Structured ML Training via Conditional Gradients

Sebastian Pokutta

Technische Universität Berlin
and
Zuse Institute Berlin

pokutta@math.tu-berlin.de
@spokutta

Deep Learning and Combinatorial Optimization
IPAM (online) · February 23, 2021

Berlin Mathematics Research Center
MATH+

What is this talk about?

Introduction

A lot about sparsity, a bit about deep learning, and combinatorial optimization mostly in passing.

(And of course lots of conditional gradients; sorry!)

What is this talk about?

Introduction

A lot about sparsity, a bit about deep learning, and combinatorial optimization mostly in passing.

(And of course lots of conditional gradients; sorry!)

Idea. We can use combinatorial polytopes (k-sparse polytopes, ℓ_p-balls etc) to 'regularize' the learning problem and induce solution structure and sparsity.

What is this talk about?

Introduction

A lot about sparsity, a bit about deep learning, and combinatorial optimization mostly in passing.

(And of course lots of conditional gradients; sorry!)

Idea. We can use combinatorial polytopes (k-sparse polytopes, ℓ_p-balls etc) to 'regularize' the learning problem and induce solution structure and sparsity.

Today. A brief overview and two examples.

What is this talk about?

Introduction

A lot about sparsity, a bit about deep learning, and combinatorial optimization mostly in passing.

(And of course lots of conditional gradients; sorry!)

Idea. We can use combinatorial polytopes (k-sparse polytopes, ℓ_p-balls etc) to 'regularize' the learning problem and induce solution structure and sparsity.

Today. A brief overview and two examples.

Outline.

- Quick recall: Conditional Gradients a.k.a. the Frank-Wolfe algorithm
- Learning dynamics via Conditional Gradients
- Training Deep Neural Networks with Conditional Gradients

What is this talk about?

Introduction

A lot about sparsity, a bit about deep learning, and combinatorial optimization mostly in passing.

(And of course lots of conditional gradients; sorry!)

Idea. We can use combinatorial polytopes (k-sparse polytopes, ℓ_p-balls etc) to 'regularize' the learning problem and induce solution structure and sparsity.

Today. A brief overview and two examples.

Outline.

- Quick recall: Conditional Gradients a.k.a. the Frank-Wolfe algorithm
- Learning dynamics via Conditional Gradients
- Training Deep Neural Networks with Conditional Gradients

(Hyperlinked) References are not exhaustive; check references contained therein.

Conditional Gradients
a.k.a. the Frank-Wolfe algorithm

—Quick Recap—

The basic problem

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

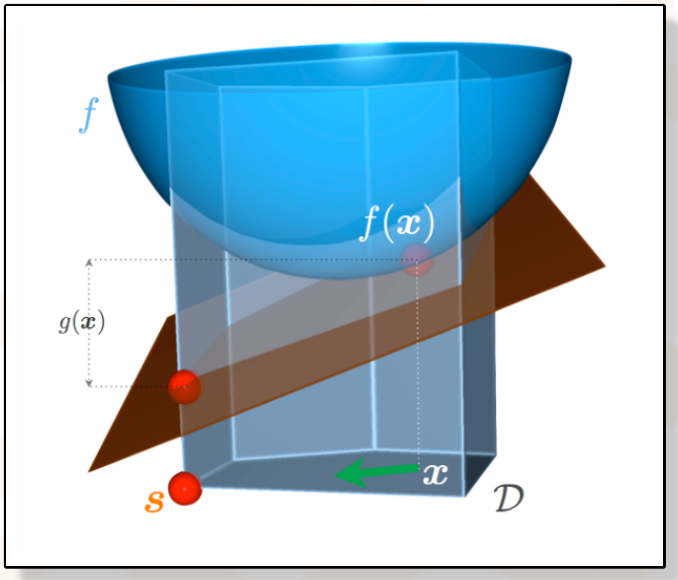
Given a smooth and convex function f and a polytope P, solve optimization problem:

The basic problem

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Given a smooth and convex function f and a polytope P, solve optimization problem:

$$\min_{x \in P} f(x) \qquad \text{(baseProblem)}$$



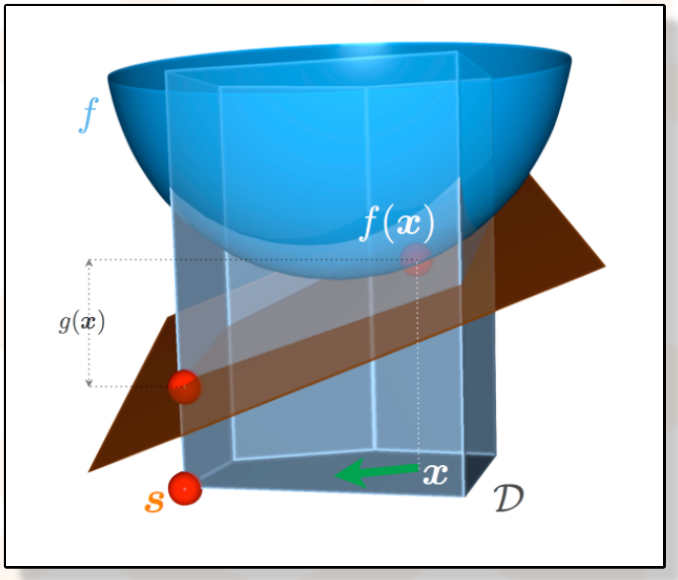
[Source: Jaggi 2013]

The basic problem

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Given a smooth and convex function f and a polytope P, solve optimization problem:

$$\min_{x \in P} f(x) \qquad \text{(baseProblem)}$$



[Source: Jaggi 2013]

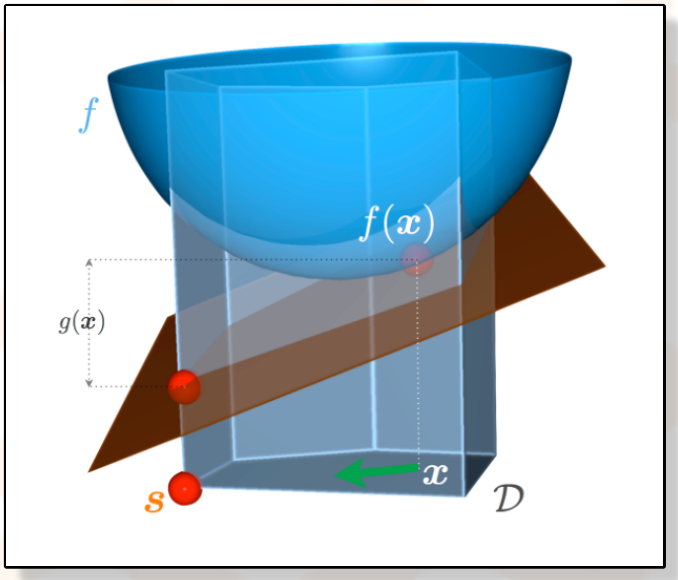
1. Very versatile model
2. Can use various types of information about both f and P
3. Works very well in (continuous) real-world applications
4. At the core of many (all?) learning algorithms (albeit mostly non-convex case)

The basic problem

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Given a smooth and convex function f and a polytope P, solve optimization problem:

$$\min_{x \in P} f(x) \qquad \text{(baseProblem)}$$



[Source: Jaggi 2013]

Our setup.

1. Access to P. Linear Optimization (LO) Oracle: Given linear objective c return

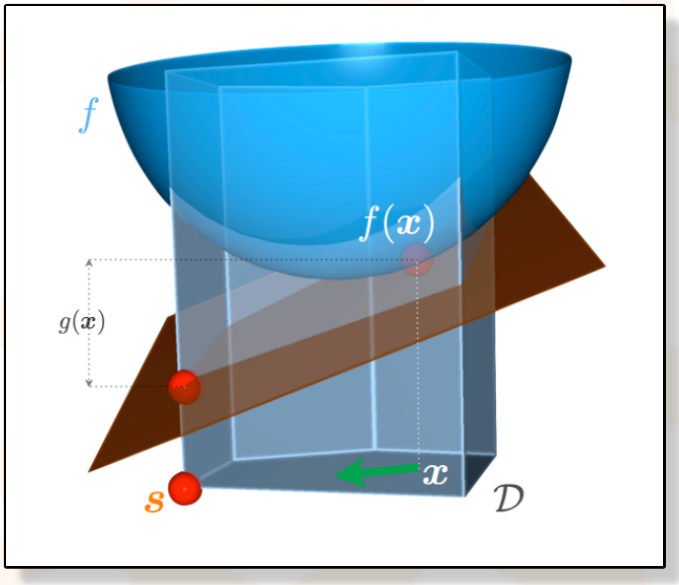
$$x \leftarrow \arg\min_{v \in P} c^T v.$$

The basic problem

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Given a smooth and convex function f and a polytope P, solve optimization problem:

$$\min_{x \in P} f(x) \qquad \text{(baseProblem)}$$



[Source: Jaggi 2013]

Our setup.

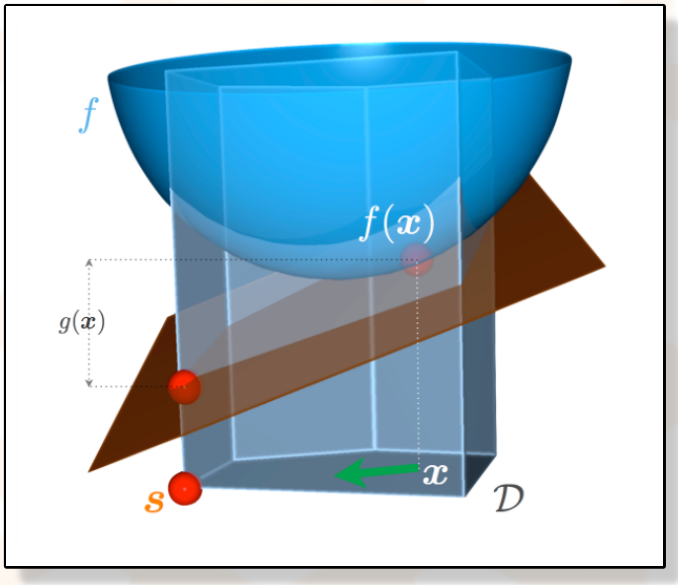
1. Access to P. Linear Optimization (LO) Oracle: Given linear objective c return
$$x \leftarrow \arg\min_{v \in P} c^T v.$$
2. Access to f. First-Order (FO) Oracle: Given x return
$$\nabla f(x) \qquad \text{and} \qquad f(x).$$

The basic problem

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Given a smooth and convex function f and a polytope P, solve optimization problem:

$$\min_{x \in P} f(x) \qquad \text{(baseProblem)}$$



[Source: Jaggi 2013]

Our setup.

1. Access to P. Linear Optimization (LO) Oracle: Given linear objective c return
$$x \leftarrow \arg\min_{v \in P} c^T v.$$
2. Access to f. First-Order (FO) Oracle: Given x return
$$\nabla f(x) \qquad \text{and} \qquad f(x).$$

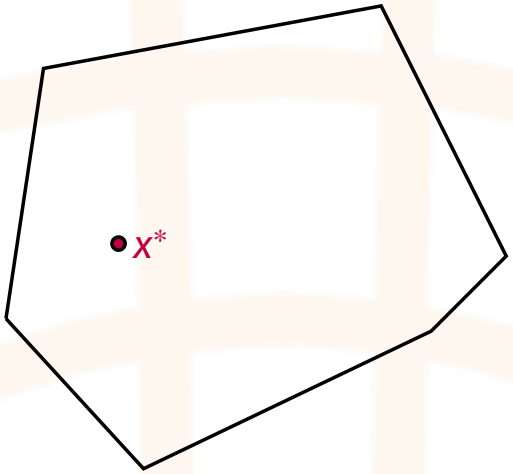
$\Rightarrow$ *Complexity of convex optimization relative to LO/FO oracle*

The Frank-Wolfe Algorithm

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Algorithm Frank-Wolfe Algorithm (FW)

1: $x_0 \in P$
2: **for** $t = 0$ **to** $T - 1$ **do**
3: $\quad v_t \leftarrow \arg\min_{v \in P} \langle \nabla f(x_t), v \rangle$
4: $\quad x_{t+1} \leftarrow x_t + \gamma_t (v_t - x_t)$
5: **end for**



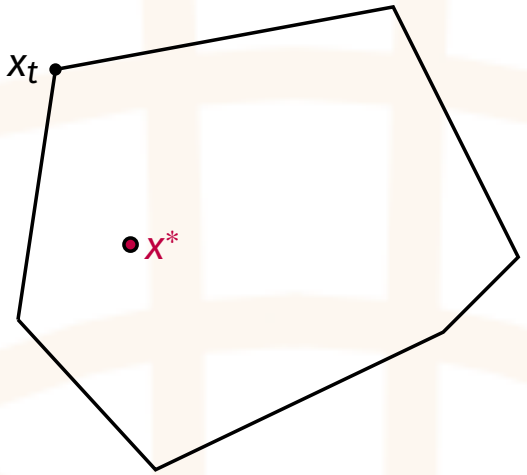
[Frank, Wolfe, 1956] [Levitin, Polyak, 1966]

The Frank-Wolfe Algorithm

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Algorithm Frank-Wolfe Algorithm (FW)

1: $x_0 \in P$
2: **for** $t = 0$ **to** $T - 1$ **do**
3: $\quad v_t \leftarrow \arg\min_{v \in P} \langle \nabla f(x_t), v \rangle$
4: $\quad x_{t+1} \leftarrow x_t + \gamma_t (v_t - x_t)$
5: **end for**



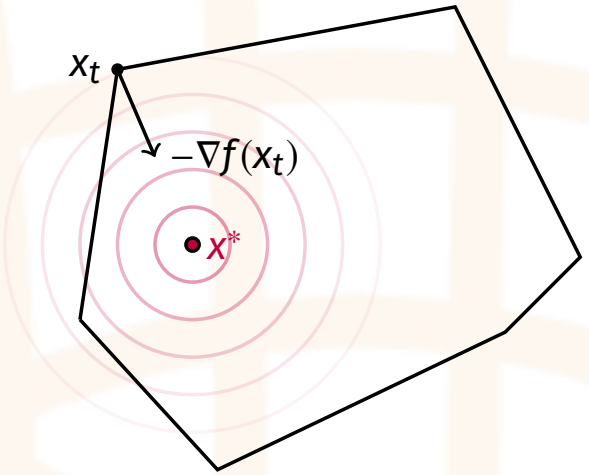
[Frank, Wolfe, 1956] [Levitin, Polyak, 1966]

The Frank-Wolfe Algorithm

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Algorithm Frank-Wolfe Algorithm (FW)

1: $x_0 \in P$
2: **for** $t = 0$ **to** $T - 1$ **do**
3: $\quad v_t \leftarrow \arg\min_{v \in P} \langle \nabla f(x_t), v \rangle$
4: $\quad x_{t+1} \leftarrow x_t + \gamma_t (v_t - x_t)$
5: **end for**



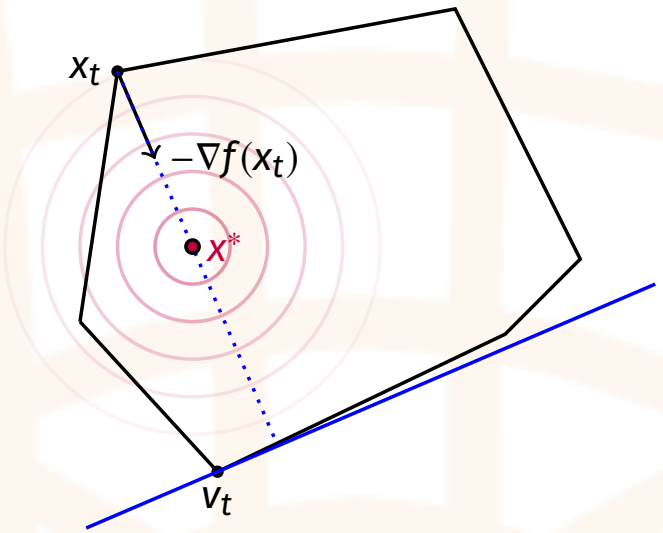
[Frank, Wolfe, 1956] [Levitin, Polyak, 1966]

The Frank-Wolfe Algorithm

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Algorithm Frank-Wolfe Algorithm (FW)

1: $x_0 \in P$
2: **for** $t = 0$ **to** $T - 1$ **do**
3: $\quad v_t \leftarrow \arg\min_{v \in P} \langle \nabla f(x_t), v \rangle$
4: $\quad x_{t+1} \leftarrow x_t + \gamma_t(v_t - x_t)$
5: **end for**



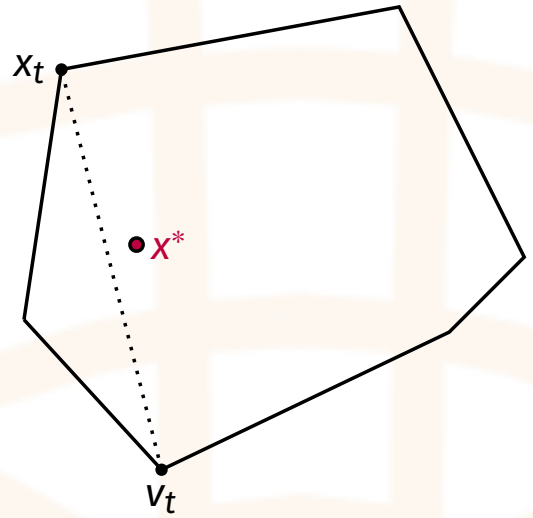
[Frank, Wolfe, 1956] [Levitin, Polyak, 1966]

The Frank-Wolfe Algorithm

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Algorithm Frank-Wolfe Algorithm (FW)

1: $x_0 \in P$
2: **for** $t = 0$ **to** $T - 1$ **do**
3: $\quad v_t \leftarrow \arg\min_{v \in P} \langle \nabla f(x_t), v \rangle$
4: $\quad x_{t+1} \leftarrow x_t + \gamma_t (v_t - x_t)$
5: **end for**



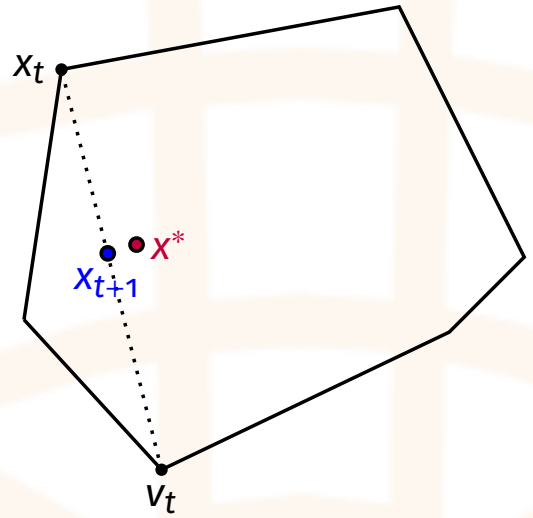
[Frank, Wolfe, 1956] [Levitin, Polyak, 1966]

The Frank-Wolfe Algorithm

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Algorithm Frank-Wolfe Algorithm (FW)

1: $x_0 \in P$
2: **for** $t = 0$ **to** $T - 1$ **do**
3: $\quad v_t \leftarrow \arg\min_{v \in P} \langle \nabla f(x_t), v \rangle$
4: $\quad x_{t+1} \leftarrow x_t + \gamma_t(v_t - x_t)$
5: **end for**



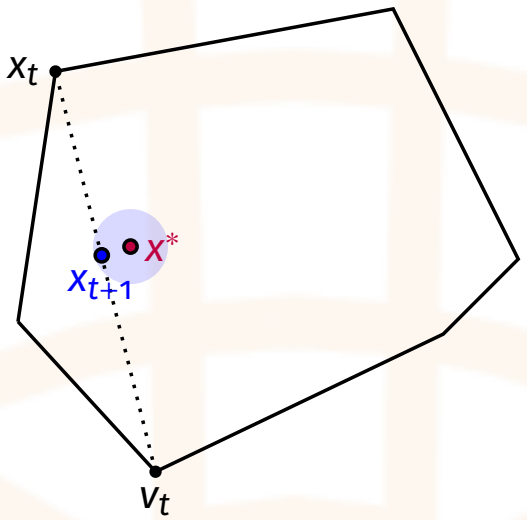
[Frank, Wolfe, 1956] [Levitin, Polyak, 1966]

The Frank-Wolfe Algorithm

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Algorithm Frank-Wolfe Algorithm (FW)

1: $x_0 \in P$
2: **for** $t = 0$ **to** $T - 1$ **do**
3: $\quad v_t \leftarrow \arg\min_{v \in P} \langle \nabla f(x_t), v \rangle$
4: $\quad x_{t+1} \leftarrow x_t + \gamma_t(v_t - x_t)$
5: **end for**



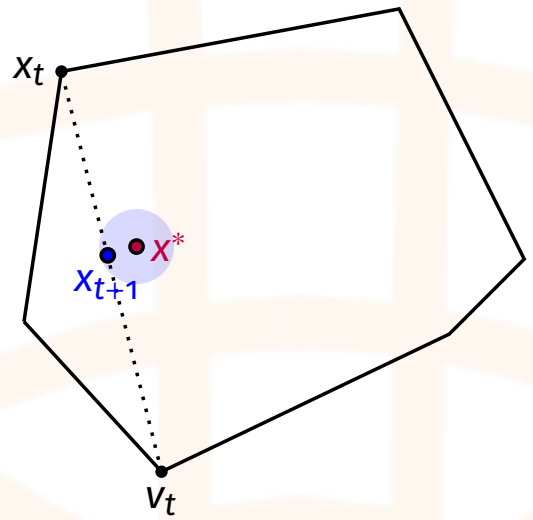
[Frank, Wolfe, 1956] [Levitin, Polyak, 1966]

The Frank-Wolfe Algorithm

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Algorithm Frank-Wolfe Algorithm (FW)

1: $x_0 \in P$
2: **for** $t = 0$ **to** $T - 1$ **do**
3: $\quad v_t \leftarrow \arg\min_{v \in P} \langle \nabla f(x_t), v \rangle$
4: $\quad x_{t+1} \leftarrow x_t + \gamma_t(v_t - x_t)$
5: **end for**



[Frank, Wolfe, 1956] [Levitin, Polyak, 1966]

Advantages:

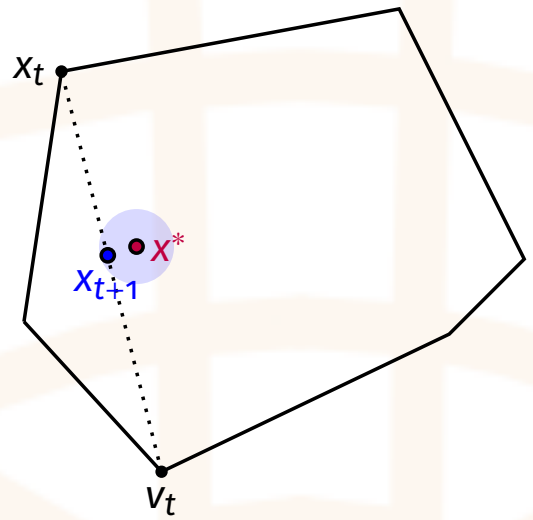
- Extremely simple and robust: no complicated data structures to maintain
- Easy to implement: requires only the two oracles
- Projection-free: feasibility convex combination and LO oracle.
- Sparsity: optimal solution is a convex combination of (usually) vertices.
- Structured update of iterates: $v_t \in P$ induces structured updates.

The Frank-Wolfe Algorithm

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Algorithm Frank-Wolfe Algorithm (FW)

1: $x_0 \in P$
2: **for** $t = 0$ **to** $T - 1$ **do**
3: $\quad v_t \leftarrow \arg\min_{v \in P} \langle \nabla f(x_t), v \rangle$
4: $\quad x_{t+1} \leftarrow x_t + \gamma_t(v_t - x_t)$
5: **end for**



[Frank, Wolfe, 1956] [Levitin, Polyak, 1966]

Advantages:

- Extremely simple and robust: no complicated data structures to maintain
- Easy to implement: requires only the two oracles
- Projection-free: feasibility convex combination and LO oracle.
- Sparsity: optimal solution is a convex combination of (usually) vertices.
- Structured update of iterates: $v_t \in P$ induces structured updates.

Disadvantages:

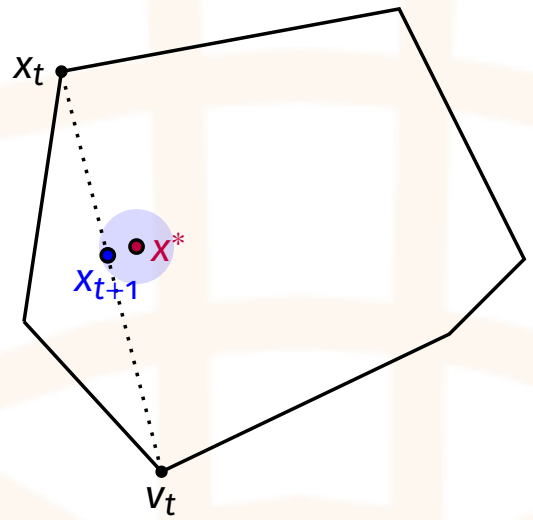
- Suboptimal convergence rate of $O(1/T)$

The Frank-Wolfe Algorithm

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Algorithm Frank-Wolfe Algorithm (FW)

1: $x_0 \in P$
2: **for** $t = 0$ **to** $T - 1$ **do**
3: $\quad v_t \leftarrow \arg\min_{v \in P} \langle \nabla f(x_t), v \rangle$
4: $\quad x_{t+1} \leftarrow x_t + \gamma_t(v_t - x_t)$
5: **end for**



[Frank, Wolfe, 1956] [Levitin, Polyak, 1966]

Advantages:

- Extremely simple and robust: no complicated data structures to maintain
- Easy to implement: requires only the two oracles
- Projection-free: feasibility convex combination and LO oracle.
- Sparsity: optimal solution is a convex combination of (usually) vertices.
- Structured update of iterates: $v_t \in P$ induces structured updates.

Disadvantages:

- Suboptimal convergence rate of $O(1/T)$

$\Rightarrow$ *Despite (theoretically) suboptimal rate heavily used in applications due to simplicity.*

Simple Convergence Proof

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Theorem (Convergence rate of the vanilla Frank-Wolfe Algorithm)

Let f be L-smooth convex, P be polytope with diameter D. With choice $\gamma_t \doteq \frac{2}{t+3}$:

$$f(x_t) - f(x^*) \leq \frac{2LD^2}{t+3}.$$

Simple Convergence Proof

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Theorem (Convergence rate of the vanilla Frank-Wolfe Algorithm)

Let f be L-smooth convex, P be polytope with diameter D. With choice $\gamma_t \doteq \frac{2}{t+3}$:

$$f(x_t) - f(x^*) \leq \frac{2LD^2}{t+3}.$$

Proof Sketch.

By smoothness:

$$f(x_{t+1}) - f(x_t) \leq \langle \nabla f(x_t), x_{t+1} - x_t \rangle + \frac{L}{2} \|x_{t+1} - x_t\|^2 = \gamma_t \langle \nabla f(x_t), v_t - x_t \rangle + \frac{L\gamma_t^2}{2} \|v_t - x_t\|^2.$$

Simple Convergence Proof

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Theorem (Convergence rate of the vanilla Frank-Wolfe Algorithm)

Let f be L-smooth convex, P be polytope with diameter D. With choice $\gamma_t \doteq \frac{2}{t+3}$:

$$f(x_t) - f(x^*) \leq \frac{2LD^2}{t+3}.$$

Proof Sketch.

By smoothness:

$$f(x_{t+1}) - f(x_t) \leq \langle \nabla f(x_t), x_{t+1} - x_t \rangle + \frac{L}{2} \|x_{t+1} - x_t\|^2 = \gamma_t \langle \nabla f(x_t), v_t - x_t \rangle + \frac{L\gamma_t^2}{2} \|v_t - x_t\|^2.$$

LP maximality and convexity: $\langle \nabla f(x_t), v_t - x_t \rangle \leq \langle \nabla f(x_t), x^* - x_t \rangle \leq f(x^*) - f(x_t)$. Moreover, $\|v_t - x_t\| \leq D$.

Simple Convergence Proof

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Theorem (Convergence rate of the vanilla Frank-Wolfe Algorithm)

Let f be L-smooth convex, P be polytope with diameter D. With choice $\gamma_t \doteq \frac{2}{t+3}$:

$$f(x_t) - f(x^*) \leq \frac{2LD^2}{t+3}.$$

Proof Sketch.

By smoothness:

$$f(x_{t+1}) - f(x_t) \leq \langle \nabla f(x_t), x_{t+1} - x_t \rangle + \frac{L}{2} \|x_{t+1} - x_t\|^2 = \gamma_t \langle \nabla f(x_t), v_t - x_t \rangle + \frac{L\gamma_t^2}{2} \|v_t - x_t\|^2.$$

LP maximality and convexity: $\langle \nabla f(x_t), v_t - x_t \rangle \leq \langle \nabla f(x_t), x^* - x_t \rangle \leq f(x^*) - f(x_t)$. Moreover, $\|v_t - x_t\| \leq D$.

Thus:

$$f(x_{t+1}) - f(x^*) \leq (1 - \gamma_t)(f(x_t) - f(x^*)) + \gamma_t^2 \frac{LD^2}{2}.$$

Simple Convergence Proof

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Theorem (Convergence rate of the vanilla Frank-Wolfe Algorithm)

Let f be L-smooth convex, P be polytope with diameter D. With choice $\gamma_t \doteq \frac{2}{t+3}$:

$$f(x_t) - f(x^*) \leq \frac{2LD^2}{t+3}.$$

Proof Sketch.

By smoothness:

$$f(x_{t+1}) - f(x_t) \leq \langle \nabla f(x_t), x_{t+1} - x_t \rangle + \frac{L}{2} \|x_{t+1} - x_t\|^2 = \gamma_t \langle \nabla f(x_t), v_t - x_t \rangle + \frac{L\gamma_t^2}{2} \|v_t - x_t\|^2.$$

LP maximality and convexity: $\langle \nabla f(x_t), v_t - x_t \rangle \leq \langle \nabla f(x_t), x^* - x_t \rangle \leq f(x^*) - f(x_t)$. Moreover, $\|v_t - x_t\| \leq D$.

Thus:

$$f(x_{t+1}) - f(x^*) \leq (1 - \gamma_t)(f(x_t) - f(x^*)) + \gamma_t^2 \frac{LD^2}{2}.$$

By Induction (plugging in the guarantee + definition of γ_t):

$$f(x_{t+1}) - f(x^*) \leq \left(1 - \frac{2}{t+3}\right) \frac{2LD^2}{t+3} + \frac{4}{(t+3)^2} \cdot \frac{LD^2}{2} = \frac{2LD^2(t+2)}{(t+3)^2} \leq \frac{2LD^2}{t+4},$$

by $(t+2)(t+4) \leq (t+3)^2$.

□

Significant progress over the recent years (incomplete list)

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

1. Strongly convex case [Garber, Hazan 2013] [Lan, Zhou 2014] [Lacoste-Julien, Jaggi 2015] [Garber, Meschi 2016]
2. Non-convex case [Lacoste-Julien 2016]
3. Online case [Hazan, Kale 2012]
4. Stochastic variants and adaptive gradients [Hazan, Luo 2016] [Reddi et al 2016] [Combettes, Spiegel, P. 2020]
5. Sharp functions and sharp regions [Kerdreux, d'Aspremont, P. 2018] [Kerdreux, d'Aspremont, P. 2020]
6. Acceleration [Diakonikolas, Carderera, P. 2019] [Bach 2020] [Carderera, Diakonikolas, Lin, P. 2021]
7. Specialized variants [Freund, Grigas, Mazumder 2015] [Braun, P., Zink 2016] [Braun, P., Tu, Wright 2018]

Conditional Gradients very competitive: simple, robust, real-world performance.

For more background etc see upcoming survey!

Conditional Gradient-based Identification of Nonlinear Dynamics (CINDy)

—Recovering Dynamics from Noisy Data—

joint work with Alejandro Carderera, Christof Schütte, Martin Weiser

Physical Systems via ODEs

CINDy: Recovering Dynamics from Noisy Data

Physical systems described by ordinary differential equation.

$$\dot{x}(t) = F\left(x(t)\right),$$

where $x(t) \in \mathbb{R}^d$ denotes the state of the system at time t.

Physical Systems via ODEs

CINDy: Recovering Dynamics from Noisy Data

Physical systems described by ordinary differential equation.

$$\dot{x}(t) = F(x(t)),$$

where $x(t) \in \mathbb{R}^d$ denotes the state of the system at time t.

Usually. $F : \mathbb{R}^d \to \mathbb{R}^d$ (usually) linear combination of simpler *ansatz functions* $\mathcal{D} = \{\psi_i \mid i \in [\![1, n]\!]\}$ with $\psi_i : \mathbb{R}^d \to \mathbb{R}$:

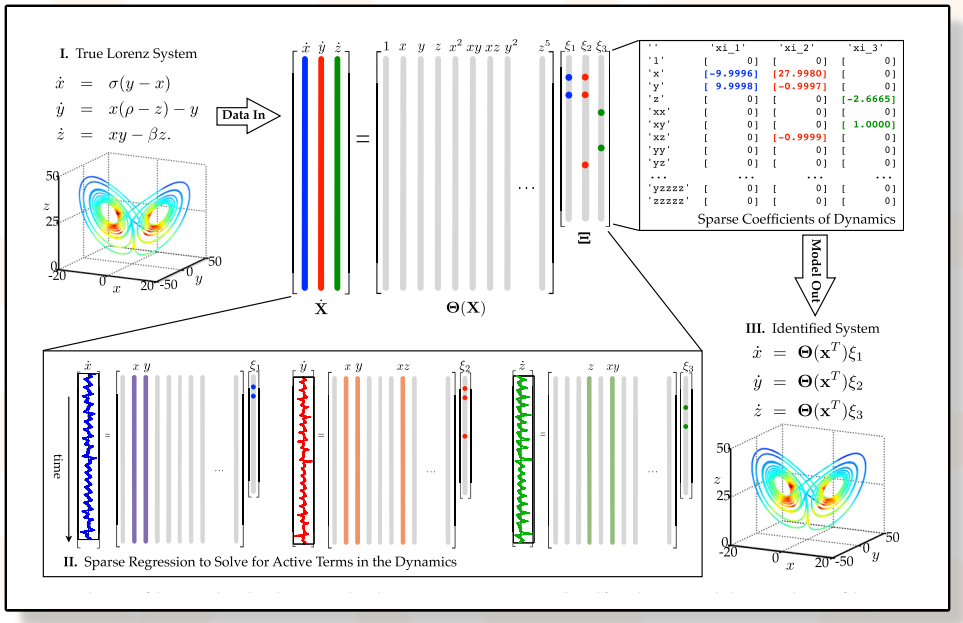
$$\dot{x}(t) = F(x(t)) = \Xi^T \psi(x(t)) = \begin{bmatrix} — & \xi_1 & — \\ & \vdots & \\ — & \xi_d & — \end{bmatrix} \begin{bmatrix} \psi_1(x(t)) \\ \vdots \\ \psi_n(x(t)) \end{bmatrix},$$

where $\Xi \in \mathbb{R}^{n \times d}$ is a typically sparse matrix and $\psi(x(t)) = [\psi_1(x(t)), \cdots, \psi_n(x(t))]^T \in \mathbb{R}^n$.

Sparse Identification of Nonlinear Dynamics (SINDy)

CINDy: Recovering Dynamics from Noisy Data

[Brunton et al 2016]



Focus on component-wise formulation of sparse recovery problem and solve a relaxation of:

$$\min_{\xi_j \in \mathbb{R}^d} \sum_{i=1}^{m} \|\dot{x}_i - \xi_j^T \psi(x_i)\|_2^2 + \alpha \|\xi_j\|_0,$$

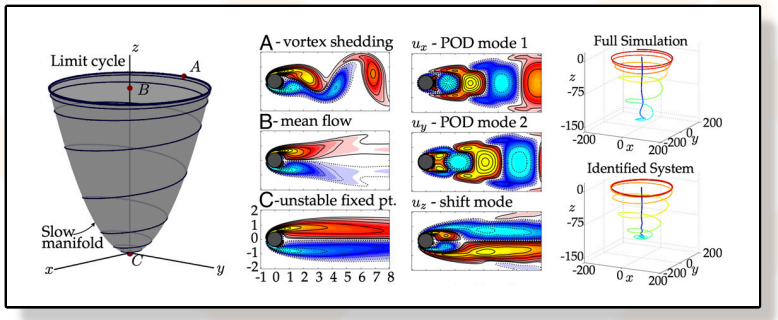
for each $j \in [\![1, d]\!]$ for a suitably chosen $\alpha \geq 0$.

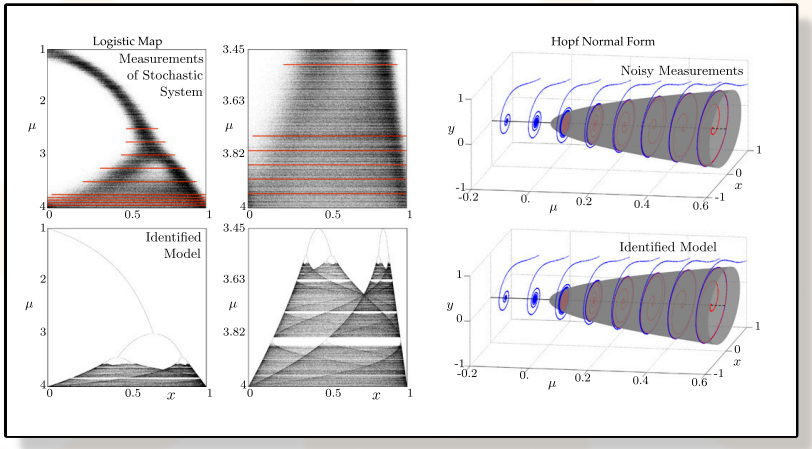
Note. Earlier approach via Gröbner/Border Bases for homogeneous case. [Heldt et al 2009]

Sparse Identification of Nonlinear Dynamics (SINDy)

CINDy: Recovering Dynamics from Noisy Data

[Brunton et al 2016]





Characteristics of SINDy.

1. Works on a very wide variety of dynamics
2. Recovers sparse dynamics very well in the noise-free case
3. However when data is noisy, picks up many auxiliary terms to *explain* noise.

The Fully-Corrective Frank-Wolfe Algorithm

CINDy: Recovering Dynamics from Noisy Data

Algorithm Fully-Corrective FW Algorithm (FCFW)

1: $x_0 \in P$, $\mathcal{S}_0 \leftarrow \{x_0\}$
2: **for** $t = 0$ **to** $T - 1$ **do**
3: $\quad v_t \leftarrow \arg\min_{v \in P} \langle \nabla f(x_t), v \rangle$
4: $\quad \mathcal{S}_{t+1} \leftarrow \mathcal{S}_t \cup \{v_t\}$
5: $\quad x_{t+1} \leftarrow \arg\min_{x \in \mathrm{conv}(\mathcal{S}_{t+1})} f(x)$
6: **end for**

[Holloway, 1974]

The Fully-Corrective Frank-Wolfe Algorithm

Algorithm Fully-Corrective FW Algorithm (FCFW)

1: $x_0 \in P$, $\mathcal{S}_0 \leftarrow \{x_0\}$
2: **for** $t = 0$ **to** $T - 1$ **do**
3: $\quad v_t \leftarrow \arg\min_{v \in P} \langle \nabla f(x_t), v \rangle$
4: $\quad \mathcal{S}_{t+1} \leftarrow \mathcal{S}_t \cup \{v_t\}$
5: $\quad x_{t+1} \leftarrow \arg\min_{x \in \mathrm{conv}(\mathcal{S}_{t+1})} f(x)$
6: **end for**

[Holloway, 1974]

The Fully-Corrective Frank-Wolfe Algorithm

CINDy: Recovering Dynamics from Noisy Data

Algorithm Fully-Corrective FW Algorithm (FCFW)

1: $x_0 \in P$, $\mathcal{S}_0 \leftarrow \{x_0\}$
2: **for** $t = 0$ **to** $T - 1$ **do**
3: $\quad v_t \leftarrow \arg\min_{v \in P} \langle \nabla f(x_t), v \rangle$
4: $\quad \mathcal{S}_{t+1} \leftarrow \mathcal{S}_t \cup \{v_t\}$
5: $\quad x_{t+1} \leftarrow \arg\min_{x \in \mathrm{conv}(\mathcal{S}_{t+1})} f(x)$
6: **end for**

[Holloway, 1974]

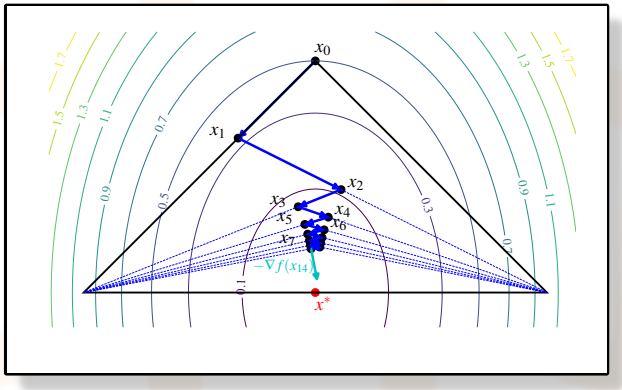
The Fully-Corrective Frank-Wolfe Algorithm

CINDy: Recovering Dynamics from Noisy Data

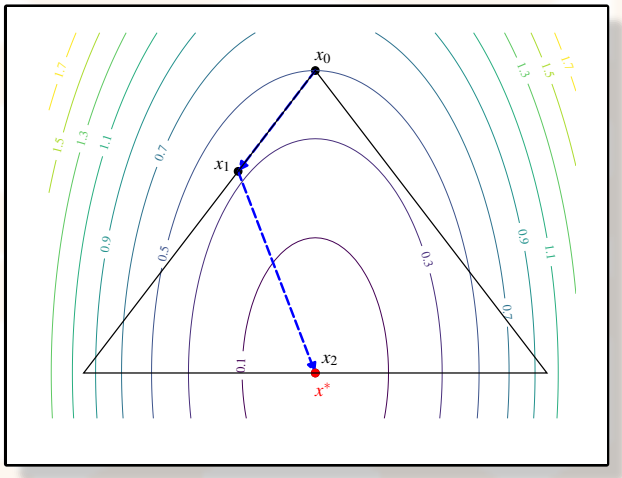
Algorithm Fully-Corrective FW Algorithm (FCFW)

1: $x_0 \in P$, $\mathcal{S}_0 \leftarrow \{x_0\}$
2: **for** $t = 0$ **to** $T - 1$ **do**
3: $\quad v_t \leftarrow \arg\min_{v \in P} \langle \nabla f(x_t), v \rangle$
4: $\quad \mathcal{S}_{t+1} \leftarrow \mathcal{S}_t \cup \{v_t\}$
5: $\quad x_{t+1} \leftarrow \arg\min_{x \in \mathrm{conv}(\mathcal{S}_{t+1})} f(x)$
6: **end for**

[Holloway, 1974]



Vanilla FW Algorithm



Fully-Corrective FW Algorithm

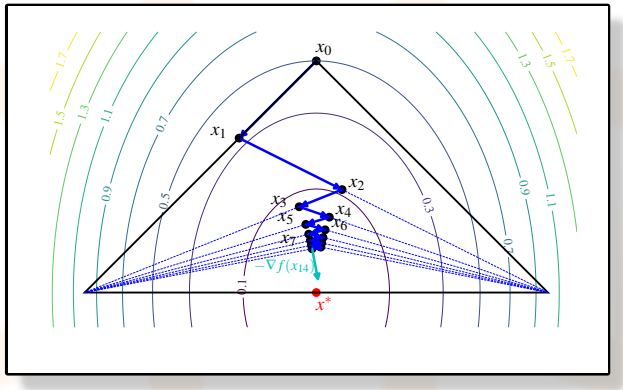
The Fully-Corrective Frank-Wolfe Algorithm

CINDy: Recovering Dynamics from Noisy Data

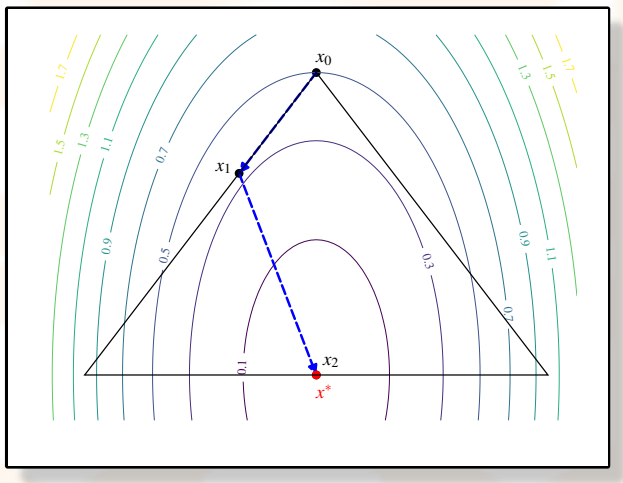
Algorithm Fully-Corrective FW Algorithm (FCFW)

1: $x_0 \in P$, $\mathcal{S}_0 \leftarrow \{x_0\}$
2: **for** $t = 0$ **to** $T - 1$ **do**
3: $\quad v_t \leftarrow \arg\min_{v \in P} \langle \nabla f(x_t), v \rangle$
4: $\quad \mathcal{S}_{t+1} \leftarrow \mathcal{S}_t \cup \{v_t\}$
5: $\quad x_{t+1} \leftarrow \arg\min_{x \in \mathrm{conv}(\mathcal{S}_{t+1})} f(x)$
6: **end for**

[Holloway, 1974]



Vanilla FW Algorithm



Fully-Corrective FW Algorithm

- Sparsity: FCFW offers much higher sparsity
- Speed: Convergence speed is (much) higher but iterations *very* costly
- Projection-free: While still projection-free requires solver for subproblems

$\Rightarrow$ *While expensive can be useful if sheer speed is not a priority but sparsity is.*

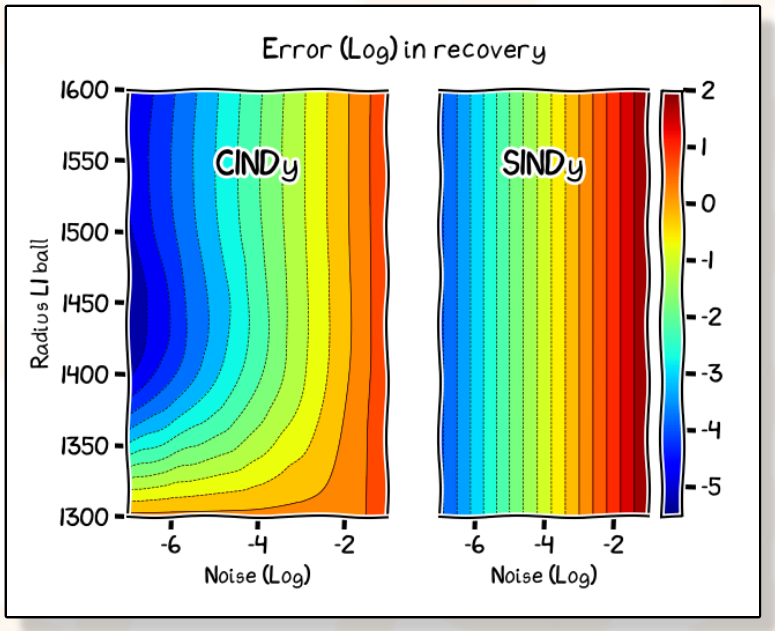
Note. Sparsity *not only* a function of formulation but also *algorithm and its trajectory*.

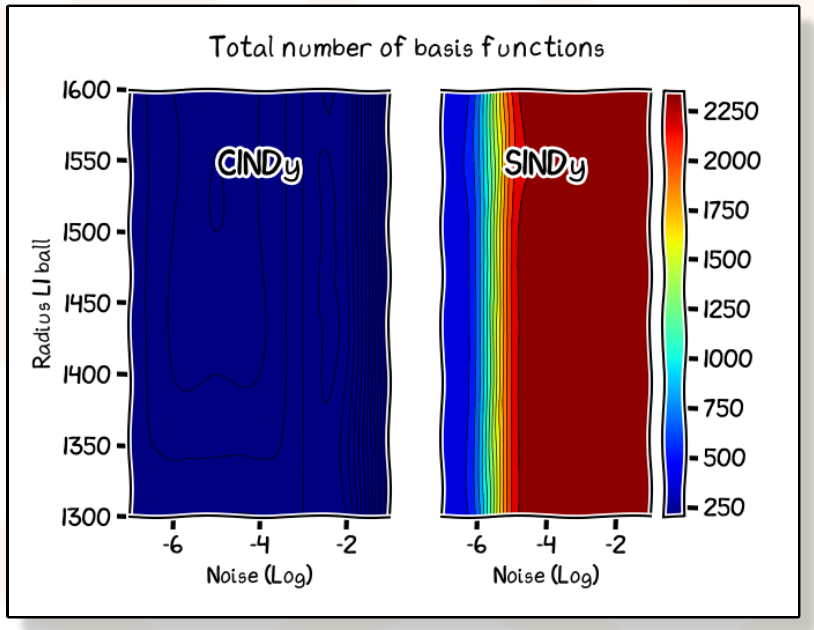
CINDy vs SINDy: a recovery example

CINDy: Recovering Dynamics from Noisy Data

Kuramoto model. $d = 10$ weakly-coupled identical oscillators. For oscillator i:

$$\dot{x}_i = \omega_i + \frac{K}{d} \sum_{j=1}^{d} \sin\left(x_j - x_i\right) + h \sin\left(x_i\right)$$



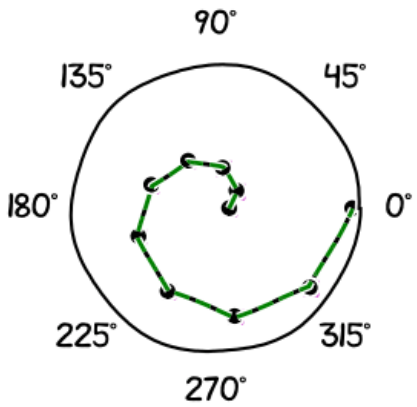


Number of data points. 3000 generated from 100 experiments (30 per experiment with additive random noise of 1.0^{-3}.

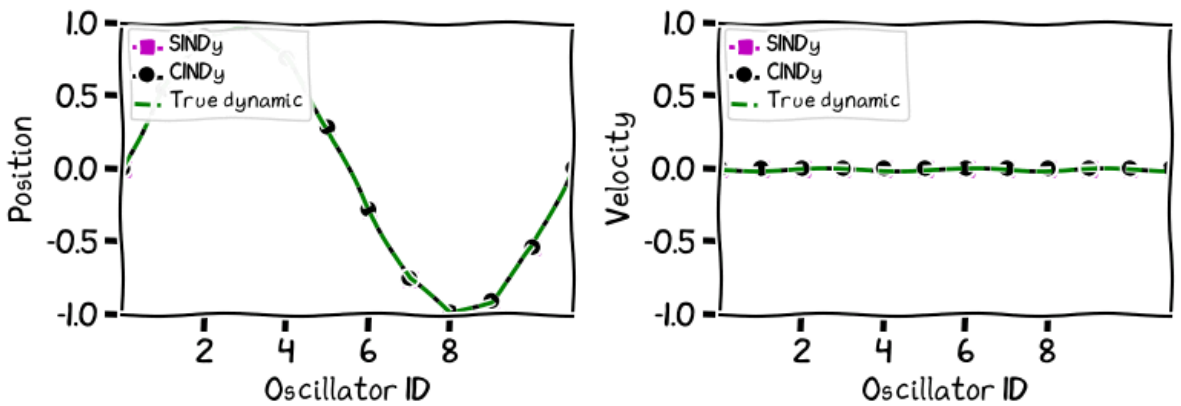
CINDy vs SINDy: sparsity matters - the most parsimonious model

CINDy: Recovering Dynamics from Noisy Data

Kuramoto model.



Fermi-Pasta-Ulam-Tsingou model.



Stochastic Conditional Gradients

—Training Neural Networks with Frank-Wolfe—

joint work with Christoph Spiegel and Max Zimmer

The Stochastic Frank-Wolfe Algorithm (with Momentum)

Training Neural Networks with Conditional Gradients

Algorithm Stochastic FW Algorithm (SFW)

$m_0 \leftarrow 0$
for $t = 0$ **to** $T - 1$ **do**
  uniformly sample i.i.d. $i_1, \ldots, i_{b_t} \sim [\![1, m]\!]$
  $\tilde{\nabla} L(\theta_t) \leftarrow \frac{1}{b_t} \sum_{j=1}^{b_t} \nabla \ell_{i_j}(\theta_t)$
  $m_t \leftarrow (1 - \rho_t)\, m_{t-1} + \rho_t\, \tilde{\nabla} L(\theta_t)$
  $v_t \leftarrow \arg\min_{v \in P} \langle m_t, v \rangle$
  $\theta_{t+1} \leftarrow \theta_t + \alpha_t (v_t - \theta_t)$
end for

[e.g., Reddi et al 2016]

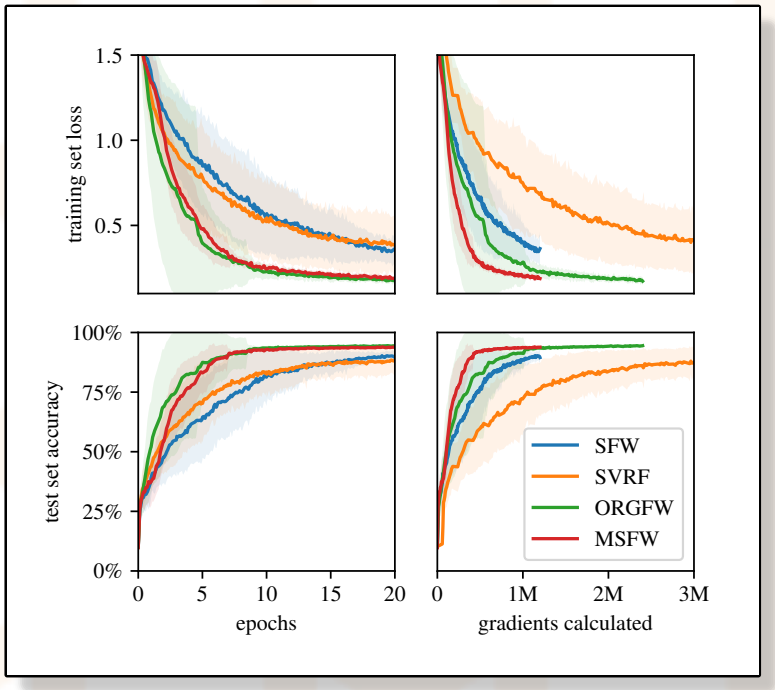
The Stochastic Frank-Wolfe Algorithm (with Momentum)

Training Neural Networks with Conditional Gradients

Algorithm Stochastic FW Algorithm (SFW)

$m_0 \leftarrow 0$
for $t = 0$ **to** $T - 1$ **do**
  uniformly sample i.i.d. $i_1, \ldots, i_{b_t} \sim [\![1, m]\!]$
  $\tilde{\nabla} L(\theta_t) \leftarrow \frac{1}{b_t} \sum_{j=1}^{b_t} \nabla \ell_{i_j}(\theta_t)$
  $m_t \leftarrow (1 - \rho_t)\, m_{t-1} + \rho_t\, \tilde{\nabla} L(\theta_t)$
  $v_t \leftarrow \arg\min_{v \in P} \langle m_t, v \rangle$
  $\theta_{t+1} \leftarrow \theta_t + \alpha_t (v_t - \theta_t)$
end for

[e.g., Reddi et al 2016]



SFW variants

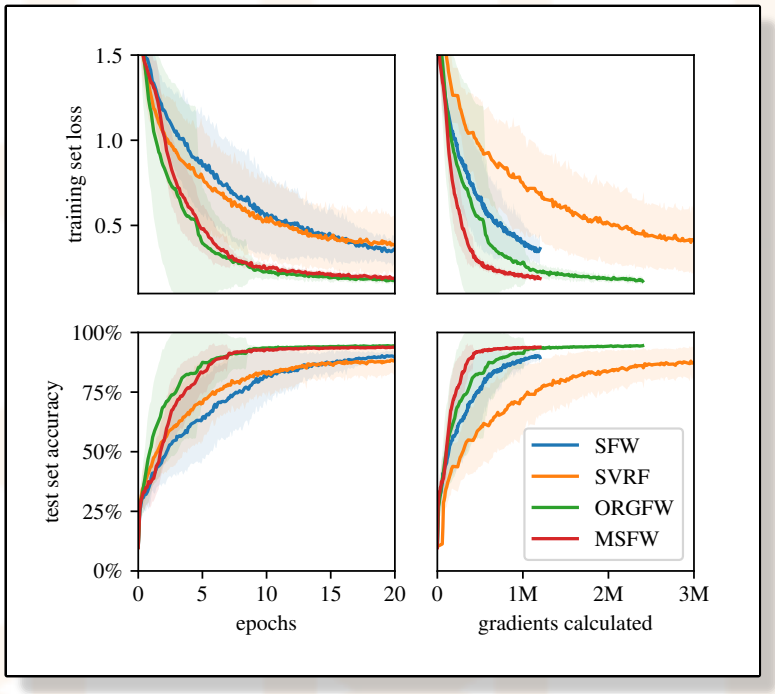
The Stochastic Frank-Wolfe Algorithm (with Momentum)

Training Neural Networks with Conditional Gradients

Algorithm Stochastic FW Algorithm (SFW)

$m_0 \leftarrow 0$
for $t = 0$ **to** $T - 1$ **do**
  uniformly sample i.i.d. $i_1, \ldots, i_{b_t} \sim [\![1, m]\!]$
  $\tilde{\nabla} L(\theta_t) \leftarrow \frac{1}{b_t} \sum_{j=1}^{b_t} \nabla \ell_{i_j}(\theta_t)$
  $m_t \leftarrow (1 - \rho_t)\, m_{t-1} + \rho_t\, \tilde{\nabla} L(\theta_t)$
  $v_t \leftarrow \arg\min_{v \in P} \langle m_t, v \rangle$
  $\theta_{t+1} \leftarrow \theta_t + \alpha_t (v_t - \theta_t)$
end for

[e.g., Reddi et al 2016]

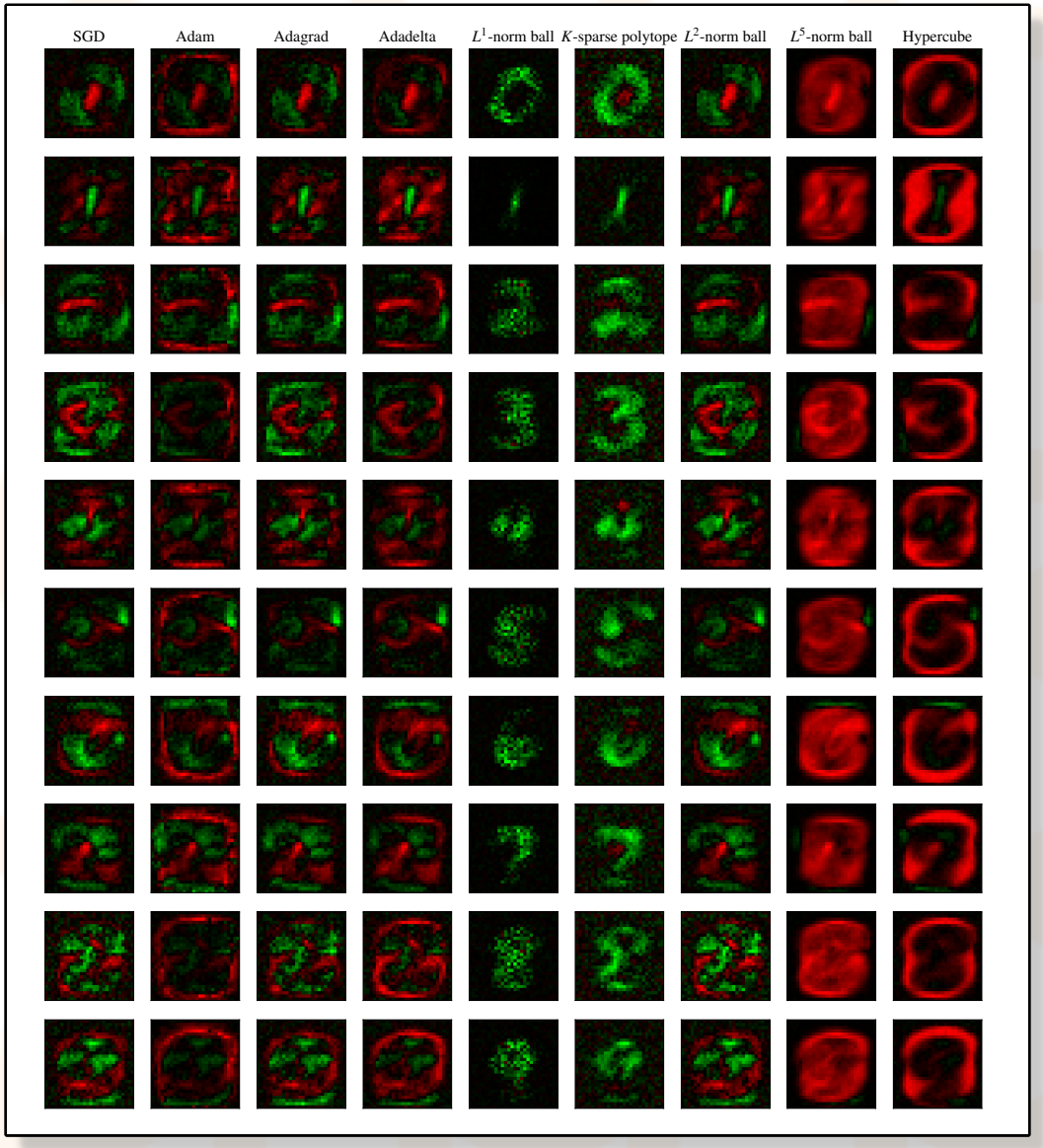


SFW variants

- Convergence rate: In the non-convex stochastic smooth case $O(1/\sqrt{T})$-rate
- Speed: Works well for very large data sets due to mini-batched gradients
- Projection-free: Remains projection-free and allows for constraints

Relevance maps under different optimizers / feasible regions

Training Neural Networks with Conditional Gradients



Iterative Magnitude Pruning (IPM) for Conditional Gradients

Training Neural Networks with Conditional Gradients

Basic Idea. Train network to relativly high accuracy. Prune small weights. Repeat.

[see e.g., Robert Lange's Blog Post for a great overview]

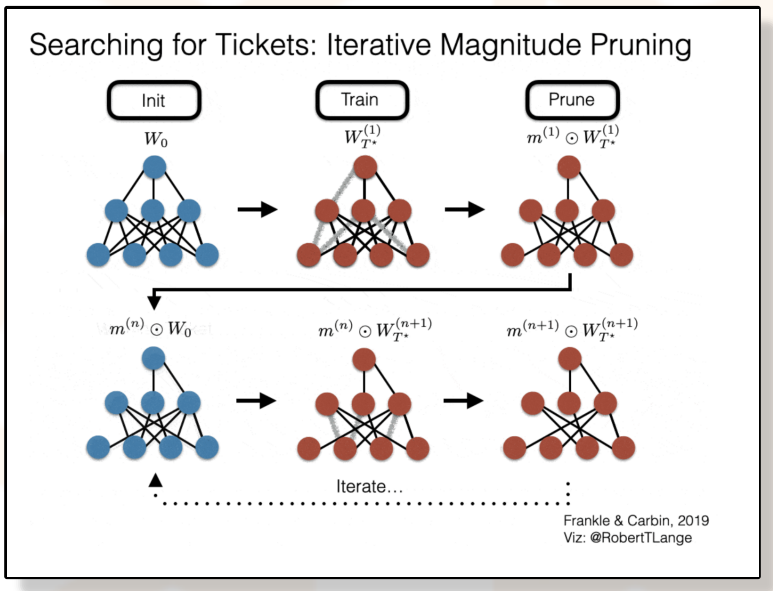
Iterative Magnitude Pruning (IPM) for Conditional Gradients

Training Neural Networks with Conditional Gradients

Basic Idea. Train network to relativly high accuracy. Prune small weights. Repeat.

Note. There is tons of variants out there (beyond scope!) with

1. Learning Rate Rewinding
2. Weight Rewinding
3. Fine-Tuning

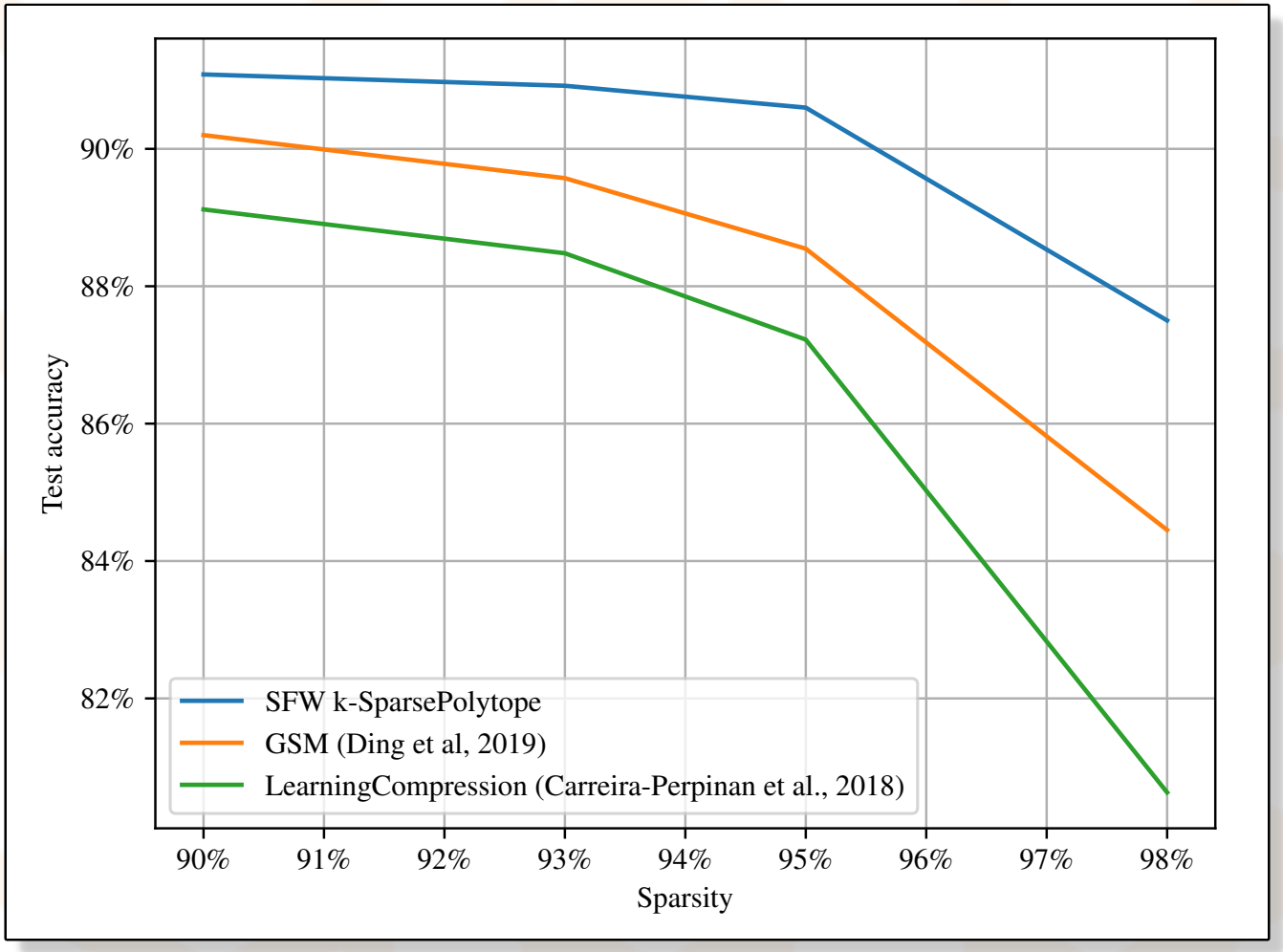


[see e.g., Robert Lange's Blog Post for a great overview]

Iterative Magnitude Pruning (IPM) for Conditional Gradients

Training Neural Networks with Conditional Gradients

Experiment. Training + Pruning + Fine-Tuning (CIFAR-10)



Thank you!