Frank-Wolfe Algorithms for Saddle Point problems



Gauthier Gidel[1] Tony Jebara[2] Simon Lacoste-Julien[3]

[1]INRIA Paris, Sierra Team [2]Department of CS, Columbia University

[3]Department of CS & OR (DIRO) Université de Montréal

10th December 2016

Overview

- Frank-Wolfe algorithm (FW) gained in popularity in the last couple of years.
- Main advantage: FW only needs LMO.
- Extend FW properties to solve saddle point problem.
- **Straightforward** extension but **Non trivial** analysis.

Saddle point and link with variational inequalities

Let $\mathcal{L} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, where $\mathcal{X}$ and $\mathcal{Y}$ are convex and compact.

$$\boxed{\text{Saddle point problem:} \quad \text{solve } \min_{\boldsymbol{x} \in \mathcal{X}} \max_{\boldsymbol{y} \in \mathcal{Y}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{y})}$$

A solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is called a ***Saddle Point***.

Saddle point and link with variational inequalities

Let $\mathcal{L} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, where $\mathcal{X}$ and $\mathcal{Y}$ are convex and compact.

$$\boxed{\text{Saddle point problem:} \quad \text{solve } \min_{\boldsymbol{x} \in \mathcal{X}} \max_{\boldsymbol{y} \in \mathcal{Y}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{y})}$$

A solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is called a ***Saddle Point***.

- **Necessary *stationary conditions:***

$$\langle \boldsymbol{x} - \boldsymbol{x}^*, \quad \nabla_x \mathcal{L}(\boldsymbol{x}^*, \boldsymbol{y}^*) \rangle \geq 0$$

Saddle point and link with variational inequalities

Let $\mathcal{L} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, where $\mathcal{X}$ and $\mathcal{Y}$ are convex and compact.

$$\boxed{\text{Saddle point problem:} \qquad \text{solve } \min_{\boldsymbol{x} \in \mathcal{X}} \max_{\boldsymbol{y} \in \mathcal{Y}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{y})}$$

A solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is called a ***Saddle Point***.

- **Necessary *stationary conditions:***

$$\begin{aligned} \langle \boldsymbol{x} - \boldsymbol{x}^*, \quad \nabla_x \mathcal{L}(\boldsymbol{x}^*, \boldsymbol{y}^*) \rangle &\geq 0 \\ \langle \boldsymbol{y} - \boldsymbol{y}^*, -\nabla_y \mathcal{L}(\boldsymbol{x}^*, \boldsymbol{y}^*) \rangle &\geq 0 \end{aligned}$$

Saddle point and link with variational inequalities

Let $\mathcal{L} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, where $\mathcal{X}$ and $\mathcal{Y}$ are convex and compact.

$$\boxed{\text{Saddle point problem:} \quad \text{solve } \min_{\boldsymbol{x} \in \mathcal{X}} \max_{\boldsymbol{y} \in \mathcal{Y}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{y})}$$

A solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is called a ***Saddle Point***.

- **Necessary *stationary conditions:***

$$\langle \boldsymbol{x} - \boldsymbol{x}^*, \quad \nabla_x \mathcal{L}(\boldsymbol{x}^*, \boldsymbol{y}^*) \rangle \geq 0$$
$$\langle \boldsymbol{y} - \boldsymbol{y}^*, -\nabla_y \mathcal{L}(\boldsymbol{x}^*, \boldsymbol{y}^*) \rangle \geq 0$$

- ***Variational inequality:***

$$\forall \boldsymbol{z} \in \mathcal{X} \times \mathcal{Y} \quad \langle \boldsymbol{z} - \boldsymbol{z}^*, g(\boldsymbol{z}^*) \rangle \geq 0$$

where $(\boldsymbol{x}^*, \boldsymbol{y}^*) = \boldsymbol{z}^*$ and $g(\boldsymbol{z}) = (\nabla_x \mathcal{L}(\boldsymbol{z}), -\nabla_y \mathcal{L}(\boldsymbol{z}))$

Saddle point and link with variational inequalities

Let $\mathcal{L} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, where $\mathcal{X}$ and $\mathcal{Y}$ are convex and compact.

$$\boxed{\text{Saddle point problem:} \quad \text{solve } \min_{\boldsymbol{x} \in \mathcal{X}} \max_{\boldsymbol{y} \in \mathcal{Y}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{y})}$$

A solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is called a ***Saddle Point***.

- **Necessary *stationary conditions:***

$$\langle \boldsymbol{x} - \boldsymbol{x}^*, \quad \nabla_x \mathcal{L}(\boldsymbol{x}^*, \boldsymbol{y}^*) \rangle \geq 0$$
$$\langle \boldsymbol{y} - \boldsymbol{y}^*, -\nabla_y \mathcal{L}(\boldsymbol{x}^*, \boldsymbol{y}^*) \rangle \geq 0$$

- ***Variational inequality:***

$$\forall \boldsymbol{z} \in \mathcal{X} \times \mathcal{Y} \quad \langle \boldsymbol{z} - \boldsymbol{z}^*, g(\boldsymbol{z}^*) \rangle \geq 0$$

  where $(\boldsymbol{x}^*, \boldsymbol{y}^*) = \boldsymbol{z}^*$ and $g(\boldsymbol{z}) = (\nabla_x \mathcal{L}(\boldsymbol{z}), -\nabla_y \mathcal{L}(\boldsymbol{z}))$

- **Sufficient *condition***: ***Global solution*** if $\mathcal{L}$ *convex-concave.* $\forall (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X} \times \mathcal{Y}$

$$\boldsymbol{x}' \mapsto \mathcal{L}(\boldsymbol{x}', \boldsymbol{y}) \text{ is convex} \quad \text{and} \quad \boldsymbol{y}' \mapsto \mathcal{L}(\boldsymbol{x}, \boldsymbol{y}') \text{ is concave.}$$

Motivations: games and robust learning

- ***Zero-sum games with two players:***

$$\min_{x \in \Delta(I)} \max_{y \in \Delta(J)} \boldsymbol{x}^\top M \boldsymbol{y}$$

[1] J. Wen, C. Yu, and R. Greiner. "Robust Learning under Uncertain Test Distributions: Relating Covariate Shift to Model Misspecification." In: *ICML*. 2014.

Motivations: games and robust learning

- ***Zero-sum games with two players:***

$$\min_{x \in \Delta(I)} \max_{y \in \Delta(J)} \boldsymbol{x}^\top M \boldsymbol{y}$$

- ***Generative Adversarial Network*** (GAN)

[1] J. Wen, C. Yu, and R. Greiner. "Robust Learning under Uncertain Test Distributions: Relating Covariate Shift to Model Misspecification." In: *ICML*. 2014.

Motivations: games and robust learning

- ***Zero-sum games with two players:***

$$\min_{x \in \Delta(I)} \max_{y \in \Delta(J)} \boldsymbol{x}^\top M \boldsymbol{y}$$

- ***Generative Adversarial Network*** (GAN)
- ***Robust learning:***[1] We want to learn

$$\min_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} \ell\left(f_\theta(x_i), y_i\right) + \lambda \Omega(\theta)$$

with an uncertainty regarding the data:

$$\min_{\theta \in \Theta} \max_{w \in \Delta_n} \sum_{i=1}^{n} \omega_i \ell\left(f_\theta(x_i), y_i\right) + \lambda \Omega(\theta)$$

$\boxed{\text{Minimize the } \textbf{worst case} \rightarrow \text{gives } \textbf{robustness}}$

[1] J. Wen, C. Yu, and R. Greiner. "Robust Learning under Uncertain Test Distributions: Relating Covariate Shift to Model Misspecification." In: *ICML*. 2014.

Problem with Hard projection

The ***structured SVM:***

$$\min_{\omega \in \mathbb{R}^d} \lambda \Omega(\omega) + \frac{1}{n} \sum_{i=1}^{n} \underbrace{\max_{y \in \mathcal{Y}_i} \left(L_i(y) - \langle \omega, \phi_i(y) \rangle \right)}_{\text{structured hinge loss}}$$

Problem with Hard projection

The ***structured SVM:***

$$\min_{\omega \in \mathbb{R}^d} \lambda \Omega(\omega) + \frac{1}{n} \sum_{i=1}^{n} \underbrace{\max_{y \in \mathcal{Y}_i} \left(L_i(y) - \langle \omega, \phi_i(y) \rangle \right)}_{\text{structured hinge loss}}$$

Regularization: penalized $\to$ constrained.

$$\min_{\Omega(\omega) \leq \beta} \max_{\alpha \in \Delta(|\mathcal{Y}|)} b^T \alpha - \omega^T M \alpha$$

Problem with Hard projection

The ***structured SVM:***

$$\min_{\omega \in \mathbb{R}^d} \lambda\Omega(\omega) + \frac{1}{n}\sum_{i=1}^{n} \underbrace{\max_{y \in \mathcal{Y}_i} \left(L_i(y) - \langle \omega, \phi_i(y) \rangle\right)}_{\text{structured hinge loss}}$$

Regularization: penalized $\rightarrow$ constrained.

$$\min_{\Omega(\omega) \leq \beta} \max_{\alpha \in \Delta(|\mathcal{Y}|)} b^T \alpha - \omega^T M \alpha$$

Problem with Hard projection

The ***structured SVM:***

$$\min_{\omega \in \mathbb{R}^d} \lambda\Omega(\omega) + \frac{1}{n}\sum_{i=1}^{n} \underbrace{\max_{y \in \mathcal{Y}_i} \left(L_i(y) - \langle \omega, \phi_i(y) \rangle\right)}_{\text{structured hinge loss}}$$

Regularization: penalized $\rightarrow$ constrained.

$$\min_{\Omega(\omega) \leq \beta} \max_{\alpha \in \Delta(|\mathcal{Y}|)} b^T\alpha - \omega^T M\alpha$$

Problem with Hard projection

The ***structured SVM:***

$$\min_{\omega \in \mathbb{R}^d} \lambda \Omega(\omega) + \frac{1}{n} \sum_{i=1}^{n} \underbrace{\max_{y \in \mathcal{Y}_i} \left(L_i(y) - \langle \omega, \phi_i(y) \rangle \right)}_{\text{structured hinge loss}}$$

Regularization: penalized $\rightarrow$ constrained.

$$\min_{\Omega(\omega) \leq \beta} \max_{\alpha \in \Delta(|\mathcal{Y}|)} b^T \alpha - \omega^T M \alpha$$

Hard to project when:

Problem with Hard projection

The ***structured SVM:***

$$\min_{\omega \in \mathbb{R}^d} \lambda \Omega(\omega) + \frac{1}{n} \sum_{i=1}^{n} \underbrace{\max_{y \in \mathcal{Y}_i} \left(L_i(y) - \langle \omega, \phi_i(y) \rangle \right)}_{\text{structured hinge loss}}$$

Regularization: penalized $\rightarrow$ constrained.

$$\min_{\Omega(\omega) \leq \beta} \max_{\alpha \in \Delta(|\mathcal{Y}|)} b^T \alpha - \omega^T M \alpha$$

Hard to project when:

- ***Structured sparsity*** norm (group lasso norm).

Problem with Hard projection

The ***structured SVM:***

$$\min_{\omega \in \mathbb{R}^d} \lambda \Omega(\omega) + \frac{1}{n} \sum_{i=1}^{n} \underbrace{\max_{y \in \mathcal{Y}_i} \left(L_i(y) - \langle \omega, \phi_i(y) \rangle \right)}_{\text{structured hinge loss}}$$

Regularization: penalized $\rightarrow$ constrained.

$$\min_{\Omega(\omega) \leq \beta} \max_{\alpha \in \Delta(|\mathcal{Y}|)} b^T \alpha - \omega^T M \alpha$$

Hard to project when:

- ***Structured sparsity*** norm (group lasso norm).
- The output $\mathcal{Y}$ is structured: ***exponential*** size.

Standard approaches in literature

Simplest algorithm to solve Saddle point problems is the *projected gradient algorithm.*

$$
\begin{aligned}
\boldsymbol{x}^{(t+1)} &= P_{\mathcal{X}}(\boldsymbol{x}^{(t)} - \eta \nabla_x \mathcal{L}(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)})) \\
\boldsymbol{y}^{(t+1)} &= P_{\mathcal{Y}}(\boldsymbol{y}^{(t)} + \eta \nabla_y \mathcal{L}(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)}))
\end{aligned}
$$

For non-smooth optimization,

$$
\frac{1}{T} \sum_{t=1}^{T} \left(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)} \right) \underset{T \to \infty}{\longrightarrow} (\boldsymbol{x}^*, \boldsymbol{y}^*)
$$

[2] N. He and Z. Harchaoui. "Semi-proximal Mirror-Prox for Nonsmooth Composite Minimization". In: *NIPS*. 2015.

Standard approaches in literature

Simplest algorithm to solve Saddle point problems is the *projected gradient algorithm*.

$$
\begin{aligned}
\boldsymbol{x}^{(t+1)} &= P_{\mathcal{X}}(\boldsymbol{x}^{(t)} - \eta \nabla_x \mathcal{L}(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)})) \\
\boldsymbol{y}^{(t+1)} &= P_{\mathcal{Y}}(\boldsymbol{y}^{(t)} + \eta \nabla_y \mathcal{L}(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)}))
\end{aligned}
$$

For non-smooth optimization,

$$
\frac{1}{T} \sum_{t=1}^{T} \left(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)} \right) \underset{T \to \infty}{\longrightarrow} (\boldsymbol{x}^*, \boldsymbol{y}^*)
$$

Faster algorithm: *projected extra-gradient algorithm*.

[2] N. He and Z. Harchaoui. "Semi-proximal Mirror-Prox for Nonsmooth Composite Minimization". In: *NIPS*. 2015.

Standard approaches in literature

Simplest algorithm to solve Saddle point problems is the *projected gradient algorithm*.

$$\boldsymbol{x}^{(t+1)} = P_{\mathcal{X}}(\boldsymbol{x}^{(t)} - \eta \nabla_x \mathcal{L}(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)}))$$
$$\boldsymbol{y}^{(t+1)} = P_{\mathcal{Y}}(\boldsymbol{y}^{(t)} + \eta \nabla_y \mathcal{L}(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)}))$$

For non-smooth optimization,

$$\frac{1}{T}\sum_{t=1}^{T}\left(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)}\right) \underset{T\to\infty}{\longrightarrow} (\boldsymbol{x}^*, \boldsymbol{y}^*)$$

Faster algorithm: *projected extra-gradient algorithm*.

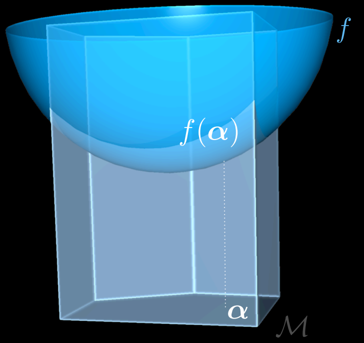
Can use LMO to compute approximate projections[2].

[2] N. He and Z. Harchaoui. "Semi-proximal Mirror-Prox for Nonsmooth Composite Minimization". In: *NIPS*. 2015.

The FW algorithm

Algorithm Frank-Wolfe algorithm

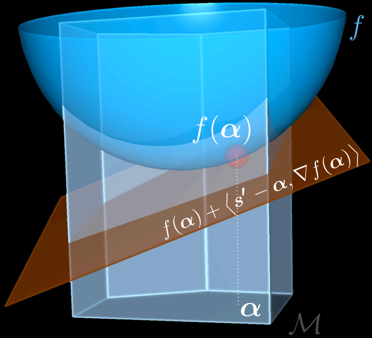
Let $\boldsymbol{x}^{(0)} \in \mathcal{X}$
for $t = 0 \dots T$ **do**
  Compute $\boldsymbol{r}^{(t)} = \nabla f(\boldsymbol{x}^{(t)})$
  Compute $\boldsymbol{s}^{(t)} \in \underset{\boldsymbol{s} \in \mathcal{X}}{\operatorname{argmin}} \left\langle \boldsymbol{s}, \boldsymbol{r}^{(t)} \right\rangle$
  Compute $g_t := \left\langle \boldsymbol{x}^{(t)} - \boldsymbol{s}^{(t)}, \boldsymbol{r}^{(t)} \right\rangle$
  if $g_t \leq \epsilon$ **then return** $\boldsymbol{x}^{(t)}$
  Let $\gamma = \frac{2}{2+t}$ (or do line-search)
  Update $\boldsymbol{x}^{(t+1)} := (1-\gamma)\boldsymbol{x}^{(t)} + \gamma \boldsymbol{s}^{(t)}$
end for



The FW algorithm

Algorithm Frank-Wolfe algorithm

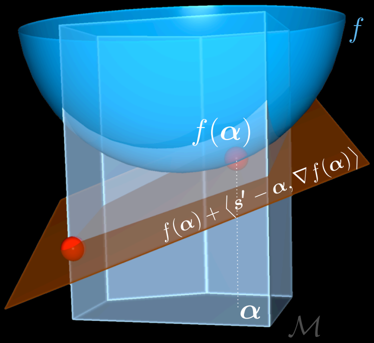
Let $\boldsymbol{x}^{(0)} \in \mathcal{X}$
for $t = 0 \ldots T$ **do**
  Compute $\boldsymbol{r}^{(t)} = \nabla f(\boldsymbol{x}^{(t)})$
  Compute $\boldsymbol{s}^{(t)} \in \underset{\boldsymbol{s} \in \mathcal{X}}{\operatorname{argmin}} \left\langle \boldsymbol{s}, \boldsymbol{r}^{(t)} \right\rangle$
  Compute $g_t := \left\langle \boldsymbol{x}^{(t)} - \boldsymbol{s}^{(t)}, \boldsymbol{r}^{(t)} \right\rangle$
  if $g_t \leq \epsilon$ **then return** $\boldsymbol{x}^{(t)}$
  Let $\gamma = \frac{2}{2+t}$ (or do line-search)
  Update $\boldsymbol{x}^{(t+1)} := (1-\gamma)\boldsymbol{x}^{(t)} + \gamma \boldsymbol{s}^{(t)}$
end for



The FW algorithm

Algorithm Frank-Wolfe algorithm

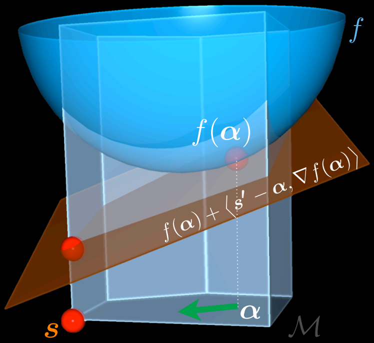
Let $\boldsymbol{x}^{(0)} \in \mathcal{X}$
for $t = 0 \dots T$ **do**
  Compute $\boldsymbol{r}^{(t)} = \nabla f(\boldsymbol{x}^{(t)})$
  Compute $\boldsymbol{s}^{(t)} \in \underset{\boldsymbol{s} \in \mathcal{X}}{\operatorname{argmin}} \left\langle \boldsymbol{s}, \boldsymbol{r}^{(t)} \right\rangle$
  Compute $g_t := \left\langle \boldsymbol{x}^{(t)} - \boldsymbol{s}^{(t)}, \boldsymbol{r}^{(t)} \right\rangle$
  if $g_t \le \epsilon$ **then return** $\boldsymbol{x}^{(t)}$
  Let $\gamma = \frac{2}{2+t}$ (or do line-search)
  Update $\boldsymbol{x}^{(t+1)} := (1-\gamma)\boldsymbol{x}^{(t)} + \gamma \boldsymbol{s}^{(t)}$
end for



The FW algorithm

Algorithm Frank-Wolfe algorithm

Let $\boldsymbol{x}^{(0)} \in \mathcal{X}$
for $t = 0 \dots T$ **do**
    Compute $\boldsymbol{r}^{(t)} = \nabla f(\boldsymbol{x}^{(t)})$
    Compute $\boldsymbol{s}^{(t)} \in \underset{\boldsymbol{s} \in \mathcal{X}}{\operatorname{argmin}} \left\langle \boldsymbol{s}, \boldsymbol{r}^{(t)} \right\rangle$
    Compute $g_t := \left\langle \boldsymbol{x}^{(t)} - \boldsymbol{s}^{(t)}, \boldsymbol{r}^{(t)} \right\rangle$
    if $g_t \leq \epsilon$ **then return** $\boldsymbol{x}^{(t)}$
    Let $\gamma = \frac{2}{2+t}$ (or do line-search)
    Update $\boldsymbol{x}^{(t+1)} := (1-\gamma)\boldsymbol{x}^{(t)} + \gamma \boldsymbol{s}^{(t)}$
end for



SP-FW

Algorithm Saddle point FW algorithm

Let $\boldsymbol{z}^{(0)} = (\boldsymbol{x}^{(0)}, \boldsymbol{y}^{(0)}) \in \mathcal{X} \times \mathcal{Y}$
for $t = 0 \dots T$ **do**
  Compute $\boldsymbol{r}^{(t)} := \begin{pmatrix} \nabla_x \mathcal{L}(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)}) \\ -\nabla_y \mathcal{L}(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)}) \end{pmatrix}$
  Compute $\boldsymbol{s}^{(t)} \in \underset{z \in \mathcal{X} \times \mathcal{Y}}{\operatorname{argmin}} \left\langle \boldsymbol{z}, \boldsymbol{r}^{(t)} \right\rangle$
  Compute $g_t := \left\langle \boldsymbol{z}^{(t)} - \boldsymbol{s}^{(t)}, \boldsymbol{r}^{(t)} \right\rangle$
  if $g_t \leq \epsilon$ **then return** $\boldsymbol{z}^{(t)}$
  Let $\gamma = \min\left(1, \frac{\nu}{C} g_t\right)$ **or** $\gamma = \frac{2}{2+t}$
  Update $\boldsymbol{z}^{(t+1)} := (1-\gamma)\boldsymbol{z}^{(t)} + \gamma \boldsymbol{s}^{(t)}$
end for

SP-FW

Algorithm Saddle point FW algorithm

1: Let $\boldsymbol{z}^{(0)} = (\boldsymbol{x}^{(0)}, \boldsymbol{y}^{(0)}) \in \mathcal{X} \times \mathcal{Y}$
2: **for** $t = 0 \dots T$ **do**
3: Compute $\boldsymbol{r}^{(t)} := \begin{pmatrix} \nabla_x \mathcal{L}(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)}) \\ -\nabla_y \mathcal{L}(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)}) \end{pmatrix}$
4: Compute $\boldsymbol{s}^{(t)} \in \underset{z \in \mathcal{X} \times \mathcal{Y}}{\operatorname{argmin}} \left\langle \boldsymbol{z}, \boldsymbol{r}^{(t)} \right\rangle$
5: Compute $g_t := \left\langle \boldsymbol{z}^{(t)} - \boldsymbol{s}^{(t)}, \boldsymbol{r}^{(t)} \right\rangle$
6: **if** $g_t \leq \epsilon$ **then return** $\boldsymbol{z}^{(t)}$
7: Let $\gamma = \min\left(1, \frac{\nu}{C} g_t\right)$ **or** $\gamma = \frac{2}{2+t}$
8: Update $\boldsymbol{z}^{(t+1)} := (1-\gamma)\boldsymbol{z}^{(t)} + \gamma \boldsymbol{s}^{(t)}$
9: **end for**

SP-FW

Algorithm Saddle point FW algorithm

Let $\boldsymbol{z}^{(0)} = (\boldsymbol{x}^{(0)}, \boldsymbol{y}^{(0)}) \in \mathcal{X} \times \mathcal{Y}$
for $t = 0 \dots T$ **do**
  Compute $\boldsymbol{r}^{(t)} := \begin{pmatrix} \nabla_x \mathcal{L}(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)}) \\ -\nabla_y \mathcal{L}(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)}) \end{pmatrix}$
  Compute $\boldsymbol{s}^{(t)} \in \underset{z \in \mathcal{X} \times \mathcal{Y}}{\operatorname{argmin}} \left\langle \boldsymbol{z}, \boldsymbol{r}^{(t)} \right\rangle$
  Compute $g_t := \left\langle \boldsymbol{z}^{(t)} - \boldsymbol{s}^{(t)}, \boldsymbol{r}^{(t)} \right\rangle$
  if $g_t \leq \epsilon$ **then return** $\boldsymbol{z}^{(t)}$
  Let $\gamma = \min\left(1, \frac{\nu}{C} g_t\right)$ **or** $\gamma = \frac{2}{2+t}$
  Update $\boldsymbol{z}^{(t+1)} := (1 - \gamma)\boldsymbol{z}^{(t)} + \gamma \boldsymbol{s}^{(t)}$
end for

SP-FW

Algorithm Saddle point FW algorithm

Let $\boldsymbol{z}^{(0)} = (\boldsymbol{x}^{(0)}, \boldsymbol{y}^{(0)}) \in \mathcal{X} \times \mathcal{Y}$
for $t = 0 \ldots T$ **do**
  Compute $\boldsymbol{r}^{(t)} := \begin{pmatrix} \nabla_x \mathcal{L}(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)}) \\ -\nabla_y \mathcal{L}(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)}) \end{pmatrix}$
  Compute $\boldsymbol{s}^{(t)} \in \operatorname*{argmin}_{z \in \mathcal{X} \times \mathcal{Y}} \left\langle \boldsymbol{z}, \boldsymbol{r}^{(t)} \right\rangle$
  Compute $g_t := \left\langle \boldsymbol{z}^{(t)} - \boldsymbol{s}^{(t)}, \boldsymbol{r}^{(t)} \right\rangle$
  if $g_t \leq \epsilon$ **then return** $\boldsymbol{z}^{(t)}$
  Let $\gamma = \min\left(1, \frac{\nu}{C} g_t\right)$ **or** $\gamma = \frac{2}{2+t}$
  Update $\boldsymbol{z}^{(t+1)} := (1-\gamma)\boldsymbol{z}^{(t)} + \gamma \boldsymbol{s}^{(t)}$
end for

- One can define FW extension with **away** step.

SP-FW

Algorithm Saddle point FW algorithm

Let $\boldsymbol{z}^{(0)} = (\boldsymbol{x}^{(0)}, \boldsymbol{y}^{(0)}) \in \mathcal{X} \times \mathcal{Y}$
for $t = 0 \ldots T$ **do**
  Compute $\boldsymbol{r}^{(t)} := \begin{pmatrix} \nabla_x \mathcal{L}(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)}) \\ -\nabla_y \mathcal{L}(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)}) \end{pmatrix}$
  Compute $\boldsymbol{s}^{(t)} \in \operatorname*{argmin}_{z \in \mathcal{X} \times \mathcal{Y}} \left\langle \boldsymbol{z}, \boldsymbol{r}^{(t)} \right\rangle$
  Compute $g_t := \left\langle \boldsymbol{z}^{(t)} - \boldsymbol{s}^{(t)}, \boldsymbol{r}^{(t)} \right\rangle$
  if $g_t \leq \epsilon$ **then return** $\boldsymbol{z}^{(t)}$
  Let $\gamma = \min\left(1, \frac{\nu}{C} g_t\right)$ **or** $\gamma = \frac{2}{2+t}$
  Update $\boldsymbol{z}^{(t+1)} := (1-\gamma)\boldsymbol{z}^{(t)} + \gamma \boldsymbol{s}^{(t)}$
end for

- One can define FW extension with **away** step.
- $\gamma_t = \frac{1}{1+t} \Rightarrow \boldsymbol{z}^{(t)} = \frac{1}{t} \sum_{i=0}^{t} \boldsymbol{s}^{(i)}$.
- $(\gamma_t = \frac{1}{1+t})$ + Bilinear objective $\leftrightarrow$ *fictitious play algorithm.*

Advantages of SP-FW

Same main property as FW:

$\boxed{\text{Only LMO (linear minimization oracle).}}$

Advantages of SP-FW

Same main property as FW:

$$\boxed{\text{Only LMO (linear minimization oracle).}}$$

Same other **advantages** as FW:

- Convergence certificate g_t for free.

Advantages of SP-FW

Same main property as FW:

$\boxed{\text{Only LMO (linear minimization oracle).}}$

Same other **advantages** as FW:

- Convergence certificate g_t for free.
- Affine invariance of the algorithm.

Advantages of SP-FW

Same main property as FW:

$\boxed{\text{Only LMO (linear minimization oracle).}}$

Same other **advantages** as FW:

- Convergence certificate g_t for free.
- Affine invariance of the algorithm.
- ***Sparsity*** of the iterates.

Advantages of SP-FW

Same main property as FW:

$$\boxed{\text{Only LMO (linear minimization oracle).}}$$

Same other **advantages** as FW:

- Convergence certificate g_t for free.
- Affine invariance of the algorithm.
- ***Sparsity*** of the iterates.
- Universal step size $\gamma_t := \frac{2}{2+t}$, adaptive step size $\gamma_t := \frac{\nu}{C} g_t$.

Advantages of SP-FW

Same main property as FW:

$$\boxed{\text{Only LMO (linear minimization oracle).}}$$

Same other **advantages** as FW:

- Convergence certificate g_t for free.
- Affine invariance of the algorithm.
- ***Sparsity*** of the iterates.
- Universal step size $\gamma_t := \frac{2}{2+t}$, adaptive step size $\gamma_t := \frac{\nu}{C} g_t$.

Main **difference** with SP:

- **No** line-search.

Advantages of SP-FW

Same main property as FW:

$\boxed{\text{Only LMO (linear minimization oracle).}}$

Same other **advantages** as FW:

- Convergence certificate g_t for free.
- Affine invariance of the algorithm.
- ***Sparsity*** of the iterates.
- Universal step size $\gamma_t := \frac{2}{2+t}$, adaptive step size $\gamma_t := \frac{\nu}{C} g_t$.

Main **difference** with SP:

- **No** line-search.

When constraints set is a "complicated" ***structured*** polytope projections can be ***hard*** whereas LMO might be ***tractable***.

Theoretical contribution

SP extension of FW with *away step*:

Convergence: $\boxed{\begin{array}{c}\textbf{\textit{Linear}} \text{ rate with } \textbf{\textit{adaptive}} \text{ step size.} \\ \textbf{\textit{Sublinear}} \text{ rate with } \textbf{\textit{universal}} \text{ step size.}\end{array}}$

[3] J. Hammond. "Solving asymmetric variational inequality problems and systems of equations with generalized nonlinear programming algorithms". PhD thesis. MIT, 1984.

Theoretical contribution

SP extension of FW with *away step*:

Convergence: $\boxed{\begin{array}{c}\textbf{\textit{Linear}}\text{ rate with }\textbf{\textit{adaptive}}\text{ step size.}\\ \textbf{\textit{Sublinear}}\text{ rate with }\textbf{\textit{universal}}\text{ step size.}\end{array}}$

- Similar hypothesis as AFW for linear convergence:
  1. Strong convexity and smoothness of the function.
  2. $\mathcal{X}$ and $\mathcal{Y}$ polytopes.

[3] J. Hammond. "Solving asymmetric variational inequality problems and systems of equations with generalized nonlinear programming algorithms". PhD thesis. MIT, 1984.

Theoretical contribution

SP extension of FW with *away step*:

Convergence: $\boxed{\begin{array}{c}\textbf{\textit{Linear}}\text{ rate with }\textbf{\textit{adaptive}}\text{ step size.}\\ \textbf{\textit{Sublinear}}\text{ rate with }\textbf{\textit{universal}}\text{ step size.}\end{array}}$

- Similar hypothesis as AFW for linear convergence:
  1. Strong convexity and smoothness of the function.
  2. $\mathcal{X}$ and $\mathcal{Y}$ polytopes.
- Additional assumption on the bilinearity.

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}) = f(\boldsymbol{x}) + \boldsymbol{x}^\top M \boldsymbol{y} - g(\boldsymbol{y})$$

$\|M\|$ smaller than the strong convexity constant.

[3] J. Hammond. "Solving asymmetric variational inequality problems and systems of equations with generalized nonlinear programming algorithms". PhD thesis. MIT, 1984.

Theoretical contribution

SP extension of FW with *away step*:

Convergence: $\boxed{\begin{array}{c}\textbf{\textit{Linear}} \text{ rate with } \textbf{\textit{adaptive}} \text{ step size.} \\ \textbf{\textit{Sublinear}} \text{ rate with } \textbf{\textit{universal}} \text{ step size.}\end{array}}$

- Similar hypothesis as AFW for linear convergence:
    1. Strong convexity and smoothness of the function.
    2. $\mathcal{X}$ and $\mathcal{Y}$ polytopes.
- Additional assumption on the bilinearity.

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}) = f(\boldsymbol{x}) + \boldsymbol{x}^\top M \boldsymbol{y} - g(\boldsymbol{y})$$

  $\|M\|$ smaller than the strong convexity constant.
- Proof use recent advances on AFW.
- Partially answering a **30 years old conjecture**[3].

[3] J. Hammond. "Solving asymmetric variational inequality problems and systems of equations with generalized nonlinear programming algorithms". PhD thesis. MIT, 1984.

Difficulties for saddle point

Usual **descent Lemma**:

$$h_{t+1} \leq h_t - \underbrace{\gamma_t g_t}_{\geq 0} + \gamma_t^2 \frac{L\|\boldsymbol{d}^{(t)}\|^2}{2}$$

With γ_t small enough the sequence decreases.

Difficulties for saddle point

Usual **descent Lemma**:

$$h_{t+1} \leq h_t - \underbrace{\gamma_t g_t}_{\geq 0} + \gamma_t^2 \frac{L\|\boldsymbol{d}^{(t)}\|^2}{2}$$

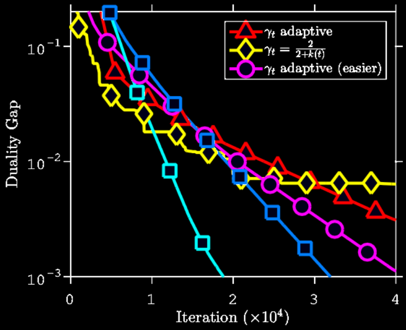
With γ_t small enough the sequence decreases.

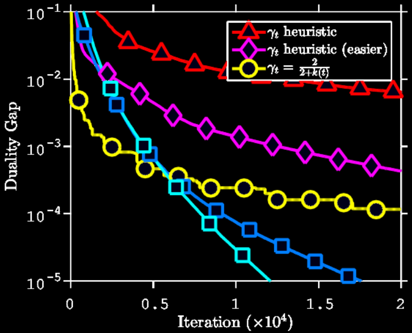
For saddle point problem the Lipschitz gradient property gives

$$\mathcal{L}_{t+1} - \mathcal{L}^* \leq \mathcal{L}_t - \mathcal{L}^* - \underbrace{\gamma_t \left(g_t^{(x)} - g_t^{(y)} \right)}_{\text{arbitrary sign}} + \gamma_t^2 \frac{L\|\boldsymbol{d}^{(t)}\|^2}{2}.$$

- Cannot control the oscillation of the sequence.
- Must introduce other quantities to establish convergence.

Toy experiments



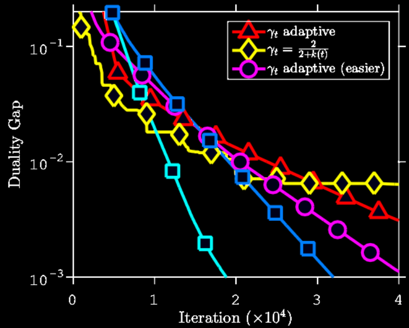


SP-AFW on a toy example $d = 30$. with theoretical step-size $\gamma_t = \frac{\nu}{C} g_t$.

$C = 2LD^2$

Figure: SP-AFW on a toy example $d = 30$ with heuristic step-size. $\gamma_t = \frac{g_t}{C + 2\frac{\|M\|^2 D^2}{\mu}}$

Toy experiments



SP-AFW on a toy example $d = 30$. with theoretical step-size $\gamma_t = \frac{\nu}{C} g_t$.

$C = 2LD^2$

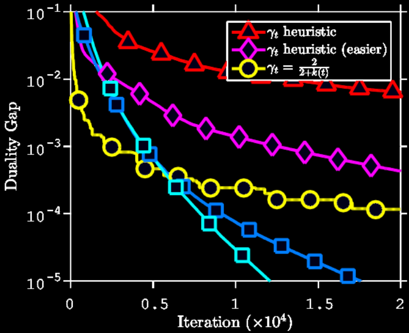


Figure: SP-AFW on a toy example $d = 30$ with heuristic step-size. $\gamma_t = \frac{g_t}{C + 2\frac{\|M\|^2 D^2}{\mu}}$

Conclusion

- SP-FW one of the first SP solver only working with LMO.

Conclusion

- SP-FW one of the first SP solver only working with LMO.
- FW resurgence lead to new *structured* problems.

Conclusion

- SP-FW one of the first SP solver only working with LMO.
- FW resurgence lead to new *structured* problems.
- Same hope as FW for SP-FW

Conclusion

- SP-FW one of the first SP solver only working with LMO.
- FW resurgence lead to new *structured* problems.
- Same hope as FW for SP-FW ↴

$\boxed{\text{Call for applications !}}$

Conclusion

- SP-FW one of the first SP solver only working with LMO.
- FW resurgence lead to new *structured* problems.
- Same hope as FW for SP-FW ↴

$\boxed{\text{Call for applications !}}$

- Still many theoretical opened questions.

Conclusion

- SP-FW one of the first SP solver only working with LMO.
- FW resurgence lead to new *structured* problems.
- Same hope as FW for SP-FW $\lrcorner$

$$\boxed{\text{Call for applications !}}$$

- Still many theoretical opened questions.
- With a bilinear objective this algorithm is ***highly related*** to the *fictitious play algorithm.*

Conclusion

- SP-FW one of the first SP solver only working with LMO.
- FW resurgence lead to new *structured* problems.
- Same hope as FW for SP-FW $\lnot$

$$\boxed{\text{Call for applications !}}$$

- Still many theoretical opened questions.
- With a bilinear objective this algorithm is ***highly related*** to the *fictitious play algorithm.*
- Rich interplay tapping into this game theory literature.

Thank You !

Slides available on `www.di.ens.fr/~gidel`.