Boosting Frank-Wolfe by Chasing Gradients

Cyrille W. Combettes

with Sebastian Pokutta

School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, GA, USA

37th International Conference on Machine Learning
July 12–18, 2020

Outline

Introduction

Let $\mathcal{H}$ be a Euclidean space (e.g., $\mathbb{R}^n$ or $\mathbb{R}^{m\times n}$) and consider

$$\begin{aligned} &\min f(x) \\ &\text{s.t. } x \in \mathcal{C} \end{aligned}$$

where

- $f : \mathcal{H} \to \mathbb{R}$ is a smooth convex function
- $\mathcal{C} \subset \mathcal{H}$ is a compact convex set, $\mathcal{C} = \mathrm{conv}(\mathcal{V})$

Introduction

Let $\mathcal{H}$ be a Euclidean space (e.g., $\mathbb{R}^n$ or $\mathbb{R}^{m\times n}$) and consider

$$\begin{aligned} &\min f(x) \\ &\text{ s.t. } x \in \mathcal{C} \end{aligned}$$

where

- $f : \mathcal{H} \to \mathbb{R}$ is a smooth convex function
- $\mathcal{C} \subset \mathcal{H}$ is a compact convex set, $\mathcal{C} = \text{conv}(\mathcal{V})$

Example

- Sparse logistic regression

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} \frac{1}{m}\sum_{i=1}^{m} \ln(1+\exp(-y_i a_i^\top x)) \\ &\text{s.t. } \|x\|_1 \leqslant \tau \end{aligned}$$

- Low-rank matrix completion

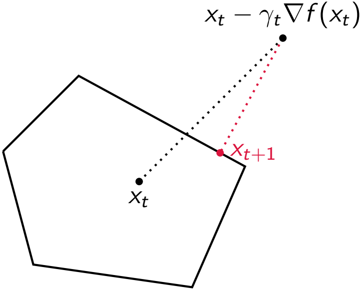
$$\begin{aligned} &\min_{X\in\mathbb{R}^{m\times n}} \frac{1}{2|\mathcal{I}|}\sum_{(i,j)\in\mathcal{I}} (Y_{i,j} - X_{i,j})^2 \\ &\text{s.t. } \|X\|_{\text{nuc}} \leqslant \tau \end{aligned}$$

Introduction

- A natural approach is to use any efficient method and add projections back onto $\mathcal{C}$ to ensure feasibility

Introduction

- A natural approach is to use any efficient method and add projections back onto $\mathcal{C}$ to ensure feasibility



Introduction

- A natural approach is to use any efficient method and add projections back onto $\mathcal{C}$ to ensure feasibility
- However, in many situations projections onto $\mathcal{C}$ are very expensive

Introduction

- A natural approach is to use any efficient method and add projections back onto $\mathcal{C}$ to ensure feasibility
- However, in many situations projections onto $\mathcal{C}$ are very expensive
- This is an issue with the method of projections, not necessarily with the geometry of $\mathcal{C}$: linear minimizations over $\mathcal{C}$ can still be relatively cheap

Introduction

- A natural approach is to use any efficient method and add projections back onto $\mathcal{C}$ to ensure feasibility
- However, in many situations projections onto $\mathcal{C}$ are very expensive
- This is an issue with the method of projections, not necessarily with the geometry of $\mathcal{C}$: linear minimizations over $\mathcal{C}$ can still be relatively cheap

Feasible region $\mathcal{C}$	**Linear minimization**	**Projection**
$\ell_1/\ell_2/\ell_\infty$-ball	$\mathcal{O}(n)$	$\mathcal{O}(n)$
ℓ_p-ball, $p \in]1, \infty[\setminus \{2\}$	$\mathcal{O}(n)$	N/A
Nuclear norm-ball	$\mathcal{O}(\text{nnz})$	$\mathcal{O}(mn \min\{m, n\})$
Flow polytope	$\mathcal{O}(n)$	$\mathcal{O}(n^{3.5})$
Birkhoff polytope	$\mathcal{O}(n^3)$	N/A
Matroid polytope	$\mathcal{O}(n \ln(n))$	$\mathcal{O}(\text{poly}(n))$

N/A: no closed-form exists and solution must be computed via nontrivial optimization

Introduction

- A natural approach is to use any efficient method and add projections back onto $\mathcal{C}$ to ensure feasibility
- However, in many situations projections onto $\mathcal{C}$ are very expensive
- This is an issue with the method of projections, not necessarily with the geometry of $\mathcal{C}$: linear minimizations over $\mathcal{C}$ can still be relatively cheap

Feasible region $\mathcal{C}$	**Linear minimization**	**Projection**
$\ell_1/\ell_2/\ell_\infty$-ball	$\mathcal{O}(n)$	$\mathcal{O}(n)$
ℓ_p-ball, $p \in]1,\infty[\setminus\{2\}$	$\mathcal{O}(n)$	N/A
Nuclear norm-ball	$\mathcal{O}(\text{nnz})$	$\mathcal{O}(mn\min\{m,n\})$
Flow polytope	$\mathcal{O}(n)$	$\mathcal{O}(n^{3.5})$
Birkhoff polytope	$\mathcal{O}(n^3)$	N/A
Matroid polytope	$\mathcal{O}(n\ln(n))$	$\mathcal{O}(\text{poly}(n))$

N/A: no closed-form exists and solution must be computed via nontrivial optimization

Introduction

- A natural approach is to use any efficient method and add projections back onto $\mathcal{C}$ to ensure feasibility
- However, in many situations projections onto $\mathcal{C}$ are very expensive
- This is an issue with the method of projections, not necessarily with the geometry of $\mathcal{C}$: linear minimizations over $\mathcal{C}$ can still be relatively cheap

Feasible region $\mathcal{C}$	**Linear minimization**	**Projection**
$\ell_1/\ell_2/\ell_\infty$-ball	$\mathcal{O}(n)$	$\mathcal{O}(n)$
ℓ_p-ball, $p \in]1,\infty[\setminus \{2\}$	$\mathcal{O}(n)$	N/A
Nuclear norm-ball	$\mathcal{O}(\text{nnz})$	$\mathcal{O}(mn\min\{m,n\})$
Flow polytope	$\mathcal{O}(n)$	$\mathcal{O}(n^{3.5})$
Birkhoff polytope	$\mathcal{O}(n^3)$	N/A
Matroid polytope	$\mathcal{O}(n\ln(n))$	$\mathcal{O}(\text{poly}(n))$

N/A: no closed-form exists and solution must be computed via nontrivial optimization

Introduction

- A natural approach is to use any efficient method and add projections back onto $\mathcal{C}$ to ensure feasibility
- However, in many situations projections onto $\mathcal{C}$ are very expensive
- This is an issue with the method of projections, not necessarily with the geometry of $\mathcal{C}$: linear minimizations over $\mathcal{C}$ can still be relatively cheap

Feasible region $\mathcal{C}$	**Linear minimization**	**Projection**
$\ell_1/\ell_2/\ell_\infty$-ball	$\mathcal{O}(n)$	$\mathcal{O}(n)$
ℓ_p-ball, $p \in]1, \infty[\setminus \{2\}$	$\mathcal{O}(n)$	N/A
Nuclear norm-ball	$\mathcal{O}(\text{nnz})$	$\mathcal{O}(mn\min\{m, n\})$
Flow polytope	$\mathcal{O}(n)$	$\mathcal{O}(n^{3.5})$
Birkhoff polytope	$\mathcal{O}(n^3)$	N/A
Matroid polytope	$\mathcal{O}(n\ln(n))$	$\mathcal{O}(\text{poly}(n))$

N/A: no closed-form exists and solution must be computed via nontrivial optimization

Introduction

- A natural approach is to use any efficient method and add projections back onto $\mathcal{C}$ to ensure feasibility
- However, in many situations projections onto $\mathcal{C}$ are very expensive
- This is an issue with the method of projections, not necessarily with the geometry of $\mathcal{C}$: linear minimizations over $\mathcal{C}$ can still be relatively cheap

Feasible region $\mathcal{C}$	**Linear minimization**	**Projection**
$\ell_1/\ell_2/\ell_\infty$-ball	$\mathcal{O}(n)$	$\mathcal{O}(n)$
ℓ_p-ball, $p \in]1,\infty[\,\setminus\{2\}$	$\mathcal{O}(n)$	N/A
Nuclear norm-ball	$\mathcal{O}(\text{nnz})$	$\mathcal{O}(mn\min\{m,n\})$
Flow polytope	$\mathcal{O}(n)$	$\mathcal{O}(n^{3.5})$
Birkhoff polytope	$\mathcal{O}(n^3)$	N/A
Matroid polytope	$\mathcal{O}(n\ln(n))$	$\mathcal{O}(\text{poly}(n))$

N/A: no closed-form exists and solution must be computed via nontrivial optimization

- Can we avoid projections?

The Frank-Wolfe algorithm

The Frank-Wolfe algorithm (Frank & Wolfe, 1956) a.k.a. conditional gradient algorithm (Levitin & Polyak, 1966):

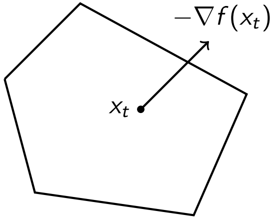
Algorithm Frank-Wolfe (FW)

Input: $x_0 \in \mathcal{C}$, $\gamma_t \in [0, 1]$.

1: **for** $t = 0$ **to** $T - 1$ **do**

2: $\quad v_t \leftarrow \arg\min_{v \in \mathcal{V}} \langle \nabla f(x_t), v \rangle$

3: $\quad x_{t+1} \leftarrow x_t + \gamma_t (v_t - x_t)$



The Frank-Wolfe algorithm

The Frank-Wolfe algorithm (Frank & Wolfe, 1956) a.k.a. conditional gradient algorithm (Levitin & Polyak, 1966):

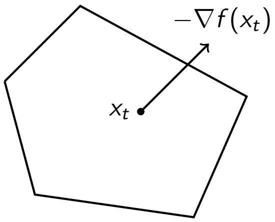
Algorithm Frank-Wolfe (FW)

Input: $x_0 \in \mathcal{C}$, $\gamma_t \in [0, 1]$.

1: **for** $t = 0$ **to** $T - 1$ **do**

2: $\quad v_t \leftarrow \arg\min_{v \in \mathcal{V}} \langle \nabla f(x_t), v \rangle$

3: $\quad x_{t+1} \leftarrow x_t + \gamma_t (v_t - x_t)$



The Frank-Wolfe algorithm

The Frank-Wolfe algorithm (Frank & Wolfe, 1956) a.k.a. conditional gradient algorithm (Levitin & Polyak, 1966):

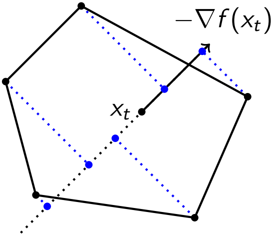
Algorithm Frank-Wolfe (FW)

Input: $x_0 \in \mathcal{C}$, $\gamma_t \in [0, 1]$.

1: **for** $t = 0$ **to** $T - 1$ **do**

2: $\quad v_t \leftarrow \arg\min_{v \in \mathcal{V}} \langle \nabla f(x_t), v \rangle$

3: $\quad x_{t+1} \leftarrow x_t + \gamma_t (v_t - x_t)$



The Frank-Wolfe algorithm

The Frank-Wolfe algorithm (Frank & Wolfe, 1956) a.k.a. conditional gradient algorithm (Levitin & Polyak, 1966):

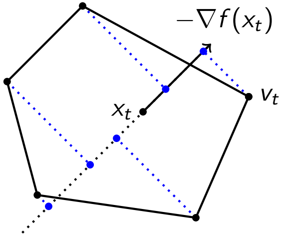
Algorithm Frank-Wolfe (FW)

Input: $x_0 \in \mathcal{C}$, $\gamma_t \in [0, 1]$.

1: **for** $t = 0$ **to** $T - 1$ **do**

2: $\quad v_t \leftarrow \arg\min_{v \in \mathcal{V}} \langle \nabla f(x_t), v \rangle$

3: $\quad x_{t+1} \leftarrow x_t + \gamma_t (v_t - x_t)$



The Frank-Wolfe algorithm

The Frank-Wolfe algorithm (Frank & Wolfe, 1956) a.k.a. conditional gradient algorithm (Levitin & Polyak, 1966):

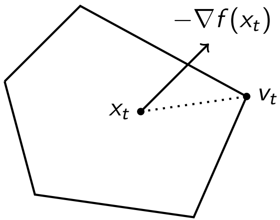
Algorithm Frank-Wolfe (FW)

Input: $x_0 \in \mathcal{C}$, $\gamma_t \in [0, 1]$.

1: **for** $t = 0$ **to** $T - 1$ **do**

2: $\quad v_t \leftarrow \arg\min_{v \in \mathcal{V}} \langle \nabla f(x_t), v \rangle$

3: $\quad x_{t+1} \leftarrow x_t + \gamma_t (v_t - x_t)$



The Frank-Wolfe algorithm

The Frank-Wolfe algorithm (Frank & Wolfe, 1956) a.k.a. conditional gradient algorithm (Levitin & Polyak, 1966):

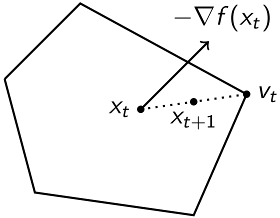
Algorithm Frank-Wolfe (FW)

Input: $x_0 \in \mathcal{C}$, $\gamma_t \in [0, 1]$.

1: **for** $t = 0$ **to** $T - 1$ **do**

2: $\quad v_t \leftarrow \arg\min_{v \in \mathcal{V}} \langle \nabla f(x_t), v \rangle$

3: $\quad x_{t+1} \leftarrow x_t + \gamma_t (v_t - x_t)$



The Frank-Wolfe algorithm

The Frank-Wolfe algorithm (Frank & Wolfe, 1956) a.k.a. conditional gradient algorithm (Levitin & Polyak, 1966):

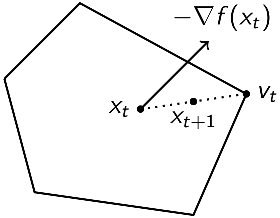
Algorithm Frank-Wolfe (FW)

Input: $x_0 \in \mathcal{C}$, $\gamma_t \in [0, 1]$.

1: **for** $t = 0$ **to** $T - 1$ **do**

2: $\quad v_t \leftarrow \arg\min_{v \in \mathcal{V}} \langle \nabla f(x_t), v \rangle$

3: $\quad x_{t+1} \leftarrow x_t + \gamma_t (v_t - x_t)$



- x_{t+1} is obtained by convex combination of $x_t \in \mathcal{C}$ and $v_t \in \mathcal{C}$, thus $x_{t+1} \in \mathcal{C}$

The Frank-Wolfe algorithm

The Frank-Wolfe algorithm (Frank & Wolfe, 1956) a.k.a. conditional gradient algorithm (Levitin & Polyak, 1966):

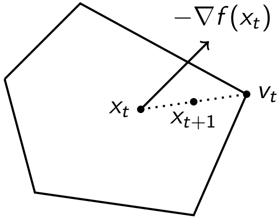
Algorithm Frank-Wolfe (FW)

Input: $x_0 \in \mathcal{C}$, $\gamma_t \in [0, 1]$.

1: **for** $t = 0$ **to** $T - 1$ **do**

2: $\quad v_t \leftarrow \underset{v \in \mathcal{V}}{\arg\min} \langle \nabla f(x_t), v \rangle$

3: $\quad x_{t+1} \leftarrow x_t + \gamma_t (v_t - x_t)$



- x_{t+1} is obtained by convex combination of $x_t \in \mathcal{C}$ and $v_t \in \mathcal{C}$, thus $x_{t+1} \in \mathcal{C}$
- FW uses linear minimizations (the "FW oracle") instead of projections

The Frank-Wolfe algorithm

The Frank-Wolfe algorithm (Frank & Wolfe, 1956) a.k.a. conditional gradient algorithm (Levitin & Polyak, 1966):

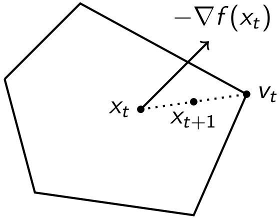
Algorithm Frank-Wolfe (FW)

Input: $x_0 \in \mathcal{C}$, $\gamma_t \in [0, 1]$.

1: **for** $t = 0$ **to** $T - 1$ **do**

2: $\quad v_t \leftarrow \underset{v \in \mathcal{V}}{\arg\min} \langle \nabla f(x_t), v \rangle$

3: $\quad x_{t+1} \leftarrow x_t + \gamma_t (v_t - x_t)$



- x_{t+1} is obtained by convex combination of $x_t \in \mathcal{C}$ and $v_t \in \mathcal{C}$, thus $x_{t+1} \in \mathcal{C}$
- FW uses linear minimizations (the "FW oracle") instead of projections
- FW = pick a vertex (using gradient information) and move in that direction

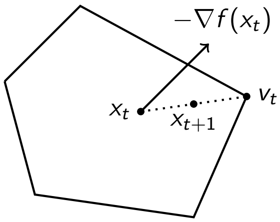
The Frank-Wolfe algorithm

The Frank-Wolfe algorithm (Frank & Wolfe, 1956) a.k.a. conditional gradient algorithm (Levitin & Polyak, 1966):

Algorithm Frank-Wolfe (FW)

Input: $x_0 \in \mathcal{C}$, $\gamma_t \in [0, 1]$.

1: **for** $t = 0$ **to** $T - 1$ **do**
2: $\quad v_t \leftarrow \arg\min_{v \in \mathcal{V}} \langle \nabla f(x_t), v \rangle$
3: $\quad x_{t+1} \leftarrow x_t + \gamma_t (v_t - x_t)$



- x_{t+1} is obtained by convex combination of $x_t \in \mathcal{C}$ and $v_t \in \mathcal{C}$, thus $x_{t+1} \in \mathcal{C}$
- FW uses linear minimizations (the "FW oracle") instead of projections
- FW = pick a vertex (using gradient information) and move in that direction
- Successfully applied to: traffic assignment, computer vision, optimal transport, adversarial learning, etc.

The Frank-Wolfe algorithm

Theorem (Levitin & Polyak, 1966; Jaggi, 2013)

Let $\mathcal{C} \subset \mathcal{H}$ *be a compact convex set with diameter* D *and* $f : \mathcal{H} \to \mathbb{R}$ *be a* L*-smooth convex function, and let* $x_0 \in \arg\min_{v \in \mathcal{V}} \langle \nabla f(y), v \rangle$ *for some* $y \in \mathcal{C}$*. If* $\gamma_t = \frac{2}{t+2}$ *(default) or* $\gamma_t = \min\left\{ \frac{\langle \nabla f(x_t), x_t - v_t \rangle}{L\|x_t - v_t\|^2}, 1 \right\}$ *("short step"), then*

$$f(x_t) - \min_{\mathcal{C}} f \leqslant \frac{4LD^2}{t+2}$$

The Frank-Wolfe algorithm

Theorem (Levitin & Polyak, 1966; Jaggi, 2013)

Let $\mathcal{C} \subset \mathcal{H}$ be a compact convex set with diameter D and $f : \mathcal{H} \to \mathbb{R}$ be a L-smooth convex function, and let $x_0 \in \arg\min_{v \in \mathcal{V}} \langle \nabla f(y), v \rangle$ for some $y \in \mathcal{C}$. If $\gamma_t = \frac{2}{t+2}$ (default) or $\gamma_t = \min\left\{\frac{\langle \nabla f(x_t), x_t - v_t \rangle}{L\|x_t - v_t\|^2}, 1\right\}$ ("short step"), then

$$f(x_t) - \min_{\mathcal{C}} f \leqslant \frac{4LD^2}{t+2}$$

- The convergence rate cannot be improved (Canon & Cullum, 1968; Jaggi, 2013; Lan, 2013)

The Frank-Wolfe algorithm

Theorem (Levitin & Polyak, 1966; Jaggi, 2013)

Let $\mathcal{C} \subset \mathcal{H}$ be a compact convex set with diameter D and $f : \mathcal{H} \to \mathbb{R}$ be a L-smooth convex function, and let $x_0 \in \arg\min_{v \in \mathcal{V}} \langle \nabla f(y), v \rangle$ for some $y \in \mathcal{C}$. If $\gamma_t = \frac{2}{t+2}$ (default) or $\gamma_t = \min\left\{ \frac{\langle \nabla f(x_t), x_t - v_t \rangle}{L\|x_t - v_t\|^2}, 1 \right\}$ ("short step"), then

$$f(x_t) - \min_{\mathcal{C}} f \leqslant \frac{4LD^2}{t+2}$$

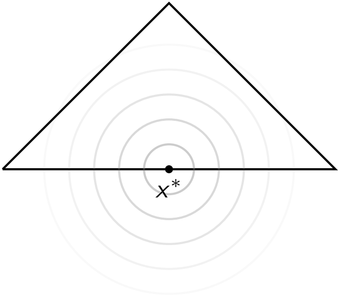
- The convergence rate cannot be improved (Canon & Cullum, 1968; Jaggi, 2013; Lan, 2013)
- Why?

The Frank-Wolfe algorithm

Consider the simple problem

$$\min \frac{1}{2}\|x\|_2^2$$
$$\text{s.t. } x \in \operatorname{conv}\left(\begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}-1\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}\right)$$

and $x^* = \begin{pmatrix}0\\0\end{pmatrix}$

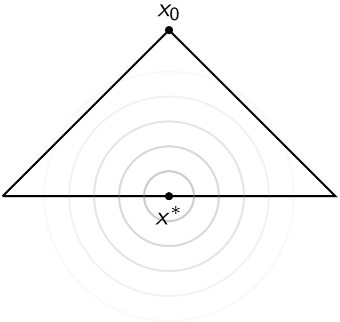


The Frank-Wolfe algorithm

Consider the simple problem

$$\min \frac{1}{2}\|x\|_2^2$$
$$\text{s.t. } x \in \operatorname{conv}\left(\begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}-1\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}\right)$$

and $x^* = \begin{pmatrix}0\\0\end{pmatrix}$



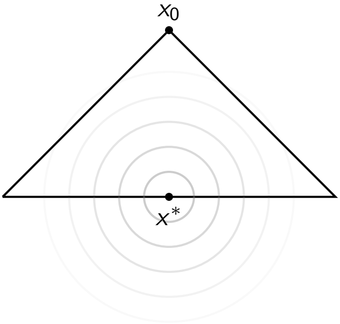
- Let $x_0 = \begin{pmatrix}0\\1\end{pmatrix}$

The Frank-Wolfe algorithm

Consider the simple problem

$$\min \frac{1}{2}\|x\|_2^2$$
$$\text{s.t. } x \in \operatorname{conv}\left(\begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}-1\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}\right)$$

and $x^* = \begin{pmatrix}0\\0\end{pmatrix}$



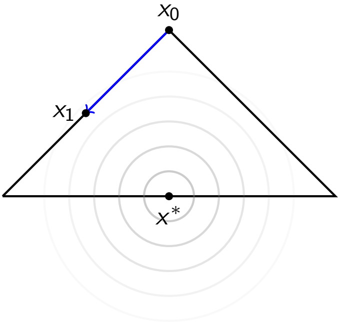
- Let $x_0 = \begin{pmatrix}0\\1\end{pmatrix}$
- FW tries to reach x^* by moving towards vertices

The Frank-Wolfe algorithm

Consider the simple problem

$$\min \frac{1}{2}\|x\|_2^2$$
$$\text{s.t. } x \in \text{conv}\left(\begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}-1\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}\right)$$

and $x^* = \begin{pmatrix}0\\0\end{pmatrix}$



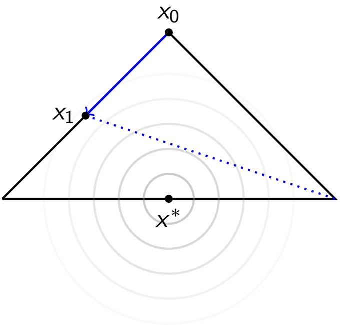
- Let $x_0 = \begin{pmatrix}0\\1\end{pmatrix}$
- FW tries to reach x^* by moving towards vertices

The Frank-Wolfe algorithm

Consider the simple problem

$$\min \frac{1}{2}\|x\|_2^2$$
$$\text{s.t. } x \in \operatorname{conv}\left(\begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}-1\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}\right)$$

and $x^* = \begin{pmatrix}0\\0\end{pmatrix}$



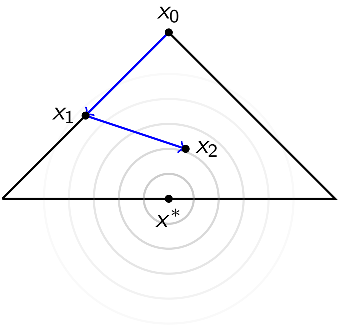
- Let $x_0 = \begin{pmatrix}0\\1\end{pmatrix}$
- FW tries to reach x^* by moving towards vertices

The Frank-Wolfe algorithm

Consider the simple problem

$$\min \frac{1}{2}\|x\|_2^2$$
$$\text{s.t. } x \in \operatorname{conv}\left(\begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}-1\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}\right)$$

and $x^* = \begin{pmatrix}0\\0\end{pmatrix}$



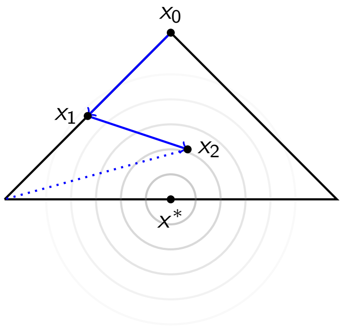
- Let $x_0 = \begin{pmatrix}0\\1\end{pmatrix}$
- FW tries to reach x^* by moving towards vertices

The Frank-Wolfe algorithm

Consider the simple problem

$$\min \frac{1}{2}\|x\|_2^2$$
$$\text{s.t. } x \in \operatorname{conv}\left(\begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}-1\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}\right)$$

and $x^* = \begin{pmatrix}0\\0\end{pmatrix}$



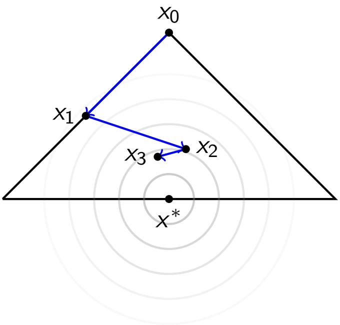
- Let $x_0 = \begin{pmatrix}0\\1\end{pmatrix}$
- FW tries to reach x^* by moving towards vertices

The Frank-Wolfe algorithm

Consider the simple problem

$$\min \frac{1}{2}\|x\|_2^2$$
$$\text{s.t. } x \in \operatorname{conv}\left(\begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}-1\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}\right)$$

and $x^* = \begin{pmatrix}0\\0\end{pmatrix}$



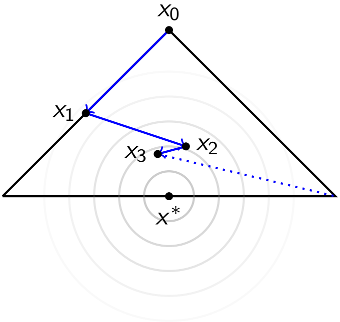
- Let $x_0 = \begin{pmatrix}0\\1\end{pmatrix}$
- FW tries to reach x^* by moving towards vertices

The Frank-Wolfe algorithm

Consider the simple problem

$$\min \frac{1}{2}\|x\|_2^2$$
$$\text{s.t. } x \in \operatorname{conv}\left(\begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}-1\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}\right)$$

and $x^* = \begin{pmatrix}0\\0\end{pmatrix}$



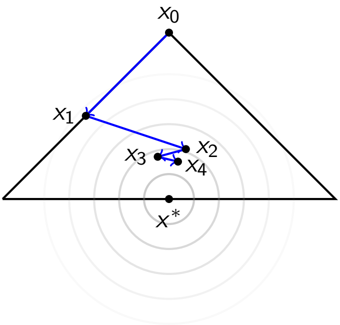
- Let $x_0 = \begin{pmatrix}0\\1\end{pmatrix}$
- FW tries to reach x^* by moving towards vertices

The Frank-Wolfe algorithm

Consider the simple problem

$$\min \frac{1}{2}\|x\|_2^2$$
$$\text{s.t. } x \in \operatorname{conv}\left(\begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}-1\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}\right)$$

and $x^* = \begin{pmatrix}0\\0\end{pmatrix}$



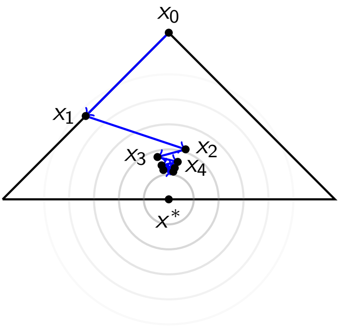
- Let $x_0 = \begin{pmatrix}0\\1\end{pmatrix}$
- FW tries to reach x^* by moving towards vertices

The Frank-Wolfe algorithm

Consider the simple problem

$$\min \frac{1}{2}\|x\|_2^2$$
$$\text{s.t. } x \in \text{conv}\left(\begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}-1\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}\right)$$

and $x^* = \begin{pmatrix}0\\0\end{pmatrix}$



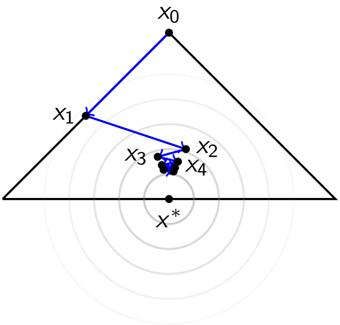
- Let $x_0 = \begin{pmatrix}0\\1\end{pmatrix}$
- FW tries to reach x^* by moving towards vertices

The Frank-Wolfe algorithm

Consider the simple problem

$$\min \frac{1}{2}\|x\|_2^2$$
$$\text{s.t. } x \in \operatorname{conv}\left(\begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}-1\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}\right)$$

and $x^* = \begin{pmatrix}0\\0\end{pmatrix}$



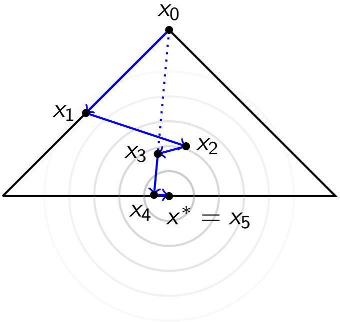
- Let $x_0 = \begin{pmatrix}0\\1\end{pmatrix}$
- FW tries to reach x^* by moving towards vertices
- This yields an inefficient zig-zagging trajectory

Improved Frank-Wolfe variants

- Away-Step Frank-Wolfe (AFW) (Wolfe, 1970; Lacoste-Julien & Jaggi, 2015): enhances FW by allowing to move away from vertices

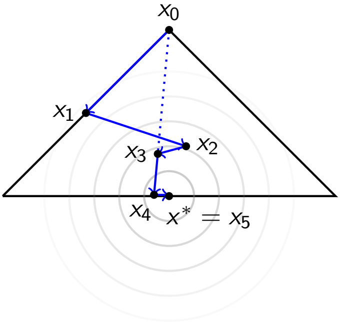
Improved Frank-Wolfe variants

- Away-Step Frank-Wolfe (AFW) (Wolfe, 1970; Lacoste-Julien & Jaggi, 2015): enhances FW by allowing to move away from vertices



Improved Frank-Wolfe variants

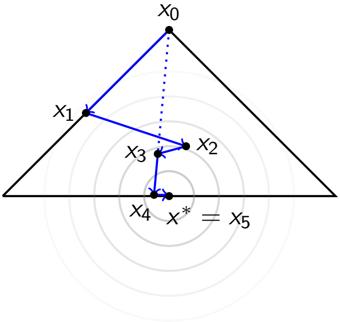
- Away-Step Frank-Wolfe (AFW) (Wolfe, 1970; Lacoste-Julien & Jaggi, 2015): enhances FW by allowing to move away from vertices



- Decomposition-Invariant Pairwise Conditional Gradient (DICG) (Garber & Meshi, 2016): memory-free variant of AFW

Improved Frank-Wolfe variants

- Away-Step Frank-Wolfe (AFW) (Wolfe, 1970; Lacoste-Julien & Jaggi, 2015): enhances FW by allowing to move away from vertices



- Decomposition-Invariant Pairwise Conditional Gradient (DICG) (Garber & Meshi, 2016): memory-free variant of AFW
- Blended Conditional Gradients (BCG) (Braun et al., 2019): blends FCFW and FW

Boosting Frank-Wolfe

- Can we speed up FW in a simple way?

Boosting Frank-Wolfe

- Can we speed up FW in a simple way?
- Rule of thumb in optimization: follow the steepest direction

Boosting Frank-Wolfe

- Can we speed up FW in a simple way?
- Rule of thumb in optimization: follow the steepest direction

Idea:

- Speed up FW by moving in a direction better aligned with $-\nabla f(x_t)$

Boosting Frank-Wolfe

- Can we speed up FW in a simple way?
- Rule of thumb in optimization: follow the steepest direction

Idea:

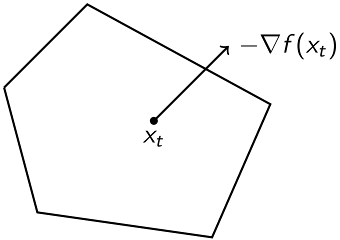
- Speed up FW by moving in a direction better aligned with $-\nabla f(x_t)$
- Build this direction by using $\mathcal{V}$ to maintain the projection-free property

Boosting Frank-Wolfe

- How can we build a direction better aligned with $-\nabla f(x_t)$ and that allows to update x_{t+1} without projection?

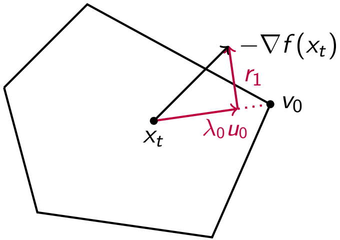
Boosting Frank-Wolfe

- How can we build a direction better aligned with $-\nabla f(x_t)$ and that allows to update x_{t+1} without projection?



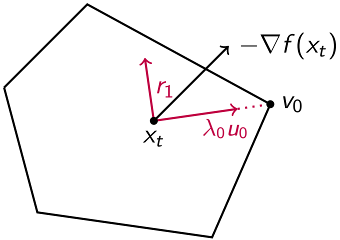
Boosting Frank-Wolfe

- How can we build a direction better aligned with $-\nabla f(x_t)$ and that allows to update x_{t+1} without projection?
- $v_0 \in \arg\max_{v \in \mathcal{V}} \langle -\nabla f(x_t), v \rangle$
  $\lambda_0 u_0 = \frac{\langle -\nabla f(x_t), v_0 - x_t \rangle}{\|v_0 - x_t\|^2}(v_0 - x_t)$
  $r_1 = -\nabla f(x_t) - \lambda_0 u_0$



Boosting Frank-Wolfe

- How can we build a direction better aligned with $-\nabla f(x_t)$ and that allows to update x_{t+1} without projection?
- $v_0 \in \arg\max_{v \in \mathcal{V}} \langle -\nabla f(x_t), v \rangle$
  $\lambda_0 u_0 = \frac{\langle -\nabla f(x_t), v_0 - x_t \rangle}{\|v_0 - x_t\|^2}(v_0 - x_t)$
  $r_1 = -\nabla f(x_t) - \lambda_0 u_0$

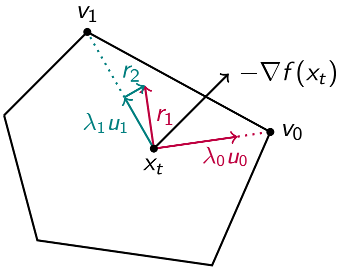


Boosting Frank-Wolfe

- How can we build a direction better aligned with $-\nabla f(x_t)$ and that allows to update x_{t+1} without projection?

- $v_0 \in \arg\max_{v\in\mathcal{V}}\langle -\nabla f(x_t), v\rangle$
  $\lambda_0 u_0 = \frac{\langle -\nabla f(x_t), v_0 - x_t\rangle}{\|v_0 - x_t\|^2}(v_0 - x_t)$
  $r_1 = -\nabla f(x_t) - \lambda_0 u_0$

- $v_1 \in \arg\max_{v\in\mathcal{V}}\langle r_1, v\rangle$
  $\lambda_1 u_1 = \frac{\langle r_1, v_1 - x_t\rangle}{\|v_1 - x_t\|^2}(v_1 - x_t)$
  $r_2 = r_1 - \lambda_1 u_1$



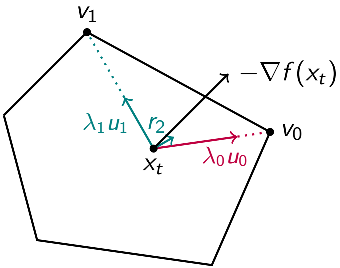
Boosting Frank-Wolfe

- How can we build a direction better aligned with $-\nabla f(x_t)$ and that allows to update x_{t+1} without projection?

- $v_0 \in \arg\max_{v \in \mathcal{V}} \langle -\nabla f(x_t), v \rangle$
  $\lambda_0 u_0 = \frac{\langle -\nabla f(x_t), v_0 - x_t \rangle}{\|v_0 - x_t\|^2}(v_0 - x_t)$
  $r_1 = -\nabla f(x_t) - \lambda_0 u_0$

- $v_1 \in \arg\max_{v \in \mathcal{V}} \langle r_1, v \rangle$
  $\lambda_1 u_1 = \frac{\langle r_1, v_1 - x_t \rangle}{\|v_1 - x_t\|^2}(v_1 - x_t)$
  $r_2 = r_1 - \lambda_1 u_1$

- We could continue:
  $v_2 \in \arg\max_{v \in \mathcal{V}} \langle r_2, v \rangle$



Boosting Frank-Wolfe

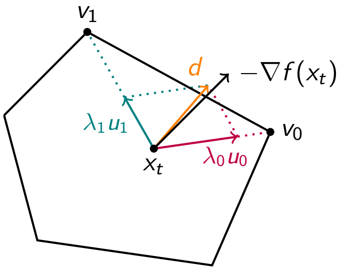
- How can we build a direction better aligned with $-\nabla f(x_t)$ and that allows to update x_{t+1} without projection?

- $v_0 \in \arg\max_{v \in \mathcal{V}} \langle -\nabla f(x_t), v \rangle$
  $\lambda_0 u_0 = \frac{\langle -\nabla f(x_t), v_0 - x_t \rangle}{\|v_0 - x_t\|^2}(v_0 - x_t)$
  $r_1 = -\nabla f(x_t) - \lambda_0 u_0$

- $v_1 \in \arg\max_{v \in \mathcal{V}} \langle r_1, v \rangle$
  $\lambda_1 u_1 = \frac{\langle r_1, v_1 - x_t \rangle}{\|v_1 - x_t\|^2}(v_1 - x_t)$
  $r_2 = r_1 - \lambda_1 u_1$

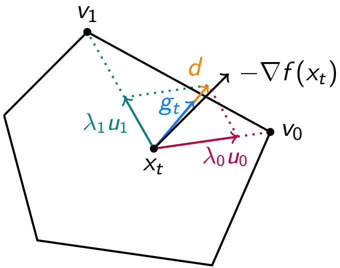
- We could continue:
  $v_2 \in \arg\max_{v \in \mathcal{V}} \langle r_2, v \rangle$

- $d = \lambda_0 u_0 + \lambda_1 u_1$



Boosting Frank-Wolfe

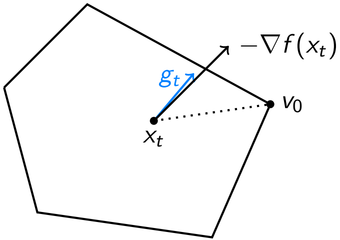
- How can we build a direction better aligned with $-\nabla f(x_t)$ and that allows to update x_{t+1} without projection?
- $v_0 \in \arg\max_{v\in\mathcal{V}} \langle -\nabla f(x_t), v\rangle$
  $\lambda_0 u_0 = \frac{\langle -\nabla f(x_t), v_0 - x_t\rangle}{\|v_0 - x_t\|^2}(v_0 - x_t)$
  $r_1 = -\nabla f(x_t) - \lambda_0 u_0$
- $v_1 \in \arg\max_{v\in\mathcal{V}} \langle r_1, v\rangle$
  $\lambda_1 u_1 = \frac{\langle r_1, v_1 - x_t\rangle}{\|v_1 - x_t\|^2}(v_1 - x_t)$
  $r_2 = r_1 - \lambda_1 u_1$
- We could continue:
  $v_2 \in \arg\max_{v\in\mathcal{V}} \langle r_2, v\rangle$
- $d = \lambda_0 u_0 + \lambda_1 u_1$
- $g_t = d/(\lambda_0 + \lambda_1)$



Boosting Frank-Wolfe

- How can we build a direction better aligned with $-\nabla f(x_t)$ and that allows to update x_{t+1} without projection?

- $v_0 \in \arg\max_{v \in \mathcal{V}} \langle -\nabla f(x_t), v \rangle$
  $\lambda_0 u_0 = \frac{\langle -\nabla f(x_t), v_0 - x_t \rangle}{\|v_0 - x_t\|^2}(v_0 - x_t)$
  $r_1 = -\nabla f(x_t) - \lambda_0 u_0$
- $v_1 \in \arg\max_{v \in \mathcal{V}} \langle r_1, v \rangle$
  $\lambda_1 u_1 = \frac{\langle r_1, v_1 - x_t \rangle}{\|v_1 - x_t\|^2}(v_1 - x_t)$
  $r_2 = r_1 - \lambda_1 u_1$
- We could continue:
  $v_2 \in \arg\max_{v \in \mathcal{V}} \langle r_2, v \rangle$
- $d = \lambda_0 u_0 + \lambda_1 u_1$
- $g_t = d/(\lambda_0 + \lambda_1)$

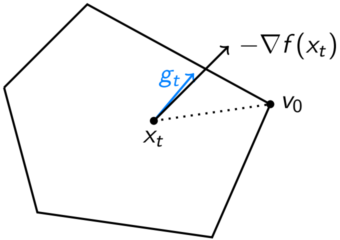


- The boosted direction g_t is better aligned with $-\nabla f(x_t)$ than is the FW direction $v_0 - x_t$

Boosting Frank-Wolfe

- How can we build a direction better aligned with $-\nabla f(x_t)$ and that allows to update x_{t+1} without projection?

- $v_0 \in \arg\max_{v \in \mathcal{V}} \langle -\nabla f(x_t), v \rangle$
  $\lambda_0 u_0 = \frac{\langle -\nabla f(x_t), v_0 - x_t \rangle}{\|v_0 - x_t\|^2}(v_0 - x_t)$
  $r_1 = -\nabla f(x_t) - \lambda_0 u_0$
- $v_1 \in \arg\max_{v \in \mathcal{V}} \langle r_1, v \rangle$
  $\lambda_1 u_1 = \frac{\langle r_1, v_1 - x_t \rangle}{\|v_1 - x_t\|^2}(v_1 - x_t)$
  $r_2 = r_1 - \lambda_1 u_1$
- We could continue:
  $v_2 \in \arg\max_{v \in \mathcal{V}} \langle r_2, v \rangle$
- $d = \lambda_0 u_0 + \lambda_1 u_1$
- $g_t = d/(\lambda_0 + \lambda_1)$



- The boosted direction g_t is better aligned with $-\nabla f(x_t)$ than is the FW direction $v_0 - x_t$ and satisfies $[x_t, x_t + g_t] \subseteq \mathcal{C}$ so we can update

$$x_{t+1} = x_t + \gamma_t g_t \quad \text{for any } \gamma_t \in [0, 1]$$

Boosting Frank-Wolfe

Why $[x_t, x_t + g_t] \subseteq \mathcal{C}$? Let K_t be the number of alignment rounds. We have

$$d = \sum_{k=0}^{K_t-1} \lambda_k (v_k - x_t) \quad \text{where } \lambda_k > 0 \text{ and } v_k \in \mathcal{V}$$

Boosting Frank-Wolfe

Why $[x_t, x_t + g_t] \subseteq \mathcal{C}$? Let K_t be the number of alignment rounds. We have

$$d = \sum_{k=0}^{K_t - 1} \lambda_k (v_k - x_t) \quad \text{where } \lambda_k > 0 \text{ and } v_k \in \mathcal{V}$$

so if $\Lambda_t = \sum_{k=0}^{K-1} \lambda_k$, then

$$g_t = \frac{1}{\Lambda_t} \sum_{k=0}^{K_t - 1} \lambda_k (v_k - x_t) = \underbrace{\left(\frac{1}{\Lambda_t} \sum_{k=0}^{K_t - 1} \lambda_k v_k \right)}_{\in \mathcal{C}} - x_t$$

Boosting Frank-Wolfe

Why $[x_t, x_t + g_t] \subseteq \mathcal{C}$? Let K_t be the number of alignment rounds. We have

$$d = \sum_{k=0}^{K_t-1} \lambda_k (v_k - x_t) \quad \text{where } \lambda_k > 0 \text{ and } v_k \in \mathcal{V}$$

so if $\Lambda_t = \sum_{k=0}^{K-1} \lambda_k$, then

$$g_t = \frac{1}{\Lambda_t} \sum_{k=0}^{K_t-1} \lambda_k (v_k - x_t) = \underbrace{\left(\frac{1}{\Lambda_t} \sum_{k=0}^{K_t-1} \lambda_k v_k \right)}_{\in \mathcal{C}} - x_t$$

Thus, $x_t + g_t \in \mathcal{C}$ so $[x_t, x_t + g_t] \subseteq \mathcal{C}$ by convexity

Boosting Frank-Wolfe

Algorithm Finding a direction g *well* aligned with ∇ from a reference point z

Input: $z \in \mathcal{C}$, $\nabla \in \mathcal{H}$, $K \in \mathbb{N}\backslash\{0\}$, $\delta \in]0,1[$.

```
d_0 ← 0, Λ ← 0
for k = 0 to K − 1 do
    r_k ← ∇ − d_k                                         ▷ k-th residual
    v_k ← arg max_{v∈𝒱} ⟨r_k, v⟩                           ▷ FW oracle
    u_k ← arg max_{u∈{v_k − z, −d_k/‖d_k‖}} ⟨r_k, u⟩
    λ_k ← ⟨r_k, u_k⟩/‖u_k‖²
    d'_k ← d_k + λ_k u_k
    if align(∇, d'_k) − align(∇, d_k) ⩾ δ then
        d_{k+1} ← d'_k
        Λ_t ← { Λ + λ_k               if u_k = v_k − z
              { Λ(1 − λ_k/‖d_k‖)      if u_k = −d_k/‖d_k‖
    else
        break                                             ▷ exit k-loop
g ← d_k/Λ                                                 ▷ normalization
```

Boosting Frank-Wolfe

Algorithm Finding a direction g *well* aligned with ∇ from a reference point z

Input: $z \in \mathcal{C}$, $\nabla \in \mathcal{H}$, $K \in \mathbb{N}\backslash\{0\}$, $\delta \in]0,1[$.

$d_0 \leftarrow 0$, $\Lambda \leftarrow 0$
for $k = 0$ **to** $K-1$ **do**
  $r_k \leftarrow \nabla - d_k$ ▷ k-th residual
  $v_k \leftarrow \arg\max_{v \in \mathcal{V}} \langle r_k, v \rangle$ ▷ FW oracle
  $u_k \leftarrow \arg\max_{u \in \{v_k - z, -d_k/\|d_k\|\}} \langle r_k, u \rangle$
  $\lambda_k \leftarrow \langle r_k, u_k \rangle / \|u_k\|^2$
  $d'_k \leftarrow d_k + \lambda_k u_k$
  if $\text{align}(\nabla, d'_k) - \text{align}(\nabla, d_k) \geqslant \delta$ **then**
    $d_{k+1} \leftarrow d'_k$
    $\Lambda_t \leftarrow \begin{cases} \Lambda + \lambda_k & \textbf{if } u_k = v_k - z \\ \Lambda(1 - \lambda_k/\|d_k\|) & \textbf{if } u_k = -d_k/\|d_k\| \end{cases}$
  else
    break ▷ exit k-loop
$g \leftarrow d_k/\Lambda$ ▷ normalization

- Technicality to ensure convergence of the procedure (Locatello et al., 2017)

Boosting Frank-Wolfe

Algorithm Finding a direction g *well* aligned with ∇ from a reference point z

Input: $z \in \mathcal{C}$, $\nabla \in \mathcal{H}$, $K \in \mathbb{N}\backslash\{0\}$, $\delta \in]0, 1[$.

$d_0 \leftarrow 0$, $\Lambda \leftarrow 0$
for $k = 0$ **to** $K - 1$ **do**
  $r_k \leftarrow \nabla - d_k$ ▷ k-th residual
  $v_k \leftarrow \arg\max_{v \in \mathcal{V}} \langle r_k, v \rangle$ ▷ FW oracle
  $u_k \leftarrow \arg\max_{u \in \{v_k - z, -d_k/\|d_k\|\}} \langle r_k, u \rangle$
  $\lambda_k \leftarrow \langle r_k, u_k \rangle / \|u_k\|^2$
  $d'_k \leftarrow d_k + \lambda_k u_k$
  if $\text{align}(\nabla, d'_k) - \text{align}(\nabla, d_k) \geqslant \delta$ **then**
    $d_{k+1} \leftarrow d'_k$
    $\Lambda_t \leftarrow \begin{cases} \Lambda + \lambda_k & \textbf{if } u_k = v_k - z \\ \Lambda(1 - \lambda_k/\|d_k\|) & \textbf{if } u_k = -d_k/\|d_k\| \end{cases}$
  else
    break ▷ exit k-loop
$g \leftarrow d_k/\Lambda$ ▷ normalization

- Technicality to ensure convergence of the procedure (Locatello et al., 2017)
- The stopping criterion is an alignment improvement condition (typically $\delta = 10^{-3}$ and $K = +\infty$)

Boosting Frank-Wolfe

Algorithm Frank-Wolfe (FW)

Input: $x_0 \in \mathcal{C}$, $\gamma_t \in [0,1]$.

1: **for** $t = 0$ **to** $T-1$ **do**

2: $\quad v_t \leftarrow \arg\min_{v \in \mathcal{V}} \langle \nabla f(x_t), v \rangle$

3: $\quad x_{t+1} \leftarrow x_t + \gamma_t (v_t - x_t)$

Algorithm Boosted Frank-Wolfe (BoostFW)

Input: $x_0 \in \mathcal{C}$, $\gamma_t \in [0,1]$, $K \in \mathbb{N}\backslash\{0\}$, $\delta \in]0,1[$.

1: **for** $t = 0$ **to** $T-1$ **do**

2: $\quad g_t \leftarrow \text{procedure}(x_t, -\nabla f(x_t), K, \delta)$

3: $\quad x_{t+1} \leftarrow x_t + \gamma_t g_t$

Algorithm Frank-Wolfe (FW)

Input: $x_0 \in \mathcal{C}$, $\gamma_t \in [0,1]$.

1: **for** $t = 0$ **to** $T-1$ **do**

2: $\quad v_t \leftarrow \underset{v \in \mathcal{V}}{\arg\min} \langle \nabla f(x_t), v \rangle$

3: $\quad x_{t+1} \leftarrow x_t + \gamma_t (v_t - x_t)$

Algorithm Boosted Frank-Wolfe (BoostFW)

Input: $x_0 \in \mathcal{C}$, $\gamma_t \in [0,1]$, $K \in \mathbb{N} \backslash \{0\}$, $\delta \in]0,1[$.

1: **for** $t = 0$ **to** $T-1$ **do**

2: $\quad g_t \leftarrow \text{procedure}(x_t, -\nabla f(x_t), K, \delta)$

3: $\quad x_{t+1} \leftarrow x_t + \gamma_t g_t$

Boosting Frank-Wolfe

Algorithm Frank-Wolfe (FW)

Input: $x_0 \in \mathcal{C}$, $\gamma_t \in [0,1]$.

1: **for** $t = 0$ **to** $T-1$ **do**

2: $\quad v_t \leftarrow \underset{v \in \mathcal{V}}{\arg\min} \langle \nabla f(x_t), v \rangle$

3: $\quad x_{t+1} \leftarrow x_t + \gamma_t (v_t - x_t)$

Algorithm Boosted Frank-Wolfe (BoostFW)

Input: $x_0 \in \mathcal{C}$, $\gamma_t \in [0,1]$, $K \in \mathbb{N} \backslash \{0\}$, $\delta \in]0,1[$.

1: **for** $t = 0$ **to** $T-1$ **do**

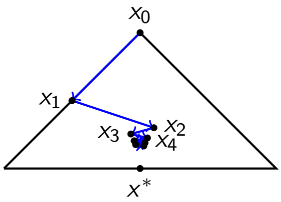
2: $\quad g_t \leftarrow \text{procedure}(x_t, -\nabla f(x_t), K, \delta)$

3: $\quad x_{t+1} \leftarrow x_t + \gamma_t g_t$

Algorithm Frank-Wolfe (FW)

Input: $x_0 \in \mathcal{C}$, $\gamma_t \in [0,1]$.

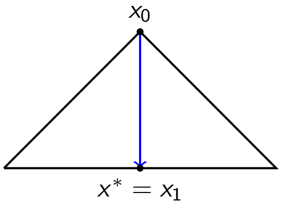
for $t = 0$ **to** $T-1$ **do**
  $v_t \leftarrow \underset{v \in \mathcal{V}}{\arg\min} \langle \nabla f(x_t), v \rangle$
  $x_{t+1} \leftarrow x_t + \gamma_t (v_t - x_t)$



Algorithm Boosted Frank-Wolfe (BoostFW)

Input: $x_0 \in \mathcal{C}$, $\gamma_t \in [0,1]$, $K \in \mathbb{N} \backslash \{0\}$, $\delta \in]0,1[$.

for $t = 0$ **to** $T-1$ **do**
  $g_t \leftarrow \text{procedure}(x_t, -\nabla f(x_t), K, \delta)$
  $x_{t+1} \leftarrow x_t + \gamma_t g_t$



Boosting Frank-Wolfe

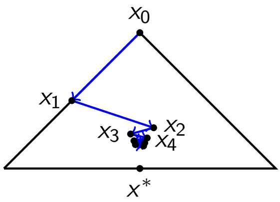
Algorithm Frank-Wolfe (FW)

Input: $x_0 \in \mathcal{C}$, $\gamma_t \in [0,1]$.

1: **for** $t = 0$ **to** $T - 1$ **do**

2: $\quad v_t \leftarrow \arg\min_{v \in \mathcal{V}} \langle \nabla f(x_t), v \rangle$

3: $\quad x_{t+1} \leftarrow x_t + \gamma_t (v_t - x_t)$



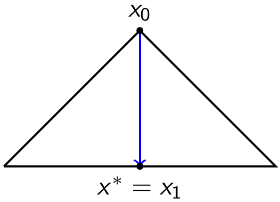
Algorithm Boosted Frank-Wolfe (BoostFW)

Input: $x_0 \in \mathcal{C}$, $\gamma_t \in [0,1]$, $K \in \mathbb{N} \backslash \{0\}$, $\delta \in]0,1[$.

1: **for** $t = 0$ **to** $T - 1$ **do**

2: $\quad g_t \leftarrow \text{procedure}(x_t, -\nabla f(x_t), K, \delta)$

3: $\quad x_{t+1} \leftarrow x_t + \gamma_t g_t$



- What is the convergence rate of BoostFW?

Boosting Frank-Wolfe

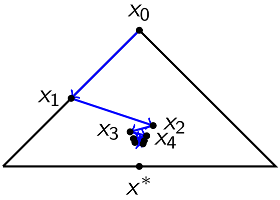
Algorithm Frank-Wolfe (FW)

Input: $x_0 \in \mathcal{C}$, $\gamma_t \in [0,1]$.

for $t = 0$ **to** $T-1$ **do**

$\quad v_t \leftarrow \arg\min_{v \in \mathcal{V}} \langle \nabla f(x_t), v \rangle$

$\quad x_{t+1} \leftarrow x_t + \gamma_t (v_t - x_t)$



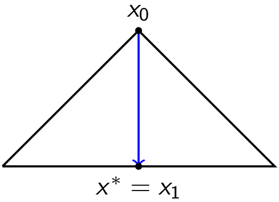
Algorithm Boosted Frank-Wolfe (BoostFW)

Input: $x_0 \in \mathcal{C}$, $\gamma_t \in [0,1]$, $K \in \mathbb{N} \setminus \{0\}$, $\delta \in]0,1[$.

for $t = 0$ **to** $T-1$ **do**

$\quad g_t \leftarrow \text{procedure}(x_t, -\nabla f(x_t), K, \delta)$

$\quad x_{t+1} \leftarrow x_t + \gamma_t g_t$



- What is the convergence rate of BoostFW?
- Is BoostFW expensive in practice?

Boosting Frank-Wolfe

Algorithm Frank-Wolfe (FW)

Input: $x_0 \in \mathcal{C}$, $\gamma_t \in [0, 1]$.

1: **for** $t = 0$ **to** $T - 1$ **do**

2: $\quad v_t \leftarrow \arg\min_{v \in \mathcal{V}} \langle \nabla f(x_t), v \rangle$

3: $\quad x_{t+1} \leftarrow x_t + \gamma_t (v_t - x_t)$

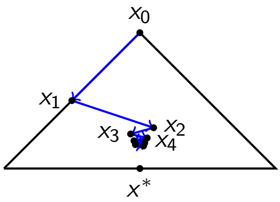
Algorithm Boosted Frank-Wolfe (BoostFW)

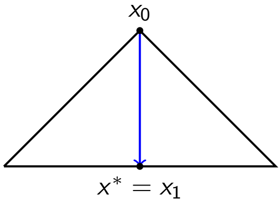
Input: $x_0 \in \mathcal{C}$, $\gamma_t \in [0, 1]$, $K \in \mathbb{N} \backslash \{0\}$, $\delta \in]0, 1[$.

1: **for** $t = 0$ **to** $T - 1$ **do**

2: $\quad g_t \leftarrow \text{procedure}(x_t, -\nabla f(x_t), K, \delta)$

3: $\quad x_{t+1} \leftarrow x_t + \gamma_t g_t$





- What is the convergence rate of BoostFW?
- Is BoostFW expensive in practice?
- How does it compare to the state-of-the-art?

Boosting Frank-Wolfe

- Let N_t be the number of iterations up to t where at least 2 rounds of alignment were performed (FW = always 1 round)

Theorem

Let $\mathcal{C} \subset \mathcal{H}$ be a compact convex set with diameter D and $f : \mathcal{H} \to \mathbb{R}$ be a L-smooth, convex, and μ-gradient dominated function, and let $x_0 \in \arg\min_{v \in \mathcal{V}} \langle \nabla f(y), v \rangle$ for some $y \in \mathcal{C}$. Set $\gamma_t = \min\left\{\frac{\langle -\nabla f(x_t), g_t \rangle}{L\|g_t\|^2}, 1\right\}$ ("short step") and suppose that $N_t \geqslant \omega t^p$ where $p \in]0,1]$. Then

$$f(x_t) - \min_{\mathcal{C}} f \leqslant \frac{LD^2}{2} \exp\left(-\delta^2 \frac{\mu}{L} \omega t^p\right)$$

Boosting Frank-Wolfe

- Let N_t be the number of iterations up to t where at least 2 rounds of alignment were performed (FW = always 1 round)

Theorem

Let $\mathcal{C} \subset \mathcal{H}$ be a compact convex set with diameter D and $f : \mathcal{H} \to \mathbb{R}$ be a L-smooth, convex, and μ-gradient dominated function, and let $x_0 \in \arg\min_{v \in \mathcal{V}} \langle \nabla f(y), v \rangle$ for some $y \in \mathcal{C}$. Set $\gamma_t = \min\left\{\frac{\langle -\nabla f(x_t), g_t \rangle}{L\|g_t\|^2}, 1\right\}$ ("short step") and suppose that $N_t \geqslant \omega t^p$ where $p \in]0, 1]$. Then

$$f(x_t) - \min_{\mathcal{C}} f \leqslant \frac{LD^2}{2} \exp\left(-\delta^2 \frac{\mu}{L} \omega t^p\right)$$

- The assumption $N_t \geqslant \omega t^p$ simply states that N_t is nonnegligeable, i.e., that the boosting procedure is active

Boosting Frank-Wolfe

- Let N_t be the number of iterations up to t where at least 2 rounds of alignment were performed (FW = always 1 round)

Theorem

Let $\mathcal{C} \subset \mathcal{H}$ be a compact convex set with diameter D and $f : \mathcal{H} \to \mathbb{R}$ be a L-smooth, convex, and μ-gradient dominated function, and let $x_0 \in \arg\min_{v \in \mathcal{V}} \langle \nabla f(y), v \rangle$ for some $y \in \mathcal{C}$. Set $\gamma_t = \min\left\{\frac{\langle -\nabla f(x_t), g_t \rangle}{L\|g_t\|^2}, 1\right\}$ ("short step") and suppose that $N_t \geqslant \omega t^p$ where $p \in]0,1]$. Then

$$f(x_t) - \min_{\mathcal{C}} f \leqslant \frac{LD^2}{2} \exp\left(-\delta^2 \frac{\mu}{L} \omega t^p\right)$$

- The assumption $N_t \geqslant \omega t^p$ simply states that N_t is nonnegligeable, i.e., that the boosting procedure is active
- Else BoostFW reduces to FW and the convergence rate is $\frac{4LD^2}{t+2}$

Boosting Frank-Wolfe

- Let N_t be the number of iterations up to t where at least 2 rounds of alignment were performed (FW = always 1 round)

Theorem

Let $\mathcal{C} \subset \mathcal{H}$ be a compact convex set with diameter D and $f : \mathcal{H} \to \mathbb{R}$ be a L-smooth, convex, and μ-gradient dominated function, and let $x_0 \in \arg\min_{v \in \mathcal{V}} \langle \nabla f(y), v \rangle$ for some $y \in \mathcal{C}$. Set $\gamma_t = \min\left\{\frac{\langle -\nabla f(x_t), g_t \rangle}{L\|g_t\|^2}, 1\right\}$ ("short step") and suppose that $N_t \geqslant \omega t^p$ where $p \in]0,1]$. Then

$$f(x_t) - \min_{\mathcal{C}} f \leqslant \frac{LD^2}{2} \exp\left(-\delta^2 \frac{\mu}{L} \omega t^p\right)$$

- The assumption $N_t \geqslant \omega t^p$ simply states that N_t is nonnegligeable, i.e., that the boosting procedure is active
- Else BoostFW reduces to FW and the convergence rate is $\frac{4LD^2}{t+2}$
- In practice, $N_t \approx t$ (so $\omega \lesssim 1$ and $p = 1$)

Computational experiments

- We compare BoostFW to AFW, BCG, and DICG on a series of experiments involving various objective functions and feasible regions

Computational experiments

- We compare BoostFW to AFW, BCG, and DICG on a series of experiments involving various objective functions and feasible regions

$$\min_{x\in\mathbb{R}^n} \|y - Ax\|_2^2 \quad \text{s.t. } \|x\|_1 \leqslant \tau$$

$$\begin{aligned}
\min_{x\in\mathbb{R}^{|\mathcal{A}|}} \ & \sum_{a\in\mathcal{A}} \tau_a x_a \left(1 + 0.03\left(\frac{x_a}{c_a}\right)^4\right) \\
\text{s.t. } & x_a = \sum_{r\in\mathcal{R}} \mathbb{1}_{\{a\in r\}} y_r && a\in\mathcal{A} \\
& \sum_{r\in\mathcal{R}_{i,j}} y_r = d_{i,j} && (i,j)\in\mathcal{S} \\
& y_r \geqslant 0 && r\in\mathcal{R}_{i,j},\ (i,j)\in\mathcal{S}
\end{aligned}$$

$$\min_{x\in\mathbb{R}^n} \frac{1}{m}\sum_{i=1}^m \ln(1+\exp(-y_i a_i^\top x)) \quad \text{s.t. } \|x\|_1 \leqslant \tau$$

$$\min_{X\in\mathbb{R}^{m\times n}} \frac{1}{|\mathcal{I}|}\sum_{(i,j)\in\mathcal{I}} h_\rho(Y_{i,j} - X_{i,j}) \quad \text{s.t. } \|X\|_{\text{nuc}} \leqslant \tau$$

Computational experiments

- We compare BoostFW to AFW, BCG, and DICG on a series of experiments involving various objective functions and feasible regions

$$\min_{x\in\mathbb{R}^n} \|y - Ax\|_2^2 \quad \text{s.t. } \|x\|_1 \leqslant \tau$$

$$\begin{aligned}
\min_{x\in\mathbb{R}^{|\mathcal{A}|}} \; & \sum_{a\in\mathcal{A}} \tau_a x_a \left(1 + 0.03\left(\frac{x_a}{c_a}\right)^4\right) \\
\text{s.t. } & x_a = \sum_{r\in\mathcal{R}} \mathbb{1}_{\{a\in r\}} y_r && a \in \mathcal{A} \\
& \sum_{r\in\mathcal{R}_{i,j}} y_r = d_{i,j} && (i,j)\in\mathcal{S} \\
& y_r \geqslant 0 && r\in\mathcal{R}_{i,j},\ (i,j)\in\mathcal{S}
\end{aligned}$$

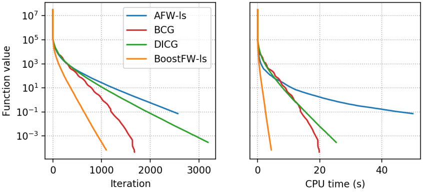
$$\min_{x\in\mathbb{R}^n} \frac{1}{m}\sum_{i=1}^{m} \ln(1+\exp(-y_i a_i^\top x)) \quad \text{s.t. } \|x\|_1 \leqslant \tau$$

$$\min_{X\in\mathbb{R}^{m\times n}} \frac{1}{|\mathcal{I}|}\sum_{(i,j)\in\mathcal{I}} h_\rho(Y_{i,j} - X_{i,j}) \quad \text{s.t. } \|X\|_{\text{nuc}} \leqslant \tau$$

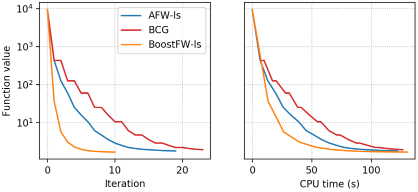
- For BoostFW and AFW we also run the line search-free variants (the "short step" strategy) and label them with an "L"

Computational experiments

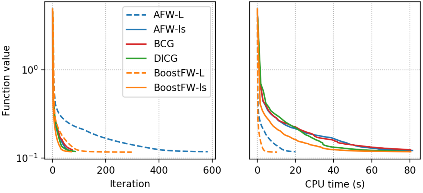
- Sparse signal recovery



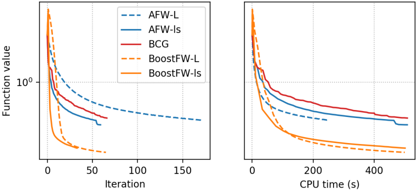
- Traffic assignment



- Sparse logistic regression on the Gisette dataset

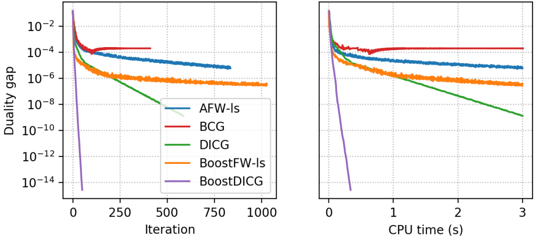


- Collaborative filtering on the MovieLens 100k dataset



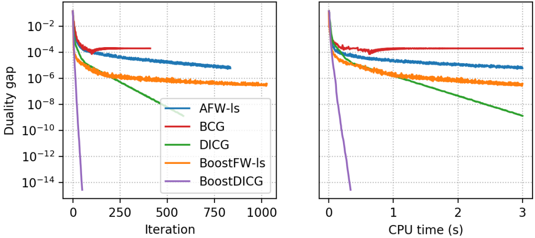
Boosting DICG

- DICG is known to perform particularly well on the video co-localization experiment (YouTube-Objects dataset)
- BoostDICG: application of our method to DICG



Boosting DICG

- DICG is known to perform particularly well on the video co-localization experiment (YouTube-Objects dataset)
- BoostDICG: application of our method to DICG



- (*details*)

DICG

$a_t \leftarrow$ away vertex

$v_t \leftarrow \arg\min_{v \in \mathcal{V}} \langle \nabla f(x_t), v \rangle$

$x_{t+1} \leftarrow x_t + \gamma_t(v_t - a_t)$

BoostDICG

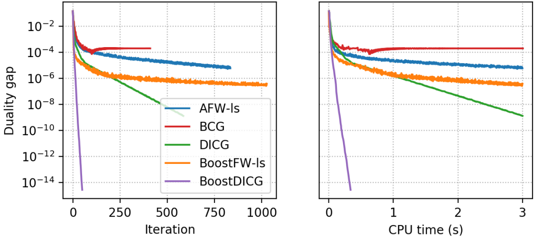
$a_t \leftarrow$ away vertex

$g_t \leftarrow \text{procedure}(a_t, -\nabla f(x_t), K, \delta)$

$x_{t+1} \leftarrow x_t + \gamma_t g_t$

Boosting DICG

- DICG is known to perform particularly well on the video co-localization experiment (YouTube-Objects dataset)
- BoostDICG: application of our method to DICG



- (*details*)

DICG

$a_t \leftarrow$ away vertex

$v_t \leftarrow \arg\min_{v \in \mathcal{V}} \langle \nabla f(x_t), v \rangle$

$x_{t+1} \leftarrow x_t + \gamma_t (v_t - a_t)$

BoostDICG

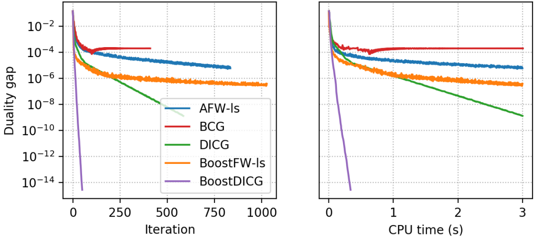
$a_t \leftarrow$ away vertex

$g_t \leftarrow \text{procedure}(a_t, -\nabla f(x_t), K, \delta)$

$x_{t+1} \leftarrow x_t + \gamma_t g_t$

Boosting DICG

- DICG is known to perform particularly well on the video co-localization experiment (YouTube-Objects dataset)
- BoostDICG: application of our method to DICG



- (*details*)

DICG

$a_t \leftarrow$ away vertex

$v_t \leftarrow \arg\min_{v \in \mathcal{V}} \langle \nabla f(x_t), v \rangle$

$x_{t+1} \leftarrow x_t + \gamma_t(v_t - a_t)$

BoostDICG

$a_t \leftarrow$ away vertex

$g_t \leftarrow \text{procedure}(a_t, -\nabla f(x_t), K, \delta)$

$x_{t+1} \leftarrow x_t + \gamma_t g_t$

Takeaways and final remarks

- Projection-free algorithms are of considerable interest in optimization

Takeaways and final remarks

- Projection-free algorithms are of considerable interest in optimization
- We have proposed an intuitive and generic boosting procedure to speed up Frank-Wolfe algorithms

Takeaways and final remarks

- Projection-free algorithms are of considerable interest in optimization
- We have proposed an intuitive and generic boosting procedure to speed up Frank-Wolfe algorithms
- Although our method may perform more linear minimizations per iteration, the progress obtained greatly overcomes their cost

Takeaways and final remarks

- Projection-free algorithms are of considerable interest in optimization
- We have proposed an intuitive and generic boosting procedure to speed up Frank-Wolfe algorithms
- Although our method may perform more linear minimizations per iteration, the progress obtained greatly overcomes their cost
- We focused on smooth convex objective functions, but we expect our method to provide significant gains in performance in other areas of optimization as well

Takeaways and final remarks

- Projection-free algorithms are of considerable interest in optimization
- We have proposed an intuitive and generic boosting procedure to speed up Frank-Wolfe algorithms
- Although our method may perform more linear minimizations per iteration, the progress obtained greatly overcomes their cost
- We focused on smooth convex objective functions, but we expect our method to provide significant gains in performance in other areas of optimization as well

  E.g., large-scale finite-sum/stochastic constrained optimization:

$$
\begin{aligned}
g_t &\leftarrow \text{procedure}(x_t, -\tilde{\nabla} f(x_t), K, \delta) \\
x_{t+1} &\leftarrow x_t + \gamma_t g_t
\end{aligned}
$$

References

G. Braun, S. Pokutta, D. Tu, and S. Wright. Blended conditional gradients: the unconditioning of conditional gradients. *ICML*, 2019.

M. D. Canon and C. D. Cullum. A tight upper bound on the rate of convergence of Frank-Wolfe algorithm. *SIAM J. Control*, 1968.

C. W. Combettes and S. Pokutta. Boosting Frank-Wolfe by chasing gradients. *ICML*, 2020.

M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Res. Logist. Q.*, 1956.

D. Garber and O. Meshi. Linear-memory and decomposition-invariant linearly convergent conditional gradient algorithm for structured polytopes. *NIPS*, 2016.

M. Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. *ICML*, 2013.

S. Lacoste-Julien and M. Jaggi. On the global linear convergence of Frank-Wolfe optimization variants. *NIPS*, 2015.

G. Lan. The complexity of large-scale convex programming under a linear optimization oracle. Technical report, University of Florida, 2013.

E. S. Levitin and B. T. Polyak. Constrained minimization methods. *USSR Comp. Math. Math. Phys.*, 1966.

F. Locatello, M. Tschannen, G. Rätsch, and M. Jaggi. Greedy algorithms for cone constrained optimization with convergence guarantees. *NIPS*, 2017.

P. Wolfe. Convergence theory in nonlinear programming. *Integer and Nonlinear Programming*. North-Holland, 1970.