Frank-Wolfe with New and Practical Descent Directions

Cyrille W. Combettes

School of Industrial and Systems Engineering
Georgia Institute of Technology

IOL & COGA Research Seminar
Zuse Institute Berlin and TU Berlin
October 27, 2020

Outline

Introduction

Consider

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

where

- $\mathcal{C} \subset \mathbb{R}^n$ is a compact convex set
- $f : \mathbb{R}^n \to \mathbb{R}$ is a smooth convex function

Introduction

Consider

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

where

- $\mathcal{C} \subset \mathbb{R}^n$ is a compact convex set
- $f : \mathbb{R}^n \to \mathbb{R}$ is a smooth convex function

Example

- Sparse logistic regression

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

- Low-rank matrix completion

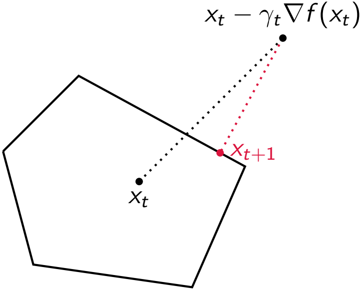
$$\min_{X \in \mathbb{R}^{m \times n}} \frac{1}{2|\mathcal{I}|} \sum_{(i,j) \in \mathcal{I}} (Y_{i,j} - X_{i,j})^2$$
$$\text{s.t. } \|X\|_{\text{nuc}} \leqslant \tau$$

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	$\mathcal{O}(\text{nonzeros})$	$\mathcal{O}(mn \min\{m, n\})$
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 general 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	$\mathcal{O}(\text{nonzeros})$	$\mathcal{O}(mn\min\{m,n\})$
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 general 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	$\mathcal{O}(\text{nonzeros})$	$\mathcal{O}(mn\min\{m,n\})$
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 general 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	$\mathcal{O}(\text{nonzeros})$	$\mathcal{O}(mn\min\{m,n\})$
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 general 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	$\mathcal{O}(\text{nonzeros})$	$\mathcal{O}(mn \min\{m, n\})$
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 general 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):

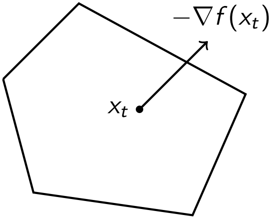
Algorithm Frank-Wolfe (FW)

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

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

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

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):

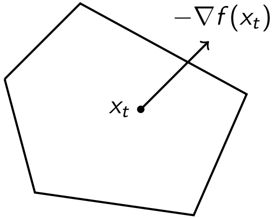
Algorithm Frank-Wolfe (FW)

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

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

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

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):

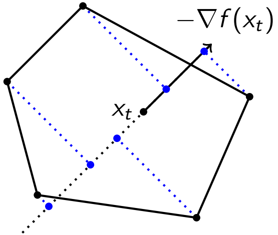
Algorithm Frank-Wolfe (FW)

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

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

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

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):

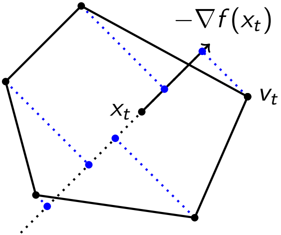
Algorithm Frank-Wolfe (FW)

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

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

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

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):

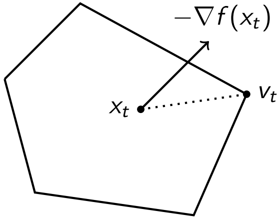
Algorithm Frank-Wolfe (FW)

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

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

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

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):

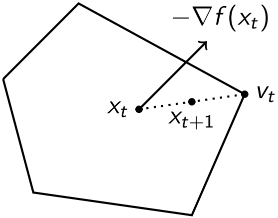
Algorithm Frank-Wolfe (FW)

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

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

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

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):

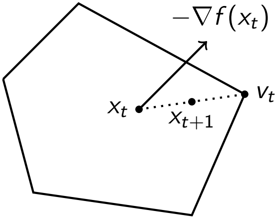
Algorithm Frank-Wolfe (FW)

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

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

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

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):

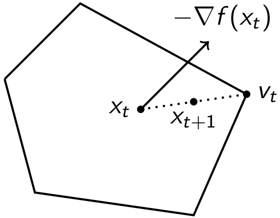
Algorithm Frank-Wolfe (FW)

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

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

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

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):

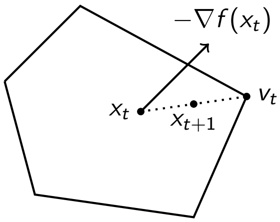
Algorithm Frank-Wolfe (FW)

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

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

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

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):

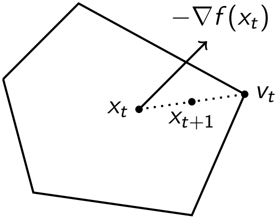
Algorithm Frank-Wolfe (FW)

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

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

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

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
- Successfully applied to: traffic assignment, computer vision, optimal transport, adversarial learning, etc.

The Frank-Wolfe algorithm

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

Let $\mathcal{C} \subset \mathbb{R}^n$ *be a compact convex set with diameter* D *and* $f : \mathbb{R}^n \to \mathbb{R}$ *be a* L*-smooth convex function, and let* $x_0 \in \arg\min_{v \in \mathcal{C}} \langle \nabla f(y), v \rangle$ *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 $\mathcal{C} \subset \mathbb{R}^n$ be a compact convex set with diameter D and $f : \mathbb{R}^n \to \mathbb{R}$ be a L-smooth convex function, and let $x_0 \in \arg\min_{v \in \mathcal{C}} \langle \nabla f(y), v \rangle$ 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 $\mathcal{C} \subset \mathbb{R}^n$ be a compact convex set with diameter D and $f : \mathbb{R}^n \to \mathbb{R}$ be a L-smooth convex function, and let $x_0 \in \arg\min_{v \in \mathcal{C}} \langle \nabla f(y), v \rangle$ 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

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

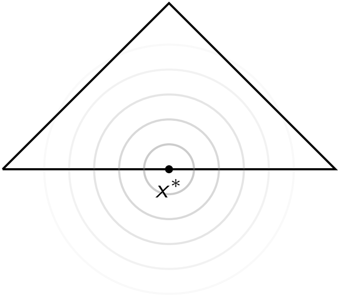
- The convergence rate cannot be improved (Canon & Cullum, 1968; Jaggi, 2013; Lan, 2013)
- Why?

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

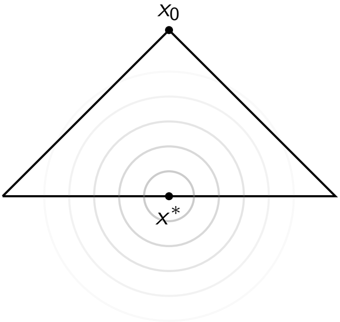


Consider the simple problem

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

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

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

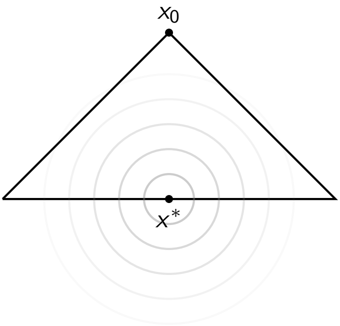


The Frank-Wolfe algorithm

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



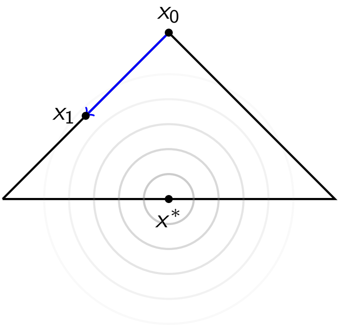
- Let $x_0 = \begin{pmatrix}0\\1\end{pmatrix}$
- FW tries to reach x^* by moving towards vertices

The Frank-Wolfe algorithm

Consider the simple problem

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

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



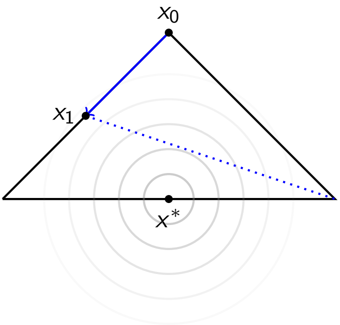
- Let $x_0 = \begin{pmatrix}0\\1\end{pmatrix}$
- FW tries to reach x^* by moving towards vertices

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



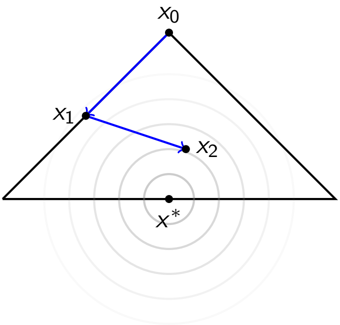
- Let $x_0 = \begin{pmatrix}0\\1\end{pmatrix}$
- FW tries to reach x^* by moving towards vertices

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



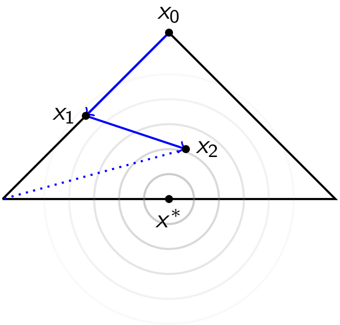
- Let $x_0 = \begin{pmatrix}0\\1\end{pmatrix}$
- FW tries to reach x^* by moving towards vertices

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



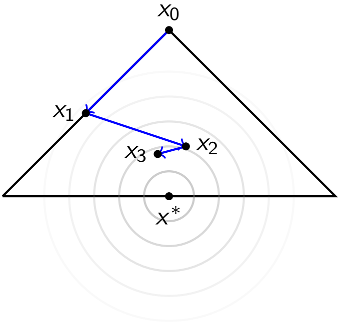
- Let $x_0 = \begin{pmatrix}0\\1\end{pmatrix}$
- FW tries to reach x^* by moving towards vertices

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

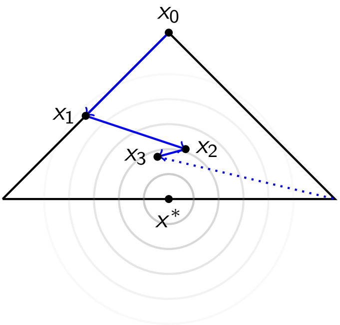


- Let $x_0 = \begin{pmatrix}0\\1\end{pmatrix}$
- FW tries to reach x^* by moving towards vertices

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



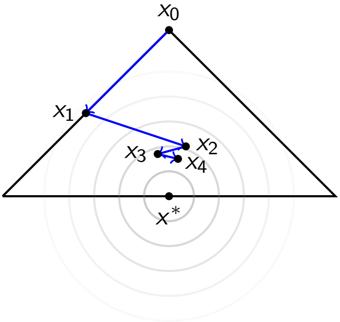
- Let $x_0 = \begin{pmatrix}0\\1\end{pmatrix}$
- FW tries to reach x^* by moving towards vertices

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



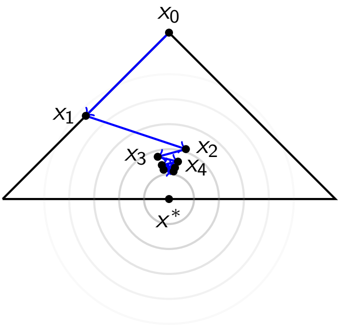
- Let $x_0 = \begin{pmatrix}0\\1\end{pmatrix}$
- FW tries to reach x^* by moving towards vertices

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



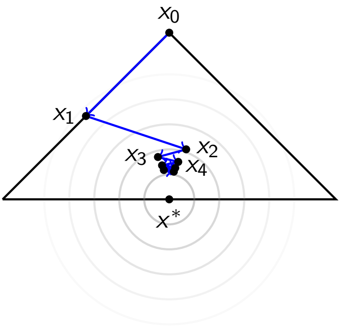
- Let $x_0 = \begin{pmatrix}0\\1\end{pmatrix}$
- FW tries to reach x^* by moving towards vertices

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



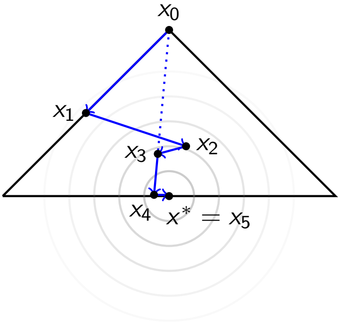
- Let $x_0 = \begin{pmatrix}0\\1\end{pmatrix}$
- FW tries to reach x^* by moving towards vertices
- This yields an inefficient zig-zagging trajectory

Improved Frank-Wolfe variants

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

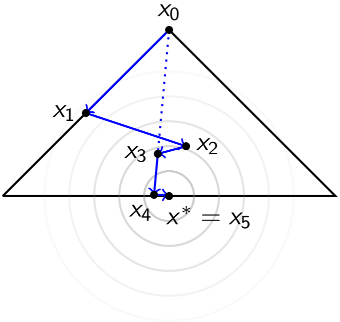
Improved Frank-Wolfe variants

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



Improved Frank-Wolfe variants

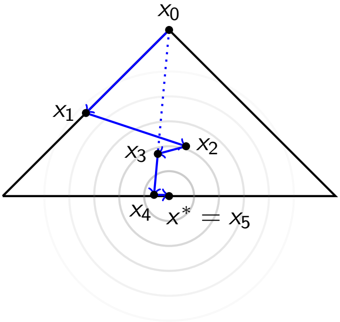
- Away-Step Frank-Wolfe (AFW) (Wolfe, 1970; Lacoste-Julien & Jaggi, 2015): enhances FW by allowing to move away from vertices



- 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 (C & Pokutta, 2020):

- 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 (C & Pokutta, 2020):

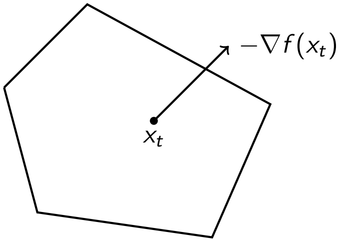
- Speed up FW by moving in a direction better aligned with $-\nabla f(x_t)$
- Build this direction by using $\mathcal{C}$ to maintain the projection-free property

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

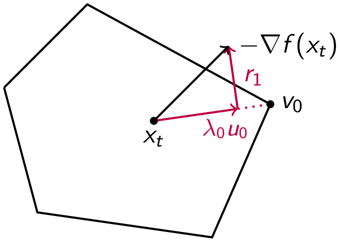
- How can we build a direction better aligned with $-\nabla f(x_t)$ and that allows to update x_{t+1} without projection?



Boosting Frank-Wolfe

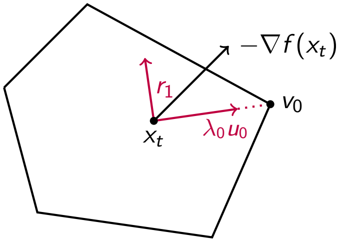
- How can we build a direction better aligned with $-\nabla f(x_t)$ and that allows to update x_{t+1} without projection?

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



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

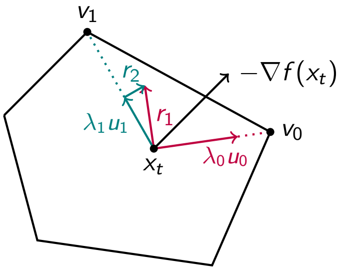


Boosting Frank-Wolfe

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

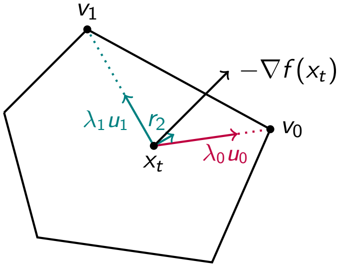
- $v_0 \in \arg\max_{v \in \mathcal{C}} \langle -\nabla f(x_t), v \rangle$
  $\lambda_0 u_0 = \frac{\langle -\nabla f(x_t), v_0 - x_t \rangle}{\|v_0 - x_t\|^2}(v_0 - x_t)$
  $r_1 = -\nabla f(x_t) - \lambda_0 u_0$

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



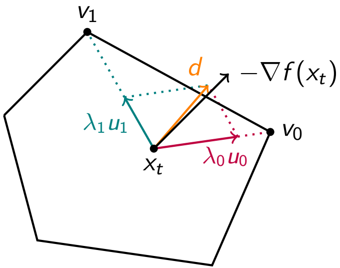
Boosting Frank-Wolfe

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



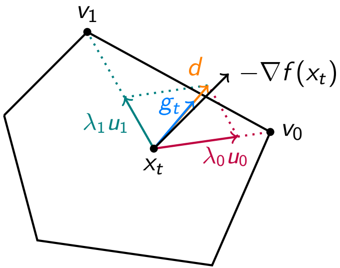
Boosting Frank-Wolfe

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



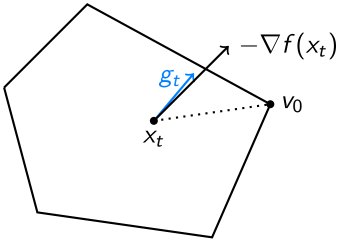
Boosting Frank-Wolfe

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



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



- 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?

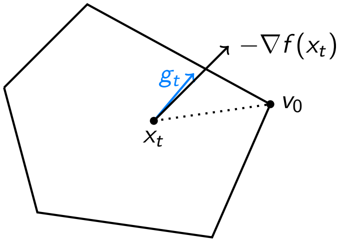
- $v_0 \in \arg\max_{v \in \mathcal{C}} \langle -\nabla f(x_t), v \rangle$
  $\lambda_0 u_0 = \frac{\langle -\nabla f(x_t), v_0 - x_t \rangle}{\|v_0 - x_t\|^2}(v_0 - x_t)$
  $r_1 = -\nabla f(x_t) - \lambda_0 u_0$

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

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

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

- $g_t = d/(\lambda_0 + \lambda_1)$



- 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

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

Boosting Frank-Wolfe

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

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

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

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

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

$d_0 \leftarrow 0$, $\Lambda \leftarrow 0$
for $k = 0$ **to** $K - 1$ **do**
  $r_k \leftarrow \nabla - d_k$ ▷ k-th residual
  $v_k \leftarrow \arg\max_{v \in \mathcal{C}} \langle r_k, v \rangle$ ▷ 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 $\operatorname{align}(\nabla, d'_k) - \operatorname{align}(\nabla, d_k) \geqslant \delta$ **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

Boosting Frank-Wolfe

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

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

$d_0 \leftarrow 0$, $\Lambda \leftarrow 0$
for $k = 0$ **to** $K-1$ **do**
  $r_k \leftarrow \nabla - d_k$ ▷ k-th residual
  $v_k \leftarrow \arg\max_{v \in \mathcal{C}} \langle r_k, v \rangle$ ▷ 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: $z \in \mathcal{C}$, $\nabla \in \mathbb{R}^n$, $K \in \mathbb{N}\backslash\{0\}$, $\delta \in]0, 1[$.

$d_0 \leftarrow 0$, $\Lambda \leftarrow 0$
for $k = 0$ **to** $K - 1$ **do**
  $r_k \leftarrow \nabla - d_k$ ▷ k-th residual
  $v_k \leftarrow \arg\max_{v \in \mathcal{C}} \langle r_k, v \rangle$ ▷ 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: $\quad v_t \leftarrow \operatorname*{arg\,min}_{v \in \mathcal{C}} \langle \nabla f(x_t), v \rangle$

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

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

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

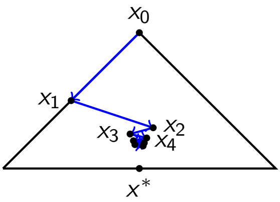
Algorithm Frank-Wolfe (FW)

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

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

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

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



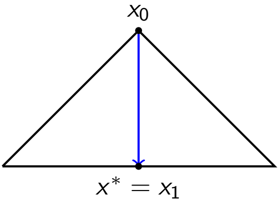
Algorithm Boosted Frank-Wolfe (BoostFW)

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



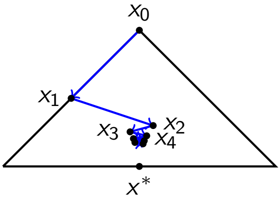
Algorithm Frank-Wolfe (FW)

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

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

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

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



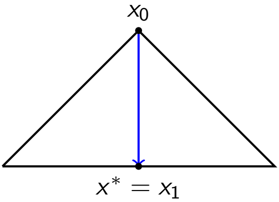
Algorithm Boosted Frank-Wolfe (BoostFW)

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

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

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

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



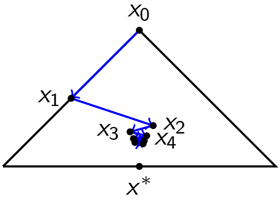
- What is the convergence rate of BoostFW?

Boosting Frank-Wolfe

Algorithm Frank-Wolfe (FW)

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

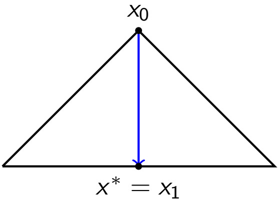
for $t = 0$ **to** $T-1$ **do**
  $v_t \leftarrow \arg\min_{v \in \mathcal{C}} \langle \nabla f(x_t), v \rangle$
  $x_{t+1} \leftarrow x_t + \gamma_t (v_t - x_t)$



Algorithm Boosted Frank-Wolfe (BoostFW)

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

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



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

Boosting Frank-Wolfe

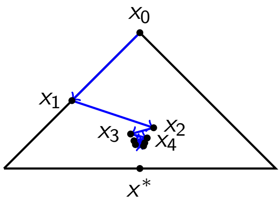
Algorithm Frank-Wolfe (FW)

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

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

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

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



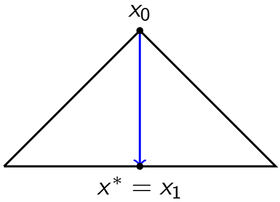
Algorithm Boosted Frank-Wolfe (BoostFW)

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



- 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 (C & Pokutta, 2020)

Let $\mathcal{C} \subset \mathbb{R}^n$ be a compact convex set with diameter D and $f : \mathbb{R}^n \to \mathbb{R}$ be a L-smooth, convex, and μ-gradient dominated function, and let $x_0 \in \arg\min_{v \in \mathcal{C}} \langle \nabla f(y), v \rangle$ 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 $N_t \geqslant \omega t$. Then

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

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 (C & Pokutta, 2020)

Let $\mathcal{C} \subset \mathbb{R}^n$ be a compact convex set with diameter D and $f : \mathbb{R}^n \to \mathbb{R}$ be a L-smooth, convex, and μ-gradient dominated function, and let $x_0 \in \arg\min_{v \in \mathcal{C}} \langle \nabla f(y), v \rangle$ 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 $N_t \geqslant \omega t$. Then

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

- The assumption $N_t \geqslant \omega t$ 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 (C & Pokutta, 2020)

Let $\mathcal{C} \subset \mathbb{R}^n$ be a compact convex set with diameter D and $f : \mathbb{R}^n \to \mathbb{R}$ be a L-smooth, convex, and μ-gradient dominated function, and let $x_0 \in \arg\min_{v \in \mathcal{C}} \langle \nabla f(y), v \rangle$ 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 $N_t \geqslant \omega t$. Then

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

- The assumption $N_t \geqslant \omega t$ 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 (C & Pokutta, 2020)

Let $\mathcal{C} \subset \mathbb{R}^n$ be a compact convex set with diameter D and $f : \mathbb{R}^n \to \mathbb{R}$ be a L-smooth, convex, and μ-gradient dominated function, and let $x_0 \in \arg\min_{v \in \mathcal{C}} \langle \nabla f(y), v \rangle$ 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 $N_t \geqslant \omega t$. Then

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

- The assumption $N_t \geqslant \omega t$ 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 $\omega \lesssim 1$)

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

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

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

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

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

Computational experiments

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

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

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

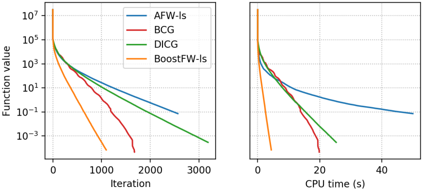
$$\begin{aligned}\min_{x\in\mathbb{R}^n}\ & \frac{1}{m}\sum_{i=1}^m \ln(1+\exp(-y_i\langle a_i,x\rangle))\\ \text{s.t. } & \|x\|_1 \leqslant \tau\end{aligned}$$

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

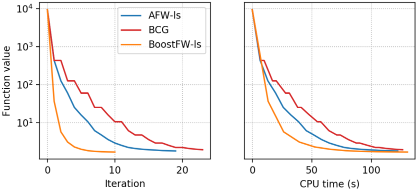
- For BoostFW and AFW we also run the line search-free variants (the "short step" strategy) and label them with an "L"

Computational experiments

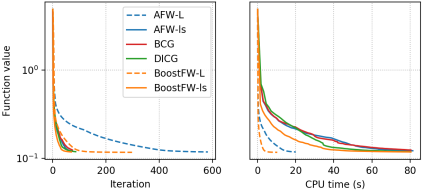
- Sparse signal recovery



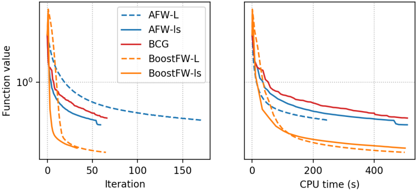
- Traffic assignment



- Sparse logistic regression on the Gisette dataset

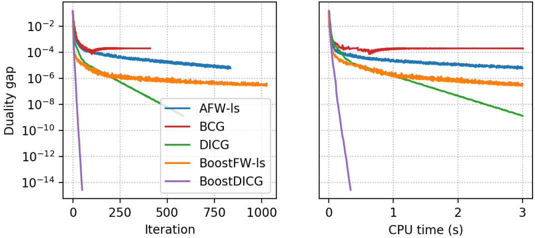


- Collaborative filtering on the MovieLens 100k dataset



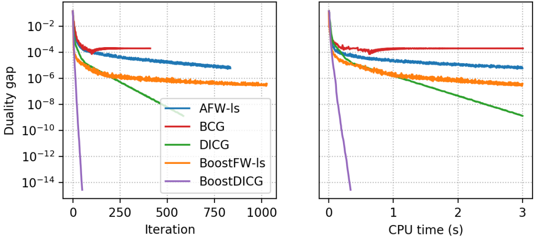
Boosting DICG

- 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

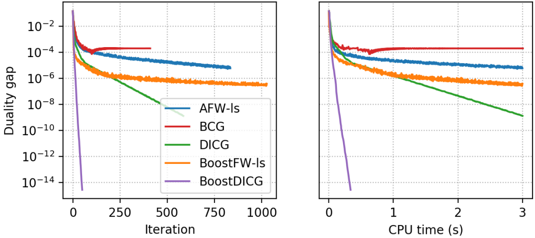


- (*details*)

DICG	BoostDICG
$a_t \leftarrow$ away vertex	$a_t \leftarrow$ away vertex
$v_t \leftarrow \arg\min_{v \in \mathcal{C}} \langle \nabla f(x_t), v \rangle$	$g_t \leftarrow \text{procedure}(a_t, -\nabla f(x_t), K, \delta)$
$x_{t+1} \leftarrow x_t + \gamma_t (v_t - a_t)$	$x_{t+1} \leftarrow x_t + \gamma_t g_t$

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

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

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

BoostDICG

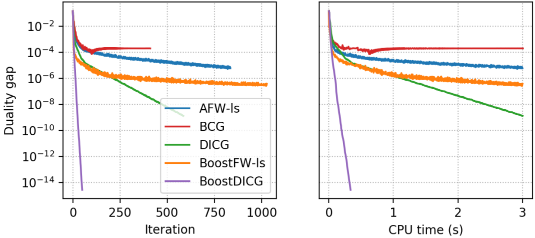
$a_t \leftarrow$ away vertex

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

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

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	BoostDICG
$a_t \leftarrow$ away vertex	$a_t \leftarrow$ away vertex
$v_t \leftarrow \arg\min_{v \in \mathcal{C}} \langle \nabla f(x_t), v \rangle$	$g_t \leftarrow \text{procedure}(a_t, -\nabla f(x_t), K, \delta)$
$x_{t+1} \leftarrow x_t + \gamma_t (v_t - a_t)$	$x_{t+1} \leftarrow x_t + \gamma_t g_t$

Takeaways

- BoostFW is an intuitive and generic procedure to speed up Frank-Wolfe algorithms

Takeaways

- BoostFW is an intuitive and generic procedure to speed up Frank-Wolfe algorithms
- Although it performs more linear minimizations per iteration, the progress obtained greatly overcomes their cost

Takeaways

- BoostFW is an intuitive and generic procedure to speed up Frank-Wolfe algorithms
- Although it performs more linear minimizations per iteration, the progress obtained greatly overcomes their cost
- The boosting procedure can be applied to any descent direction $-d_t$ (obtained from, e.g., momentum acceleration, stochasticity, etc.):

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

Large-scale optimization

Consider

$$\min \left\{ f(x) := \frac{1}{m} \sum_{i=1}^{m} f_i(x) \right\}$$
$$\text{s.t. } x \in \mathcal{C}$$

where

- $\mathcal{C} \subset \mathbb{R}^n$ is a compact convex set
- $f_1, \ldots, f_m \colon \mathbb{R}^n \to \mathbb{R}$ are smooth (non)convex functions
- $m \ggg 1$ is very large

Large-scale optimization

Consider

$$\min \left\{ f(x) := \frac{1}{m} \sum_{i=1}^{m} f_i(x) \right\}$$
$$\text{s.t. } x \in \mathcal{C}$$

where

- $\mathcal{C} \subset \mathbb{R}^n$ is a compact convex set
- $f_1, \ldots, f_m \colon \mathbb{R}^n \to \mathbb{R}$ are smooth (non)convex functions
- $m \ggg 1$ is very large

Computing $f(x)$ or $\nabla f(x)$ is too expensive

- Cannot use line search
- More efficient to use an estimator $\tilde{\nabla} f(x)$ to get approximate (but cheap) gradient information

Stochastic Frank-Wolfe algorithms

Template Stochastic Frank-Wolfe

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

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

2: Update the gradient estimator $\tilde{\nabla} f(x_t)$

3: $v_t \leftarrow \arg\min_{v \in \mathcal{C}} \langle \tilde{\nabla} f(x_t), v \rangle$

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

Stochastic Frank-Wolfe algorithms

Template Stochastic Frank-Wolfe

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

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

2: Update the gradient estimator $\tilde{\nabla} f(x_t)$

3: $v_t \leftarrow \arg\min_{v \in \mathcal{C}} \langle \tilde{\nabla} f(x_t), v \rangle$

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

Stochastic Frank-Wolfe algorithms

Template Stochastic Frank-Wolfe

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

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

2: Update the gradient estimator $\tilde{\nabla} f(x_t)$

3: $v_t \leftarrow \arg\min_{v \in \mathcal{C}} \langle \tilde{\nabla} f(x_t), v \rangle$

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

Stochastic Frank-Wolfe algorithms

Template Stochastic Frank-Wolfe

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

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

2: Update the gradient estimator $\tilde{\nabla} f(x_t)$

3: $v_t \leftarrow \arg\min_{v \in \mathcal{C}} \langle \tilde{\nabla} f(x_t), v \rangle$

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

Typical analysis: let $\varepsilon_t := f(x_t) - \min_{\mathcal{C}} f$, then by smoothness, convexity, and Cauchy-Schwarz,

$$\mathrm{E}[\varepsilon_{t+1}] \leqslant (1 - \gamma_t)\mathrm{E}[\varepsilon_t] + \gamma_t \mathrm{E}[\|\tilde{\nabla} f(x_t) - \nabla f(x_t)\|] D + \frac{L}{2}\gamma_t^2 D^2$$

Stochastic Frank-Wolfe algorithms

Template Stochastic Frank-Wolfe

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

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

2: Update the gradient estimator $\tilde{\nabla} f(x_t)$

3: $v_t \leftarrow \underset{v \in \mathcal{C}}{\arg\min} \langle \tilde{\nabla} f(x_t), v \rangle$

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

Typical analysis: let $\varepsilon_t := f(x_t) - \min_{\mathcal{C}} f$, then by smoothness, convexity, and Cauchy-Schwarz,

$$\mathrm{E}[\varepsilon_{t+1}] \leqslant (1-\gamma_t)\mathrm{E}[\varepsilon_t] + \gamma_t \mathrm{E}[\|\tilde{\nabla} f(x_t) - \nabla f(x_t)\|] D + \frac{L}{2}\gamma_t^2 D^2$$

Stochastic Frank-Wolfe algorithms

Template Stochastic Frank-Wolfe

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

1: **for** $t = 0$ **to** $T - 1$ **do**
2: Update the gradient estimator $\tilde{\nabla} f(x_t)$
3: $v_t \leftarrow \arg\min_{v \in \mathcal{C}} \langle \tilde{\nabla} f(x_t), v \rangle$
4: $x_{t+1} \leftarrow x_t + \gamma_t (v_t - x_t)$

Typical analysis: let $\varepsilon_t := f(x_t) - \min_{\mathcal{C}} f$, then by smoothness, convexity, and Cauchy-Schwarz,

$$\mathrm{E}[\varepsilon_{t+1}] \leqslant (1 - \gamma_t)\mathrm{E}[\varepsilon_t] + \gamma_t \mathrm{E}[\|\tilde{\nabla} f(x_t) - \nabla f(x_t)\|] D + \frac{L}{2}\gamma_t^2 D^2$$

By Jensen's inequality,

$$\mathrm{E}[\|\tilde{\nabla} f(x_t) - \nabla f(x_t)\|] \leqslant \sqrt{\mathrm{E}[\|\tilde{\nabla} f(x_t) - \nabla f(x_t)\|^2]}$$

Stochastic Frank-Wolfe algorithms

Template Stochastic Frank-Wolfe

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

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

2: Update the gradient estimator $\tilde{\nabla} f(x_t)$

3: $v_t \leftarrow \arg\min_{v\in\mathcal{C}} \langle \tilde{\nabla} f(x_t), v\rangle$

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

Typical analysis: let $\varepsilon_t := f(x_t) - \min_{\mathcal{C}} f$, then by smoothness, convexity, and Cauchy-Schwarz,

$$\mathrm{E}[\varepsilon_{t+1}] \leqslant (1-\gamma_t)\mathrm{E}[\varepsilon_t] + \gamma_t \mathrm{E}[\|\tilde{\nabla} f(x_t) - \nabla f(x_t)\|] D + \frac{L}{2}\gamma_t^2 D^2$$

By Jensen's inequality,

$$\mathrm{E}[\|\tilde{\nabla} f(x_t) - \nabla f(x_t)\|] \leqslant \sqrt{\mathrm{E}[\|\tilde{\nabla} f(x_t) - \nabla f(x_t)\|^2]}$$

To obtain $\mathrm{E}[\varepsilon_t] = \mathcal{O}(1/t)$, we need $\mathrm{E}[\|\tilde{\nabla} f(x_t) - \nabla f(x_t)\|^2] = \mathcal{O}(1/t^2)$

Stochastic Frank-Wolfe algorithms

- The *vanilla* Stochastic Frank-Wolfe algorithm (SFW) estimates the gradient by averaging over a minibatch of size b_t:

$$\tilde{\nabla} f(x_t) \leftarrow \frac{1}{b_t} \sum_{i=i_1}^{i_{b_t}} \nabla f_i(x_t) \quad \text{where} \quad i_1, \ldots, i_{b_t} \overset{\text{i.i.d.}}{\sim} \mathcal{U}([\![1, m]\!])$$

Stochastic Frank-Wolfe algorithms

- The *vanilla* Stochastic Frank-Wolfe algorithm (SFW) estimates the gradient by averaging over a minibatch of size b_t:

$$\tilde{\nabla} f(x_t) \leftarrow \frac{1}{b_t} \sum_{i=i_1}^{i_{b_t}} \nabla f_i(x_t) \quad \text{where} \quad i_1, \ldots, i_{b_t} \overset{\text{i.i.d.}}{\sim} \mathcal{U}([\![1, m]\!])$$

This estimator is unbiased and its variance is bounded by

$$\mathrm{E}[\|\tilde{\nabla} f(x_t) - \nabla f(x_t)\|^2] \leqslant \frac{G^2}{b_t} \quad \text{where} \quad G := \max_{i \in [\![1,m]\!]} \max_{x \in \mathcal{C}} \|\nabla f_i(x)\|$$

Stochastic Frank-Wolfe algorithms

- The *vanilla* Stochastic Frank-Wolfe algorithm (SFW) estimates the gradient by averaging over a minibatch of size b_t:

$$\tilde{\nabla} f(x_t) \leftarrow \frac{1}{b_t} \sum_{i=i_1}^{i_{b_t}} \nabla f_i(x_t) \quad \text{where} \quad i_1, \ldots, i_{b_t} \overset{\text{i.i.d.}}{\sim} \mathcal{U}([\![1, m]\!])$$

This estimator is unbiased and its variance is bounded by

$$\mathrm{E}[\|\tilde{\nabla} f(x_t) - \nabla f(x_t)\|^2] \leqslant \frac{G^2}{b_t} \quad \text{where} \quad G := \max_{i \in [\![1,m]\!]} \max_{x \in \mathcal{C}} \|\nabla f_i(x)\|$$

so $b_t = \Theta(t^2)$ works

Stochastic Frank-Wolfe algorithms

- The *vanilla* Stochastic Frank-Wolfe algorithm (SFW) estimates the gradient by averaging over a minibatch of size b_t:

$$\tilde{\nabla} f(x_t) \leftarrow \frac{1}{b_t} \sum_{i=i_1}^{i_{b_t}} \nabla f_i(x_t) \quad \text{where} \quad i_1, \ldots, i_{b_t} \overset{\text{i.i.d.}}{\sim} \mathcal{U}(\llbracket 1, m \rrbracket)$$

  This estimator is unbiased and its variance is bounded by

$$\mathrm{E}[\|\tilde{\nabla} f(x_t) - \nabla f(x_t)\|^2] \leqslant \frac{G^2}{b_t} \quad \text{where} \quad G := \max_{i \in \llbracket 1, m \rrbracket} \max_{x \in \mathcal{C}} \|\nabla f_i(x)\|$$

  so $b_t = \Theta(t^2)$ works

- Using variance reduction, the Stochastic Variance-Reduced Frank-Wolfe algorithm (SVRF) (Hazan & Luo, 2016) satisfies

$$\mathrm{E}[\|\tilde{\nabla} f(x_t) - \nabla f(x_t)\|^2] \leqslant \frac{4L}{b_t}(\mathrm{E}[\varepsilon_t] + \mathrm{E}[\tilde{\varepsilon}_t])$$

Stochastic Frank-Wolfe algorithms

- The *vanilla* Stochastic Frank-Wolfe algorithm (SFW) estimates the gradient by averaging over a minibatch of size b_t:

$$\tilde{\nabla} f(x_t) \leftarrow \frac{1}{b_t} \sum_{i=i_1}^{i_{b_t}} \nabla f_i(x_t) \quad \text{where} \quad i_1, \ldots, i_{b_t} \overset{\text{i.i.d.}}{\sim} \mathcal{U}(\llbracket 1, m \rrbracket)$$

This estimator is unbiased and its variance is bounded by

$$\mathrm{E}[\|\tilde{\nabla} f(x_t) - \nabla f(x_t)\|^2] \leqslant \frac{G^2}{b_t} \quad \text{where} \quad G := \max_{i \in \llbracket 1, m \rrbracket} \max_{x \in \mathcal{C}} \|\nabla f_i(x)\|$$

so $b_t = \Theta(t^2)$ works

- Using variance reduction, the Stochastic Variance-Reduced Frank-Wolfe algorithm (SVRF) (Hazan & Luo, 2016) satisfies

$$\mathrm{E}[\|\tilde{\nabla} f(x_t) - \nabla f(x_t)\|^2] \leqslant \frac{4L}{b_t}(\mathrm{E}[\varepsilon_t] + \mathrm{E}[\tilde{\varepsilon}_t])$$

and $b_t = \Theta(t)$ works (by induction)

Stochastic Frank-Wolfe algorithms

- The *vanilla* Stochastic Frank-Wolfe algorithm (SFW) estimates the gradient by averaging over a minibatch of size b_t:

$$\tilde{\nabla} f(x_t) \leftarrow \frac{1}{b_t} \sum_{i=i_1}^{i_{b_t}} \nabla f_i(x_t) \quad \text{where} \quad i_1, \dots, i_{b_t} \overset{\text{i.i.d.}}{\sim} \mathcal{U}(\llbracket 1, m \rrbracket)$$

  This estimator is unbiased and its variance is bounded by

$$\mathrm{E}[\|\tilde{\nabla} f(x_t) - \nabla f(x_t)\|^2] \leqslant \frac{G^2}{b_t} \quad \text{where} \quad G := \max_{i \in \llbracket 1, m \rrbracket} \max_{x \in \mathcal{C}} \|\nabla f_i(x)\|$$

  so $b_t = \Theta(t^2)$ works

- Using variance reduction, the Stochastic Variance-Reduced Frank-Wolfe algorithm (SVRF) (Hazan & Luo, 2016) satisfies

$$\mathrm{E}[\|\tilde{\nabla} f(x_t) - \nabla f(x_t)\|^2] \leqslant \frac{4L}{b_t}(\mathrm{E}[\varepsilon_t] + \mathrm{E}[\tilde{\varepsilon}_t])$$

  and $b_t = \Theta(t)$ works (by induction)

- See also, e.g., Shen et al. (2019), Yurtsever et al. (2019), Xie et al. (2020), Zhang et al. (2020), Négiar et al. (2020)

Stochastic Frank-Wolfe algorithms

Algorithm	**Update** $\tilde{\nabla} f(x_t)$
SFW	$\frac{1}{b_t}\sum_{i=i_1}^{i_{b_t}} \nabla f_i(x_t)$
SVRF	$\nabla f(\tilde{x}_t) + \frac{1}{b_t}\sum_{i=i_1}^{i_{b_t}} (\nabla f_i(x_t) - \nabla f_i(\tilde{x}_t))$
SPIDER-FW	$\nabla f(\tilde{x}_t) + \frac{1}{b_t}\sum_{i=i_1}^{i_{b_t}} (\nabla f_i(x_t) - \nabla f_i(x_{t-1}))$
ORGFW	$\frac{1}{b_t}\sum_{i=i_1}^{i_{b_t}} \nabla f_i(x_t) + (1-\rho_t)\left(\tilde{\nabla} f(x_{t-1}) - \frac{1}{b_t}\sum_{i=i_1}^{i_{b_t}} \nabla f_i(x_{t-1})\right)$
CSFW	$\tilde{\nabla} f(x_{t-1}) + \sum_{i=i_1}^{i_{b_t}} \left(\frac{1}{m} f_i'(\langle a_i, x_t\rangle) - [\alpha_{t-1}]_i\right) a_i$
	and $[\alpha_t]_i \leftarrow \begin{cases} (1/m) f_i'(\langle a_i, x_t\rangle) & \text{if } i \in \{i_1, \ldots, i_{b_t}\} \\ [\alpha_{t-1}]_i & \text{else} \end{cases}$

Stochastic Frank-Wolfe algorithms

Algorithm	**Update** $\tilde{\nabla} f(x_t)$
SFW	$\frac{1}{b_t}\sum_{i=i_1}^{i_{b_t}} \nabla f_i(x_t)$
SVRF	$\nabla f(\tilde{x}_t) + \frac{1}{b_t}\sum_{i=i_1}^{i_{b_t}}(\nabla f_i(x_t) - \nabla f_i(\tilde{x}_t))$
SPIDER-FW	$\nabla f(\tilde{x}_t) + \frac{1}{b_t}\sum_{i=i_1}^{i_{b_t}}(\nabla f_i(x_t) - \nabla f_i(x_{t-1}))$
ORGFW	$\frac{1}{b_t}\sum_{i=i_1}^{i_{b_t}} \nabla f_i(x_t) + (1-\rho_t)\left(\tilde{\nabla} f(x_{t-1}) - \frac{1}{b_t}\sum_{i=i_1}^{i_{b_t}} \nabla f_i(x_{t-1})\right)$
CSFW	$\tilde{\nabla} f(x_{t-1}) + \sum_{i=i_1}^{i_{b_t}}\left(\frac{1}{m} f_i'(\langle a_i, x_t\rangle) - [\alpha_{t-1}]_i\right) a_i$ and $[\alpha_t]_i \leftarrow \begin{cases}(1/m) f_i'(\langle a_i, x_t\rangle) & \text{if } i \in \{i_1, \ldots, i_{b_t}\}\\ [\alpha_{t-1}]_i & \text{else}\end{cases}$

Stochastic Frank-Wolfe algorithms

Algorithm	**Update** $\tilde{\nabla} f(x_t)$
SFW	$\frac{1}{b_t}\sum_{i=i_1}^{i_{b_t}} \nabla f_i(x_t)$
SVRF	$\nabla f(\tilde{x}_t) + \frac{1}{b_t}\sum_{i=i_1}^{i_{b_t}} (\nabla f_i(x_t) - \nabla f_i(\tilde{x}_t))$
SPIDER-FW	$\nabla f(\tilde{x}_t) + \frac{1}{b_t}\sum_{i=i_1}^{i_{b_t}} (\nabla f_i(x_t) - \nabla f_i(x_{t-1}))$
ORGFW	$\frac{1}{b_t}\sum_{i=i_1}^{i_{b_t}} \nabla f_i(x_t) + (1-\rho_t)\left(\tilde{\nabla} f(x_{t-1}) - \frac{1}{b_t}\sum_{i=i_1}^{i_{b_t}} \nabla f_i(x_{t-1})\right)$
CSFW	$\tilde{\nabla} f(x_{t-1}) + \sum_{i=i_1}^{i_{b_t}} \left(\frac{1}{m} f_i'(\langle a_i, x_t\rangle) - [\alpha_{t-1}]_i\right) a_i$ and $[\alpha_t]_i \leftarrow \begin{cases} (1/m) f_i'(\langle a_i, x_t\rangle) & \text{if } i \in \{i_1, \ldots, i_{b_t}\} \\ [\alpha_{t-1}]_i & \text{else} \end{cases}$

Stochastic Frank-Wolfe algorithms

Algorithm	Update $\tilde{\nabla} f(x_t)$
SFW	$\displaystyle\frac{1}{b_t}\sum_{i=i_1}^{i_{b_t}} \nabla f_i(x_t)$
SVRF	$\displaystyle\nabla f(\tilde{x}_t) + \frac{1}{b_t}\sum_{i=i_1}^{i_{b_t}} (\nabla f_i(x_t) - \nabla f_i(\tilde{x}_t))$
SPIDER-FW	$\displaystyle\nabla f(\tilde{x}_t) + \frac{1}{b_t}\sum_{i=i_1}^{i_{b_t}} (\nabla f_i(x_t) - \nabla f_i(x_{t-1}))$
ORGFW	$\displaystyle\frac{1}{b_t}\sum_{i=i_1}^{i_{b_t}} \nabla f_i(x_t) + (1-\rho_t)\left(\tilde{\nabla} f(x_{t-1}) - \frac{1}{b_t}\sum_{i=i_1}^{i_{b_t}} \nabla f_i(x_{t-1})\right)$
CSFW	$\displaystyle\tilde{\nabla} f(x_{t-1}) + \sum_{i=i_1}^{i_{b_t}} \left(\frac{1}{m} f_i'(\langle a_i, x_t\rangle) - [\alpha_{t-1}]_i\right) a_i$ and $[\alpha_t]_i \leftarrow \begin{cases} (1/m) f_i'(\langle a_i, x_t\rangle) & \text{if } i \in \{i_1, \ldots, i_{b_t}\} \\ [\alpha_{t-1}]_i & \text{else} \end{cases}$

Stochastic Frank-Wolfe algorithms

Algorithm	**Update** $\tilde{\nabla} f(x_t)$
SFW	$\displaystyle \frac{1}{b_t}\sum_{i=i_1}^{i_{b_t}} \nabla f_i(x_t)$
SVRF	$\displaystyle \nabla f(\tilde{x}_t) + \frac{1}{b_t}\sum_{i=i_1}^{i_{b_t}} (\nabla f_i(x_t) - \nabla f_i(\tilde{x}_t))$
SPIDER-FW	$\displaystyle \nabla f(\tilde{x}_t) + \frac{1}{b_t}\sum_{i=i_1}^{i_{b_t}} (\nabla f_i(x_t) - \nabla f_i(x_{t-1}))$
ORGFW	$\displaystyle \frac{1}{b_t}\sum_{i=i_1}^{i_{b_t}} \nabla f_i(x_t) + (1-\rho_t)\left(\tilde{\nabla} f(x_{t-1}) - \frac{1}{b_t}\sum_{i=i_1}^{i_{b_t}} \nabla f_i(x_{t-1})\right)$
CSFW	$\displaystyle \tilde{\nabla} f(x_{t-1}) + \sum_{i=i_1}^{i_{b_t}} \left(\frac{1}{m} f_i'(\langle a_i, x_t\rangle) - [\alpha_{t-1}]_i\right) a_i$ and $[\alpha_t]_i \leftarrow \begin{cases} (1/m) f_i'(\langle a_i, x_t\rangle) & \text{if } i \in \{i_1, \dots, i_{b_t}\} \\ [\alpha_{t-1}]_i & \text{else} \end{cases}$

Stochastic Frank-Wolfe algorithms

Algorithm	**Update** $\tilde{\nabla} f(x_t)$
SFW	$\displaystyle \frac{1}{b_t}\sum_{i=i_1}^{i_{b_t}} \nabla f_i(x_t)$
SVRF	$\displaystyle \nabla f(\tilde{x}_t) + \frac{1}{b_t}\sum_{i=i_1}^{i_{b_t}} (\nabla f_i(x_t) - \nabla f_i(\tilde{x}_t))$
SPIDER-FW	$\displaystyle \nabla f(\tilde{x}_t) + \frac{1}{b_t}\sum_{i=i_1}^{i_{b_t}} (\nabla f_i(x_t) - \nabla f_i(x_{t-1}))$
ORGFW	$\displaystyle \frac{1}{b_t}\sum_{i=i_1}^{i_{b_t}} \nabla f_i(x_t) + (1-\rho_t)\left(\tilde{\nabla} f(x_{t-1}) - \frac{1}{b_t}\sum_{i=i_1}^{i_{b_t}} \nabla f_i(x_{t-1})\right)$
CSFW	$\displaystyle \tilde{\nabla} f(x_{t-1}) + \sum_{i=i_1}^{i_{b_t}} \left(\frac{1}{m} f_i'(\langle a_i, x_t\rangle) - [\alpha_{t-1}]_i\right) a_i$
	and $[\alpha_t]_i \leftarrow \begin{cases} (1/m) f_i'(\langle a_i, x_t\rangle) & \text{if } i \in \{i_1, \ldots, i_{b_t}\} \\ [\alpha_{t-1}]_i & \text{else} \end{cases}$

The Adaptive Gradient algorithm

Simultaneously proposed by Duchi et al. (2011) and McMahan & Streeter (2010):

Algorithm Adaptive Gradient (AdaGrad)

Input: $x_0 \in \mathcal{C}$, $\delta > 0$, $\eta > 0$.

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

2: Update the gradient estimator $\tilde{\nabla} f(x_t)$

3: $H_t \leftarrow \operatorname{diag}\left(\delta 1 + \sqrt{\sum_{s=0}^{t} \tilde{\nabla} f(x_s)^2}\right)$

4: $x_{t+1} \leftarrow \operatorname*{arg\,min}_{x \in \mathcal{C}} \eta \langle \tilde{\nabla} f(x_t), x \rangle + \frac{1}{2} \|x - x_t\|_{H_t}^2$

The Adaptive Gradient algorithm

Simultaneously proposed by Duchi et al. (2011) and McMahan & Streeter (2010):

Algorithm Adaptive Gradient (AdaGrad)

Input: $x_0 \in \mathcal{C}$, $\delta > 0$, $\eta > 0$.

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

2: Update the gradient estimator $\tilde{\nabla} f(x_t)$

3: $H_t \leftarrow \operatorname{diag}\left(\delta 1 + \sqrt{\sum_{s=0}^{t} \tilde{\nabla} f(x_s)^2}\right)$

4: $x_{t+1} \leftarrow \arg\min_{x \in \mathcal{C}} \eta \langle \tilde{\nabla} f(x_t), x \rangle + \frac{1}{2} \|x - x_t\|_{H_t}^2$

The Adaptive Gradient algorithm

Simultaneously proposed by Duchi et al. (2011) and McMahan & Streeter (2010):

Algorithm Adaptive Gradient (AdaGrad)

Input: $x_0 \in \mathcal{C}$, $\delta > 0$, $\eta > 0$.

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

2: Update the gradient estimator $\tilde{\nabla} f(x_t)$

3: $H_t \leftarrow \text{diag}\left(\delta 1 + \sqrt{\sum_{s=0}^{t} \tilde{\nabla} f(x_s)^2}\right)$

4: $x_{t+1} \leftarrow \arg\min_{x \in \mathcal{C}} \eta \langle \tilde{\nabla} f(x_t), x \rangle + \frac{1}{2} \|x - x_t\|_{H_t}^2$

The Adaptive Gradient algorithm

Simultaneously proposed by Duchi et al. (2011) and McMahan & Streeter (2010):

Algorithm Adaptive Gradient (AdaGrad)

Input: $x_0 \in \mathcal{C}$, $\delta > 0$, $\eta > 0$.

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

2: Update the gradient estimator $\tilde{\nabla} f(x_t)$

3: $H_t \leftarrow \operatorname{diag}\left(\delta 1 + \sqrt{\sum_{s=0}^{t} \tilde{\nabla} f(x_s)^2}\right)$

4: $x_{t+1} \leftarrow \underset{x \in \mathcal{C}}{\arg\min}\, \eta \langle \tilde{\nabla} f(x_t), x \rangle + \frac{1}{2} \|x - x_t\|_{H_t}^2$

The Adaptive Gradient algorithm

Simultaneously proposed by Duchi et al. (2011) and McMahan & Streeter (2010):

Algorithm Adaptive Gradient (AdaGrad)

Input: $x_0 \in \mathcal{C}$, $\delta > 0$, $\eta > 0$.

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

2: Update the gradient estimator $\tilde{\nabla} f(x_t)$

3: $$H_t \leftarrow \operatorname{diag}\left(\delta \mathbb{1} + \sqrt{\sum_{s=0}^{t} \tilde{\nabla} f(x_s)^2}\right)$$

4: $$x_{t+1} \leftarrow \operatorname*{arg\,min}_{x \in \mathcal{C}} \eta \langle \tilde{\nabla} f(x_t), x \rangle + \frac{1}{2} \|x - x_t\|_{H_t}^2$$

- We denote $\|u\|_{H_t}^2 = \langle u, H_t u \rangle$

The Adaptive Gradient algorithm

Simultaneously proposed by Duchi et al. (2011) and McMahan & Streeter (2010):

Algorithm Adaptive Gradient (AdaGrad)

Input: $x_0 \in \mathcal{C}$, $\delta > 0$, $\eta > 0$.

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

2: Update the gradient estimator $\tilde{\nabla} f(x_t)$

3: $$H_t \leftarrow \operatorname{diag}\left(\delta \mathbb{1} + \sqrt{\sum_{s=0}^{t} \tilde{\nabla} f(x_s)^2}\right)$$

4: $$x_{t+1} \leftarrow \arg\min_{x \in \mathcal{C}} \eta \langle \tilde{\nabla} f(x_t), x \rangle + \frac{1}{2} \|x - x_t\|_{H_t}^2$$

- We denote $\|u\|_{H_t}^2 = \langle u, H_t u \rangle$
- The default value for the offset is $\delta \leftarrow 10^{-8}$

The Adaptive Gradient algorithm

Simultaneously proposed by Duchi et al. (2011) and McMahan & Streeter (2010):

Algorithm Adaptive Gradient (AdaGrad)

Input: $x_0 \in \mathcal{C}$, $\delta > 0$, $\eta > 0$.

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

2: Update the gradient estimator $\tilde{\nabla} f(x_t)$

3: $$H_t \leftarrow \operatorname{diag}\left(\delta \mathbb{1} + \sqrt{\sum_{s=0}^{t} \tilde{\nabla} f(x_s)^2}\right)$$

4: $$x_{t+1} \leftarrow \operatorname*{arg\,min}_{x \in \mathcal{C}} \eta \langle \tilde{\nabla} f(x_t), x \rangle + \frac{1}{2} \|x - x_t\|_{H_t}^2$$

- We denote $\|u\|_{H_t}^2 = \langle u, H_t u \rangle$
- The default value for the offset is $\delta \leftarrow 10^{-8}$

The Adaptive Gradient algorithm

By first-order optimality condition (Polyak, 1987),

$$x_{t+1} \leftarrow \arg\min_{x \in \mathcal{C}} \|x - (x_t - \eta H_t^{-1} \tilde{\nabla} f(x_t))\|_{H_t}$$

The Adaptive Gradient algorithm

By first-order optimality condition (Polyak, 1987),

$$x_{t+1} \leftarrow \arg\min_{x \in \mathcal{C}} \|x - (x_t - \eta H_t^{-1} \tilde{\nabla} f(x_t))\|_{H_t}$$

Ignoring the constraint set $\mathcal{C}$ for ease of exposition, we obtain

$$x_{t+1} \leftarrow x_t - \eta H_t^{-1} \tilde{\nabla} f(x_t)$$

The Adaptive Gradient algorithm

By first-order optimality condition (Polyak, 1987),

$$x_{t+1} \leftarrow \arg\min_{x \in \mathcal{C}} \|x - (x_t - \eta H_t^{-1} \tilde{\nabla} f(x_t))\|_{H_t}$$

Ignoring the constraint set $\mathcal{C}$ for ease of exposition, we obtain

$$x_{t+1} \leftarrow x_t - \eta H_t^{-1} \tilde{\nabla} f(x_t)$$

i.e., for every feature $i \in [\![1, n]\!]$,

$$[x_{t+1}]_i \leftarrow [x_t]_i - \frac{\eta [\tilde{\nabla} f(x_t)]_i}{\delta + \sqrt{\sum_{s=0}^{t} [\tilde{\nabla} f(x_s)]_i^2}}$$

The Adaptive Gradient algorithm

By first-order optimality condition (Polyak, 1987),

$$x_{t+1} \leftarrow \arg\min_{x \in \mathcal{C}} \|x - (x_t - \eta H_t^{-1} \tilde{\nabla} f(x_t))\|_{H_t}$$

Ignoring the constraint set $\mathcal{C}$ for ease of exposition, we obtain

$$x_{t+1} \leftarrow x_t - \eta H_t^{-1} \tilde{\nabla} f(x_t)$$

i.e., for every feature $i \in [\![1, n]\!]$,

$$[x_{t+1}]_i \leftarrow [x_t]_i - \frac{\eta [\tilde{\nabla} f(x_t)]_i}{\delta + \sqrt{\sum_{s=0}^{t} [\tilde{\nabla} f(x_s)]_i^2}}$$

- The offset δ prevents from dividing by zero

The Adaptive Gradient algorithm

By first-order optimality condition (Polyak, 1987),

$$x_{t+1} \leftarrow \arg\min_{x \in \mathcal{C}} \|x - (x_t - \eta H_t^{-1} \tilde{\nabla} f(x_t))\|_{H_t}$$

Ignoring the constraint set $\mathcal{C}$ for ease of exposition, we obtain

$$x_{t+1} \leftarrow x_t - \eta H_t^{-1} \tilde{\nabla} f(x_t)$$

i.e., for every feature $i \in [\![1, n]\!]$,

$$[x_{t+1}]_i \leftarrow [x_t]_i - \frac{\eta [\tilde{\nabla} f(x_t)]_i}{\delta + \sqrt{\sum_{s=0}^{t} [\tilde{\nabla} f(x_s)]_i^2}}$$

- The offset δ prevents from dividing by zero
- The step-size automatically adjusts to the geometry of the problem

The Adaptive Gradient algorithm

We have

$$[x_{t+1}]_i \leftarrow [x_t]_i - \frac{\eta[\tilde{\nabla} f(x_t)]_i}{\delta + \sqrt{\sum_{s=0}^{t} [\tilde{\nabla} f(x_s)]_i^2}}$$

The Adaptive Gradient algorithm

We have

$$[x_{t+1}]_i \leftarrow [x_t]_i - \frac{\eta[\tilde{\nabla} f(x_t)]_i}{\delta + \sqrt{\sum_{s=0}^{t}[\tilde{\nabla} f(x_s)]_i^2}}$$

so

- If $[\tilde{\nabla} f(x_0)]_i = \ldots = [\tilde{\nabla} f(x_{t-1})]_i = 0$ and $[\tilde{\nabla} f(x_t)]_i > 0$ (feature i is "rare") then

$$[x_{t+1}]_i \approx [x_t]_i - \eta$$

The Adaptive Gradient algorithm

We have

$$[x_{t+1}]_i \leftarrow [x_t]_i - \frac{\eta[\tilde{\nabla} f(x_t)]_i}{\delta + \sqrt{\sum_{s=0}^{t}[\tilde{\nabla} f(x_s)]_i^2}}$$

so

- If $[\tilde{\nabla} f(x_0)]_i = \ldots = [\tilde{\nabla} f(x_{t-1})]_i = 0$ and $[\tilde{\nabla} f(x_t)]_i > 0$ (feature i is "rare") then

$$[x_{t+1}]_i \approx [x_t]_i - \eta$$

- If $[\tilde{\nabla} f(x_0)]_i = \ldots = [\tilde{\nabla} f(x_t)]_i = 1$ (feature i is "common") then

$$[x_{t+1}]_i \approx [x_t]_i - \frac{\eta}{\sqrt{t+1}}$$

The Adaptive Gradient algorithm

We have

$$[x_{t+1}]_i \leftarrow [x_t]_i - \frac{\eta[\tilde{\nabla} f(x_t)]_i}{\delta + \sqrt{\sum_{s=0}^{t}[\tilde{\nabla} f(x_s)]_i^2}}$$

so

- If $[\tilde{\nabla} f(x_0)]_i = \ldots = [\tilde{\nabla} f(x_{t-1})]_i = 0$ and $[\tilde{\nabla} f(x_t)]_i > 0$ (feature i is "rare") then

$$[x_{t+1}]_i \approx [x_t]_i - \eta$$

- If $[\tilde{\nabla} f(x_0)]_i = \ldots = [\tilde{\nabla} f(x_t)]_i = 1$ (feature i is "common") then

$$[x_{t+1}]_i \approx [x_t]_i - \frac{\eta}{\sqrt{t+1}}$$

Larger step-sizes are given to infrequent (but potentially very informative) features whenever they appear so that they do not go unnoticed. This adjusts the trajectory of the iterates

Frank-Wolfe with adaptive gradients

- How can we use adaptive gradients in FW?

Frank-Wolfe with adaptive gradients

- How can we use adaptive gradients in FW?
- Let $G_t = H_t^{-1}\tilde{\nabla} f(x_t)$, then unconstrained AdaGrad is

$$x_{t+1} \leftarrow x_t - \eta G_t$$

Frank-Wolfe with adaptive gradients

- How can we use adaptive gradients in FW?
- Let $G_t = H_t^{-1}\tilde{\nabla} f(x_t)$, then unconstrained AdaGrad is

$$x_{t+1} \leftarrow x_t - \eta G_t$$

Could we do

$$v_t \leftarrow \underset{v \in \mathcal{C}}{\arg\min} \langle G_t, v \rangle$$
$$x_{t+1} \leftarrow x_t + \gamma_t (v_t - x_t)$$

as did FW for unconstrained gradient descent (for which $G_t = \nabla f(x_t)$)?

Frank-Wolfe with adaptive gradients

- How can we use adaptive gradients in FW?
- Let $G_t = H_t^{-1}\tilde{\nabla} f(x_t)$, then unconstrained AdaGrad is

$$x_{t+1} \leftarrow x_t - \eta G_t$$

  Could we do

$$v_t \leftarrow \arg\min_{v \in \mathcal{C}} \langle G_t, v \rangle$$
$$x_{t+1} \leftarrow x_t + \gamma_t (v_t - x_t)$$

  as did FW for unconstrained gradient descent (for which $G_t = \nabla f(x_t)$)?

- We would likely lose the precious properties of the descent directions of AdaGrad

Frank-Wolfe with adaptive gradients

- Instead, consider the constrained subproblem occurring at every iteration:

$$x_{t+1} \leftarrow \arg\min_{x \in \mathcal{C}} \eta \langle \tilde{\nabla} f(x_t), x \rangle + \frac{1}{2} \|x - x_t\|_{H_t}^2$$

Frank-Wolfe with adaptive gradients

- Instead, consider the constrained subproblem occurring at every iteration:

$$x_{t+1} \leftarrow \operatorname*{arg\,min}_{x \in \mathcal{C}} \eta \langle \tilde{\nabla} f(x_t), x \rangle + \frac{1}{2} \|x - x_t\|_{H_t}^2$$

- This can become quite expensive and inefficient overall

Frank-Wolfe with adaptive gradients

- Instead, consider the constrained subproblem occurring at every iteration:

$$x_{t+1} \leftarrow \arg\min_{x \in \mathcal{C}} \eta \langle \tilde{\nabla} f(x_t), x \rangle + \frac{1}{2} \|x - x_t\|_{H_t}^2$$

- This can become quite expensive and inefficient overall
- Note that AdaGrad is usually used for unconstrained optimization

Frank-Wolfe with adaptive gradients

- Instead, consider the constrained subproblem occurring at every iteration:

$$x_{t+1} \leftarrow \underset{x \in \mathcal{C}}{\arg\min}\, \eta \langle \tilde{\nabla} f(x_t), x \rangle + \frac{1}{2} \|x - x_t\|_{H_t}^2$$

- This can become quite expensive and inefficient overall
- Note that AdaGrad is usually used for unconstrained optimization

Idea (C et al., 2020):

- Solve the subproblem using FW (*sliding* technique (Lan & Zhou, 2016))

Frank-Wolfe with adaptive gradients

- Instead, consider the constrained subproblem occurring at every iteration:

$$x_{t+1} \leftarrow \operatorname*{arg\,min}_{x \in \mathcal{C}} \eta \langle \tilde{\nabla} f(x_t), x \rangle + \frac{1}{2} \|x - x_t\|_{H_t}^2$$

- This can become quite expensive and inefficient overall
- Note that AdaGrad is usually used for unconstrained optimization

Idea (C et al., 2020):

- Solve the subproblem using FW (*sliding* technique (Lan & Zhou, 2016))
- Run only a small and fixed number K of iterations of FW ($K \sim 5$)

Frank-Wolfe with adaptive gradients

- Instead, consider the constrained subproblem occurring at every iteration:

$$x_{t+1} \leftarrow \arg\min_{x \in \mathcal{C}} \eta \langle \tilde{\nabla} f(x_t), x \rangle + \frac{1}{2} \|x - x_t\|_{H_t}^2$$

- This can become quite expensive and inefficient overall
- Note that AdaGrad is usually used for unconstrained optimization

Idea (C et al., 2020):

- Solve the subproblem using FW (*sliding* technique (Lan & Zhou, 2016))
- Run only a small and fixed number K of iterations of FW ($K \sim 5$)
- We claim that leveraging just a small amount of information from the adaptive metric H_t is enough

Frank-Wolfe with adaptive gradients

Template Frank-Wolfe with adaptive gradients

Input: $x_0 \in \mathcal{C}$, $0 < \lambda_t^- \leqslant \lambda_{t+1}^- \leqslant \lambda_{t+1}^+ \leqslant \lambda_t^+$, $K \in \mathbb{N}\backslash\{0\}$, $\eta > 0$, $\gamma_t \in [0,1]$.

for $t = 0$ **to** $T-1$ **do**
  Update the gradient estimator $\tilde{\nabla} f(x_t)$
  Update the diagonal matrix H_t and clip its entries to $[\lambda_t^-, \lambda_t^+]$
  $y_0^{(t)} \leftarrow x_t$
  for $k = 0$ **to** $K-1$ **do**
    $\nabla Q_t(y_k^{(t)}) \leftarrow \tilde{\nabla} f(x_t) + \dfrac{1}{\eta_t} H_t(y_k^{(t)} - x_t)$
    $v_k^{(t)} \leftarrow \operatorname*{arg\,min}_{v \in \mathcal{C}} \langle \nabla Q_t(y_k^{(t)}), v \rangle$
    $\gamma_k^{(t)} \leftarrow \min\left\{ \eta_t \dfrac{\langle \nabla Q_t(y_k^{(t)}), y_k^{(t)} - v_k^{(t)} \rangle}{\|y_k^{(t)} - v_k^{(t)}\|_{H_t}^2}, \gamma_t \right\}$
    $y_{k+1} \leftarrow y_k^{(t)} + \gamma_k^{(t)}(v_k^{(t)} - y_k^{(t)})$
  $x_{t+1} \leftarrow y_K^{(t)}$

Frank-Wolfe with adaptive gradients

Template Frank-Wolfe with adaptive gradients

Input: $x_0 \in \mathcal{C}$, $0 < \lambda_t^- \leqslant \lambda_{t+1}^- \leqslant \lambda_{t+1}^+ \leqslant \lambda_t^+$, $K \in \mathbb{N} \backslash \{0\}$, $\eta > 0$, $\gamma_t \in [0, 1]$.

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

  Update the gradient estimator $\tilde{\nabla} f(x_t)$

  Update the diagonal matrix H_t and clip its entries to $[\lambda_t^-, \lambda_t^+]$

  $y_0^{(t)} \leftarrow x_t$

  for $k = 0$ **to** $K - 1$ **do**

    $\nabla Q_t(y_k^{(t)}) \leftarrow \tilde{\nabla} f(x_t) + \dfrac{1}{\eta_t} H_t(y_k^{(t)} - x_t)$

    $v_k^{(t)} \leftarrow \underset{v \in \mathcal{C}}{\arg\min} \langle \nabla Q_t(y_k^{(t)}), v \rangle$

    $\gamma_k^{(t)} \leftarrow \min \left\{ \eta_t \dfrac{\langle \nabla Q_t(y_k^{(t)}), y_k^{(t)} - v_k^{(t)} \rangle}{\|y_k^{(t)} - v_k^{(t)}\|_{H_t}^2}, \gamma_t \right\}$

    $y_{k+1} \leftarrow y_k^{(t)} + \gamma_k^{(t)} (v_k^{(t)} - y_k^{(t)})$

  $x_{t+1} \leftarrow y_K^{(t)}$

Frank-Wolfe with adaptive gradients

Template Frank-Wolfe with adaptive gradients

Input: $x_0 \in \mathcal{C}$, $0 < \lambda_t^- \leqslant \lambda_{t+1}^- \leqslant \lambda_{t+1}^+ \leqslant \lambda_t^+$, $K \in \mathbb{N} \backslash \{0\}$, $\eta > 0$, $\gamma_t \in [0, 1]$.

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

  Update the gradient estimator $\tilde{\nabla} f(x_t)$

  Update the diagonal matrix H_t and clip its entries to $[\lambda_t^-, \lambda_t^+]$

  $y_0^{(t)} \leftarrow x_t$

  for $k = 0$ **to** $K - 1$ **do**

$$\nabla Q_t(y_k^{(t)}) \leftarrow \tilde{\nabla} f(x_t) + \frac{1}{\eta_t} H_t(y_k^{(t)} - x_t)$$

$$v_k^{(t)} \leftarrow \arg\min_{v \in \mathcal{C}} \langle \nabla Q_t(y_k^{(t)}), v \rangle$$

$$\gamma_k^{(t)} \leftarrow \min \left\{ \eta_t \frac{\langle \nabla Q_t(y_k^{(t)}), y_k^{(t)} - v_k^{(t)} \rangle}{\|y_k^{(t)} - v_k^{(t)}\|_{H_t}^2}, \gamma_t \right\}$$

$$y_{k+1} \leftarrow y_k^{(t)} + \gamma_k^{(t)}(v_k^{(t)} - y_k^{(t)})$$

  $x_{t+1} \leftarrow y_K^{(t)}$

Frank-Wolfe with adaptive gradients

Template Frank-Wolfe with adaptive gradients

Input: $x_0 \in \mathcal{C}$, $0 < \lambda_t^- \leqslant \lambda_{t+1}^- \leqslant \lambda_{t+1}^+ \leqslant \lambda_t^+$, $K \in \mathbb{N}\backslash\{0\}$, $\eta > 0$, $\gamma_t \in [0,1]$.

1: **for** $t = 0$ **to** $T-1$ **do**
2: Update the gradient estimator $\tilde{\nabla} f(x_t)$
3: Update the diagonal matrix H_t and clip its entries to $[\lambda_t^-, \lambda_t^+]$
4: $y_0^{(t)} \leftarrow x_t$
5: **for** $k = 0$ **to** $K-1$ **do**
6: $\nabla Q_t(y_k^{(t)}) \leftarrow \tilde{\nabla} f(x_t) + \frac{1}{\eta_t} H_t(y_k^{(t)} - x_t)$
7: $v_k^{(t)} \leftarrow \arg\min_{v \in \mathcal{C}} \langle \nabla Q_t(y_k^{(t)}), v \rangle$
8: $\gamma_k^{(t)} \leftarrow \min \left\{ \eta_t \frac{\langle \nabla Q_t(y_k^{(t)}), y_k^{(t)} - v_k^{(t)} \rangle}{\|y_k^{(t)} - v_k^{(t)}\|_{H_t}^2}, \gamma_t \right\}$
9: $y_{k+1} \leftarrow y_k^{(t)} + \gamma_k^{(t)}(v_k^{(t)} - y_k^{(t)})$
10: $x_{t+1} \leftarrow y_K^{(t)}$

- Lines 4-9 apply K iterations of FW to $\min_{x \in \mathcal{C}} \left\{ Q_t(x) := f(x_t) + \langle \tilde{\nabla} f(x_t), x - x_t \rangle + \frac{1}{2\eta_t} \|x - x_t\|_{H_t}^2 \right\}$

Frank-Wolfe with adaptive gradients

Template Frank-Wolfe with adaptive gradients

Input: $x_0 \in \mathcal{C}$, $0 < \lambda_t^- \leqslant \lambda_{t+1}^- \leqslant \lambda_{t+1}^+ \leqslant \lambda_t^+$, $K \in \mathbb{N}\backslash\{0\}$, $\eta > 0$, $\gamma_t \in [0, 1]$.

1: **for** $t = 0$ **to** $T - 1$ **do**
2: Update the gradient estimator $\tilde{\nabla} f(x_t)$
3: Update the diagonal matrix H_t and clip its entries to $[\lambda_t^-, \lambda_t^+]$
4: $y_0^{(t)} \leftarrow x_t$
5: **for** $k = 0$ **to** $K - 1$ **do**
6: $\nabla Q_t(y_k^{(t)}) \leftarrow \tilde{\nabla} f(x_t) + \frac{1}{\eta_t} H_t(y_k^{(t)} - x_t)$
7: $v_k^{(t)} \leftarrow \arg\min_{v \in \mathcal{C}} \langle \nabla Q_t(y_k^{(t)}), v \rangle$
8: $\gamma_k^{(t)} \leftarrow \min \left\{ \eta_t \frac{\langle \nabla Q_t(y_k^{(t)}), y_k^{(t)} - v_k^{(t)} \rangle}{\|y_k^{(t)} - v_k^{(t)}\|_{H_t}^2}, \gamma_t \right\}$
9: $y_{k+1} \leftarrow y_k^{(t)} + \gamma_k^{(t)}(v_k^{(t)} - y_k^{(t)})$
10: $x_{t+1} \leftarrow y_K^{(t)}$

- Lines 4-9 apply K iterations of FW to $\min_{x \in \mathcal{C}} \left\{ Q_t(x) := f(x_t) + \langle \tilde{\nabla} f(x_t), x - x_t \rangle + \frac{1}{2\eta_t} \|x - x_t\|_{H_t}^2 \right\}$

Frank-Wolfe with adaptive gradients

Template Frank-Wolfe with adaptive gradients

Input: $x_0 \in \mathcal{C}$, $0 < \lambda_t^- \leqslant \lambda_{t+1}^- \leqslant \lambda_{t+1}^+ \leqslant \lambda_t^+$, $K \in \mathbb{N}\backslash\{0\}$, $\eta > 0$, $\gamma_t \in [0, 1]$.

1: **for** $t = 0$ **to** $T - 1$ **do**
2: Update the gradient estimator $\tilde{\nabla} f(x_t)$
3: Update the diagonal matrix H_t and clip its entries to $[\lambda_t^-, \lambda_t^+]$
4: $y_0^{(t)} \leftarrow x_t$
5: **for** $k = 0$ **to** $K - 1$ **do**
6: $\nabla Q_t(y_k^{(t)}) \leftarrow \tilde{\nabla} f(x_t) + \frac{1}{\eta_t} H_t(y_k^{(t)} - x_t)$
7: $v_k^{(t)} \leftarrow \arg\min_{v \in \mathcal{C}} \langle \nabla Q_t(y_k^{(t)}), v \rangle$
8: $\gamma_k^{(t)} \leftarrow \min\left\{ \eta_t \frac{\langle \nabla Q_t(y_k^{(t)}), y_k^{(t)} - v_k^{(t)} \rangle}{\|y_k^{(t)} - v_k^{(t)}\|_{H_t}^2}, \gamma_t \right\}$
9: $y_{k+1} \leftarrow y_k^{(t)} + \gamma_k^{(t)}(v_k^{(t)} - y_k^{(t)})$
10: $x_{t+1} \leftarrow y_K^{(t)}$

- Lines 4-9 apply K iterations of FW to $\min_{x \in \mathcal{C}} \left\{ Q_t(x) := f(x_t) + \langle \tilde{\nabla} f(x_t), x - x_t \rangle + \frac{1}{2\eta_t} \|x - x_t\|_{H_t}^2 \right\}$
- AdaX depending on the strategy for $\tilde{\nabla} f(x_t)$: AdaSFW, AdaSVRF, etc.

Frank-Wolfe with adaptive gradients

Theorem (C et al., 2020)

Let $\mathcal{C} \subset \mathbb{R}^n$ be a compact convex set with diameter D and $f_1, \ldots, f_m \colon \mathbb{R}^n \to \mathbb{R}$ be L-smooth convex functions. Then AdaSFW with $b_t \leftarrow \big(G(t+2)/(LD)\big)^2$, $\eta_t \leftarrow \lambda_t^-/L$, and $\gamma_t \leftarrow 2/(t+2)$ satisfies

$$\mathbb{E}[f(x_t)] - \min_{\mathcal{C}} f \leqslant \frac{2LD^2(K+1+\kappa)}{t+1}$$

where $\kappa \coloneqq \lambda_0^+/\lambda_0^-$

Frank-Wolfe with adaptive gradients

Theorem (C et al., 2020)

Let $\mathcal{C} \subset \mathbb{R}^n$ be a compact convex set with diameter D and $f_1, \ldots, f_m \colon \mathbb{R}^n \to \mathbb{R}$ be L-smooth convex functions. Then AdaSFW with $b_t \leftarrow \big(G(t+2)/(LD)\big)^2$, $\eta_t \leftarrow \lambda_t^- / L$, and $\gamma_t \leftarrow 2/(t+2)$ satisfies

$$\mathbb{E}[f(x_t)] - \min_{\mathcal{C}} f \leqslant \frac{2LD^2(K+1+\kappa)}{t+1}$$

where $\kappa := \lambda_0^+ / \lambda_0^-$

- In practice, no need to know G, L, D and simply set $b_t = \Theta(t^2)$

Frank-Wolfe with adaptive gradients

Theorem (C et al., 2020)

Let $\mathcal{C} \subset \mathbb{R}^n$ be a compact convex set with diameter D and $f_1, \ldots, f_m \colon \mathbb{R}^n \to \mathbb{R}$ be L-smooth convex functions. Then AdaSFW with $b_t \leftarrow \big(G(t+2)/(LD)\big)^2$, $\eta_t \leftarrow \lambda_t^-/L$, and $\gamma_t \leftarrow 2/(t+2)$ satisfies

$$\mathbb{E}[f(x_t)] - \min_{\mathcal{C}} f \leqslant \frac{2LD^2(K+1+\kappa)}{t+1}$$

where $\kappa := \lambda_0^+/\lambda_0^-$

- In practice, no need to know G, L, D and simply set $b_t = \Theta(t^2)$
- Also no need for λ_t^-, λ_t^+ and can set η_t to a constant value

Frank-Wolfe with adaptive gradients

Theorem (C et al., 2020)

Let $\mathcal{C} \subset \mathbb{R}^n$ be a compact convex set with diameter D and $f_1, \ldots, f_m \colon \mathbb{R}^n \to \mathbb{R}$ be L-smooth convex functions. Then AdaSFW with $b_t \leftarrow \big(G(t+2)/(LD)\big)^2$, $\eta_t \leftarrow \lambda_t^-/L$, and $\gamma_t \leftarrow 2/(t+2)$ satisfies

$$\mathbb{E}[f(x_t)] - \min_{\mathcal{C}} f \leqslant \frac{2LD^2(K+1+\kappa)}{t+1}$$

where $\kappa := \lambda_0^+/\lambda_0^-$

- In practice, no need to know G, L, D and simply set $b_t = \Theta(t^2)$
- Also no need for λ_t^-, λ_t^+ and can set η_t to a constant value
- AdaSVRF and AdaCSFW also yield $\mathcal{O}(1/t)$ convergence

Frank-Wolfe with adaptive gradients

Theorem (C et al., 2020)

Let $\mathcal{C} \subset \mathbb{R}^n$ be a compact convex set with diameter D and $f_1, \ldots, f_m : \mathbb{R}^n \to \mathbb{R}$ be L-smooth convex functions. Then AdaSFW with $b_t \leftarrow \big(G(t+2)/(LD)\big)^2$, $\eta_t \leftarrow \lambda_t^- / L$, and $\gamma_t \leftarrow 2/(t+2)$ satisfies

$$\mathbb{E}[f(x_t)] - \min_{\mathcal{C}} f \leqslant \frac{2LD^2(K+1+\kappa)}{t+1}$$

where $\kappa \coloneqq \lambda_0^+ / \lambda_0^-$

- In practice, no need to know G, L, D and simply set $b_t = \Theta(t^2)$
- Also no need for λ_t^-, λ_t^+ and can set η_t to a constant value
- AdaSVRF and AdaCSFW also yield $\mathcal{O}(1/t)$ convergence
- If $f_1, \ldots, f_m$ are nonconvex, then AdaSFW converges to a stationary point at a rate $\mathcal{O}(1/\sqrt{t})$

Computational experiments

- We compare our method to SFW, SVRF, SPIDER-FW, ORGFW, and CSFW on a wide range of experiments

Computational experiments

- We compare our method to SFW, SVRF, SPIDER-FW, ORGFW, and CSFW on a wide range of experiments
- For the experiments with convex objectives, we run AdaX where X is the best performing variant

Computational experiments

- We compare our method to SFW, SVRF, SPIDER-FW, ORGFW, and CSFW on a wide range of experiments
- For the experiments with convex objectives, we run AdaX where X is the best performing variant
- For the neural network experiments, CSFW is not applicable and we run AdaSFW only

Computational experiments

- We compare our method to SFW, SVRF, SPIDER-FW, ORGFW, and CSFW on a wide range of experiments
- For the experiments with convex objectives, we run AdaX where X is the best performing variant
- For the neural network experiments, CSFW is not applicable and we run AdaSFW only
- In addition, we run AdamSFW, a variant of AdaSFW with momentum inspired by Kingma & Ba (2015); Reddi et al. (2018)

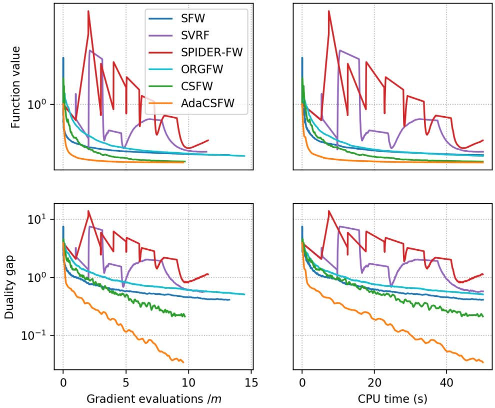
Computational experiments

- We compare our method to SFW, SVRF, SPIDER-FW, ORGFW, and CSFW on a wide range of experiments
- For the experiments with convex objectives, we run AdaX where X is the best performing variant
- For the neural network experiments, CSFW is not applicable and we run AdaSFW only
- In addition, we run AdamSFW, a variant of AdaSFW with momentum inspired by Kingma & Ba (2015); Reddi et al. (2018)
- We set $K \sim 5$

Support vector classification on a synthetic dataset

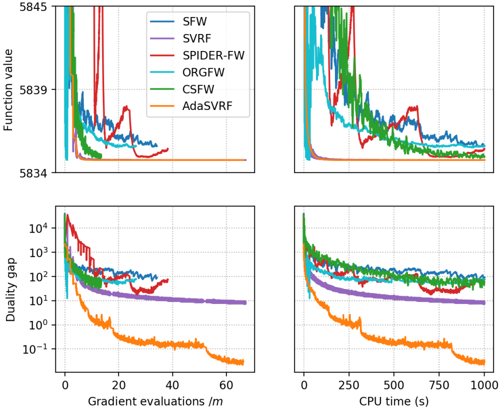
$$\min_{x \in \mathbb{R}^n} \frac{1}{m} \sum_{i=1}^{m} \max\{0, 1 - y_i \langle a_i, x \rangle\}^2$$

$$\text{s.t. } \|x\|_\infty \leqslant \tau$$



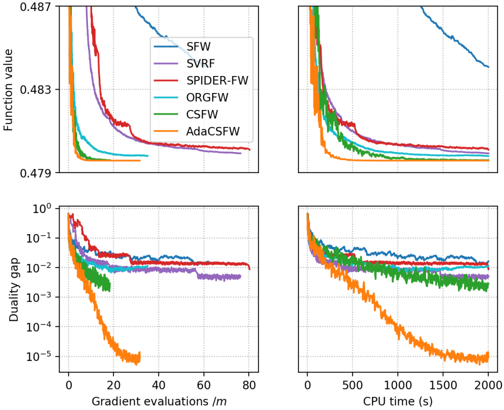
Linear regression on the YearPredictionMSD dataset

$$\min_{x\in\mathbb{R}^n} \frac{1}{m}\sum_{i=1}^{m}(y_i - \langle a_i, x\rangle)^2$$
$$\text{s.t. } \|x\|_1 \leqslant \tau$$



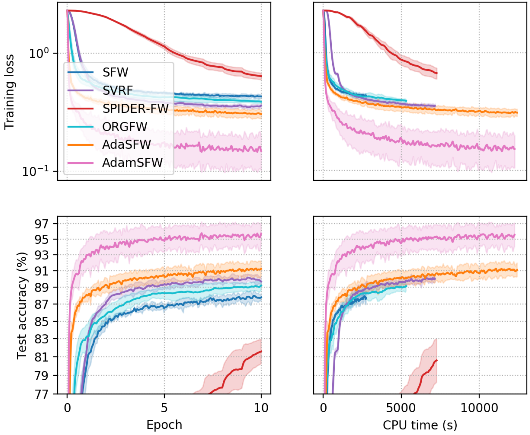
Logistic regression on the RCV1 dataset

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



Convolutional neural network on the MNIST dataset

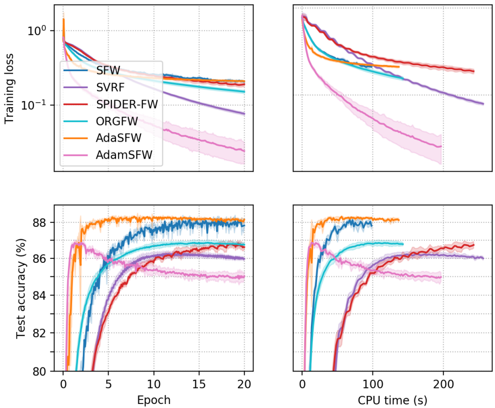
- Each layer of the neural network is constrained into an ℓ_1-ball



- AdamSFW strongly outperforms the other methods

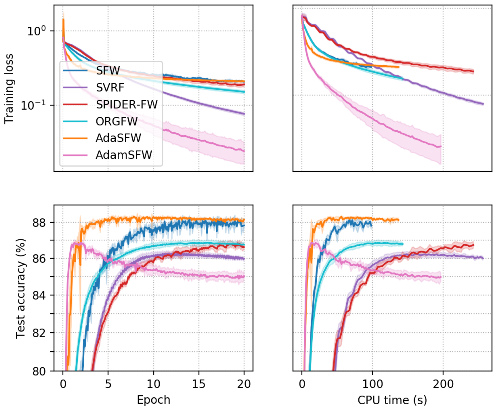
Neural network with one hidden layer on the IMDB dataset

- Each layer is constrained into an ℓ_∞-ball



Neural network with one hidden layer on the IMDB dataset

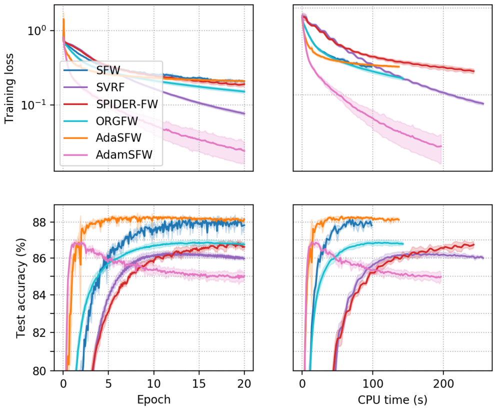
- Each layer is constrained into an ℓ_∞-ball



- AdaSFW and AdamSFW are the only ones to outperform SFW

Neural network with one hidden layer on the IMDB dataset

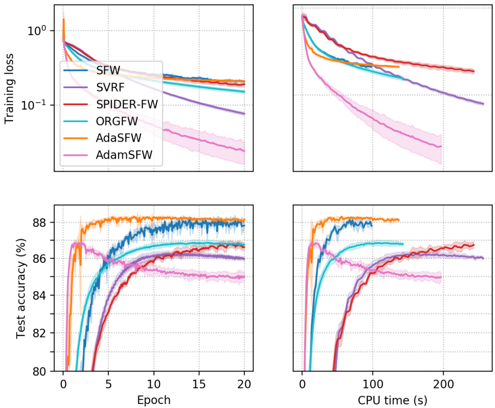
- Each layer is constrained into an ℓ_∞-ball



- AdaSFW and AdamSFW are the only ones to outperform SFW
- AdamSFW reaches its maximum test accuracy very fast (good for early stopping)

Neural network with one hidden layer on the IMDB dataset

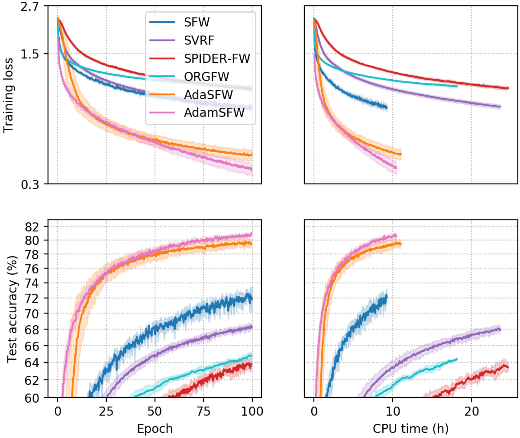
- Each layer is constrained into an ℓ_∞-ball



- AdaSFW and AdamSFW are the only ones to outperform SFW
- AdamSFW reaches its maximum test accuracy very fast (good for early stopping)
- AdaSFW yields the best test performance, despite optimizing slowly over the training set

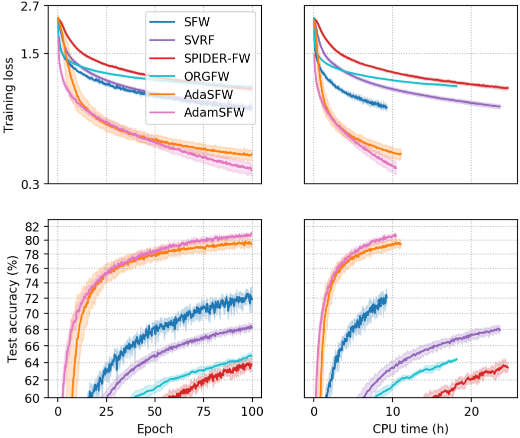
Convolutional network on the CIFAR-10 dataset

- Each layer is constrained into an ℓ_∞-ball



Convolutional network on the CIFAR-10 dataset

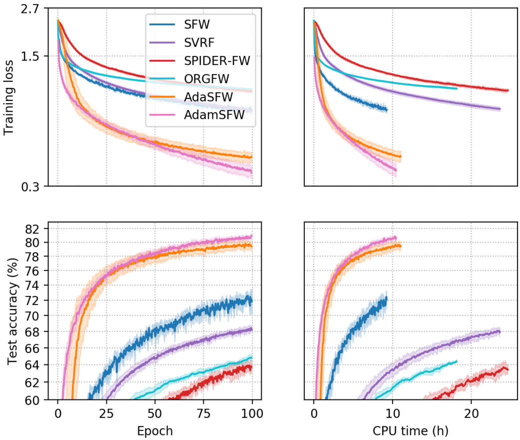
- Each layer is constrained into an ℓ_∞-ball



- AdaSFW and AdamSFW strongly outperform the other methods

Convolutional network on the CIFAR-10 dataset

- Each layer is constrained into an ℓ_∞-ball



- AdaSFW and AdamSFW strongly outperform the other methods
- AdaSFW and AdamSFW are the only ones to outperform SFW

Thank you!

References (1/3)

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.

C. W. Combettes, C. Spiegel, and S. Pokutta. Projection-free adaptive gradients for large-scale optimization. *arXiv*, 2020.

J. C. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 2011.

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.

E. Hazan and H. Luo. Variance-reduced and projection-free stochastic optimization. *ICML*, 2016.

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

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

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

References (2/3)

G. Lan. The complexity of large-scale convex programming under a linear optimization oracle. *arXiv*, 2013.

G. Lan and Y. Zhou. Conditional gradient sliding for convex optimization. *SIAM J. Optim.*, 2016.

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.

H. B. McMahan and M. Streeter. Adaptive bound optimization for online convex optimization. *COLT*, 2010.

G. Négiar, G. Dresdner, A. Y.-T. Tsai, L. El Ghaoui, F. Locatello, R. M. Freund, and F. Pedregosa. Stochastic Frank-Wolfe for constrained finite-sum minimization. *ICML*, 2020.

S. J. Reddi, S. Kale, and S. Kumar. On the convergence of Adam and beyond. *ICLR*, 2018.

Z. Shen, C. Fang, P. Zhao, J. Huang, and H. Qian. Complexities in projection-free stochastic non-convex minimization. *AISTATS*, 2019.

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

J. Xie, Z. Shen, C. Zhang, H. Qian, and B. Wang. Efficient projection-free online methods with stochastic recursive gradient. *AAAI*, 2020.

References (3/3)

A. Yurtsever, S. Sra, and V. Cevher. Conditional gradient methods via stochastic path-integrated differential estimator. *ICML*, 2019.

M. Zhang, Z. Shen, A. Mokhtari, H. Hassani, A. Karbasi. One Sample Stochastic Frank-Wolfe. *AISTATS*, 2020.