x) \geq f(x) - f(x^*) \geq 0, \quad \forall\, x \in C.$$

- Since primal error (i.e., $f(x) - f(x^\star)$) is not computable (x^* unknown), duality gap represents a good optimality measure, e.g. as a stopping criterion.
- IMPORTANT: We have $g(x)$ for free at each iteration (from problem at Step 3).
- So, if we want a primal gap $0 \leq f(x) - f(x^*) \leq \epsilon$, we need

$$f(x) - f(x^*) \leq g(x) \leq \epsilon.$$

- Thus we can stop the method when

$$\nabla f(x_k)^\top (\hat{x}_k - x_k) \geq -\epsilon.$$

Why Using Frank-Wolfe Method?

The Frank-Wolfe method is really appealing in the machine learning context for two main reasons:

- the cost per iteration is much smaller than the one we have for projected gradient method (Frank-Wolfe method is a projection-free algorithm);
- the iterates keep desirable structure (like, e.g., sparsity).

Cost per Iteration

Easy to see that solving problem at Step 2 of the algorithm costs less than projecting over C!

Example. If we think about a polyhedral feasible set, at each iteration

- Frank-Wolfe solves a linear program;
- Projection is equivalent to solve a quadratic programming problem.

Frank-Wolfe Iteration

At each iteration of the algorithm we solve the problem:

$$\hat{x}_k = \underset{x \in C}{\text{Argmin}}\, \nabla f(x_k)^\top (x - x_k)$$

- When C is a polytope, by means of the fundamental theorem of linear programming, one of the vertices is solution of the linear program.
- Frank-Wolfe iteration in some cases has linear cost.

Unit Simplex

Unit Simplex

feasible set is

$$C = \{x \in R^n : \; e^\top x = 1, \; x \geq 0\} = conv(\{e_i, \; i = 1, \ldots, n\});$$

solution in this case is

$$\hat{x}_k = e_{i_k},$$

with $i_k = \operatorname*{Argmin}_i \nabla_i f(x_k)$.

- It is easy to see that cost per iteration is $\mathcal{O}(n)$.

A 3D Simplex

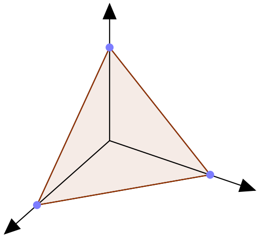

Figure: Unit simplex.

Scheme of Frank-Wolfe Method for Unit Simplex

Algorithm 2 `Frank-Wolfe method for unit simplex`

1 Set $x_1 = e_i$, with $i = 1, \ldots, n$
2 For $k = 1, \ldots$
3 Set $\hat{x}_k = e_{i_k}$, with $i_k = \operatorname{Argmin}_i \nabla_i f(x_k)$.
4 If $\hat{x}_k$ satisfies some specific condition, then STOP
5 Set $x_{k+1} = x_k + \alpha_k(\hat{x}_k - x_k)$, with $\alpha_k = \frac{2}{k+1}$
6 End for

ℓ_1-ball

ℓ_1-ball

Feasible set is

$$C = \{x \in R^n : \|x\|_1 \leq 1\} = conv(\{\pm e_i,\ i = 1, \ldots, n\});$$

solution in this case is

$$\hat{x}_k = sign(-\nabla_{i_k} f(x_k)) \cdot e_{i_k},$$

with $i_k = \operatorname*{Argmax}_i |\nabla_i f(x_k)|$.

- It is easy to see that cost per iteration is $\mathcal{O}(n)$.

A 3D ℓ_1 Ball

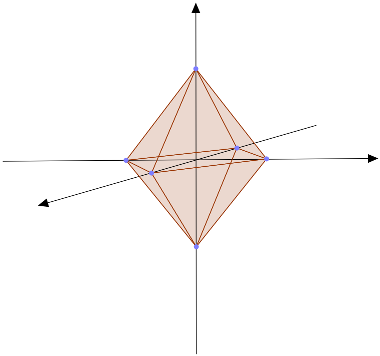

Figure: ℓ_1 ball.

Scheme of Frank-Wolfe Method for ℓ_1 Ball

Algorithm 3 `Frank-Wolfe method for` ℓ_1 `ball`

1 Set $x_1 = \pm e_i$, with $i = 1, \ldots, n$

2 For $k = 1, \ldots$

3 Set $\hat{x}_k = sign(-\nabla_{i_k} f(x_k)) \cdot e_{i_k}$, with $i_k = \operatorname{Argmax}_i |\nabla_i f(x_k)|$.

4 If $\hat{x}_k$ satisfies some specific condition, then STOP

5 Set $x_{k+1} = x_k + \alpha_k(\hat{x}_k - x_k)$, with $\alpha_k = \frac{2}{k+1}$

6 End for

Comments

- It is easy to see that at most one new nonzero is included at each iteration in both cases.
- Support of the solution (i.e., number of nonzero component of x_k) is upper bounded by k.
- Frank-Wolfe can be seen in this case as a variant of coordinate descent (we use coordinate directions at each step).

Re-parameterization of Feasible Set

Re-parameterization of C

If we choose a re-parameterization of C by a surjective linear or affine map $M : \hat{C} \to C$ then

$$\min_{x \in C} f(x) \equiv \min_{y \in \hat{C}} \hat{f}(y)$$

with $\hat{f}(y) = f(My)$.

- Every iteration of FW Algorithm remains the same (thanks to $\nabla \hat{f}(y) = M^\top \nabla f(My)$).
- Frank-Wolfe is invariant under "distortion" (existing approaches in optimization strongly depend on distortion of the domain).
- To better understand this concept, we consider problem

$$\begin{aligned} &\min f(x) \\ &x \in C = conv\{v_1, \ldots, v_p\}. \end{aligned} \tag{5}$$

- We use *bary-centric coordinates* to reparameterize the problem.
- In this case M contains the vertices as columns

$$M = [v_1 \ldots v_p]$$

and $\hat{C}$ is just the unit simplex.

Re-parameterization of Feasible Set

- Solving the original problem using the Frank-Wolfe method is equivalent to solve (by means of Frank-Wolfe) the reparameterized problem

$$\begin{array}{ll} \min & f(My) \\ s.t. & e^\top y = 1 \\ & y \geq 0 \end{array} \tag{6}$$

- Indeed, at each iteration we have

$$\begin{array}{ll} \min & \nabla \hat{f}(y_k)^\top (y - y_k) \\ s.t. & e^\top y = 1 \\ & y \geq 0 \end{array} \tag{7}$$

and by taking into account the expression of $\hat{f}$, we get

$$\nabla \hat{f}(y_k)^\top (\hat{y}_k - y_k) = \Big(M^\top \nabla f(My_k)\Big)^\top (\hat{y}_k - y_k) = \nabla f(My_k)^\top (M\hat{y}_k - My_k).$$

- By further considering that $x_k = My_k$ and $\hat{x}_k = My_k$, we get

$$\nabla \hat{f}(y_k)^\top (\hat{y}_k - y_k) = \nabla f(x_k)^\top (\hat{x}_k - x_k).$$

Why Do We Get Sublinear Rate?

- Sublinear rate is due to the use a *linear minimization oracle* over C that is defined as follows

$$LMO(y) = \operatorname*{Argmin}_{x \in C} y^\top x.$$

An example getting sublinear rate

- C polytope and x^* on the boundary of the feasible set.
- REASON: Iterates of the algorithm start to zig-zag between the vertices defining the face containing the solution x^*.

Zig-zagging

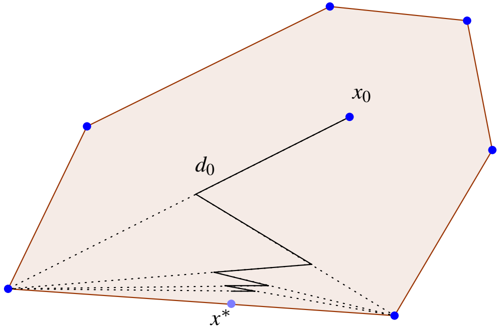

Figure: zig-zagging phenomenon.

Frank-Wolfe Variants

- **General case:** not possible to improve the sublinear rate of the Frank-Wolfe algorithm.
- **Polyhedral case:** there exist some variants of the Frank-Wolfe algorithm that guarantee (under suitable assumptions like, e.g. σ-strong convexity) convergence at a linear rate.
- We report here two well known variants.

Away-step Frank-Wolfe method

- This modification of the Frank-Wolfe method was proposed by Wolfe (1970).
- Frank-Wolfe directions are always directed towards extreme points.
- When close to the optimum (and the optimum is on the boundary) directions get more and more orthogonal to the gradient thus getting the zig-zagging phenomenon.
- In order to avoid this, Wolfe suggested to include directions pointing away from extreme points.
- Linear convergence can be obtained in case f is σ-strongly convex and C is polyhedral.
- Here, we consider a problem of the form

$$\begin{aligned} &\min f(x) \\ &x \in C = conv\{v_1, \ldots, v_p\} \end{aligned} \tag{8}$$

If we call $V = \{v_1, \ldots, v_p\}$, we know that, at step k, iterate is represented as a sparse convex combination of at most k vertices $S_k \subseteq V$.

Scheme of the Method

Algorithm 4 `Away-step Frank-Wolfe method`

Choose a point $x_1 \in C$

For $k = 1, \ldots$

  Set $\hat{x}_k^{FW} = \operatorname*{Argmin}_{x \in C} \nabla f(x_k)^\top (x - x_k)$

  If $\hat{x}_k^{FW}$ satisfies some specific condition, then STOP

  Set $\hat{x}_k^{AS} = \operatorname*{Argmax}_{x \in S_k} \nabla f(x_k)^\top (x - x_k)$

  Set $d_k^{FW} = \hat{x}_k^{FW} - x_k$ and $d_k^{AS} = x_k - \hat{x}_k^{AS}$

  If $\nabla f(x_k)^\top d_k^{FW} \leq \nabla f(x_k)^\top d_k^{AS}$

   Then set $d_k = d_k^{FW}$ and $\bar{\alpha} = 1$

   Else set $d_k = d_k^{AS}$ and $\bar{\alpha} = \max_\beta \{x_k + \beta d_k^{AS} \in C\}$

  End If

  Set $x_{k+1} = x_k + \alpha_k d_k$, with $\alpha_k \in (0, \bar{\alpha}]$ suitably chosen stepsize

  Calculate S_{k+1} set of currently used vertices.

End for

Comments

- At each iteration we calculate the classic Frank-Wolfe direction and the so-called away-step direction.
- away-step direction points away from the worst vertex (i.e., the one with highest value of the linearized function) describing the current iterate.
- Then we choose the best between the two and perform a line search along that direction (See Step 9).
- Finally, we update S_k.

Remark

Storing and updating S_k might be costly in practice. Furthermore, there might be multiple ways to represent iterate k as combination of vertices.

Behavior of the Away-step Frank-Wolfe Method

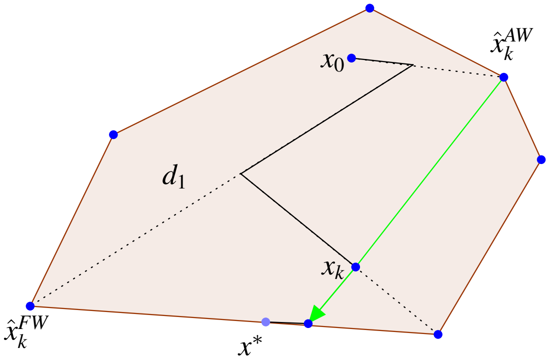

Figure: Behavior of the away-step Frank-Wolfe method.

Pairwise Frank-Wolfe

- First described by Mitchel et al. for the polytope distance problem.
- This method is strongly related to classic SMO algorithms in machine learning.
- The main idea is moving weight from the away vertex to the Frank-Wolfe vertex.
- In practice, at each iteration we use the search direction

$$d_k = d_k^{FW} + d_k^{AS}.$$

- Linear convergence can be obtained under similar assumptions as the away-step Frank-Wolfe method.
- Linear rate is more loose than away-step variant.
- Anyway, the method is very efficient in practice.

Scheme of Pairwise Frank-Wolfe

Algorithm 5 `Pairwise Frank-Wolfe method`

Choose a point $x_1 \in C$

For $k = 1, \dots$

    Set $d_k = d_k^{FW} + d_k^{AS}$ and $\bar{\alpha} = \max_\beta \{x_k + \beta d_k \in C\}$

    Set $x_{k+1} = x_k + \alpha_k d_k$, with $\alpha_k \in (0, \bar{\alpha}]$

      suitably chosen stepsize

    Calculate S_{k+1} set of currently used vertices.

End for

Behavior of the Pairwise Frank-Wolfe Method

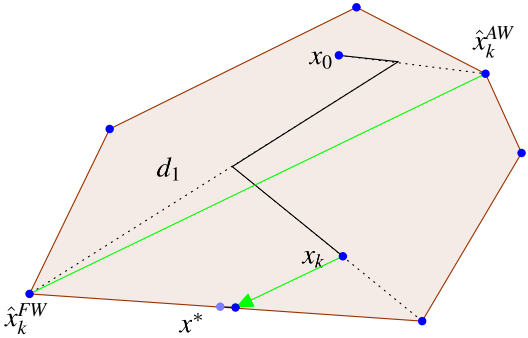

Figure: Behavior of the pairwise Frank-Wolfe method.

Fully Corrective Frank-Wolfe

Problem to Be Solved

We consider the more general problem:

$$\begin{aligned} &\min f(x) \\ &x \in C \end{aligned} \tag{9}$$

where f is continuously differentiable and C is a compact convex set.

- The fully corrective Frank Wolfe method (aka Simplicial Decomposition) represents a class of methods used for dealing with convex problems.
- It was first introduced by Holloway (1970) and then further studied in other papers.
- The method basically uses an iterative *inner approximation* of the feasible set C.
- The feasible set is approximated with the convex hull of an ever expanding finite set $C_k = \{\hat{x}_0, \hat{x}_2, \ldots, \hat{x}_k\}$ where $\hat{x}_i$, $i = 0, \ldots, k$ are extreme points of C.
- We denote this set with $conv(C_k)$:

$$conv(C_k) = \{x \mid x = \sum_{i=0}^{k} \lambda_i \hat{x}_i, \sum_{i=0}^{k} \lambda_i = 1, \lambda_i \geq 0\} \tag{10}$$

Details

- At each iteration, add new extreme points to C_k in such a way that a function reduction is guaranteed when minimizing the objective function over the convex hull of the new (enlarged) set of extreme points.
- If the algorithm does not find at least one new point, the solution is optimal and the algorithm terminates.
- Use of the proposed method indicated when following two conditions satisfied:
  1. Minimizing a linear function over C is much simpler than solving the original nonlinear problem;
  2. Minimizing the original objective function over the convex hull of a relatively small set of extreme points is much simpler than solving the original nonlinear problem.
- First condition needed due to the way a new extreme point is generated. Indeed, this new point is the solution of the following linear programming problem

$$\begin{array}{ll} \min & \nabla f(x_k)^\top (x - x_k) \\ \text{s.t.} & x \in C \end{array} \tag{11}$$

where a linear approximation calculated at the last iterate x_k (i.e. the solution obtained by minimizing f over $conv(C_{k-1})$) is minimized over the original feasible set C.

Scheme of Fully corrective Frank-Wolfe

Algorithm 6 `Fully corrective Frank-Wolfe method`

Choose an extreme point $\hat{x}_0$ of C, then set $C_0 = \{\hat{x}_0\}$ and $x_1 = \hat{x}_0$
For $k = 1, \ldots$
  Set $\hat{x}_k = \underset{x \in C}{\operatorname{Argmin}} \nabla f(x_k)^\top (x - x_k)$
  If $\hat{x}_k$ satisfies some specific condition, then STOP
  Set $C_k = C_{k-1} \cup \{\hat{x}_k\}$
  Set $x_{k+1} = \underset{x \in conv(C_k)}{\operatorname{Argmin}} f(x)$
End for

Finite Convergence of Fully Corrective Frank-Wolfe

Proposition [Finite Convergence]

Fully corrective Frank-Wolfe algorithm obtains a solution of Problem (9) (with C polytope) in a finite number of iterations.

Proof

Extreme point $\hat{x}_k$, obtained by approximately solving linear problem at Step 3, can only satisfy one of the following conditions

1. $\nabla f(x_k)^\top(\hat{x}_k - x_k) \geq 0$. Hence we get

$$\min_{x \in C} \nabla f(x_k)^\top(x - x_k) = \nabla f(x_k)^\top(\hat{x}_k - x_k) \geq 0,$$

   that is necessary and sufficient optimality conditions are satisfied and x_k minimizes f over the feasible set C;

2. $\nabla f(x_k)^\top(\hat{x}_k - x_k) < 0$, hence direction $d_k = \hat{x}_k - x_k$ is descent direction and

$$\hat{x}_k \notin conv(C_{k-1}). \tag{12}$$

   Indeed, since x_k minimizes f over $conv(C_{k-1})$ it satisfies necessary and sufficient optimality conditions, that is $\nabla f(x_k)^\top(x - x_k) \geq 0$ for all $x \in conv(C_{k-1})$.

From (12) we thus have $\hat{x}_k \notin C_{k-1}$.
Since our feasible set C has a finite number of extreme points, case 2 occurs only a finite number of times, and case 1) will eventually occur. $\square$

PROs and CONs

PROs

- Method makes more progress per iteration than other variants.
- Iterates combination of fewer vertices (better sparsity) with respect to other variants.

CONs

- Solving the inner problem, in some cases, is as hard as solving the original one.

Behavior of the Fully Corrective Frank-Wolfe Method

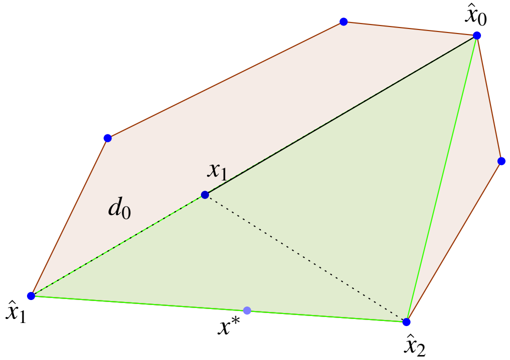

Figure: Behavior of the fully corrective Frank-Wolfe method.