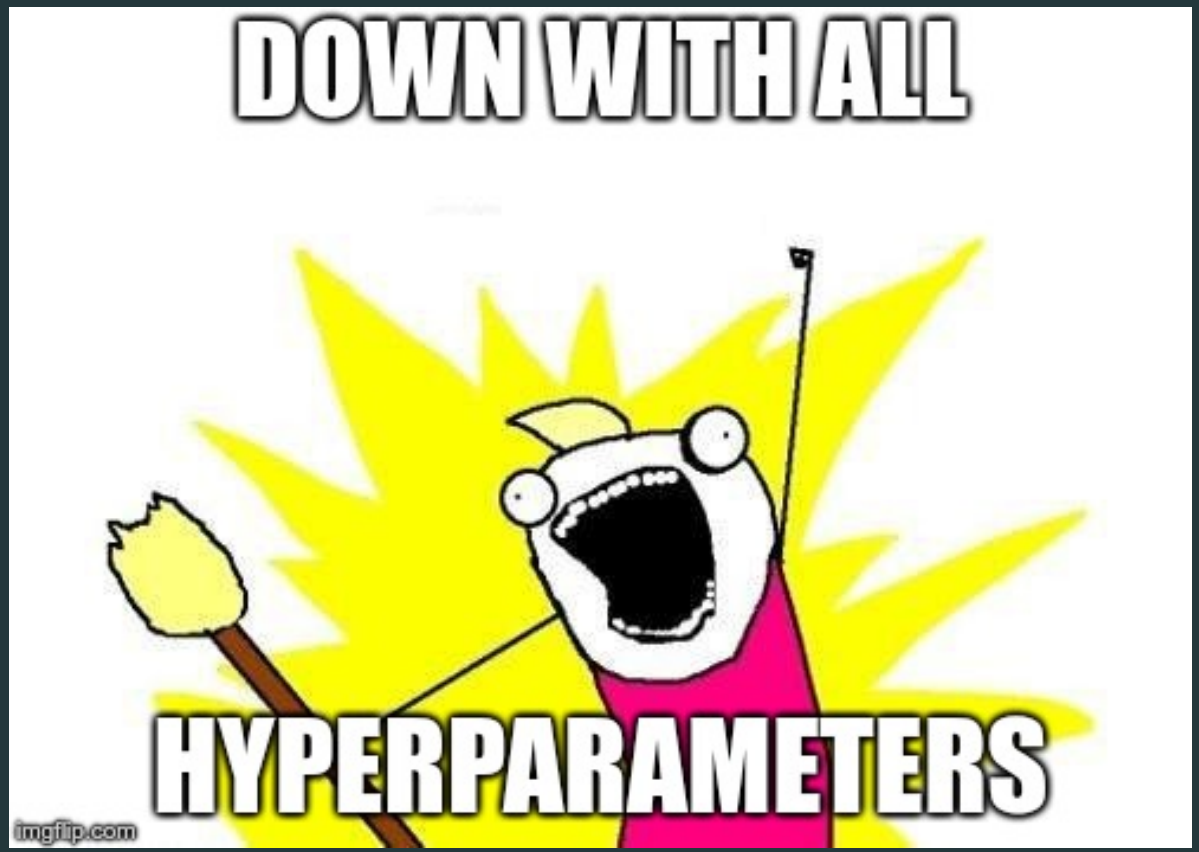
Sufficient decrease is all you need

Fabian Pedregosa

DALI 2019

Motivation



Problem Setting

Most first-order optimization methods are of the form

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \gamma_t \boldsymbol{p}_t \qquad \text{where}$$

- $\boldsymbol{p}_t$ is the update direction, determined by the algorithm.
- γ_t is a the step-size, free parameter.

How to choose the step-size?

- Chosen **in advance**, i.e., $\gamma_t = 1/L$ or $\gamma_t = L/\sqrt{t+1}$

  \+ Simple, - Suboptimal

Problem Setting

Most first-order optimization methods are of the form

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \gamma_t \boldsymbol{p}_t \qquad \text{where}$$

- $\boldsymbol{p}_t$ is the update direction, determined by the algorithm.
- γ_t is a the step-size, free parameter.

How to choose the step-size?

- Chosen **in advance**, i.e., $\gamma_t = 1/L$ or $\gamma_t = L/\sqrt{t+1}$
  + Simple, - Suboptimal
- Exact line-search: $\gamma_t \in \mathbf{arg\,min}_{\gamma \geq 0} f(\boldsymbol{x}_t + \gamma \boldsymbol{p}_t)$
  + Optimal, - Expensive

Problem Setting

Most first-order optimization methods are of the form

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \gamma_t \boldsymbol{p}_t \qquad \text{where}$$

- $\boldsymbol{p}_t$ is the update direction, determined by the algorithm.
- γ_t is a the step-size, free parameter.

How to choose the step-size?

- Chosen **in advance**, i.e., $\gamma_t = 1/L$ or $\gamma_t = L/\sqrt{t+1}$
  + Simple, - Suboptimal
- Exact line-search: $\gamma_t \in \mathbf{arg\,min}_{\gamma \geq 0} f(\boldsymbol{x}_t + \gamma \boldsymbol{p}_t)$
  + Optimal, - Expensive
- **Adaptive** step-size (aka bactracking, inexact LS): Choose γ_t based on local sufficient decrease condition
  +Efficient, +Simple

Adaptive Step-size Selection

For gradient descent ($\boldsymbol{p}_t = -\nabla f(\boldsymbol{x}_t)$), Armijo backtracking (Armijo, 1966) simple and efficient condition:

$$f(\boldsymbol{x}_{t+1}) \leq f(\boldsymbol{x}_t) - \gamma_t \|\nabla f(\boldsymbol{x}_t)\|^2$$

What about other methods?

$\Longrightarrow$ **This talk**

Outline

1. Adaptive **Three Operator Splitting** and structured saddle-point problems.

2. Adaptive **Frank-Wolfe** (and linearly-convergent variants).

3. **Perspectives**: stochastic optimization.

Three Operator Splitting (TOS) (Davis and Yin, 2017)

Solves optimization problems of the form

$$\underset{x \in \mathbb{R}^d}{\text{minimize}}\, f(x) + g(x) + h(x)\ ,$$

where f is convex and L-smooth, g, h are convex with access to $\mathbf{prox}_{\gamma g}(x) \stackrel{\text{def}}{=} \mathbf{arg\,min}_z\, g(z) + \frac{1}{2\gamma}\|x - z\|^2$, $\mathbf{prox}_{\gamma h}$.

Iterates on $y_t \in \mathbb{R}^d$ given by

$$\begin{aligned} z_t &= \mathbf{prox}_{\gamma h}(y_t) \\ x_t &= \mathbf{prox}_{\gamma g}(2y_t - z_t - \gamma \nabla f(z_t)) \\ y_{t+1} &= y_t - z_t + x_t \end{aligned}$$

Three Operator Splitting (TOS) (Davis and Yin, 2017)

Solves optimization problems of the form

$$\underset{x \in \mathbb{R}^d}{\text{minimize}}\, f(x) + g(x) + h(x)\ ,$$

where f is convex and L-smooth, g, h are convex with access to $\mathbf{prox}_{\gamma g}(x) \stackrel{\text{def}}{=} \mathbf{arg\,min}_z\, g(z) + \frac{1}{2\gamma}\|x - z\|^2$, $\mathbf{prox}_{\gamma h}$.

Iterates on $y_t \in \mathbb{R}^d$ given by

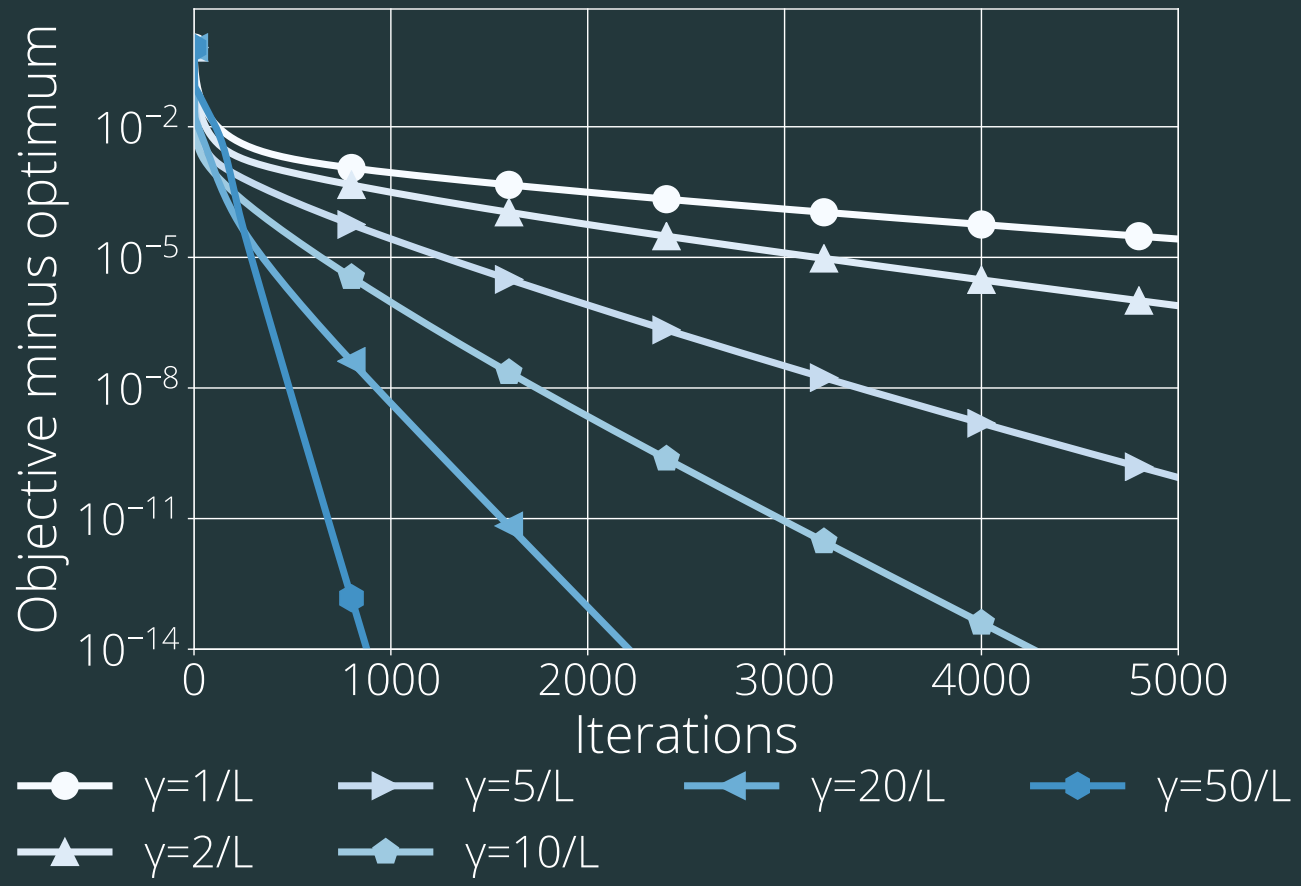
$$\begin{aligned} z_t &= \mathbf{prox}_{\gamma h}(y_t) \\ x_t &= \mathbf{prox}_{\gamma g}(2y_t - z_t - \gamma \nabla f(z_t)) \\ y_{t+1} &= y_t - z_t + x_t \end{aligned}$$

Generalizes proximal-gradient (FB, ISTA) and Douglas-Rachford.

Importance of step-size

Guaranteed convergence $\gamma < 2/L$, with $L =$ Lipschitz const. of ∇f.

In practice, best performance is when $\gamma \gg 2/L$



What can we do about it?

Revisiting the Three Operator Splitting

Saddle-point reformulation of original problem

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} f(\boldsymbol{x}) + g(\boldsymbol{x}) + h(\boldsymbol{x})$$

$$= \min_{\boldsymbol{x}\in\mathbb{R}^d} f(\boldsymbol{x}) + g(\boldsymbol{x}) + \max_{\boldsymbol{u}\in\mathbb{R}^d} \{\langle \boldsymbol{x}, \boldsymbol{u}\rangle - h^*(\boldsymbol{u})\}$$

Revisiting the Three Operator Splitting

Saddle-point reformulation of original problem

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} f(\boldsymbol{x}) + g(\boldsymbol{x}) + h(\boldsymbol{x})$$

$$= \min_{\boldsymbol{x}\in\mathbb{R}^d} \max_{\boldsymbol{u}\in\mathbb{R}^d} \underbrace{f(\boldsymbol{x}) + g(\boldsymbol{x}) + \langle \boldsymbol{x}, \boldsymbol{u}\rangle - h^*(\boldsymbol{u})}_{:=\mathcal{L}(\boldsymbol{x},\boldsymbol{u})}$$

We can rewrite the three operator splitting as

$$\begin{aligned}
\boldsymbol{x}_{t+1} &= \mathbf{prox}_{\gamma g}(\boldsymbol{z}_t - \gamma(\nabla f(\boldsymbol{z}_t) + \boldsymbol{u}_t)) \\
\boldsymbol{u}_{t+1} &= \mathbf{prox}_{h^*/\gamma}(\boldsymbol{u}_t + \boldsymbol{x}_{t+1}/\gamma)\ , \\
\boldsymbol{z}_{t+1} &= \boldsymbol{x}_{t+1} - \gamma(\boldsymbol{u}_{t+1} - \boldsymbol{u}_t)
\end{aligned}$$

Minimizing with respect to primal variable

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} \mathcal{L}(\boldsymbol{x}, \boldsymbol{u}_t) = \min_{\boldsymbol{x}\in\mathbb{R}^d} \underbrace{f(\boldsymbol{x}) + \langle \boldsymbol{x}, \boldsymbol{u}_t \rangle}_{\text{smooth}} + \underbrace{g(\boldsymbol{x})}_{\text{proximal}}$$

Minimizing with respect to primal variable

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} \mathcal{L}(\boldsymbol{x}, \boldsymbol{u}_t) = \min_{\boldsymbol{x}\in\mathbb{R}^d} \underbrace{f(\boldsymbol{x}) + \langle \boldsymbol{x}, \boldsymbol{u}_t \rangle}_{\text{smooth}} + \underbrace{g(\boldsymbol{x})}_{\text{proximal}}$$

- Proximal-gradient iteration, with $\boldsymbol{x} = \boldsymbol{z}_t$ as starting point:

$$\boldsymbol{x}_{t+1} = \mathbf{prox}_{\gamma g}(\boldsymbol{z}_t - \gamma(\nabla f(\boldsymbol{z}_t) + \boldsymbol{u}_t))$$

Minimizing with respect to primal variable

$$\min_{x \in \mathbb{R}^d} \mathcal{L}(x, u_t) = \min_{x \in \mathbb{R}^d} \underbrace{f(x) + \langle x, u_t \rangle}_{\text{smooth}} + \underbrace{g(x)}_{\text{proximal}}$$

- Proximal-gradient iteration, with $x = z_t$ as starting point:

$$x_{t+1} = \mathbf{prox}_{\gamma g}(z_t - \gamma(\nabla f(z_t) + u_t))$$

$=$ first step of TOS

Minimizing with respect to the dual variable

$$\min_{\boldsymbol{u} \in \mathbb{R}^d} \mathcal{L}(\boldsymbol{x}_t, \boldsymbol{u}) = \min_{\boldsymbol{u} \in \mathbb{R}^d} \underbrace{h^*(\boldsymbol{u}) - \langle \boldsymbol{x}_t, \boldsymbol{u} \rangle}_{\text{proximal}}$$

Minimizing with respect to the dual variable

$$\min_{\boldsymbol{u}\in\mathbb{R}^d} \mathcal{L}(\boldsymbol{x}_t, \boldsymbol{u}) = \min_{\boldsymbol{u}\in\mathbb{R}^d} \underbrace{h^*(\boldsymbol{u}) - \langle \boldsymbol{x}_t, \boldsymbol{u}\rangle}_{\text{proximal}}$$

- Proximal-point iteration:

$$\boldsymbol{u}_{t+1} = \mathbf{prox}_{\sigma h^*}(\boldsymbol{u}_t + \sigma \boldsymbol{x}_{t+1})$$

$=$ second update in TOS with $\sigma = 1/\gamma$

Extrapolation Step

Last update:

$$\boldsymbol{z}_{t+1} = \boldsymbol{x}_{t+1} - \gamma(\boldsymbol{u}_{t+1} - \boldsymbol{u}_t) \qquad \text{(extrapolation step)}$$

Extrapolation Step

Last update:

$$\boldsymbol{z}_{t+1} = \boldsymbol{x}_{t+1} - \gamma(\boldsymbol{u}_{t+1} - \boldsymbol{u}_t) \qquad \text{(extrapolation step)}$$

Verifies $\boldsymbol{z}_{t+1} \in \partial h^\star(\boldsymbol{u}_{t+1})$.

Extrapolation Step

Last update:

$$\boldsymbol{z}_{t+1} = \boldsymbol{x}_{t+1} - \gamma(\boldsymbol{u}_{t+1} - \boldsymbol{u}_t) \qquad \text{(extrapolation step)}$$

Verifies $\boldsymbol{z}_{t+1} \in \partial h^\star(\boldsymbol{u}_{t+1})$.

At optimum, we have $\boldsymbol{x}^\star \in \partial h^\star(\boldsymbol{u}^\star)$.

$\implies$ Solving the KKT conditions at $\boldsymbol{u}_{t+1}$.

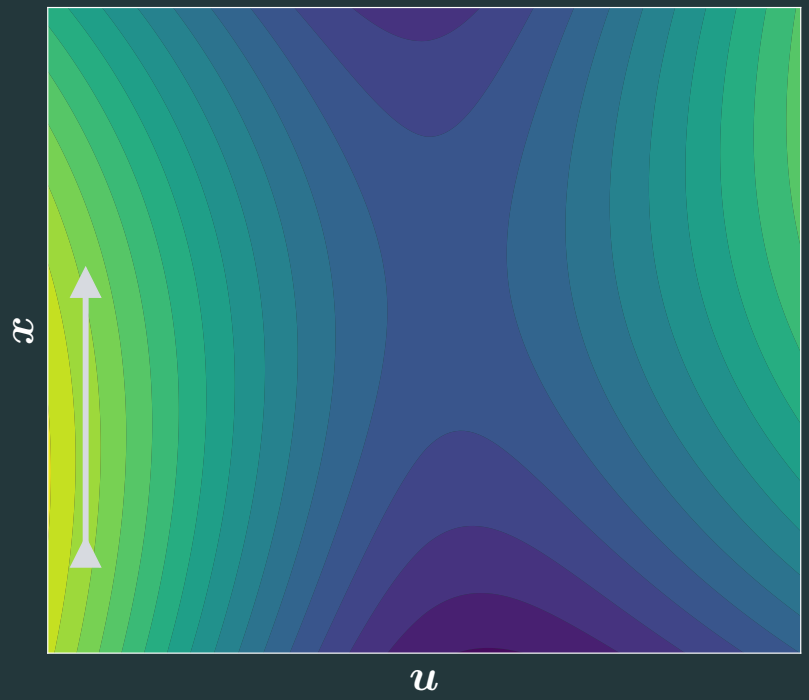
Revisiting the three operator splitting

Iteration 1: proximal-gradient step

$\rightarrow \boldsymbol{x}_{t+1} = \mathbf{prox}_{\gamma g}(\boldsymbol{z}_t - \gamma(\nabla f(\boldsymbol{z}_t) + \boldsymbol{u}_t))$

$\boldsymbol{u}_{t+1} = \mathbf{prox}_{h^\star/\gamma}(\boldsymbol{u}_t + \boldsymbol{x}_{t+1}/\gamma)$

$\boldsymbol{z}_{t+1} = \boldsymbol{x}_{t+1} - \gamma(\boldsymbol{u}_{t+1} - \boldsymbol{u}_t)$



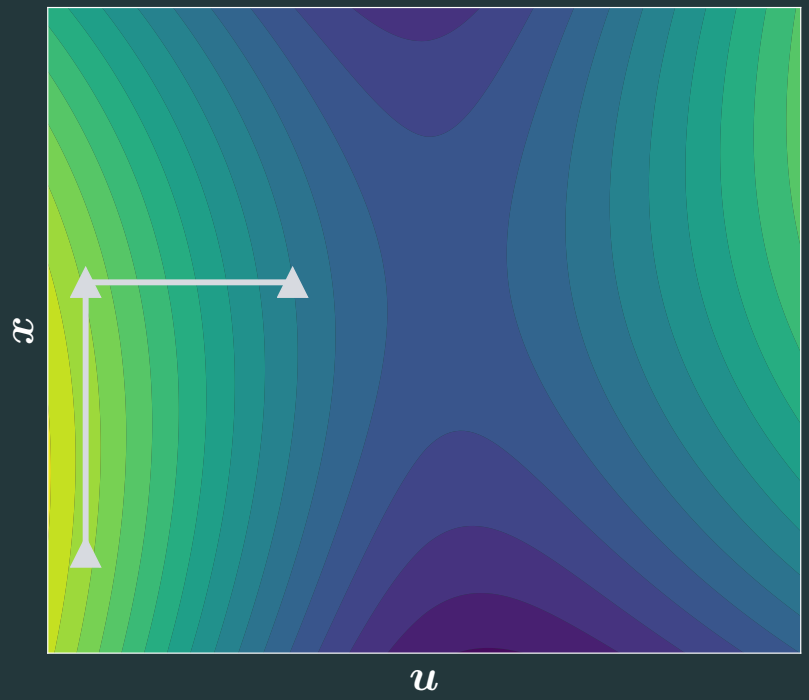
Revisiting the three operator splitting

Iteration 1: proximal-point step

$$\boldsymbol{x}_{t+1} = \mathbf{prox}_{\gamma g}(\boldsymbol{z}_t - \gamma(\nabla f(\boldsymbol{z}_t) + \boldsymbol{u}_t))$$

$$\rightarrow \boldsymbol{u}_{t+1} = \mathbf{prox}_{h^\star/\gamma}(\boldsymbol{u}_t + \boldsymbol{x}_{t+1}/\gamma)$$

$$\boldsymbol{z}_{t+1} = \boldsymbol{x}_{t+1} - \gamma(\boldsymbol{u}_{t+1} - \boldsymbol{u}_t)$$



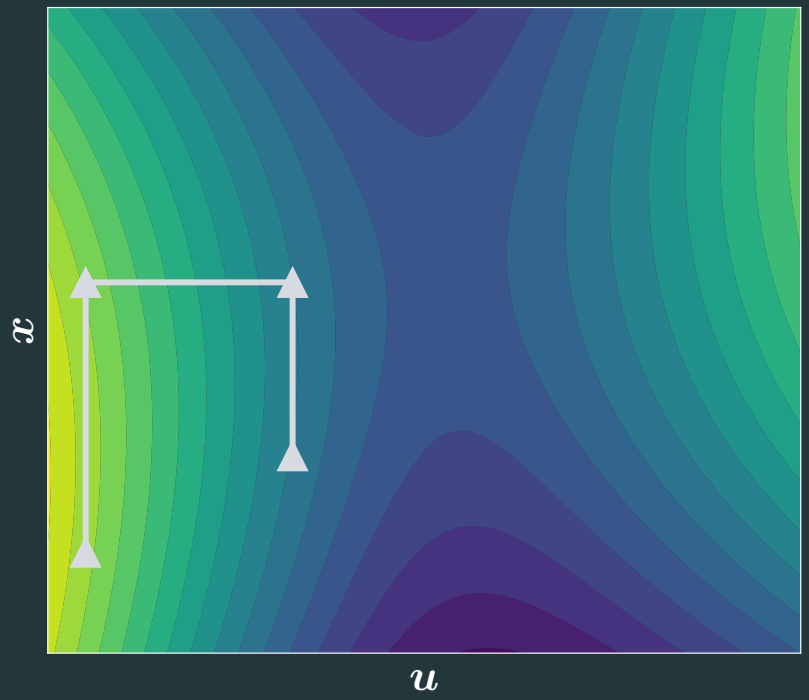
Revisiting the three operator splitting

Iteration 2: extrapolation

$$\boldsymbol{x}_{t+1} = \mathbf{prox}_{\gamma g}(\boldsymbol{z}_t - \gamma(\nabla f(\boldsymbol{z}_t) + \boldsymbol{u}_t))$$

$$\boldsymbol{u}_{t+1} = \mathbf{prox}_{h^\star/\gamma}(\boldsymbol{u}_t + \boldsymbol{x}_{t+1}/\gamma)$$

$$\rightarrow \boldsymbol{z}_{t+1} = \boldsymbol{x}_{t+1} - \gamma(\boldsymbol{u}_{t+1} - \boldsymbol{u}_t)$$



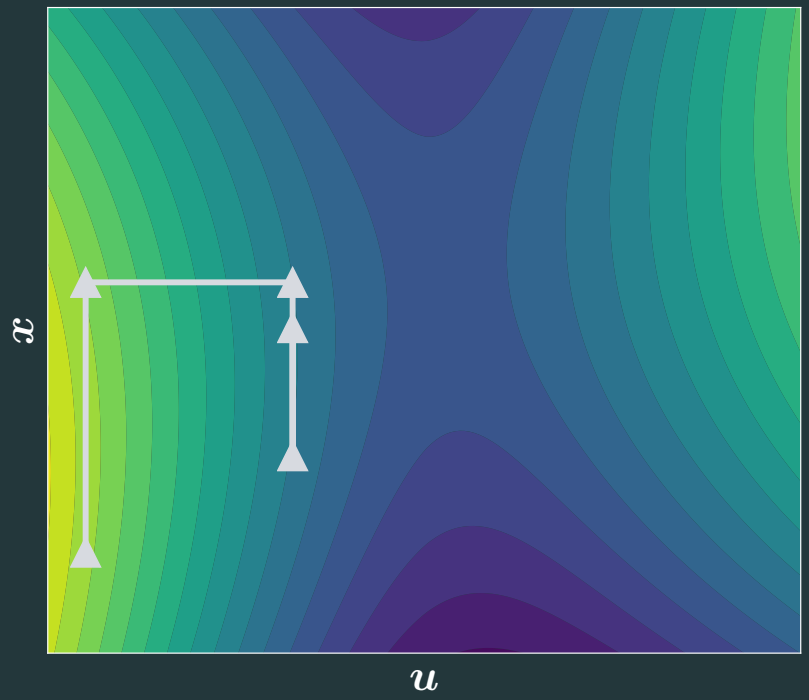
Revisiting the three operator splitting

Iteration 2: proximal-gradient step

$\rightarrow \boldsymbol{x}_{t+1} = \mathbf{prox}_{\gamma g}(\boldsymbol{z}_t - \gamma(\nabla f(\boldsymbol{z}_t) + \boldsymbol{u}_t))$

$\boldsymbol{u}_{t+1} = \mathbf{prox}_{h^\star/\gamma}(\boldsymbol{u}_t + \boldsymbol{x}_{t+1}/\gamma)$

$\boldsymbol{z}_{t+1} = \boldsymbol{x}_{t+1} - \gamma(\boldsymbol{u}_{t+1} - \boldsymbol{u}_t)$



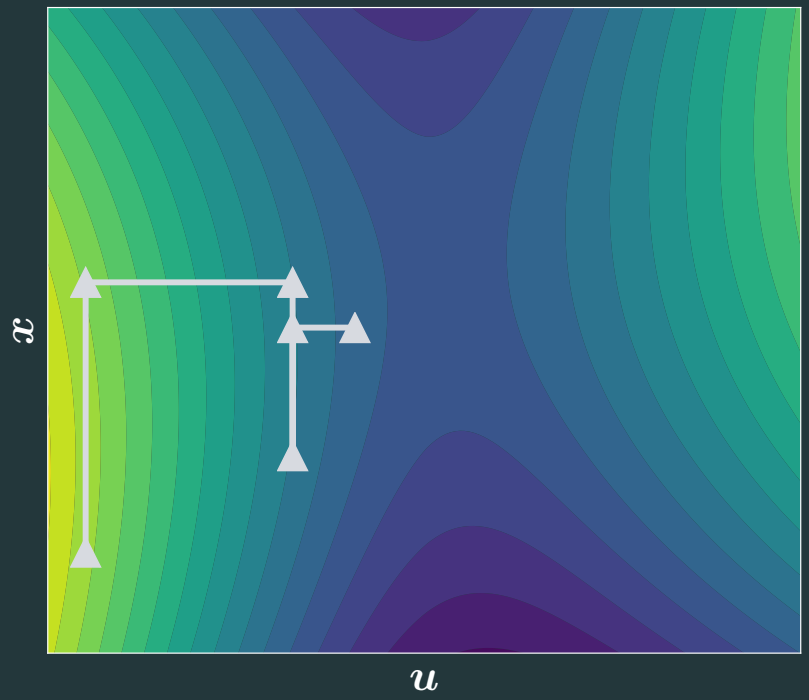
Revisiting the three operator splitting

Iteration 3: proximal-point step

$$\boldsymbol{x}_{t+1} = \mathbf{prox}_{\gamma g}(\boldsymbol{z}_t - \gamma(\nabla f(\boldsymbol{z}_t) + \boldsymbol{u}_t))$$

$$\rightarrow \boldsymbol{u}_{t+1} = \mathbf{prox}_{h^\star/\gamma}(\boldsymbol{u}_t + \boldsymbol{x}_{t+1}/\gamma)$$

$$\boldsymbol{z}_{t+1} = \boldsymbol{x}_{t+1} - \gamma(\boldsymbol{u}_{t+1} - \boldsymbol{u}_t)$$



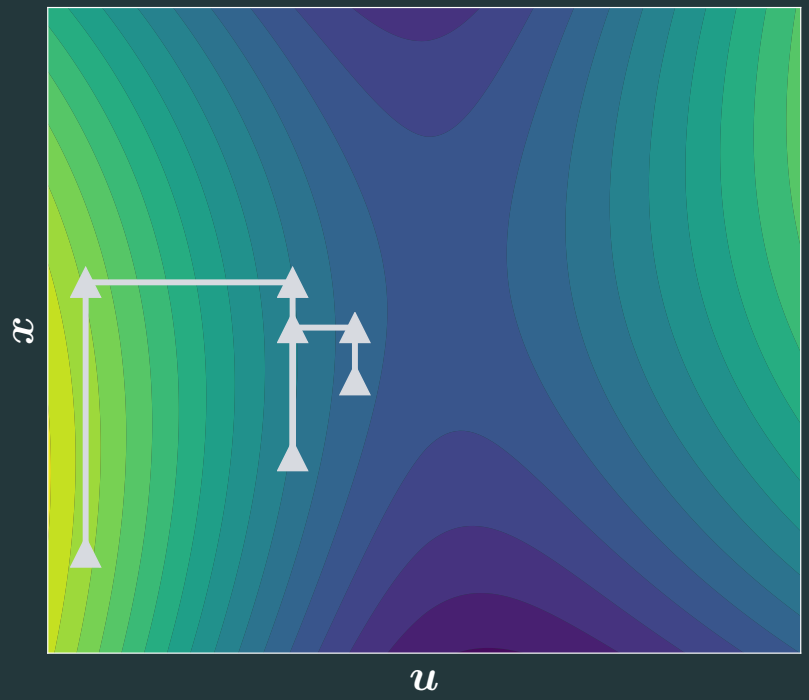
Revisiting the three operator splitting

Iteration 3: extrapolation

$$\boldsymbol{x}_{t+1} = \mathbf{prox}_{\gamma g}(\boldsymbol{z}_t - \gamma(\nabla f(\boldsymbol{z}_t) + \boldsymbol{u}_t))$$

$$\boldsymbol{u}_{t+1} = \mathbf{prox}_{h^\star/\gamma}(\boldsymbol{u}_t + \boldsymbol{x}_{t+1}/\gamma)$$

$$\rightarrow \boldsymbol{z}_{t+1} = \boldsymbol{x}_{t+1} - \gamma(\boldsymbol{u}_{t+1} - \boldsymbol{u}_t)$$



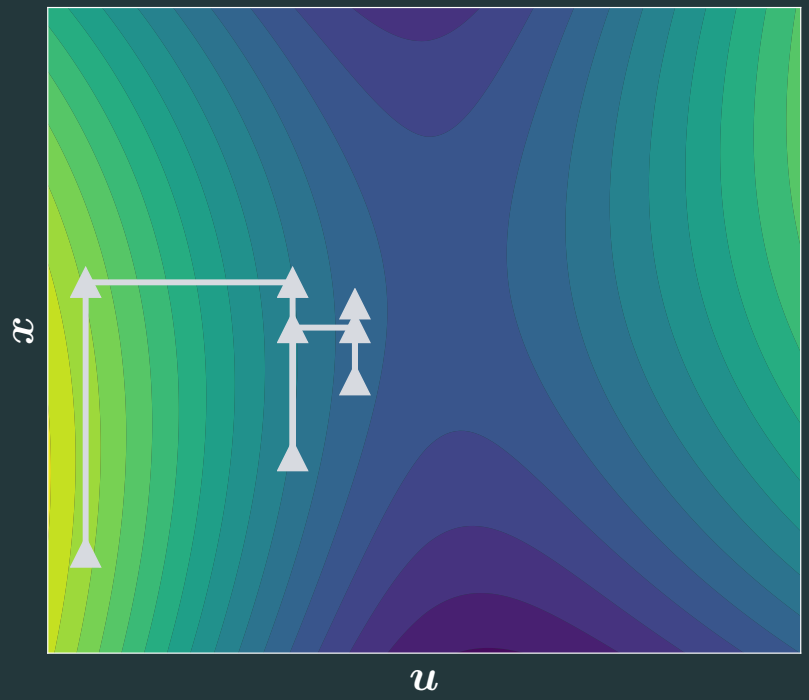
Revisiting the three operator splitting

Iteration 4: proximal-gradient step

$\rightarrow \boldsymbol{x}_{t+1} = \mathbf{prox}_{\gamma g}(\boldsymbol{z}_t - \gamma(\nabla f(\boldsymbol{z}_t) + \boldsymbol{u}_t))$

$\boldsymbol{u}_{t+1} = \mathbf{prox}_{h^\star/\gamma}(\boldsymbol{u}_t + \boldsymbol{x}_{t+1}/\gamma)$

$\boldsymbol{z}_{t+1} = \boldsymbol{x}_{t+1} - \gamma(\boldsymbol{u}_{t+1} - \boldsymbol{u}_t)$



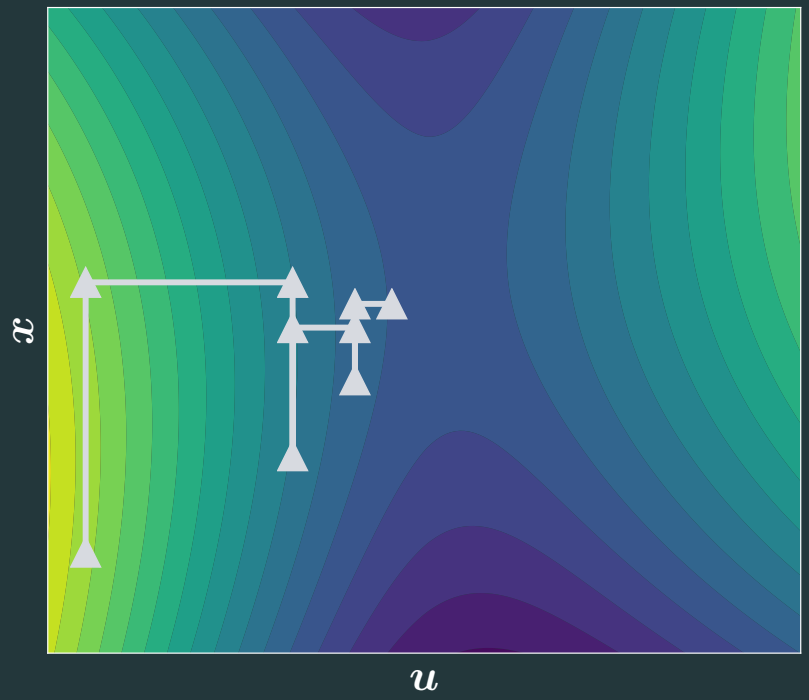
Revisiting the three operator splitting

Iteration 4: proximal-point step

$$\boldsymbol{x}_{t+1} = \mathbf{prox}_{\gamma g}(\boldsymbol{z}_t - \gamma(\nabla f(\boldsymbol{z}_t) + \boldsymbol{u}_t))$$

$$\rightarrow \boldsymbol{u}_{t+1} = \mathbf{prox}_{h^\star/\gamma}(\boldsymbol{u}_t + \boldsymbol{x}_{t+1}/\gamma)$$

$$\boldsymbol{z}_{t+1} = \boldsymbol{x}_{t+1} - \gamma(\boldsymbol{u}_{t+1} - \boldsymbol{u}_t)$$



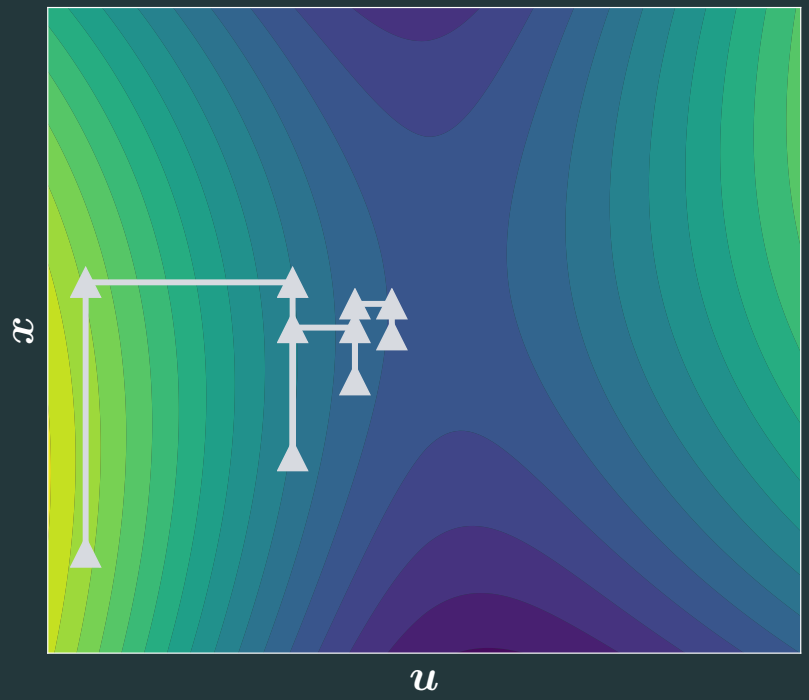
Revisiting the three operator splitting

Iteration 5: extrapolation

$$\boldsymbol{x}_{t+1} = \mathbf{prox}_{\gamma g}(\boldsymbol{z}_t - \gamma(\nabla f(\boldsymbol{z}_t) + \boldsymbol{u}_t))$$

$$\boldsymbol{u}_{t+1} = \mathbf{prox}_{h^\star/\gamma}(\boldsymbol{u}_t + \boldsymbol{x}_{t+1}/\gamma)$$

$$\rightarrow \boldsymbol{z}_{t+1} = \boldsymbol{x}_{t+1} - \gamma(\boldsymbol{u}_{t+1} - \boldsymbol{u}_t)$$



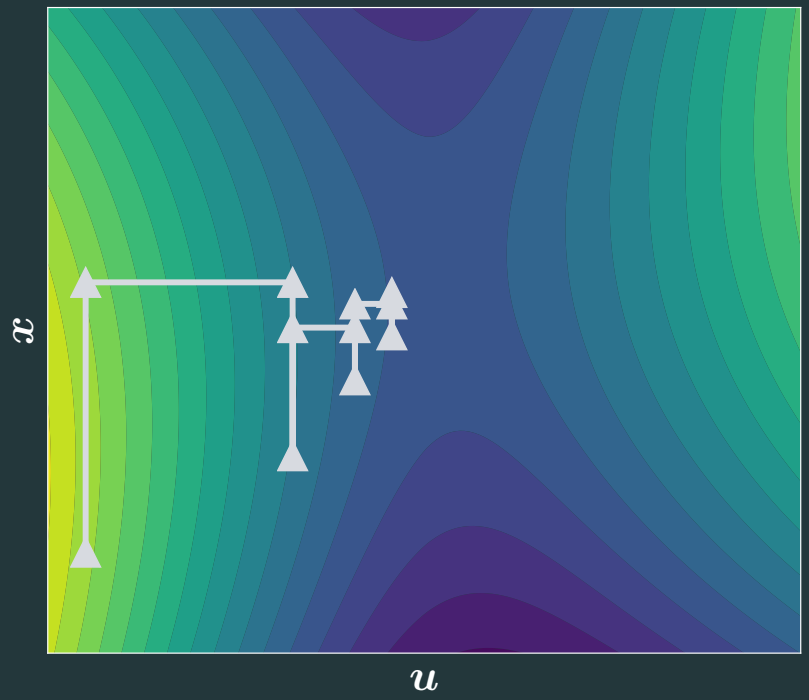
Revisiting the three operator splitting

Iteration 5: proximal-gradient step

$\rightarrow \boldsymbol{x}_{t+1} = \mathbf{prox}_{\gamma g}(\boldsymbol{z}_t - \gamma(\nabla f(\boldsymbol{z}_t) + \boldsymbol{u}_t))$

$\boldsymbol{u}_{t+1} = \mathbf{prox}_{h^\star/\gamma}(\boldsymbol{u}_t + \boldsymbol{x}_{t+1}/\gamma)$

$\boldsymbol{z}_{t+1} = \boldsymbol{x}_{t+1} - \gamma(\boldsymbol{u}_{t+1} - \boldsymbol{u}_t)$



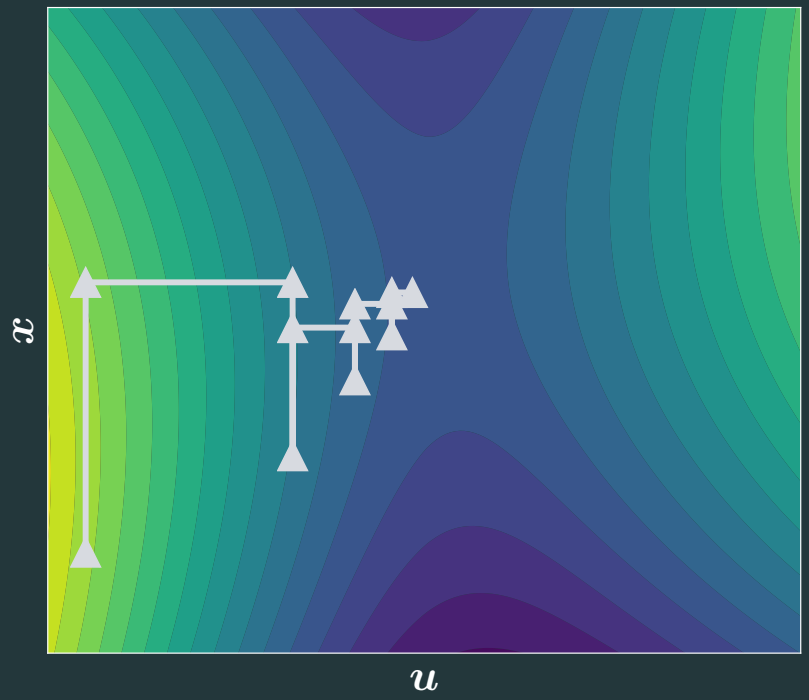
Revisiting the three operator splitting

Iteration 6: proximal-point step

$$\boldsymbol{x}_{t+1} = \mathbf{prox}_{\gamma g}(\boldsymbol{z}_t - \gamma(\nabla f(\boldsymbol{z}_t) + \boldsymbol{u}_t))$$

$$\rightarrow \boldsymbol{u}_{t+1} = \mathbf{prox}_{h^\star/\gamma}(\boldsymbol{u}_t + \boldsymbol{x}_{t+1}/\gamma)$$

$$\boldsymbol{z}_{t+1} = \boldsymbol{x}_{t+1} - \gamma(\boldsymbol{u}_{t+1} - \boldsymbol{u}_t)$$



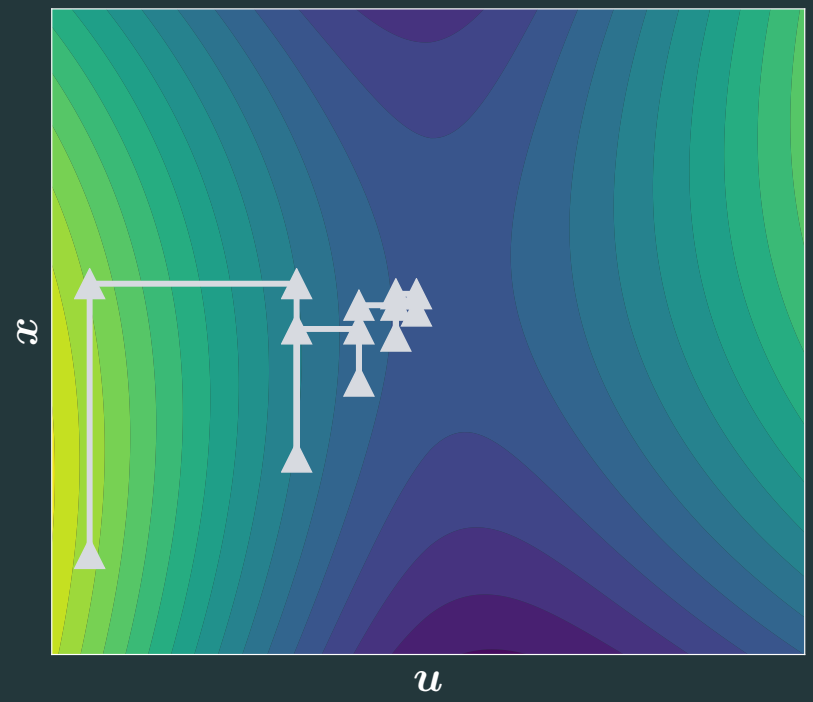
Revisiting the three operator splitting

Iteration 6: extrapolation

$$\boldsymbol{x}_{t+1} = \mathbf{prox}_{\gamma g}(\boldsymbol{z}_t - \gamma(\nabla f(\boldsymbol{z}_t) + \boldsymbol{u}_t))$$

$$\boldsymbol{u}_{t+1} = \mathbf{prox}_{h^\star/\gamma}(\boldsymbol{u}_t + \boldsymbol{x}_{t+1}/\gamma)$$

$$\rightarrow \boldsymbol{z}_{t+1} = \boldsymbol{x}_{t+1} - \gamma(\boldsymbol{u}_{t+1} - \boldsymbol{u}_t)$$



Take-Home Message

TOS is (basically) alternated proximal-gradient and proximal-point

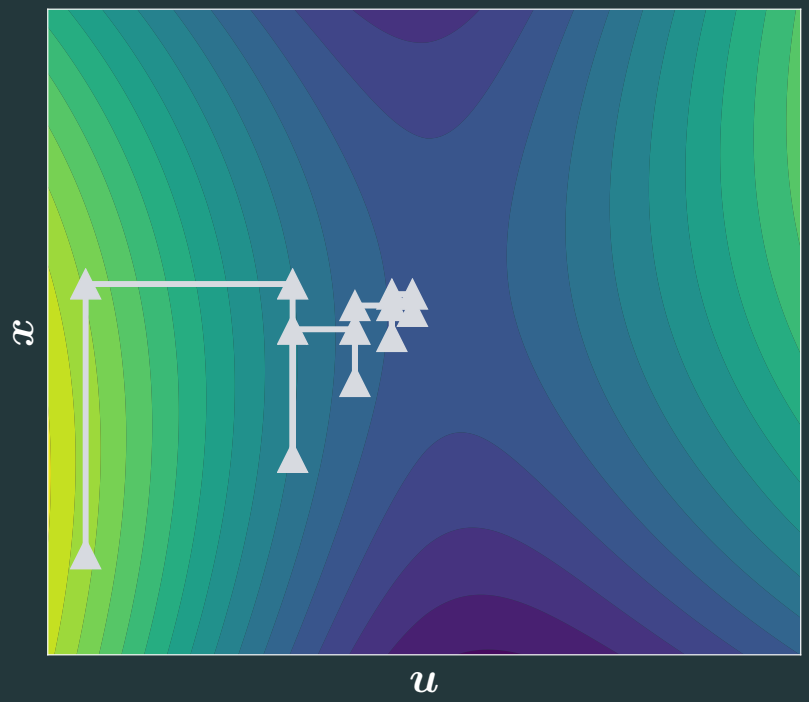
Revisiting the three operator splitting

Iteration 6: extrapolation

$$\boldsymbol{x}_{t+1} = \mathbf{prox}_{\gamma g}(\boldsymbol{z}_t - \gamma(\nabla f(\boldsymbol{z}_t) + \boldsymbol{u}_t))$$

$$\boldsymbol{u}_{t+1} = \mathbf{prox}_{h^\star/\gamma}(\boldsymbol{u}_t + \boldsymbol{x}_{t+1}/\gamma)$$

$$\rightarrow \boldsymbol{z}_{t+1} = \boldsymbol{x}_{t+1} - \gamma(\boldsymbol{u}_{t+1} - \boldsymbol{u}_t)$$



Take-Home Message

TOS is (basically) alternated proximal-gradient and proximal-point

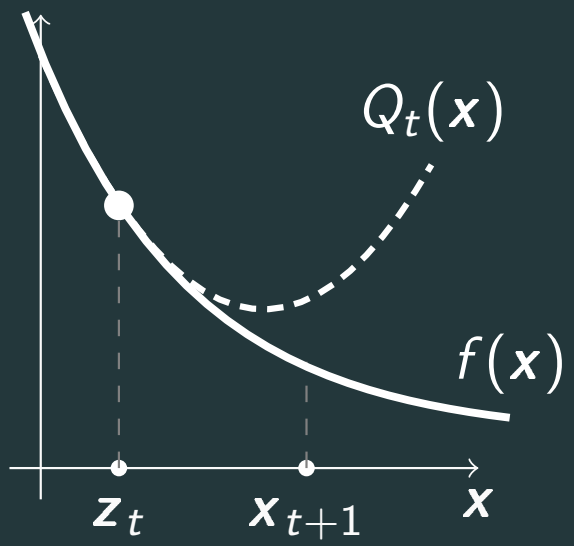
Can we use the adaptive step-size of proximal-gradient?

Adaptive TOS (Pedregosa and Gidel, 2018)

Let $Q_t(\boldsymbol{x}) \stackrel{\text{def}}{=} f(\boldsymbol{z}_t) + \langle \nabla f(\boldsymbol{z}_t), \boldsymbol{x} - \boldsymbol{z}_t \rangle + \frac{1}{2\gamma_t} \|\boldsymbol{x} - \boldsymbol{z}_t\|^2$.

Start with optimistic step-size γ_t and decrease it until:

$$f(\boldsymbol{x}_{t+1}) \leq Q(\boldsymbol{x}_{t+1}) \text{ with } \boldsymbol{x}_{t+1} = \mathbf{prox}_{\gamma_t g}(\boldsymbol{z}_t - \gamma_t(\nabla f(\boldsymbol{z}_t) + \boldsymbol{u}_t))$$

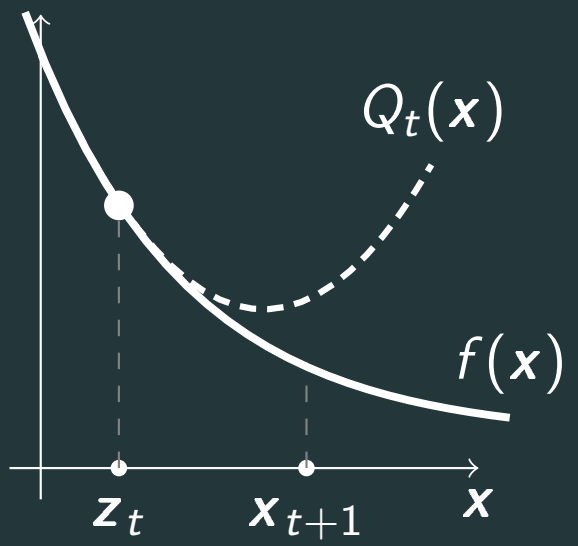


Adaptive TOS (Pedregosa and Gidel, 2018)

Let $Q_t(\boldsymbol{x}) \stackrel{\text{def}}{=} f(\boldsymbol{z}_t) + \langle \nabla f(\boldsymbol{z}_t), \boldsymbol{x} - \boldsymbol{z}_t \rangle + \frac{1}{2\gamma_t} \|\boldsymbol{x} - \boldsymbol{z}_t\|^2$.

Start with optimistic step-size γ_t and decrease it until:

$$f(\boldsymbol{x}_{t+1}) \leq Q(\boldsymbol{x}_{t+1}) \text{ with } \boldsymbol{x}_{t+1} = \mathbf{prox}_{\gamma_t g}(\boldsymbol{z}_t - \gamma_t(\nabla f(\boldsymbol{z}_t) + \boldsymbol{u}_t))$$

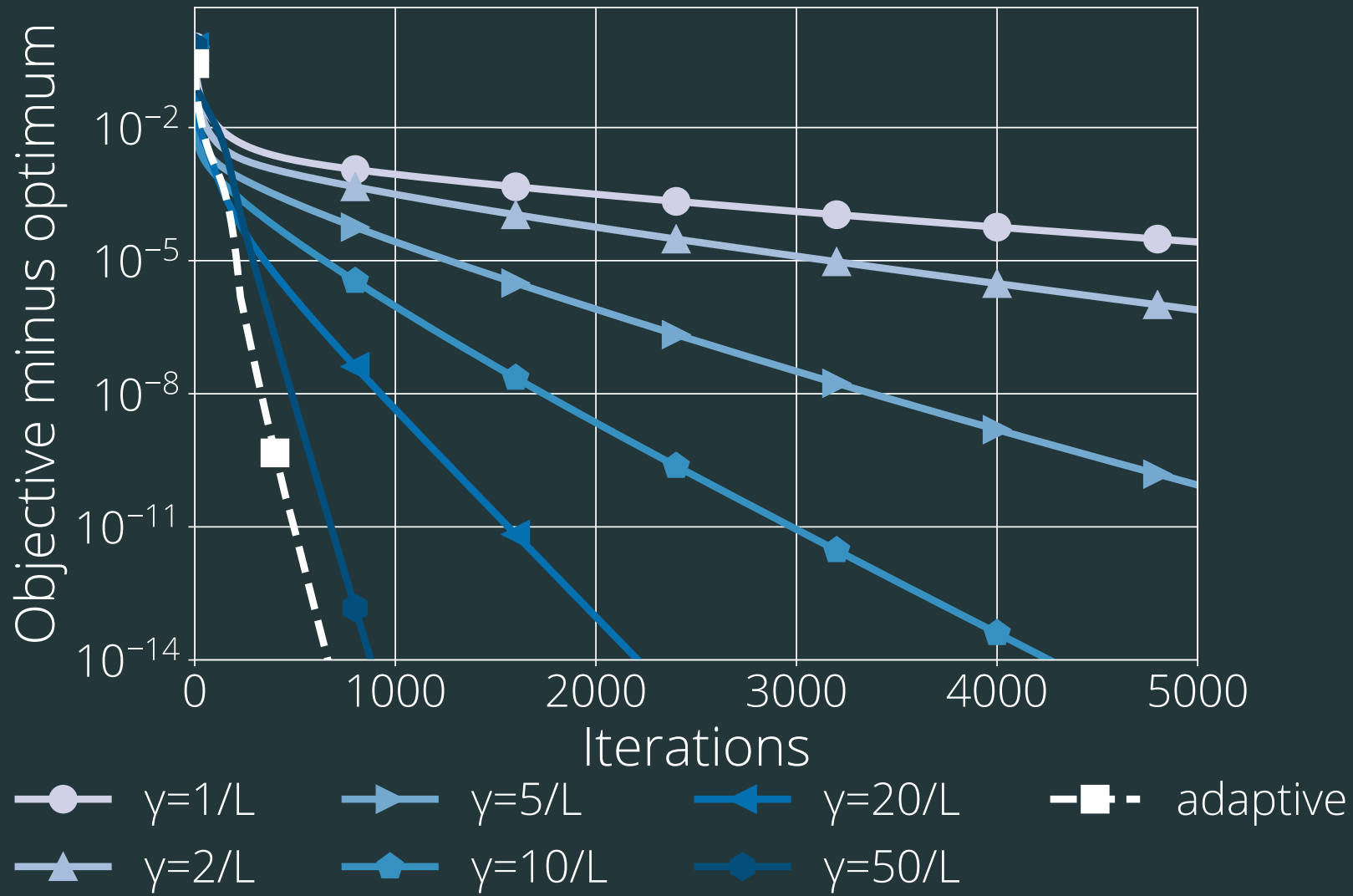


Run rest of algorithm with that step-size:

$$\boldsymbol{u}_{t+1} = \mathbf{prox}_{h^*/\gamma_t}(\boldsymbol{u}_t + \boldsymbol{x}_{t+1}/\gamma_t) \quad (1)$$

$$\boldsymbol{z}_{t+1} = \boldsymbol{x}_{t+1} - \gamma_t(\boldsymbol{u}_{t+1} - \boldsymbol{u}_t) \quad (2)$$

Performance of the adaptive step-size strategy



Performance is as good as best hand-tuned step-size

Convergence rates (informal)

As good as the original method with fixed step-size

Theorem (sublinear convergence rate)

For any $(\boldsymbol{x}, \boldsymbol{u}) \in dom\mathcal{L}$*:*

$$\mathcal{L}(\overline{\boldsymbol{x}}_t, \boldsymbol{u}) - \mathcal{L}(\boldsymbol{x}, \overline{\boldsymbol{u}}_t) \leq \tau L \frac{\|\boldsymbol{z}_0 - \boldsymbol{x}\|^2 + \gamma_0^2 \|\boldsymbol{u}_0 - \boldsymbol{u}\|^2}{2t}.$$

Theorem

If f *is* L_f*-smooth,* μ*-strongly convex and* h *is* L_h*-smooth then*

$$\|\boldsymbol{x}_{t+1} - \boldsymbol{x}^\star\|^2 \leq \left(1 - \min\left\{\tau \frac{\mu}{L_f}, \frac{1}{1 + \gamma_0 L_h}\right\}\right)^{t+1} C_0 \qquad (3)$$

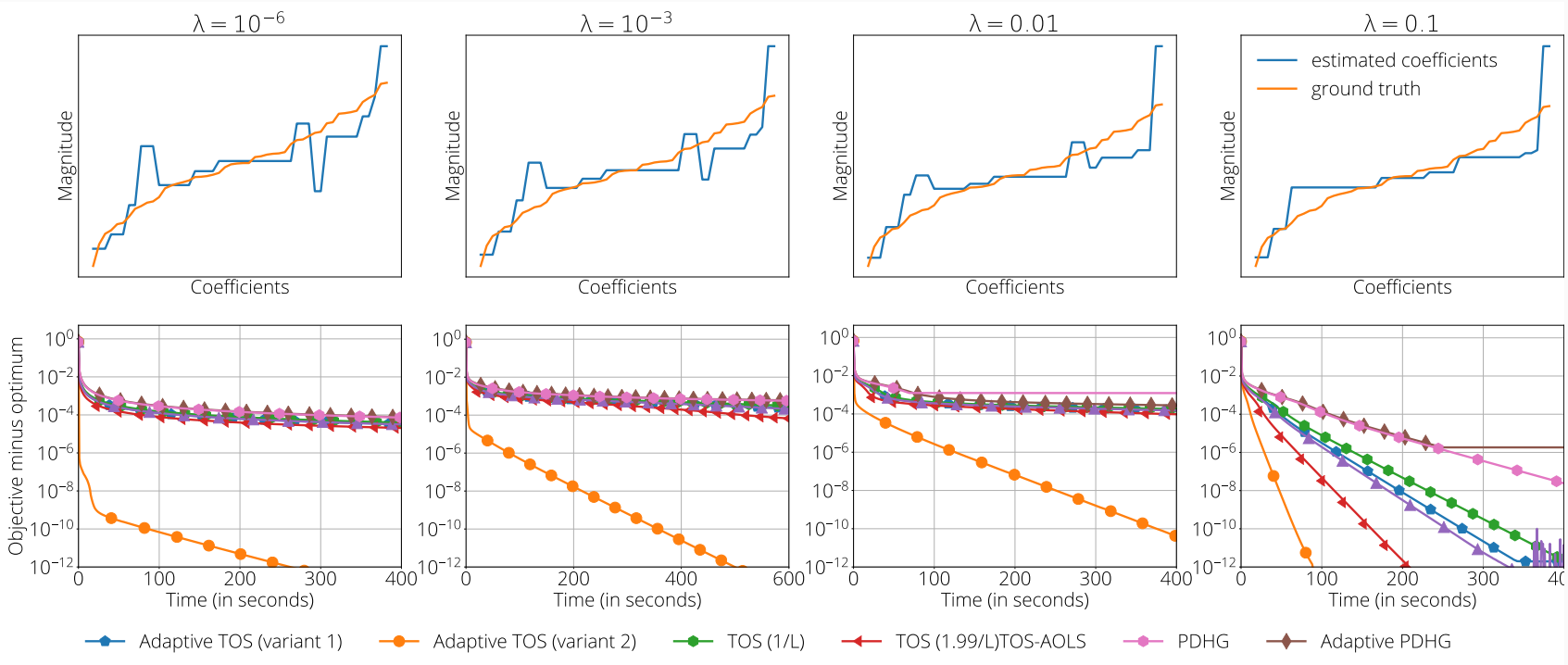
with $\tau =$ *line search decrease factor,* $C_0 =$ *only depends on initial conditions.*

Experiments

Logistic + Nearly-isotonic penalty

Problem

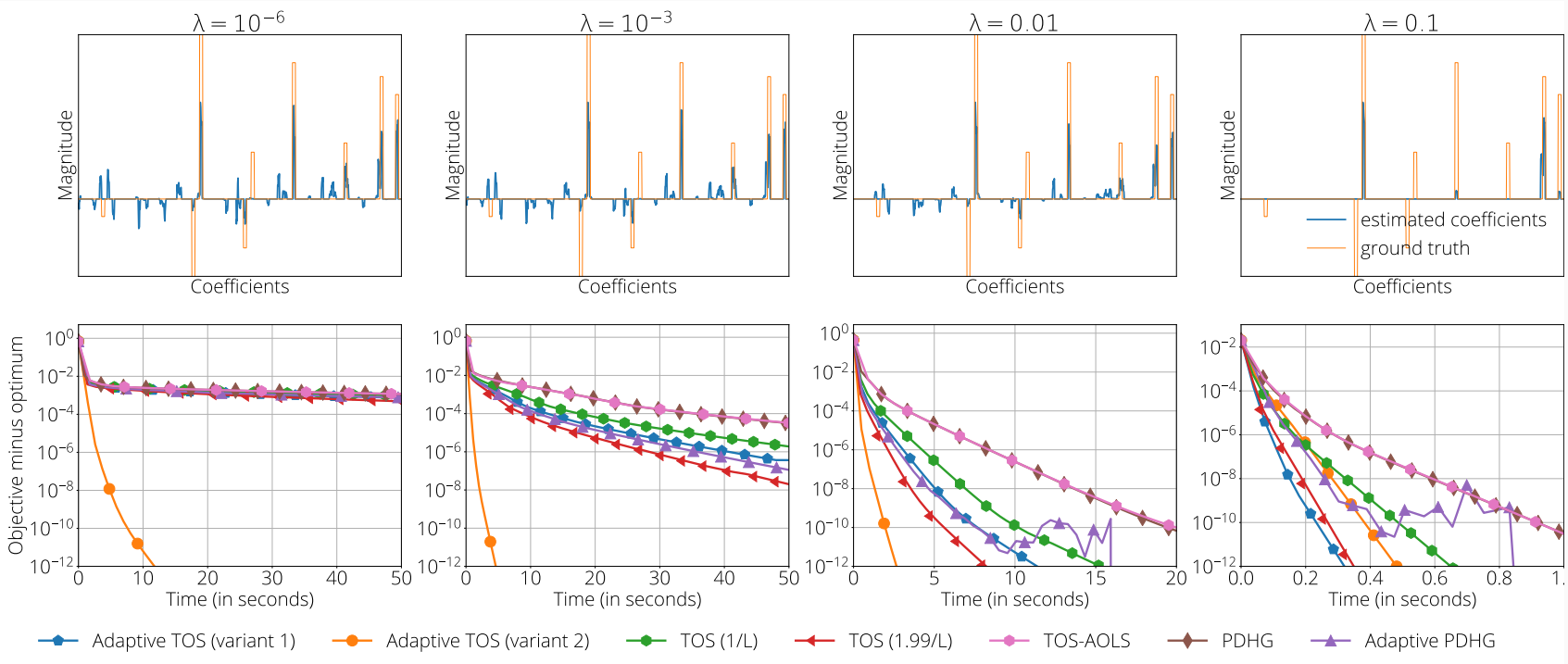
$$\mathbf{arg\,min}_{\boldsymbol{x}} \ \text{logistic}(\boldsymbol{x}) + \lambda \sum_{i=1}^{p-1} \max\{\boldsymbol{x}_i - \boldsymbol{x}_{i+1}, 0\}$$



Logistic + Overlapping group lasso penalty

Problem

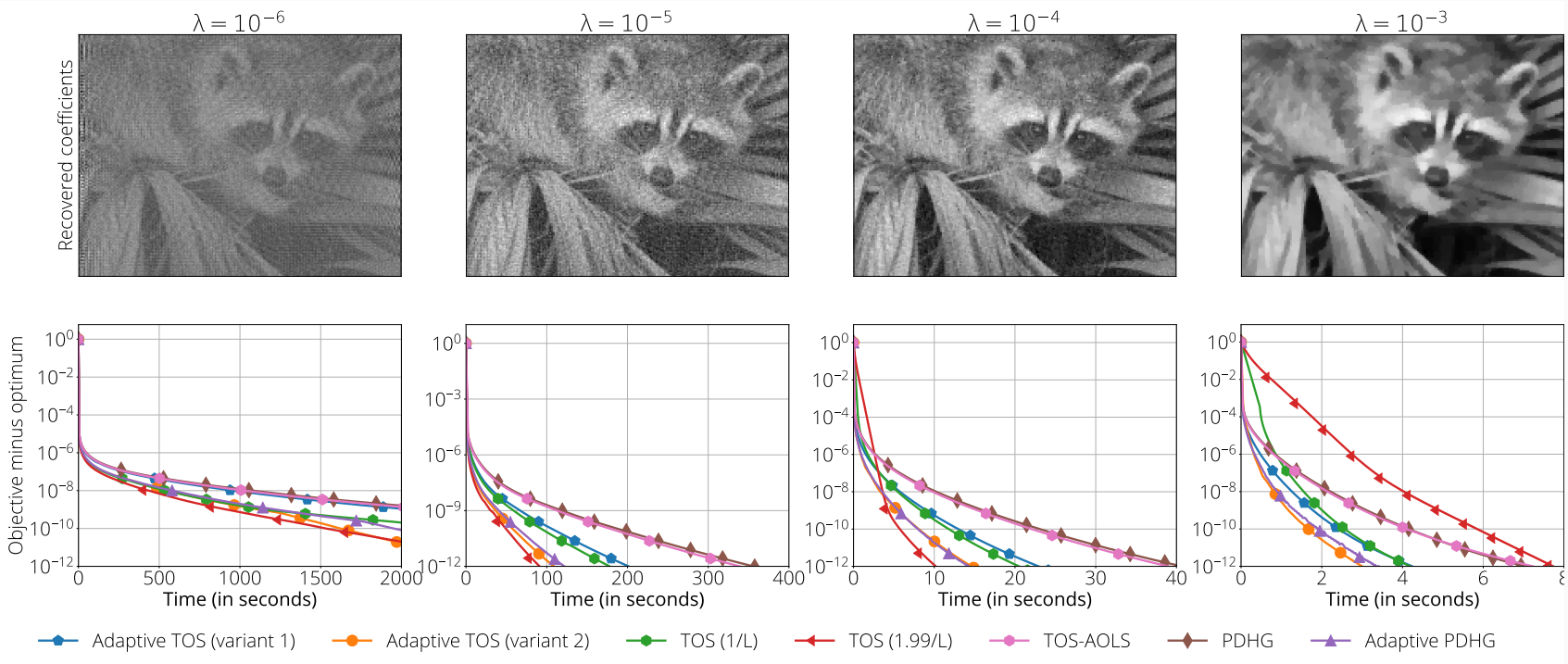
$$\arg\min_{x} \text{logistic}(x) + \lambda \sum_{g \in \mathcal{G}} \|[x]_g\|_2$$



Quadratic loss + total variation penalty

Problem

$$\underset{\boldsymbol{x}}{\textbf{arg min}}\ \text{least\_squares}(\boldsymbol{x}) + \lambda \|\boldsymbol{x}\|_{\text{TV}}$$



Adaptive Frank-Wolfe

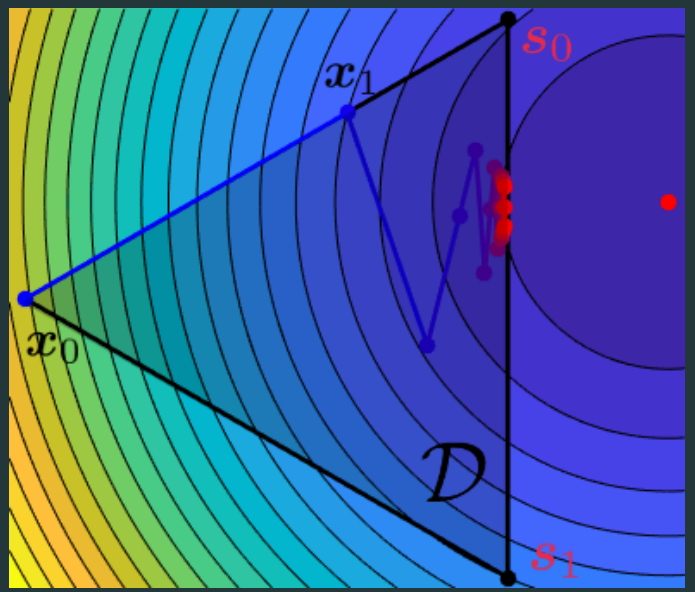
The Frank-Wolfe (FW) algorithm, aka conditional gradient

Problem: smooth f, compact $\mathcal{D}$

$$\arg\min_{x \in \mathcal{D}} f(x)$$

Algorithm 1: Frank-Wolfe (FW)

1 **for** $t = 0, 1 \ldots$ **do**
2 $\quad s_t \in \arg\min_{s \in \mathcal{D}} \langle \nabla f(x_t), s \rangle$
3 $\quad d_t = s_t - x_t$
4 $\quad$ Find γ_t, e.g., by line-search:
$\quad \gamma_t \in \arg\min_{\gamma \in [0,1]} f(x_t + \gamma d_t)$
5 $\quad x_{t+1} = x_t + \gamma d_t$



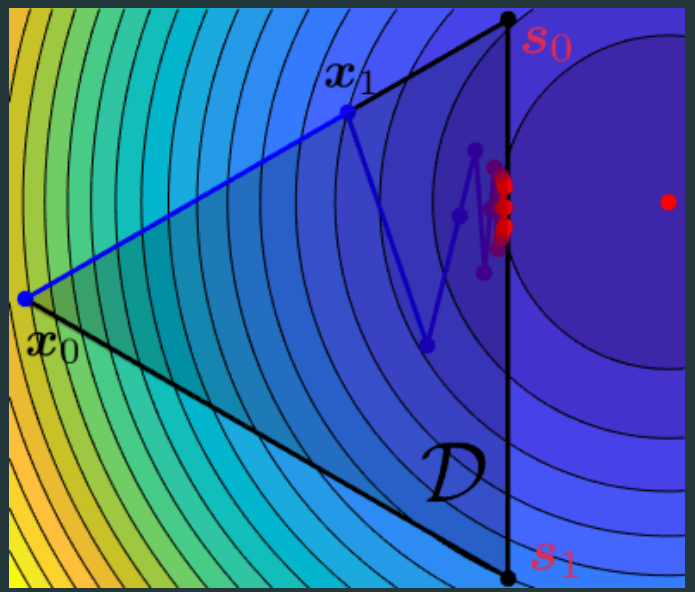
The Frank-Wolfe (FW) algorithm, aka conditional gradient

Problem: smooth f, compact $\mathcal{D}$

$$\arg\min_{x \in \mathcal{D}} f(x)$$

Algorithm 1: Frank-Wolfe (FW)

1 **for** $t = 0, 1 \ldots$ **do**
2 $\quad s_t \in \arg\min_{s \in \mathcal{D}} \langle \nabla f(x_t), s \rangle$
3 $\quad d_t = s_t - x_t$
4 $\quad$ Find γ_t, e.g., by line-search: $\gamma_t \in \arg\min_{\gamma \in [0,1]} f(x_t + \gamma d_t)$
5 $\quad x_{t+1} = x_t + \gamma d_t$



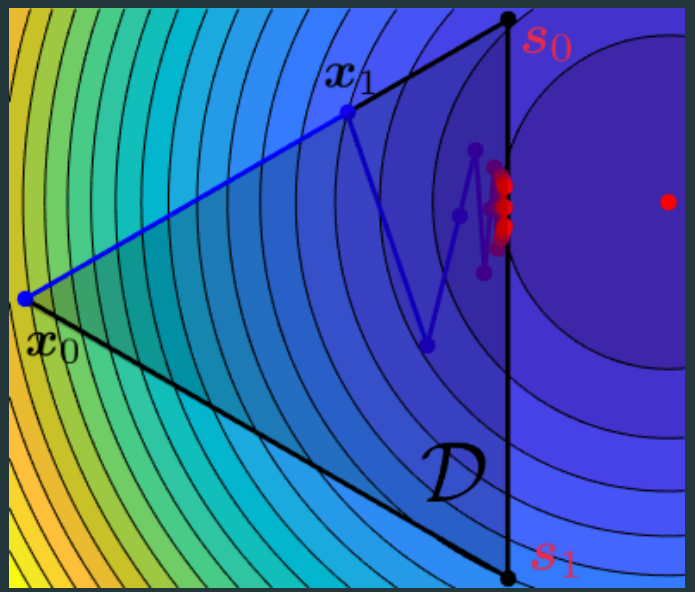
The Frank-Wolfe (FW) algorithm, aka conditional gradient

Problem: smooth f, compact $\mathcal{D}$

$$\arg\min_{x \in \mathcal{D}} f(x)$$

Algorithm 1: Frank-Wolfe (FW)

1 **for** $t = 0, 1 \ldots$ **do**
2 $\quad s_t \in \arg\min_{s \in \mathcal{D}} \langle \nabla f(x_t), s \rangle$
3 $\quad d_t = s_t - x_t$
4 $\quad$ Find γ_t, e.g., by line-search:
$\quad \gamma_t \in \arg\min_{\gamma \in [0,1]} f(x_t + \gamma d_t)$
5 $\quad x_{t+1} = x_t + \gamma d_t$



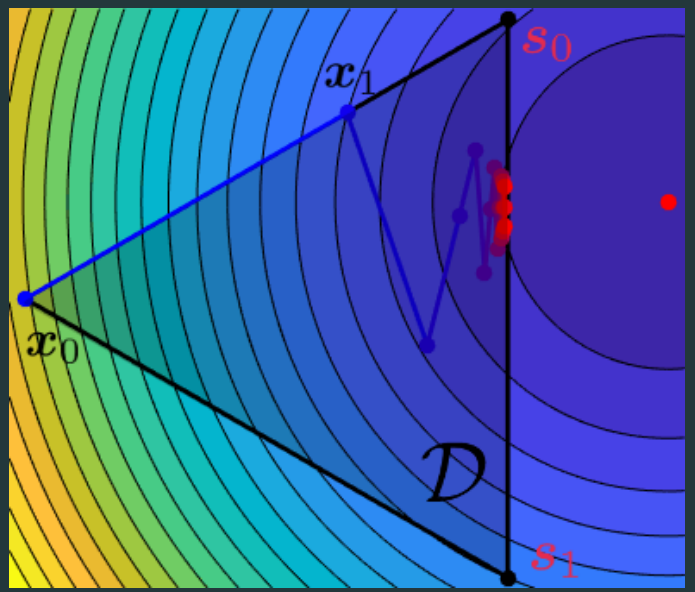
The Frank-Wolfe (FW) algorithm, aka conditional gradient

Problem: smooth f, compact $\mathcal{D}$

$$\arg\min_{x \in \mathcal{D}} f(x)$$

Algorithm 1: Frank-Wolfe (FW)

1 **for** $t = 0, 1 \ldots$ **do**
2 $\quad s_t \in \arg\min_{s \in \mathcal{D}} \langle \nabla f(x_t), s \rangle$
3 $\quad d_t = s_t - x_t$
4 $\quad$ Find γ_t, e.g., by line-search: $\gamma_t \in \arg\min_{\gamma \in [0,1]} f(x_t + \gamma d_t)$
5 $\quad x_{t+1} = x_t + \gamma d_t$



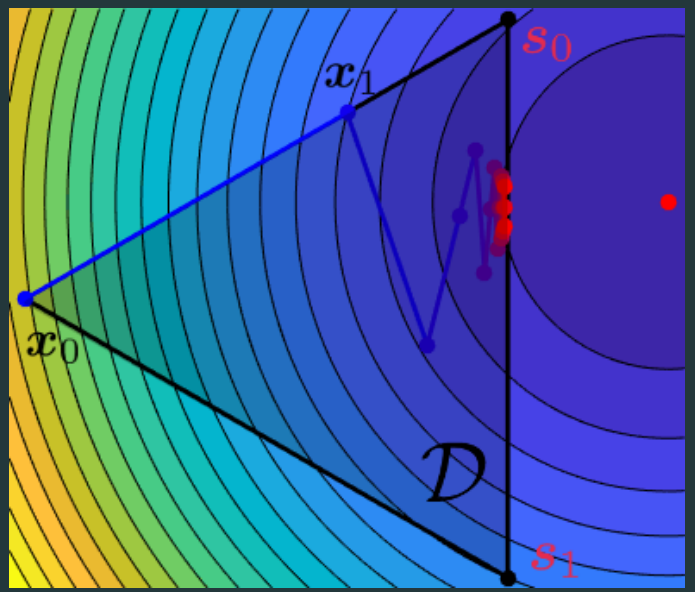
The Frank-Wolfe (FW) algorithm, aka conditional gradient

Problem: smooth f, compact $\mathcal{D}$

$$\arg\min_{x \in \mathcal{D}} f(x)$$

Algorithm 1: Frank-Wolfe (FW)

1 **for** $t = 0, 1 \ldots$ **do**
2 $\quad s_t \in \arg\min_{s \in \mathcal{D}} \langle \nabla f(x_t), s \rangle$
3 $\quad d_t = s_t - x_t$
4 $\quad$ Find γ_t, e.g., by line-search:
$\quad \gamma_t \in \arg\min_{\gamma \in [0,1]} f(x_t + \gamma d_t)$
5 $\quad x_{t+1} = x_t + \gamma d_t$



Setting the step-size in Frank-Wolfe

- Exact line-search only feasible for quadratic objective.

Setting the step-size in Frank-Wolfe

- Exact line-search only feasible for quadratic objective.
- "Oblivious" step-size $\gamma_t = 2/(t+2)$ is convergent, but slow.

Setting the step-size in Frank-Wolfe

- Exact line-search only feasible for quadratic objective.
- "Oblivious" step-size $\gamma_t = 2/(t+2)$ is convergent, but slow.
- New linearly-convergent variants (Lacoste-Julien and Jaggi, 2015) assume access to exact line search.

Setting the step-size in Frank-Wolfe

- Exact line-search only feasible for quadratic objective.
- "Oblivious" step-size $\gamma_t = 2/(t+2)$ is convergent, but slow.
- New linearly-convergent variants (Lacoste-Julien and Jaggi, 2015) assume access to exact line search.

We would like:

- Efficient.
- Adaptive to local geometry.
- That achives best possible rates in every situation.

Setting the step-size in Frank-Wolfe

- Exact line-search only feasible for quadratic objective.
- "Oblivious" step-size $\gamma_t = 2/(t+2)$ is convergent, but slow.
- New linearly-convergent variants (Lacoste-Julien and Jaggi, 2015) assume access to exact line search.

We would like:

- Efficient.
- Adaptive to local geometry.
- That achives best possible rates in every situation.

Is it possible?

A practical issue

- In a polytope, FW moves in the direction of a vertex.

A practical issue

- In a polytope, FW moves in the direction of a vertex.
- Two verteces can be far apart $\implies$ optimal step-size does not vary smoothly.

A practical issue

- In a polytope, FW moves in the direction of a vertex.
- Two verteces can be far apart $\implies$ optimal step-size does not vary smoothly.
- Directly using Armijo/Sufficient decrease conditions (Dunn, 1980) is particularly difficult because of this.

The Adaptive FW algorithm (Pedregosa et al., 2018)

Key Idea

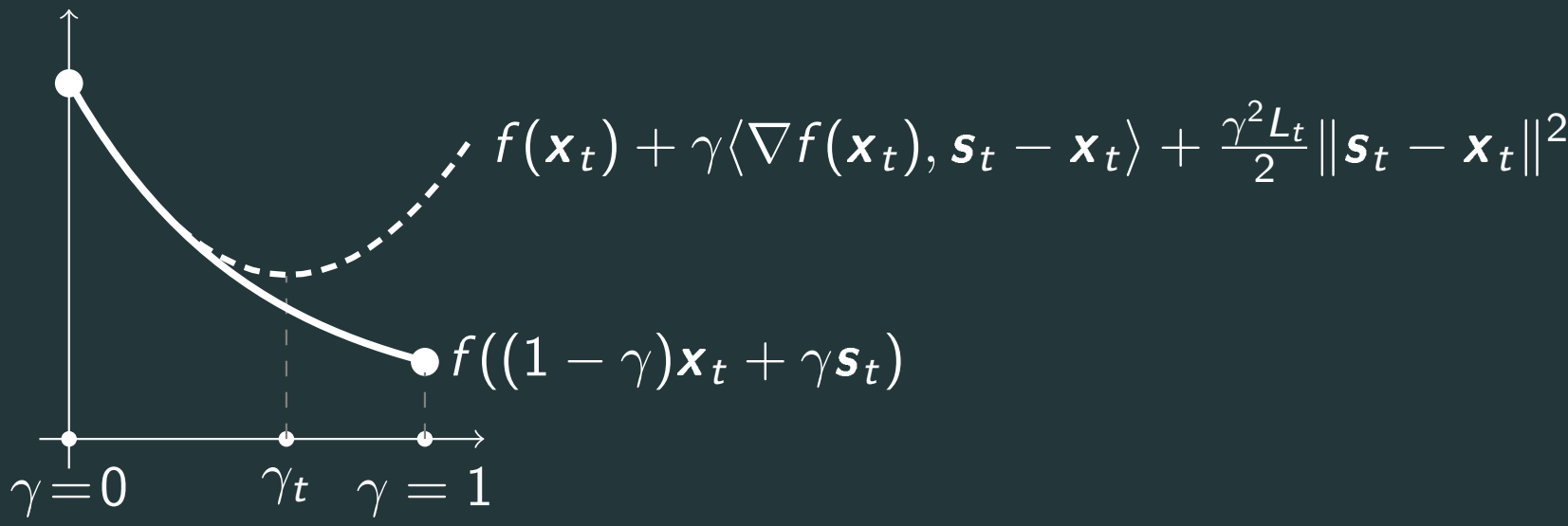
Estimate smoothness, not step-size!

Algorithm 2: The Adaptive Frank-Wolfe algorithm (AdaFW)

for $t = 0, 1 \dots$ **do**
- $\boldsymbol{s}_t \in \arg\min_{\boldsymbol{s} \in \mathcal{D}} \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{s} \rangle$
- $\boldsymbol{d}_t = \boldsymbol{s}_t - \boldsymbol{x}_t$
- Find L_t that verifies sufficient decrease (4), with
- $\gamma_t = \min \left\{ \frac{\langle -\nabla f(\boldsymbol{x}_t), \boldsymbol{d}_t \rangle}{L_t \|\boldsymbol{d}_t\|^2}, 1 \right\}$
- $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \gamma_t \boldsymbol{d}_t$

$$f(\boldsymbol{x}_{t+1}) \leq f(\boldsymbol{x}_t) + \gamma_t \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{s}_t - \boldsymbol{x}_t \rangle + \frac{\gamma_t^2 L_t}{2} \|\boldsymbol{s}_t - \boldsymbol{x}_t\|^2 \quad (4)$$

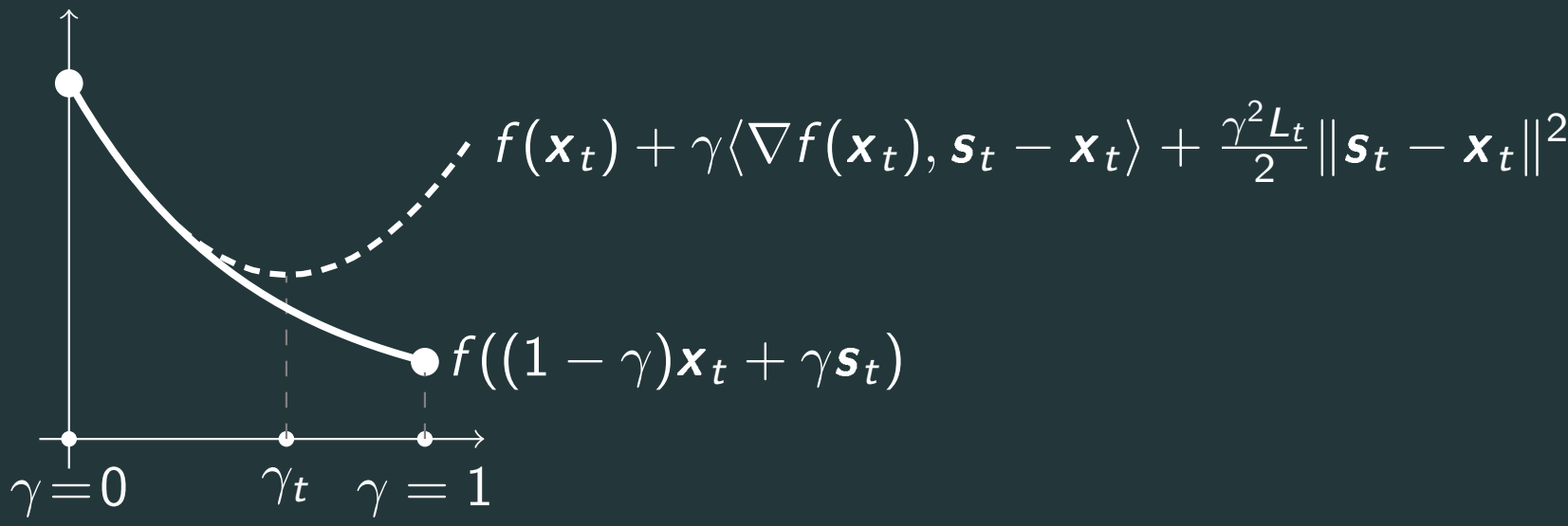
The Adaptive FW algorithm (Pedregosa et al., 2018)[1]



- Worst-case, $L_t = L$. Often $L_t \ll L \implies$ larger step-size.

[1] Fabian Pedregosa, Armin Askari, Geoffrey Negiar, and Martin Jaggi (2018). "Step-Size Adaptivity in Projection-Free Optimization". In: *ArXiv*.

The Adaptive FW algorithm (Pedregosa et al., 2018)[1]



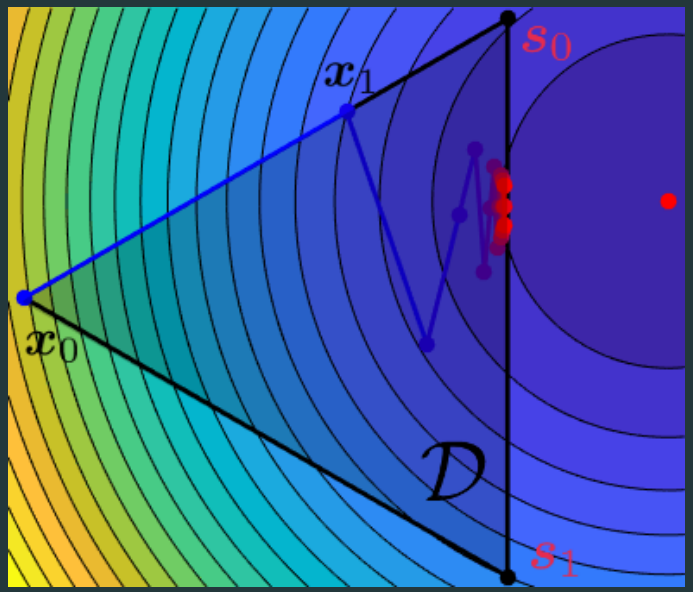
- Worst-case, $L_t = L$. Often $L_t \ll L \implies$ larger step-size.
- Two extra function evaluations per iteration. Often given as byproduct of gradient.

[1] Fabian Pedregosa, Armin Askari, Geoffrey Negiar, and Martin Jaggi (2018). "Step-Size Adaptivity in Projection-Free Optimization". In: *ArXiv*.

Extension to other FW variants

Zig-Zagging phenomena in FW

The Frank-Wolfe algorithm zig-zags when the solution lies in a face of the boundary.



Some FW variants have been developed to address this issue.

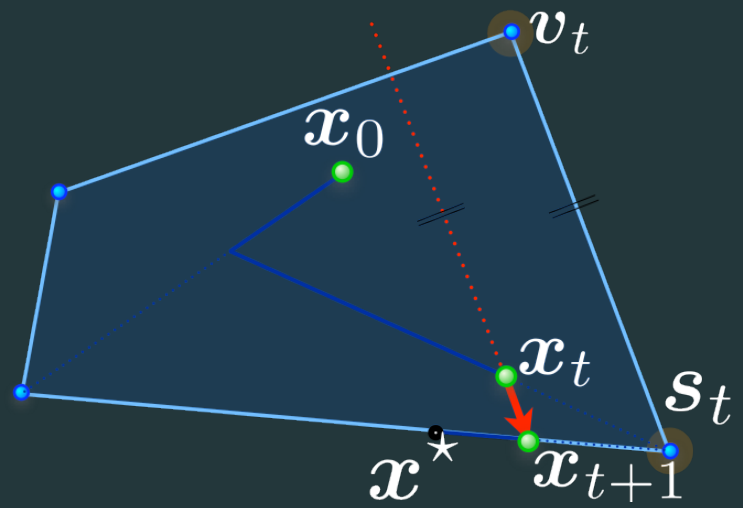
Pairwise FW

Key idea

1. Keep track of previously added vertices (activeset) $\mathcal{S}_t$.
2. Move weight mass between two vertices in each step.

Algorithm 3: Pairwise FW (Lacoste-Julien and Jaggi, 2015)

for $t = 0, 1 \ldots$ **do**
- $\boldsymbol{s}_t \in \arg\min_{\boldsymbol{s} \in \mathcal{D}} \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{s} \rangle$
- $\boldsymbol{v}_t \in \arg\max_{\boldsymbol{s} \in \mathcal{S}_t} \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{s} \rangle$
- $\boldsymbol{d}_t = \boldsymbol{s}_t - \boldsymbol{v}_t$
- Find γ_t by line-search: $\gamma_t \in \arg\min_{\gamma \in [0, \gamma_t^{\max}]} f(\boldsymbol{x}_t + \gamma \boldsymbol{d}_t)$
- $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \gamma_t \boldsymbol{d}_t$



Adaptive Away-steps and Pairwise FW (Pedregosa et al., 2018)

Convergence of Away-steps and Pairwise FW

- Linear convergence for strongly convex functions on polytopes (Lacoste-Julien and Jaggi, 2015).

Adaptive Away-steps and Pairwise FW (Pedregosa et al., 2018)

Convergence of Away-steps and Pairwise FW

- Linear convergence for strongly convex functions on polytopes (Lacoste-Julien and Jaggi, 2015).
- Can we design variants with adaptive step-size?

Adaptive Away-steps and Pairwise FW (Pedregosa et al., 2018)

Convergence of Away-steps and Pairwise FW

- Linear convergence for strongly convex functions on polytopes (Lacoste-Julien and Jaggi, 2015).
- Can we design variants with adaptive step-size?

Introducing Adaptive Away-steps and Adaptive Pairwise

Choose L_t such that it verifies

$$f(\boldsymbol{x}_t + \gamma_t \boldsymbol{d}_t) \leq f(\boldsymbol{x}_t) + \gamma_t \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{d}_t \rangle + \frac{\gamma_t^2 L_t}{2} \|\boldsymbol{d}_t\|^2$$

$$\text{with } \gamma_t = \min \left\{ \frac{\langle -\nabla f(\boldsymbol{x}_t), \boldsymbol{d}_t \rangle}{L_t \|\boldsymbol{d}_t\|^2}, \gamma_{\max} \right\}$$

Theory for Adaptive Step-size variants

Strongly convex f

Pairwise and Away-steps converge linearly on a polytope. For each "good step" we have:

$$f(\boldsymbol{x}_{t+1}) - f(\boldsymbol{x}^\star) \leq (1-\rho)(f(\boldsymbol{x}_t) - f(\boldsymbol{x}^\star)) \ , \quad \rho > 0$$

Theory for Adaptive Step-size variants

Strongly convex f

Pairwise and Away-steps converge linearly on a polytope. For each "good step" we have:

$$f(\boldsymbol{x}_{t+1}) - f(\boldsymbol{x}^\star) \leq (1-\rho)(f(\boldsymbol{x}_t) - f(\boldsymbol{x}^\star)) \ , \quad \rho > 0$$

Convex f

For all FW variants, $f(\boldsymbol{x}_t) - f(\boldsymbol{x}^\star) \leq \mathcal{O}(1/t)$

Theory for Adaptive Step-size variants

Strongly convex f

Pairwise and Away-steps converge linearly on a polytope. For each "good step" we have:

$$f(\boldsymbol{x}_{t+1}) - f(\boldsymbol{x}^\star) \leq (1-\rho)(f(\boldsymbol{x}_t) - f(\boldsymbol{x}^\star)) \ , \quad \rho > 0$$

Convex f

For all FW variants, $f(\boldsymbol{x}_t) - f(\boldsymbol{x}^\star) \leq \mathcal{O}(1/t)$

Non-Convex f

For all FW variants, $\max_{\boldsymbol{s}\in\mathcal{D}} \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{x}_t - \boldsymbol{s} \rangle \leq \mathcal{O}(1/\sqrt{t})$

Same rate as with exact line search

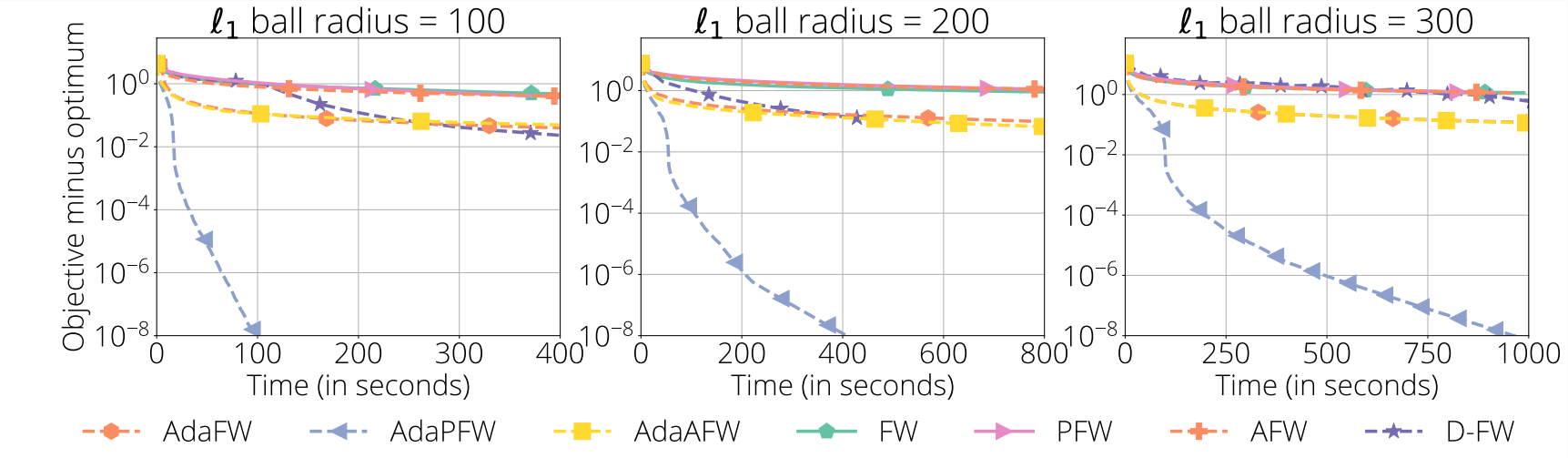
Experiments

Experiments RCV1

Problem: ℓ_1-constrained logistic regression

$$\underset{\|\boldsymbol{x}\|_1 \leq \alpha}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \varphi(\boldsymbol{a}_i^T \boldsymbol{x}, b_i) \quad \text{with } \varphi = \text{logistic loss.}$$

Dataset	dimension	density	$\overline{L}_t / L$
RCV1	47236	10^{-3}	1.3×10^{-2}

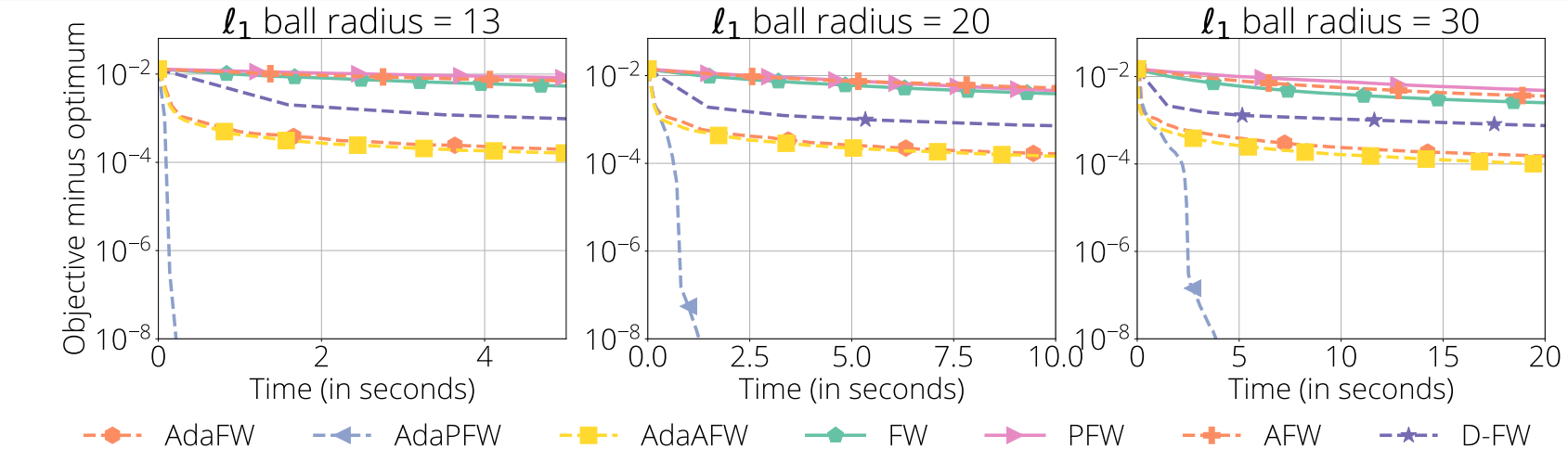


Experiments Madelon

Problem: ℓ_1-constrained logistic regression

$$\underset{\|\boldsymbol{x}\|_1 \leq \alpha}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \varphi(\boldsymbol{a}_i^T \boldsymbol{x}, b_i) \quad \text{with } \varphi = \text{logistic loss.}$$

Dataset	dimension	density	$\overline{L}_t / L$
Madelon	500	1.	3.3×10^{-3}

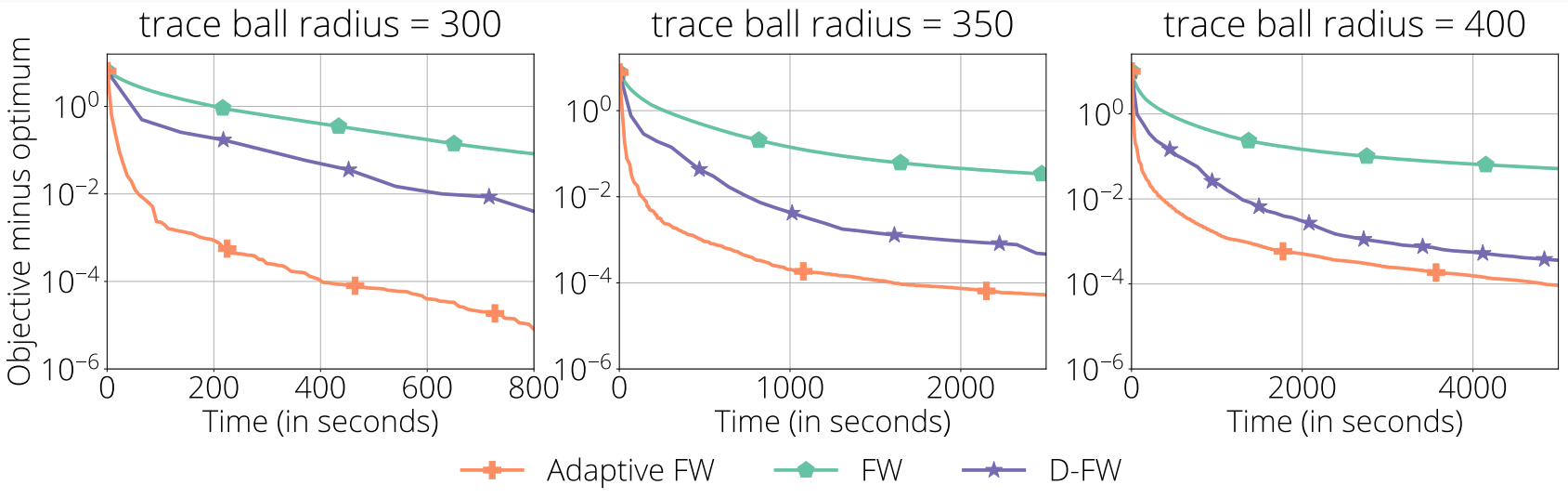


Experiments MovieLens 1M

Problem: trace-norm constrained robust matrix completion

$$\underset{\|\boldsymbol{x}\|_* \leq \alpha}{\arg\min} \frac{1}{|B|} \sum_{(i,j)\in B}^{n} h(\boldsymbol{X}_{i,j}, \boldsymbol{A}_{i,j}) \quad \text{with } h = \text{Huber loss.}$$

Dataset	dimension	density	$\bar{L}_t/L$
MovieLens 1M	22,393,987	0.04	1.1×10^{-2}



Perspectives

Stochastic optimization

Problem

$$\arg\min_{\boldsymbol{x}\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{x})$$

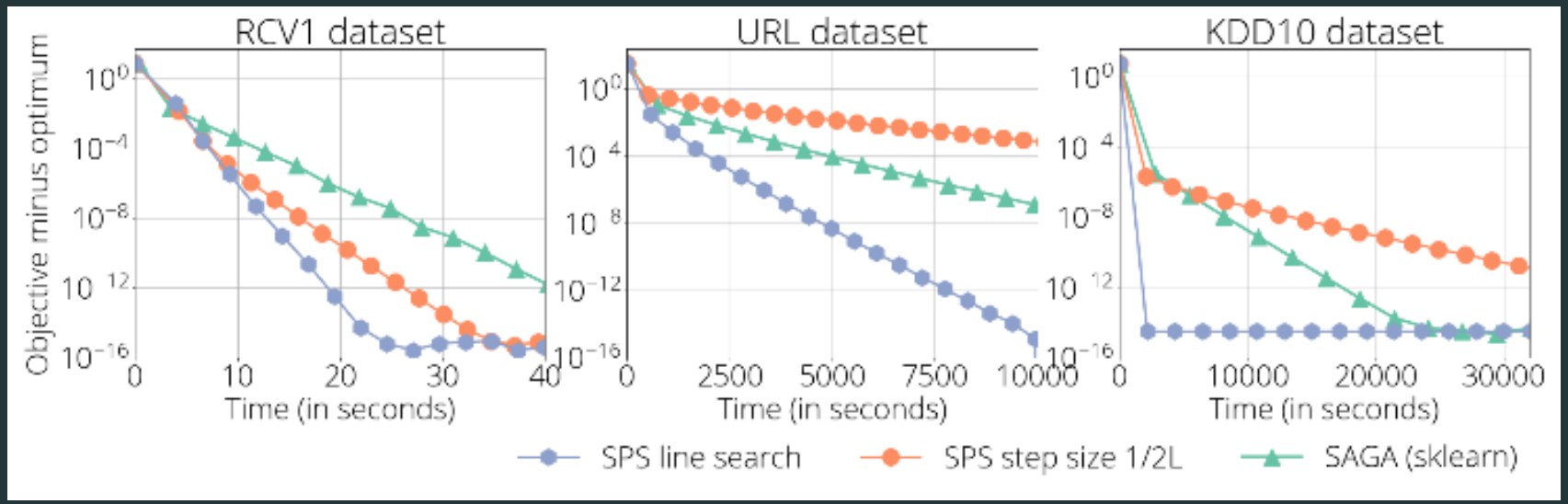
Main challenge: How to evaluate Armijo condition $f(\boldsymbol{x}_{t+1}) \leq f(\boldsymbol{x}_t) - \gamma\|\boldsymbol{p}_t\|^2$ without access to f?

Experiments stochastic line search

Heuristic from (Schmidt et al. 2017)[2] to estimate L:

$$f_i(\boldsymbol{x}_t - \frac{1}{L}\nabla f_i(\boldsymbol{x}_t)) \leq f_i(\boldsymbol{x}_t) - \frac{1}{2L}\|\nabla f_i(\boldsymbol{x}_t)\|^2$$

with i random index sampled at iter t.



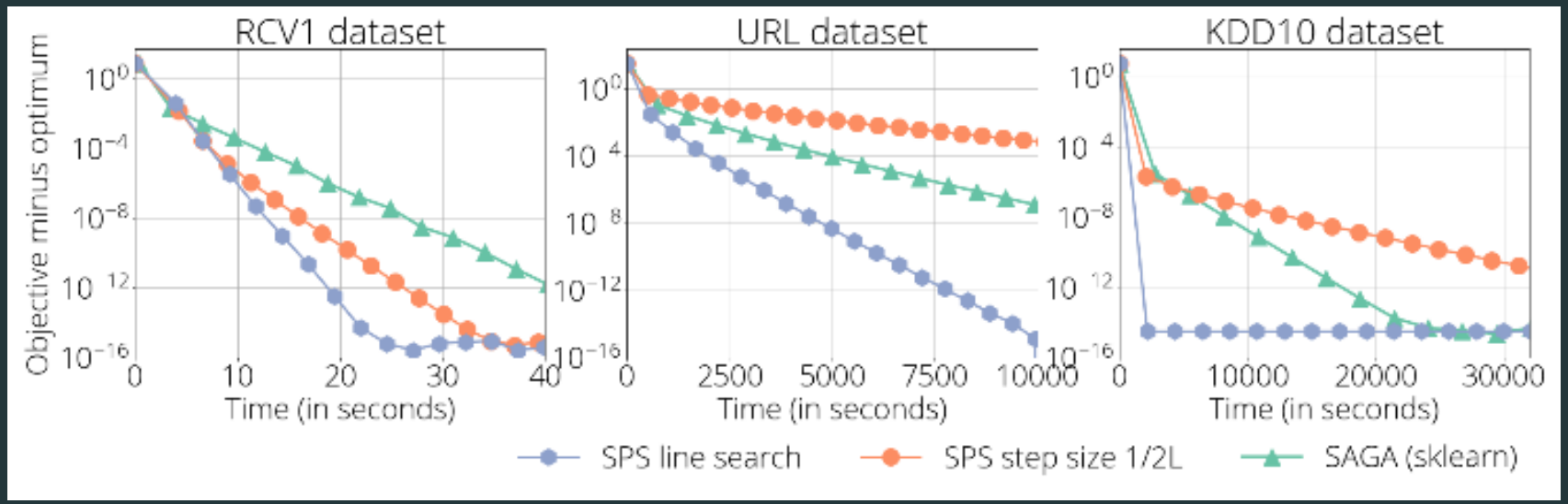
[2]Mark Schmidt, Nicolas Le Roux, and Francis Bach (2017). "Minimizing finite sums with the stochastic average gradient". In: *Mathematical Programming*.

Experiments stochastic line search

Heuristic from (Schmidt et al. 2017)[2] to estimate L:

$$f_i(\boldsymbol{x}_t - \frac{1}{L}\nabla f_i(\boldsymbol{x}_t)) \leq f_i(\boldsymbol{x}_t) - \frac{1}{2L}\|\nabla f_i(\boldsymbol{x}_t)\|^2$$

with i random index sampled at iter t.



Can we prove convergence of this (or similar) method?

[2]Mark Schmidt, Nicolas Le Roux, and Francis Bach (2017). "Minimizing finite sums with the stochastic average gradient". In: *Mathematical Programming*.

Recent developments

- (Shang et al., 2018) Adaptive step-size for SVRG. Condition is costly to evaluate.

Recent developments

- (Shang et al., 2018) Adaptive step-size for SVRG. Condition is costly to evaluate.
- (Paquette and Scheinberg, 2018) Evaluates a stochastic version of the Armijo condition. Accepts step-size based on concept of reliable/unreliable estimate.

Recent developments

- (Shang et al., 2018) Adaptive step-size for SVRG. Condition is costly to evaluate.
- (Paquette and Scheinberg, 2018) Evaluates a stochastic version of the Armijo condition. Accepts step-size based on concept of reliable/unreliable estimate.
  (step-size tends to be really small)

Conclusion

- Applicability of sufficient decrease beyond classical framework.

Conclusion

- Applicability of sufficient decrease beyond classical framework.
- Sufficient decrease condition to set step-size in TOS and FW and variants.

Conclusion

- Applicability of sufficient decrease beyond classical framework.
- Sufficient decrease condition to set step-size in TOS and FW and variants.
  - Faster
  - (Mostly) Hyperparameter-free.

Conclusion

- Applicability of sufficient decrease beyond classical framework.
- Sufficient decrease condition to set step-size in TOS and FW and variants.
  - Faster
  - (Mostly) Hyperparameter-free.
- Perspectives in stochastic optimization.

Thanks for your attention

References

- Armijo, Larry (1966). "Minimization of functions having Lipschitz continuous first partial derivatives". In: *Pacific Journal of Mathematics*.
- Davis, Damek and Wotao Yin (2017). "A three-operator splitting scheme and its optimization applications". In: *Set-valued and variational analysis*.
- Dunn, Joseph C (1980). "Convergence rates for conditional gradient sequences generated by implicit step length rules". In: *SIAM Journal on Control and Optimization*.
- Lacoste-Julien, Simon and Martin Jaggi (2015). "On the global linear convergence of Frank-Wolfe optimization variants". In: *Advances in Neural Information Processing Systems*.
- Paquette, Courtney and Katya Scheinberg (2018). "A stochastic line search method with convergence rate analysis". In: *arXiv preprint arXiv:1807.07994*.
- Pedregosa, Fabian et al. (2018). "Step-Size Adaptivity in Projection-Free Optimization". In: *ArXiv*.


Pedregosa, Fabian and Gauthier Gidel (2018). "Adaptive Three Operator Splitting". In: *Proceedings of the 35th International Conference on Machine Learning*.


Schmidt, Mark, Nicolas Le Roux, and Francis Bach (2017). "Minimizing finite sums with the stochastic average gradient". In: *Mathematical Programming*.


Shang, Fanhua et al. (2018). "Guaranteed Sufficient Decrease for Stochastic Variance Reduced Gradient Optimization". In: *arXiv preprint arXiv:1802.09933*.