Efficient Online Learning with Memory via Frank-Wolfe Optimization: Algorithms with Bounded Dynamic Regret and Applications to Control

Hongyu Zhou, Zirui Xu, Vasileios Tzoumas



MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

Examples of Online Learning Problems

Target Tracking

Goal: Minimize distance to a target that moves in a cluttered environment.

Complication: Unpredictable disturbances.

Problem: How can the drone efficiently choose optimal control actions?[1]

[1] Chen, Liu, Shen, IROS' 16

Examples of Online Learning Problems

Background Extraction

Goal: Extraction background in frames of a video.

Complication: Unpredictable moving objects.

Problem: How can we efficiently recover background in all frames?[1]

[1] Kalhan et al., TSP '21

Examples of Online Learning Problems

Video Denoising

Goal: Removing noise from frames of a video.

Complication: Unpredictable noise.

Problem: How can we efficiently remove noise in all frames?[1]

[1] Ji et al., CVPR '10

Online Convex Optimization with Memory

All above scenarios take the form of Online Convex Optimization with Memory (OCO-M)

Online Convex Optimization with Memory

All above scenarios take the form of Online Convex Optimization with Memory (OCO-M)

OCO-M is a sequential game between an optimizer and an adversary:

Online Convex Optimization with Memory

All above scenarios take the form of Online Convex Optimization with Memory (OCO-M)

OCO-M is a sequential game between an optimizer and an adversary:

Initialization: the optimizer is given a convex domain set $\mathcal{X}$;

Online Convex Optimization with Memory

All above scenarios take the form of Online Convex Optimization with Memory (OCO-M)

OCO-M is a sequential game between an optimizer and an adversary:

Initialization: the optimizer is given a convex domain set $\mathcal{X}$;
At each iteration $t = 1, \ldots, T$:

- the optimizer chooses a decision $\mathbf{x}_t \in \mathcal{X}$;

Control input

Online Convex Optimization with Memory

All above scenarios take the form of Online Convex Optimization with Memory (OCO-M)

OCO-M is a sequential game between an optimizer and an adversary:

Initialization: the optimizer is given a convex domain set $\mathcal{X}$;
At each iteration $t = 1, \ldots, T$:

- the optimizer chooses a decision $\mathbf{x}_t \in \mathcal{X}$; → Control input
- the adversary chooses a convex loss function $f_t : \mathcal{X}^t \to \mathbb{R}$; → Noise realization

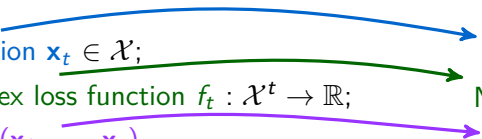
Online Convex Optimization with Memory

All above scenarios take the form of Online Convex Optimization with Memory (OCO-M)

OCO-M is a sequential game between an optimizer and an adversary:

Initialization: the optimizer is given a convex domain set $\mathcal{X}$;
At each iteration $t = 1, \ldots, T$:

- the optimizer chooses a decision $\mathbf{x}_t \in \mathcal{X}$;
- the adversary chooses a convex loss function $f_t : \mathcal{X}^t \to \mathbb{R}$;
- the optimizer suffers a loss $f_t(\mathbf{x}_1, \ldots, \mathbf{x}_t)$.



Online Convex Optimization with Memory

All above scenarios take the form of Online Convex Optimization with Memory (OCO-M)

OCO-M is a sequential game between an optimizer and an adversary:

Initialization: the optimizer is given a convex domain set $\mathcal{X}$;
At each iteration $t = 1, \ldots, T$:

- the optimizer chooses a decision $\mathbf{x}_t \in \mathcal{X}$;
- the adversary chooses a convex loss function $f_t : \mathcal{X}^t \to \mathbb{R}$;
- the optimizer suffers a loss $f_t(\mathbf{x}_1, \ldots, \mathbf{x}_t)$.

Control input

Noise realization

Tracking error
depends on past decisions

Goal: The optimizer aims to minimize cumulative loss, *i.e.*, $\sum_{t=1}^{T} f_t(\mathbf{x}_1, \ldots, \mathbf{x}_t)$.

Online Convex Optimization with Memory

Problem

Two players, an online optimizer and an adversary, choose decisions sequentially over a time horizon T. At each time step $t = 1, \ldots, T$:

- *the optimizer first chooses a decision $\mathbf{x}_t$ from a known convex set $\mathcal{X}$;*
- *then, the adversary chooses a convex loss f_t to penalize the optimizer's decision;*
- *the optimizer suffers a loss $f_t(\mathbf{x}_{t-m}, \ldots, \mathbf{x}_t)$.*

The optimizer aims to minimize $\sum_{t=1}^{T} f_t(\mathbf{x}_{t-m}, \ldots, \mathbf{x}_t)$.

Online Convex Optimization with Memory

Problem

Two players, an online optimizer and an adversary, choose decisions sequentially over a time horizon T. At each time step $t = 1, \ldots, T$:

- *the optimizer first chooses a decision $\mathbf{x}_t$ from a known convex set $\mathcal{X}$;*
- *then, the adversary chooses a convex loss f_t to penalize the optimizer's decision;*
- *the optimizer suffers a loss $f_t(\mathbf{x}_{t-m}, \ldots, \mathbf{x}_t)$.*

The optimizer aims to minimize $\sum_{t=1}^{T} f_t(\mathbf{x}_{t-m}, \ldots, \mathbf{x}_t)$.

Challenges:

- The optimizer gets to know f_t only after $\mathbf{x}_t$ has been chosen.
- The loss function depends on the past m decisions.
- State-of-the-art methods rely on projection, which can be computationally expensive.

Online Convex Optimization with Memory

Problem

Two players, an online optimizer and an adversary, choose decisions sequentially over a time horizon T. At each time step $t = 1, \ldots, T$:

- *the optimizer first chooses a decision $\mathbf{x}_t$ from a known convex set $\mathcal{X}$;*
- *then, the adversary chooses a convex loss f_t to penalize the optimizer's decision;*
- *the optimizer suffers a loss $f_t(\mathbf{x}_{t-m}, \ldots, \mathbf{x}_t)$.*

The optimizer aims to minimize $\sum_{t=1}^{T} f_t(\mathbf{x}_{t-m}, \ldots, \mathbf{x}_t)$.

Challenges:

- The optimizer gets to know f_t only after $\mathbf{x}_t$ has been chosen.
- The loss function depends on the past m decisions.
- State-of-the-art methods rely on projection, which can be computationally expensive.

Goal: Develop a projection-free algorithm for OCO-M with bounded dynamic regret.

Standard Assumptions in OCO-M

Assumption (Convex and Compact Bounded Domain Containing the Origin)

$\mathcal{X}$ is convex and compact, $\mathbf{0} \in \mathcal{X}$, and such that $\|\mathbf{x} - \mathbf{y}\| \leq D$ for all $\mathbf{x} \in \mathcal{X}, \mathbf{y} \in \mathcal{X}$.

Standard Assumptions in OCO-M

Assumption (Convex and Compact Bounded Domain Containing the Origin)

$\mathcal{X}$ is convex and compact, $\mathbf{0} \in \mathcal{X}$, and such that $\|\mathbf{x} - \mathbf{y}\| \leq D$ for all $\mathbf{x} \in \mathcal{X}, \mathbf{y} \in \mathcal{X}$.

Assumption (Convex Loss)

$\widetilde{f}_t(\mathbf{x}) \triangleq f_t(\mathbf{x}, \ldots, \mathbf{x}) : \mathcal{X}^{m+1} \to \mathbb{R}$ is convex in $\mathbf{x}$ for all t.

Standard Assumptions in OCO-M

Assumption (Convex and Compact Bounded Domain Containing the Origin)

$\mathcal{X}$ is convex and compact, $\mathbf{0} \in \mathcal{X}$, and such that $\|\mathbf{x} - \mathbf{y}\| \leq D$ for all $\mathbf{x} \in \mathcal{X}, \mathbf{y} \in \mathcal{X}$.

Assumption (Convex Loss)

$\widetilde{f}_t(\mathbf{x}) \triangleq f_t(\mathbf{x}, \ldots, \mathbf{x}) : \mathcal{X}^{m+1} \to \mathbb{R}$ is convex in $\mathbf{x}$ for all t.

Assumption (Bounded Gradient)

$\left\|\nabla \widetilde{f}_t(\mathbf{x})\right\|_2 \leq G$ for all $\mathbf{x} \in \mathcal{X}$ and t, where G is given.

Standard Assumptions in OCO-M

Assumption (Convex and Compact Bounded Domain Containing the Origin)

$\mathcal{X}$ *is convex and compact,* $\mathbf{0} \in \mathcal{X}$*, and such that* $\|\mathbf{x} - \mathbf{y}\| \leq D$ *for all* $\mathbf{x} \in \mathcal{X}, \mathbf{y} \in \mathcal{X}$*.*

Assumption (Convex Loss)

$\widetilde{f}_t(\mathbf{x}) \triangleq f_t(\mathbf{x}, \ldots, \mathbf{x}) : \mathcal{X}^{m+1} \to \mathbb{R}$ *is convex in* $\mathbf{x}$ *for all* t*.*

Assumption (Bounded Gradient)

$\left\|\nabla \widetilde{f}_t(\mathbf{x})\right\|_2 \leq G$ *for all* $\mathbf{x} \in \mathcal{X}$ *and* t*, where* G *is given.*

Assumption (Bounded Loss)

$0 \leq a \leq f_t(\mathbf{x}_0, \ldots, \mathbf{x}_m) \leq a + c$*, for all* $(\mathbf{x}_0, \ldots, \mathbf{x}_m) \in \mathcal{X}^{m+1}$ *and* t*, where* a*,* c *are given.*

Standard Assumptions in OCO-M

Assumption (Convex and Compact Bounded Domain Containing the Origin)

$\mathcal{X}$ *is convex and compact,* $\mathbf{0} \in \mathcal{X}$*, and such that* $\|\mathbf{x} - \mathbf{y}\| \leq D$ *for all* $\mathbf{x} \in \mathcal{X}, \mathbf{y} \in \mathcal{X}$.

Assumption (Convex Loss)

$\widetilde{f}_t(\mathbf{x}) \triangleq f_t(\mathbf{x}, \ldots, \mathbf{x}) : \mathcal{X}^{m+1} \to \mathbb{R}$ *is convex in* $\mathbf{x}$ *for all* t.

Assumption (Bounded Gradient)

$\left\|\nabla \widetilde{f}_t(\mathbf{x})\right\|_2 \leq G$ *for all* $\mathbf{x} \in \mathcal{X}$ *and* t*, where* G *is given.*

Assumption (Bounded Loss)

$0 \leq a \leq f_t(\mathbf{x}_0, \ldots, \mathbf{x}_m) \leq a + c$*, for all* $(\mathbf{x}_0, \ldots, \mathbf{x}_m) \in \mathcal{X}^{m+1}$ *and* t*, where* a*,* c *are given.*

Assumption (Coordinate-Wise Lipschitz)

$|f_t(\mathbf{x}_0, \ldots, \mathbf{x}_m) - f_t(\mathbf{y}_0, \ldots, \mathbf{y}_m)| \leq L \sum_{i=0}^{m} \|\mathbf{x}_i - \mathbf{y}_i\|_2$*, for all* $(\mathbf{x}_0, \ldots, \mathbf{x}_m) \in \mathcal{X}^{m+1}$*, and* $(\mathbf{y}_0, \ldots, \mathbf{y}_m) \in \mathcal{X}^{m+1}$*, and for all* t.

Dynamic Regret: Performance Metric for Online Algorithms

Definition (Dynamic Regret)

Given a time-varying comparator sequence $\mathbf{v}_1, \ldots, \mathbf{v}_T$*, the dynamic regret is*

$$\text{Regret}_T^D \triangleq \sum_{t=1}^{T} f_t(\mathbf{x}_{t-m}, \ldots, \mathbf{x}_t) - \sum_{t=1}^{T} f_t(\mathbf{v}_{t-m}, \ldots, \mathbf{v}_t).$$

$\mathbf{v}_1, \ldots, \mathbf{v}_T$ are typically chosen as the minimizers of $\sum_{t=1}^{T} f_t(\mathbf{v}_{t-m}, \ldots, \mathbf{v}_t)$

Dynamic Regret: Performance Metric for Online Algorithms

Definition (Dynamic Regret)

Given a time-varying comparator sequence $\mathbf{v}_1, \ldots, \mathbf{v}_T$*, the dynamic regret is*

$$\text{Regret}_T^D \triangleq \sum_{t=1}^{T} f_t(\mathbf{x}_{t-m}, \ldots, \mathbf{x}_t) - \sum_{t=1}^{T} f_t(\mathbf{v}_{t-m}, \ldots, \mathbf{v}_t).$$

Goal: Develop an online algorithm for OCO-M with sublinear dynamic regret, *i.e.*,

$$\lim_{T\to\infty} \text{Regret}_T^D / T \to 0.$$

Remark: When the limit is zero, then $f_t(\mathbf{x}_{t-m}, \ldots, \mathbf{x}_t) \to f_t(\mathbf{v}_{t-m}, \ldots, \mathbf{v}_t)$.

Closest Related Work

	Projection	Projection-Free
OCO-M	Weinberger et al., TIT '02 Anava, et al., NeurIPS '15 Zhao et al., AISTATS '22 ...	Our Algorithm
OCO	Hazan, '16 Zinkevich, ICML '03 Mokhtari et al., CDC '16 Jadbabaie et al., AISTATS '15 Yang et al., ICML '16 Zhang et al., NeurIPS '18 Chang et al., AAAI '21 ...	Hazan et al., ICML '12 Jaggi, ICML '13 Garber et al., ICML '15 Wan et al., AAAI '21 Kretzu et al., AISTATS '21 Kalhan et al., TSP '21; Garber et al., COLT '22 Wan et al., COLT '23 ...

projection of a candidate solution to the domain set; computationally inefficient

Projection

Projection-Free

computationally more effiecient

Preliminary: Online Frank-Wolfe (OFW)

OFW is a projection-free algorithm for OCO

Preliminary: Online Frank-Wolfe (OFW)

OFW is a projection-free algorithm for OCO

Initilization: Time horizon T, step size η, domain set $\mathcal{X}$, and $\mathbf{x}_1 \in \mathcal{X}$.

Preliminary: Online Frank-Wolfe (OFW)

OFW is a projection-free algorithm for OCO

Initilization: Time horizon T, step size η, domain set $\mathcal{X}$, and $\mathbf{x}_1 \in \mathcal{X}$.
At each iteration $t = 1, \ldots, T$:

1. Suffer a loss $f_t(\mathbf{x}_t)$;

Preliminary: Online Frank-Wolfe (OFW)

OFW is a projection-free algorithm for OCO

Initilization: Time horizon T, step size η, domain set $\mathcal{X}$, and $\mathbf{x}_1 \in \mathcal{X}$.
At each iteration $t = 1, \ldots, T$:

1. Suffer a loss $f_t(\mathbf{x}_t)$;
2. Obtain gradient $\nabla f_t(\mathbf{x}_t)$;

Preliminary: Online Frank-Wolfe (OFW)

OFW is a projection-free algorithm for OCO

Initilization: Time horizon T, step size η, domain set $\mathcal{X}$, and $\mathbf{x}_1 \in \mathcal{X}$.
At each iteration $t = 1, \ldots, T$:

1. Suffer a loss $f_t(\mathbf{x}_t)$;
2. Obtain gradient $\nabla f_t(\mathbf{x}_t)$;
3. Compute $\mathbf{x}'_t = \arg\min_{\mathbf{x} \in \mathcal{X}} \langle \nabla f_t(\mathbf{x}_t), \mathbf{x} \rangle$;

Preliminary: Online Frank-Wolfe (OFW)

OFW is a projection-free algorithm for OCO

Initilization: Time horizon T, step size η, domain set $\mathcal{X}$, and $\mathbf{x}_1 \in \mathcal{X}$.
At each iteration $t = 1, \ldots, T$:

1. Suffer a loss $f_t(\mathbf{x}_t)$;
2. Obtain gradient $\nabla f_t(\mathbf{x}_t)$;
3. Compute $\mathbf{x}'_t = \arg\min_{\mathbf{x} \in \mathcal{X}} \langle \nabla f_t(\mathbf{x}_t), \mathbf{x} \rangle$;
4. Update $\mathbf{x}_{t+1} = (1 - \eta)\mathbf{x}_t + \eta \mathbf{x}'_t$;

Preliminary: Online Frank-Wolfe (OFW)

OFW is a projection-free algorithm for OCO

Initilization: Time horizon T, step size η, domain set $\mathcal{X}$, and $\mathbf{x}_1 \in \mathcal{X}$.
At each iteration $t = 1, \ldots, T$:

1. Suffer a loss $f_t(\mathbf{x}_t)$;
2. Obtain gradient $\nabla f_t(\mathbf{x}_t)$;
3. Compute $\mathbf{x}'_t = \arg\min_{\mathbf{x} \in \mathcal{X}} \langle \nabla f_t(\mathbf{x}_t), \mathbf{x} \rangle$; → Projection-free step
4. Update $\mathbf{x}_{t+1} = (1 - \eta)\mathbf{x}_t + \eta \mathbf{x}'_t$;

Preliminary: Dynamic Regret of OFW

Theorem (Dynamic Regret Bound of OFW)

OFW achieves

$$\text{Regret}_T^D \leq \mathcal{O}\left(\frac{1+V_T}{\eta} + \sqrt{TD_T}\right),$$

where

- $V_T \triangleq \sum_{t=1}^{T} \sup_{\mathbf{x}\in\mathcal{X}} |f_t(\mathbf{x}) - f_{t-1}(\mathbf{x})|$; ⟶ Capture how fast f_t changes
- $D_T \triangleq \sum_{t=1}^{T} \|\nabla f_t(\mathbf{x}_t) - \nabla f_{t-1}(\mathbf{x}_{t-1})\|_2^2$. ⟶ Capture how fast ∇f_t changes

Preliminary: Dynamic Regret of OFW

Theorem (Dynamic Regret Bound of OFW)

OFW achieves

$$\text{Regret}_T^D \leq \mathcal{O}\left(\frac{1+V_T}{\eta} + \sqrt{TD_T}\right),$$

where

- $V_T \triangleq \sum_{t=1}^{T} \sup_{\mathbf{x}\in\mathcal{X}} |f_t(\mathbf{x}) - f_{t-1}(\mathbf{x})|$; ⟶ Capture how fast f_t changes
- $D_T \triangleq \sum_{t=1}^{T} \|\nabla f_t(\mathbf{x}_t) - \nabla f_{t-1}(\mathbf{x}_{t-1})\|_2^2$. ⟶ Capture how fast ∇f_t changes

Comparison with Kalhan et al., TSP '21: OFW with step size $\eta = \mathcal{O}\left(1/\sqrt{T}\right)$ achieves the same bound as in Theorem 1, Kalhan et al., TSP '21, but we remove the assumption that f_t needs to be smooth.

Example of non-smooth f_t: Hinge loss for classification problems

Preliminary: Dynamic Regret of OFW

Theorem (Dynamic Regret Bound of OFW)

OFW achieves

$$\text{Regret}_T^D \leq \mathcal{O}\left(\frac{1+V_T}{\eta} + \sqrt{TD_T}\right),$$

where

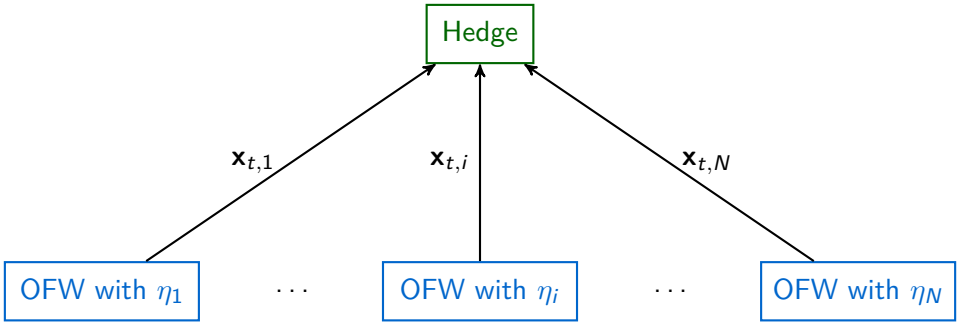
- $V_T \triangleq \sum_{t=1}^{T} \sup_{\mathbf{x}\in\mathcal{X}} |f_t(\mathbf{x}) - f_{t-1}(\mathbf{x})|$; ⟶ Capture how fast f_t changes
- $D_T \triangleq \sum_{t=1}^{T} \|\nabla f_t(\mathbf{x}_t) - \nabla f_{t-1}(\mathbf{x}_{t-1})\|_2^2$. ⟶ Capture how fast ∇f_t changes

Remark: The ideal step size that minimizes the dynamic regret bound is $\eta_* = \mathcal{O}\left(\frac{\sqrt{(1+V_T)}}{T}\right)$. However, the value of V_T is unknown a prior.

We aim to overcome it by fusing multiple OFW with different step sizes

Our Algorithm: Meta-OFW

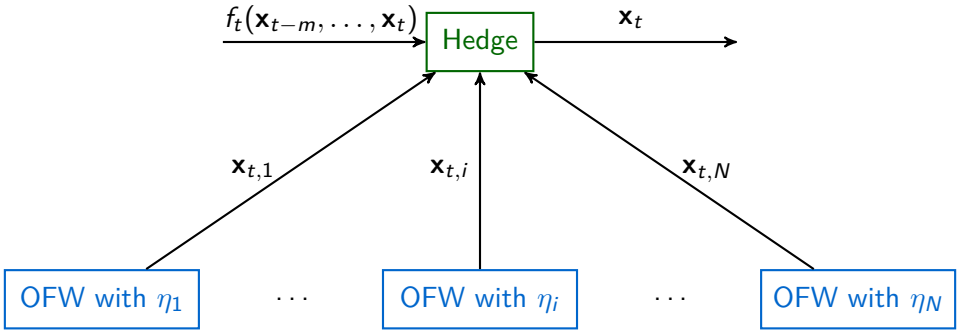
Step 1: N base-learners using OFW with appropriately chosen step sizes send $\mathbf{x}_{t,i}$ to a meta-learner using Hedge algorithm[2]



[2] Freund and Schapire, Journal of Computer and System Sciences '97

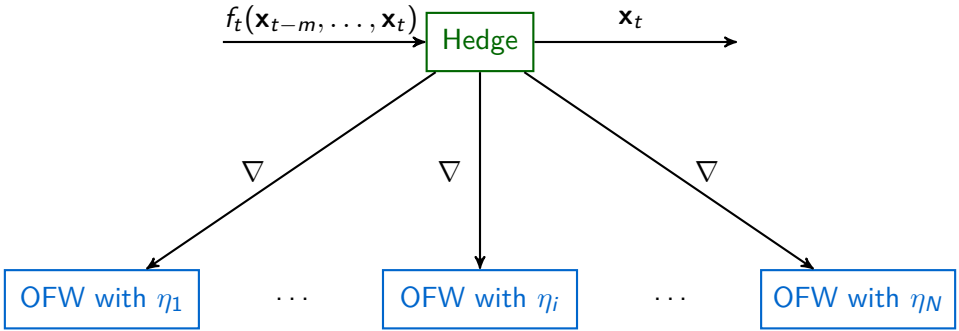
Our Algorithm: Meta-OFW

Step 2: Hedge fuses all $\mathbf{x}_{t,i}$ into a single decision $\mathbf{x}_t$ and receives loss $f_t(\mathbf{x}_{t-m}, \ldots, \mathbf{x}_t)$



Our Algorithm: Meta-OFW

Step 3: Hedge sends gradient ∇ to all base-learners to update their decisions $\mathbf{x}_{t+1,i}$



Our Algorithm: Meta-OFW

Theorem (Dynamic Regret Bound of Meta-OFW)

Meta-OFW achieves

$$\text{Regret}_T^D \leq \mathcal{O}\left(\sqrt{T\left(1 + V_T + \bar{D}_T + C_T\right)}\right),$$

where

- $V_T \triangleq \sum_{t=1}^{T} \sup_{\mathbf{x}\in\mathcal{X}} |f_t(\mathbf{x}) - f_{t-1}(\mathbf{x})|$;
- $\bar{D}_T$ *is an upper bound of* $D_{T,i} \triangleq \sum_{t=1}^{T} \|\nabla f_t(\mathbf{x}_{t,i}) - \nabla f_{t-1}(\mathbf{x}_{t-1,i})\|_2^2$;
- $C_T \triangleq \sum_{t=2}^{T} \|\mathbf{v}_t - \mathbf{v}_{t-1}\|_2$. ⟶ Capture how fast $\mathbf{v}_t$ changes

Application to Control

Linear Time-Varying (LTV) system:

$$x_{t+1} = A_t x_t + B_t u_t + w_t, \quad t = 1, \ldots, T.$$

Application to Control

Linear Time-Varying (LTV) system:

$$x_{t+1} = A_t x_t + B_t u_t + w_t, \quad t = 1, \ldots, T.$$

Disturbance-Action control policy:

$$u_t = -K_t x_t + \sum_{i=1}^{H} M_t^{[i-1]} w_{t-i},$$

where $M_t = (M_t^{[0]}, \ldots, M_t^{[H-1]})$, with bounded $M_t^{[i]} \in \mathbb{R}^{d_u \times d_x}$ being a control gain to be designed, H is the memory length, K_t is sequentially stabilizing,[2] and $w_\tau = 0$, $\forall \tau < 0$.

[2] Gradu et al., L4DC '23

Application to Control

Problem (Non-Stochastic Control (NSC))

At each $t = 1, \dots, T$,

- *first a control input $u_t \in \mathcal{U}_t$ is chosen;*
- *then, a loss function $c_t : \mathbb{R}^{d_x} \times \mathbb{R}^{d_u} \to \mathbb{R}$ is revealed;*
- *the system evolves to state x_{t+1};*
- *the controller suffers a loss $c_t(x_{t+1}, u_t)$.*

The goal is to guarantee states and control inputs that satisfy the constraints for all t and that minimize the dynamic policy regret defined as

$$\text{Regret-NSC}_T^D = \sum_{t=1}^{T} c_t\left(x_{t+1}, u_t\right) - \sum_{t=1}^{T} c_t\left(x_{t+1}^*, u_t^*\right).$$

Application to Control

Assumption (Bounded System Matrices and Noise)

$\|A_t\| \leq \kappa_A$, $\|B_t\| \leq \kappa_B$, *and* $w_t \in \mathcal{W} \triangleq \{w \mid \|w\| \leq W\}$ *where* κ_A, κ_B, *and* W *are given.*

Assumption (Convex and Bounded Loss Function with Bounded Gradient)

The loss function $c_t(x_{t+1}, u_t) : \mathbb{R}^{d_x} \times \mathbb{R}^{d_u} \to \mathbb{R}$ *is convex in* x_{t+1} *and* u_t. *Further,* $|c_t(x, u)|$, $\|\nabla_x c_t(x, u)\|$, *and* $\|\nabla_u c_t(x, u)\|$ *are bounded when* $\|x\|$ *and* $\|u\|$ *are bounded.*

Numerical Evaluation

Goal: Evaluate loss and computational efficiency against projection-based method

Setup:

- Linear system: $x_{t+1} = Ax_t + Bu_t + w_t$, where $x_t \in \mathbb{R}^{d_x}$, $u_t \in \mathbb{R}^{d_u}$;
- w_t sampled from Gaussian distribution with bounded magnitude;
- Quadratic loss function: $c_t(x_{t+1}, u_t) = q_t x_{t+1}^\top x_{t+1} + r_t u_t^\top u_t$ with q_t, r_t as sinusoidal functions;
- Total iteration $T = 200$;
- Comparison with Scream algorithms.[2]

Use projection-based algorithm as base-learners
and Hedge as a meta-learner for OCO-M

[2] Zhao et al., AISTATS '22

Numerical Evaluation

Table: Performance comparison.

$(d_x,\ d_u)$	Time (seconds)		Cumulative Loss	
	Meta-OFW	**Scream**	**Meta-OFW**	**Scream**
$(2,\ 1)$	52.49	17.64	915.52	1819.41
$(4,\ 2)$	135.04	177.47	1388.56	3231.89
$(6,\ 3)$	287.67	605.48	1235.83	2021.47
$(8,\ 4)$	504.82	1351.29	1421.05	1873.79
$(10,\ 5)$	786.87	2219.85	1202.43	1439.22
$(12,\ 6)$	998.22	3029.89	893.17	984.96
$(14,\ 7)$	2022.5	5531.36	797.80	958.09

Meta-OFW achieves lower loss

Numerical Evaluation

Table: Performance comparison.

$(d_x,\ d_u)$	Time (seconds)		Cumulative Loss	
	Meta-OFW	**Scream**	**Meta-OFW**	**Scream**
(2, 1)	52.49	17.64	915.52	1819.41
(4, 2)	135.04	177.47	1388.56	3231.89
(6, 3)	287.67	605.48	1235.83	2021.47
(8, 4)	504.82	1351.29	1421.05	1873.79
(10, 5)	786.87	2219.85	1202.43	1439.22
(12, 6)	998.22	3029.89	893.17	984.96
(14, 7)	2022.5	5531.36	797.80	958.09

Meta-OFW is about 3 times faster

Numerical Evaluation

Table: Performance comparison.

$(d_x,\ d_u)$	Time (seconds)		Cumulative Loss	
	Meta-OFW	**Scream**	**Meta-OFW**	**Scream**
$(2,\ 1)$	52.49	17.64	915.52	1819.41
$(4,\ 2)$	135.04	177.47	1388.56	3231.89
$(6,\ 3)$	287.67	605.48	1235.83	2021.47
$(8,\ 4)$	504.82	1351.29	1421.05	1873.79
$(10,\ 5)$	786.87	2219.85	1202.43	1439.22
$(12,\ 6)$	998.22	3029.89	893.17	984.96
$(14,\ 7)$	2022.5	5531.36	797.80	958.09

Meta-OFW is about 3 times faster

Remark: Kalhan et al. TSP '21 shows OFW is 20 times faster than OGD in matrix completion scenarios.

Summary

Online learning with memory algorithm that

- is projection-free, and
- provides dynamic regret guarantees, and
- can be applied to non-smooth loss functions.

Next steps:

- Extend to non-convex online optimization;
- Extend to safe non-linear control.[2]

[2] Zhou, Song, and Tzoumas. Safe Non-Stochastic Control of Control-Affine Systems: An Online Convex Optimization Approach, IEEE Robotics and Automation Letters (RA-L) '23

Recent Improvement upon Presented Results

Wan et al. show that[3]

- when f_t is smooth and convex, OFW achieves

$$\text{Regret}_T^D \leq \mathcal{O}\left(\sqrt{T(1+V_T)}\right);$$

- when f_t is smooth and convex, and $\mathcal{X}$ is strongly convex, OFW achieves

$$\text{Regret}_T^D \leq \mathcal{O}\left(T^{1/3}(1+V_T)^{2/3}\right).$$

[3] Wan et al., Improved Dynamic Regret for Online Frank-Wolfe, COLT '22