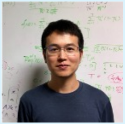
Near-Optimal No-Regret Learning for General Convex Games

Gabriele Farina*

Ioannis Anagnostides*



Haipeng Luo

Chung-Wei Lee

Christian Kroer

Tuomas Sandholm

gfarina@{cs.cmu.edu | meta.com}

Simons Institute

October 14, 2022

What this talk is about

- Recent line of work: uncoupled learning dynamics such that, when employed by all players in a game, each player's regret after T repetitions grows **polylogarithmically in T**

What this talk is about

- Recent line of work: uncoupled learning dynamics such that, when employed by all players in a game, each player's regret after T repetitions grows **polylogarithmically in T**
- However, so far these results have only been limited to certain classes of **games with structured strategy spaces**
  - Mostly normal-form games
  - Extensive-form games via **kernelized** multiplicative weights

What this talk is about

- Recent line of work: uncoupled learning dynamics such that, when employed by all players in a game, each player's regret after T repetitions grows **polylogarithmically in T**
- However, so far these results have only been limited to certain classes of **games with structured strategy spaces**
  - Mostly normal-form games
  - Extensive-form games via **kernelized** multiplicative weights
- Yet, many fundamental models in economics and multiagent systems require more **general, convex** strategy sets

What this talk is about

- Recent line of work: uncoupled learning dynamics such that, when employed by all players in a game, each player's regret after T repetitions grows **polylogarithmically in** T
- However, so far these results have only been limited to certain classes of **games with structured strategy spaces**
  - Mostly normal-form games
  - Extensive-form games via **kernelized** multiplicative weights
- Yet, many fundamental models in economics and multiagent systems require more **general, convex** strategy sets

Q: Can $O(\mathrm{polylog}\, T)$ regret be attained in general convex and compact strategy sets while retaining efficient strategy updates?

What this talk is about

Q: Can $O(\text{polylog}\, T)$ regret be attained in general convex and compact strategy sets while retaining efficient strategy updates?

- We answer in the **positive**, and give an uncoupled learning algorithm with $O(\log T)$ per-player regret in general convex games
  - In adversarial settings: usual $O(\sqrt{T})$ regret bound

What this talk is about

Q: Can $O(\text{polylog}\, T)$ regret be attained in general convex and compact strategy sets while retaining efficient strategy updates?

- We answer in the **positive**, and give an uncoupled learning algorithm with $O(\log T)$ per-player regret in general convex games
  - In adversarial settings: usual $O(\sqrt{T})$ regret bound
- Per-iteration complexity:
  - $O(\log \log T)$ with access to local proximal oracle
  - $O(\text{poly}\, T)$ with access to only a linear optimization oracle

What this talk is about

Q: Can $O(\text{polylog}\, T)$ regret be attained in general convex and compact strategy sets while retaining efficient strategy updates?

- We answer in the **positive**, and give an uncoupled learning algorithm with $O(\log T)$ per-player regret in general convex games
  - In adversarial settings: usual $O(\sqrt{T})$ regret bound
- Per-iteration complexity:
  - $O(\log \log T)$ with access to local proximal oracle
  - $O(\text{poly}\, T)$ with access to only a linear optimization oracle
- In special cases where prior results apply, our algorithm improves over the state-of-the-art regret bounds in terms of the dependence on either **the number of iterations** or **dimension of the strategy sets**

History and Context

Regret minimization

- Celebrated framework that has been central in the development of online learning and multiagent systems

Regret minimization

- Celebrated framework that has been central in the development of online learning and multiagent systems
- **Idea:** a player is "learning" how to play the game when looking back they do not strongly wish they had played differently

Regret minimization

- Celebrated framework that has been central in the development of online learning and multiagent systems
- **Idea:** a player is "learning" how to play the game when looking back they do not strongly wish they had played differently

Why care about regret minimization?

At least three different scenarios

Regret minimization

- Celebrated framework that has been central in the development of online learning and multiagent systems
- **Idea:** a player is "learning" how to play the game when looking back they do not strongly wish they had played differently

Why care about regret minimization?

1. Natural notion of performance if the learning is truly online
   - Good news: no-regret algorithms are designed for adversarial environment

Regret minimization

- Celebrated framework that has been central in the development of online learning and multiagent systems
- **Idea:** a player is "learning" how to play the game when looking back they do not strongly wish they had played differently

Why care about regret minimization?

1. Natural notion of performance if the learning is truly online
   - Good news: no-regret algorithms are designed for adversarial environment
2. May be a good model for the behavior of a modeled system

Regret minimization

- Celebrated framework that has been central in the development of online learning and multiagent systems
- **Idea:** a player is "learning" how to play the game when looking back they do not strongly wish they had played differently

Why care about regret minimization?

1. Natural notion of performance if the learning is truly online
   - Good news: no-regret algorithms are designed for adversarial environment
2. May be a good model for the behavior of a modeled system
3. Important connections to game-theoretic equilibria
   - Convergence to coarse correlated equilibrium in multi-player general-sum games
   - Approximation error is tied to maximum individual regret
   - Special case: Nash equilibrium in 2-player 0-sum games

Equilibrium finding and self play

Why care about regret minimization?

1. Natural notion of performance if the learning is truly online
2. May be a good model for the behavior of a modeled system
3. **Important connections to game-theoretic equilibria**

- Idea: players train against each other in **self play**

Equilibrium finding and self play

Why care about regret minimization?

1. Natural notion of performance if the learning is truly online
2. May be a good model for the behavior of a modeled system
3. **Important connections to game-theoretic equilibria**

- Idea: players train against each other in **self play**
- Current state-of-the-art for large games

Equilibrium finding and self play

Why care about regret minimization?

1. Natural notion of performance if the learning is truly online
2. May be a good model for the behavior of a modeled system
3. **Important connections to game-theoretic equilibria**

- Idea: players train against each other in **self play**
- Current state-of-the-art for large games
- Remarkable practical success: primary component in recent landmark results in AI

Equilibrium finding and self play

Why care about regret minimization?

1. Natural notion of performance if the learning is truly online
2. May be a good model for the behavior of a modeled system
3. **Important connections to game-theoretic equilibria**

- Idea: players train against each other in **self play**
- Current state-of-the-art for large games
- Remarkable practical success: primary component in recent landmark results in AI

Self-play learning is far from fully unpredictable/adversarial setting online learning has historically focused on...

Equilibrium finding and self play

Why care about regret minimization?

1. Natural notion of performance if the learning is truly online
2. May be a good model for the behavior of a modeled system
3. **Important connections to game-theoretic equilibria**

- Idea: players train against each other in **self play**
- Current state-of-the-art for large games
- Remarkable practical success: primary component in recent landmark results in AI

Self-play learning is far from fully unpredictable/adversarial setting online learning has historically focused on...

Q: What performance guarantees can be obtained for self-play in general games?

Prior work

What performance guarantees can be obtained for self-play in general games?

- Question first formulated by Daskalakis et al. [2011] for two-player zero-sum matrix games

Prior work

What performance guarantees can be obtained for self-play in general games?

- Question first formulated by Daskalakis et al. [2011] for two-player zero-sum matrix games
- Since then: considerable interest in extending guarantees to more general setting

Prior work

What performance guarantees can be obtained for self-play in general games?

- Question first formulated by Daskalakis et al. [2011] for two-player zero-sum matrix games
- Since then: considerable interest in extending guarantees to more general setting
- Chiang et al. [2012] and Rakhlin and Sridharan [2013] pioneered the framework of **optimism**

Prior work

What performance guarantees can be obtained for self-play in general games?

- Question first formulated by Daskalakis et al. [2011] for two-player zero-sum matrix games
- Since then: considerable interest in extending guarantees to more general setting
- Chiang et al. [2012] and Rakhlin and Sridharan [2013] pioneered the framework of **optimism**
- Syrgkanis et al. [2015] crystallized the **RVU property** and established dynamics with $O(T^{1/4})$ per-player regret in convex games

Prior work

What performance guarantees can be obtained for self-play in general games?

- Question first formulated by Daskalakis et al. [2011] for two-player zero-sum matrix games
- Since then: considerable interest in extending guarantees to more general setting
- Chiang et al. [2012] and Rakhlin and Sridharan [2013] pioneered the framework of **optimism**
- Syrgkanis et al. [2015] crystallized the **RVU property** and established dynamics with $O(T^{1/4})$ per-player regret in convex games
- Chen and Peng [2020] improves to $O(T^{1/6})$ but only in two-player games

Prior work (cont'd)

- Daskalakis et al. [2021] shows that in matrix games one can achieve $O(\log^4 T)$ by using the OMWU algorithm
  - *Exponential improvement* over the guarantees obtained using traditional techniques within the no-regret framework!

Prior work (cont'd)

- Daskalakis et al. [2021] shows that in matrix games one can achieve $O(\log^4 T)$ by using the OMWU algorithm
  - *Exponential improvement* over the guarantees obtained using traditional techniques within the no-regret framework!
- Farina et al. [2022] show that in certain classes of polyhedral games (including extensive-form games) one can run OMWU on the (exponentially many) vertices in polynomial time thanks to a **kernel trick** ($\rightarrow$ Kernelized OMWU)

Prior work (cont'd)

- Daskalakis et al. [2021] shows that in matrix games one can achieve $O(\log^4 T)$ by using the OMWU algorithm
  - ▷ *Exponential improvement* over the guarantees obtained using traditional techniques within the no-regret framework!
- Farina et al. [2022] show that in certain classes of polyhedral games (including extensive-form games) one can run OMWU on the (exponentially many) vertices in polynomial time thanks to a **kernel trick** ($\rightarrow$ Kernelized OMWU)
- Anagnostides et al. [2022] extendes the technique of Daskalakis et al. [2021] to swap regret in matrix games

Prior work (cont'd)

- Daskalakis et al. [2021] shows that in matrix games one can achieve $O(\log^4 T)$ by using the OMWU algorithm
  - *Exponential improvement* over the guarantees obtained using traditional techniques within the no-regret framework!
- Farina et al. [2022] show that in certain classes of polyhedral games (including extensive-form games) one can run OMWU on the (exponentially many) vertices in polynomial time thanks to a **kernel trick** ($\rightarrow$ Kernelized OMWU)
- Anagnostides et al. [2022] extendes the technique of Daskalakis et al. [2021] to swap regret in matrix games
- Piliouras et al. [2022] proposes learning dynamics that guarantee $O(\log T)$ regret for a **subsequence** of iterates in matrix games

Prior work (cont'd)

- Daskalakis et al. [2021] shows that in matrix games one can achieve $O(\log^4 T)$ by using the OMWU algorithm
  - ▷ *Exponential improvement* over the guarantees obtained using traditional techniques within the no-regret framework!
- Farina et al. [2022] show that in certain classes of polyhedral games (including extensive-form games) one can run OMWU on the (exponentially many) vertices in polynomial time thanks to a **kernel trick** ($\rightarrow$ Kernelized OMWU)
- Anagnostides et al. [2022] extendes the technique of Daskalakis et al. [2021] to swap regret in matrix games
- Piliouras et al. [2022] proposes learning dynamics that guarantee $O(\log T)$ regret for a **subsequence** of iterates in matrix games
- **This paper**: $O(\log T)$ regret for general convex games

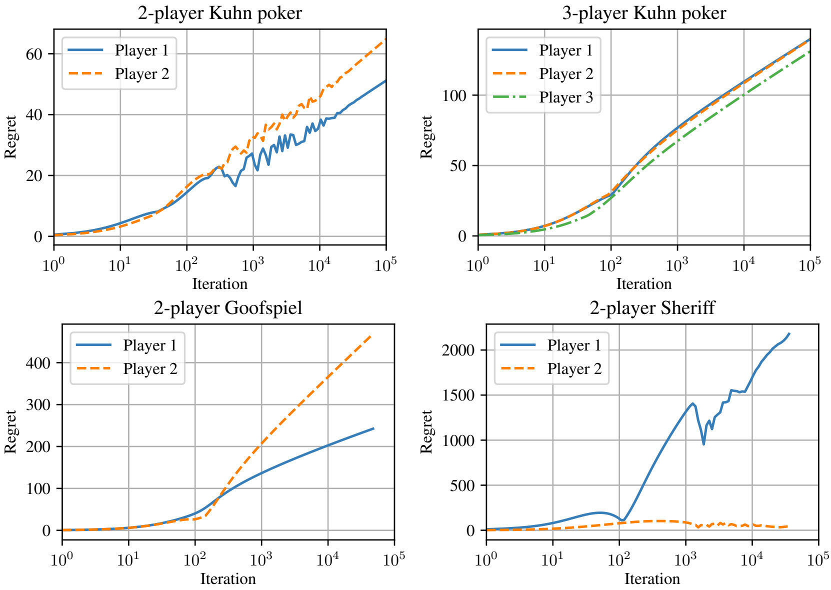
Comparison table

Method	Applies to	Regret bound	Cost per iteration
OFTRL / OMD [Syrgkanis et al., 2015]	General convex set	$O(\sqrt{n}\mathfrak{R}T^{1/4})$	Regularizer- & oracle- dependent
OMWU [Daskalakis et al., 2021]	Simplex Δ^d	$O(n \log d \log^4 T)$	$O(d)$
Clairvoyant MWU [Piliouras et al., 2022]	Simplex Δ^d	$O(n \log d \log T)$ ⚠ subsequence only!	$O(d)$
Kernelized OMWU [Farina et al., 2022]	Polytope $\Omega = \mathrm{co}\mathcal{V}$ with $\mathcal{V} \subseteq \{0,1\}^d$	$O(n \log \lvert\mathcal{V}\rvert \log^4 T)$	$d \times$ cost of kernel
`LRL-OFTRL` [**This talk**]	General convex set $\mathcal{X} \subseteq \mathbb{R}^d$	$O(nd\lVert\mathcal{X}\rVert_1^3 \log T)$	Oracle-dependent: • $O(\log \log T)$ proximal oracle calls • $O(\text{poly } T)$ linear opt. oracle calls

where:

- n: number of players
- T: number of iterations/repetitions of the game
- $\mathfrak{R}$: regularizer-dependent parameter
- $\mathrm{co}\mathcal{V}$: convex hull of $\mathcal{V}$
- $\lVert\mathcal{X}\rVert_1$: upper bound on $\max_{\boldsymbol{x}\in\mathcal{X}} \lVert\boldsymbol{x}\rVert_1$

Experimental results (log **x-axis**)



Convex Games

Convex game

In an n-player convex game:

Every player $i \in \{1, \ldots, n\}$ has a nonnempty convex and compact strategy set $\mathcal{X}_i$ (these include *mixed* strategies)

The **utility function** $u_i : \bigtimes_{j=1}^{n} \mathcal{X}_j \to \mathbb{R}$ of player i is a continuously differentiable function such that:

1. (concavity) $u_i(\boldsymbol{x}_i, \boldsymbol{x}_{-i})$ is concave in $\boldsymbol{x}_i$ for all $\boldsymbol{x}_{-i}$

Convex game

In an n-player convex game:

Every player $i \in \{1, \ldots, n\}$ has a nonnempty convex and compact strategy set $\mathcal{X}_i$ (these include *mixed* strategies)

The **utility function** $u_i : \bigtimes_{j=1}^{n} \mathcal{X}_j \to \mathbb{R}$ of player i is a continuously differentiable function such that:

1. (concavity) $u_i(\boldsymbol{x}_i, \boldsymbol{x}_{-i})$ is concave in $\boldsymbol{x}_i$ for all $\boldsymbol{x}_{-i}$
2. (bounded gradients) $\|\nabla_{\boldsymbol{x}_i} u_i(\boldsymbol{x})\|_\infty \leq B$ for all $\boldsymbol{x}$

Convex game

In an n-player convex game:

Every player $i \in \{1, \ldots, n\}$ has a nonnempty convex and compact strategy set $\mathcal{X}_i$ (these include *mixed* strategies)

The **utility function** $u_i : \bigtimes_{j=1}^{n} \mathcal{X}_j \to \mathbb{R}$ of player i is a continuously differentiable function such that:

1. (concavity) $u_i(\boldsymbol{x}_i, \boldsymbol{x}_{-i})$ is concave in $\boldsymbol{x}_i$ for all $\boldsymbol{x}_{-i}$
2. (bounded gradients) $\|\nabla_{\boldsymbol{x}_i} u_i(\boldsymbol{x})\|_\infty \leq B$ for all $\boldsymbol{x}$
3. (smoothness) $\nabla_{\boldsymbol{x}_i} u_i$ is L-Lipschitz smooth:

$$\|\nabla_{\boldsymbol{x}_i} u_i(\boldsymbol{x}) - \nabla_{\boldsymbol{x}_i} u_i(\boldsymbol{x}')\|_\infty \leq L\|\boldsymbol{x} - \boldsymbol{x}'\|_1$$

   for all $\boldsymbol{x}, \boldsymbol{x}'$.

Example: Normal-Form Games

	0	-1	$+1$
	$+1$	0	-1
	-1	$+1$	0

- Games like rock-paper-scissors
  - Simultaneous action game with finite action set $\mathcal{A}_i$ for each player i

Example: Normal-Form Games

			
	0	-1	$+1$
	$+1$	0	-1
	-1	$+1$	0

- Games like rock-paper-scissors
  - Simultaneous action game with finite action set $\mathcal{A}_i$ for each player i
- Each player's strategy set if the set of distributions over their actions $\mathcal{A}_i$

$$\mathcal{X}_i = \Delta(\mathcal{A}_i)$$

Example: Normal-Form Games

			
	0	−1	+1
	+1	0	−1
	−1	+1	0

- Games like rock-paper-scissors
  - ▷ Simultaneous action game with finite action set $\mathcal{A}_i$ for each player i
- Each player's strategy set if the set of distributions over their actions $\mathcal{A}_i$

$$\mathcal{X}_i = \Delta(\mathcal{A}_i)$$

- The utility of player i is the **multilinear** function

$$u_i(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{a}\sim\boldsymbol{x}}[U_i(\boldsymbol{a})]$$

where U_i is the payoff function of the game

Example: Normal-Form Games

	0	-1	$+1$
	$+1$	0	-1
	-1	$+1$	0

- Games like rock-paper-scissors
  - ▷ Simultaneous action game with finite action set $\mathcal{A}_i$ for each player i
- Each player's strategy set if the set of distributions over their actions $\mathcal{A}_i$

$$\mathcal{X}_i = \Delta(\mathcal{A}_i)$$

- The utility of player i is the **multilinear** function

$$u_i(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{a} \sim \boldsymbol{x}}[U_i(\boldsymbol{a})]$$

  where U_i is the payoff function of the game

- Gradients of u_i are bounded by the maximum payoff

Example: Normal-Form Games

	0	−1	+1
	+1	0	−1
	−1	+1	0

- Games like rock-paper-scissors
  - Simultaneous action game with finite action set $\mathcal{A}_i$ for each player i
- Each player's strategy set if the set of distributions over their actions $\mathcal{A}_i$

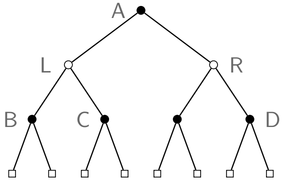
$$\mathcal{X}_i = \Delta(\mathcal{A}_i)$$

- The utility of player i is the **multilinear** function

$$u_i(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{a} \sim \boldsymbol{x}}[U_i(\boldsymbol{a})]$$

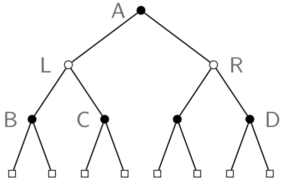
  where U_i is the payoff function of the game
- Gradients of u_i are bounded by the maximum payoff
- Smoothness of ∇u_i is known total variation lemma

Example: Extensive-Form Games



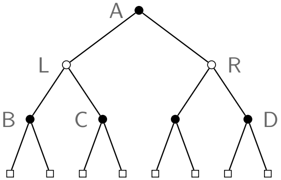
- Games played on a **game tree**

Example: Extensive-Form Games



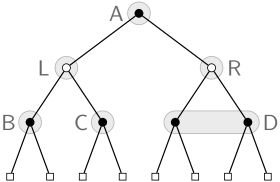
- Games played on a **game tree**
- Poker, Go, Bridge, ...

Example: Extensive-Form Games



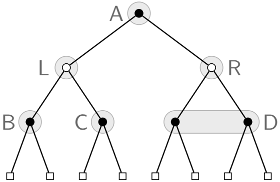
- Games played on a **game tree**
- Poker, Go, Bridge, ...
- **Turns**, simultaneous moves, **stochastic** moves

Example: Extensive-Form Games



- Games played on a **game tree**
- Poker, Go, Bridge, ...
- **Turns**, simultaneous moves, **stochastic** moves
- **Imperfect information**

Example: Extensive-Form Games

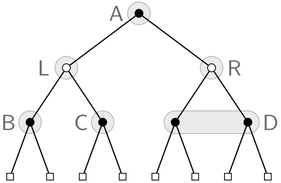


- Games played on a **game tree**
- Poker, Go, Bridge, ...
- **Turns**, simultaneous moves, **stochastic** moves
- **Imperfect information**

Extensive-form games are convex games:

- Strategy space of each player is a **sequence-form polytope** [Romanovskii, 1962, Koller et al., 1996]

Example: Extensive-Form Games

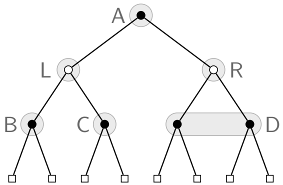


- Games played on a **game tree**
- Poker, Go, Bridge, ...
- **Turns**, simultaneous moves, **stochastic** moves
- **Imperfect information**

Extensive-form games are convex games:

- Strategy space of each player is a **sequence-form polytope** [Romanovskii, 1962, Koller et al., 1996]
- Utilities are multilinear

Example: Extensive-Form Games



- Games played on a **game tree**
- Poker, Go, Bridge, ...
- **Turns**, simultaneous moves, **stochastic** moves
- **Imperfect information**

Extensive-form games are convex games:

- Strategy space of each player is a **sequence-form polytope** [Romanovskii, 1962, Koller et al., 1996]
- Utilities are multilinear
- Hence gradients are smooth and bounded similarly to normal-form games

Example: Splittable Routing Games

- Every player has to route a flow f_i from a source to a destination in an undirected graph

Example: Splittable Routing Games

- Every player has to route a flow f_i from a source to a destination in an undirected graph
- Every edge is associated with a latency function $\ell_e(f_e)$ mapping the amount of flow through the edge to some latency

Example: Splittable Routing Games

- Every player has to route a flow f_i from a source to a destination in an undirected graph
- Every edge is associated with a latency function $\ell_e(f_e)$ mapping the amount of flow through the edge to some latency
- Strategy set of each player is all possible ways of splitting f_i into paths from source to destination

Example: Splittable Routing Games

- Every player has to route a flow f_i from a source to a destination in an undirected graph
- Every edge is associated with a latency function $\ell_e(f_e)$ mapping the amount of flow through the edge to some latency
- Strategy set of each player is all possible ways of splitting f_i into paths from source to destination
- Under suitable restrictions on the latency functions, these games satisfy our convex game definition [Syrgkanis et al., 2015, Roughgarden and Schoppmann, 2015]

Example: Cournot Competition

- Games played among n firms (players)

Example: Cournot Competition

- Games played among n firms (players)
- Every firm i decides the quantity s_i of a good to produce from an interval

Example: Cournot Competition

- Games played among n firms (players)
- Every firm i decides the quantity s_i of a good to produce from an interval
- A cost function c_i assigns a production cost to the given quantity

Example: Cournot Competition

- Games played among n firms (players)
- Every firm i decides the quantity s_i of a good to produce from an interval
- A cost function c_i assigns a production cost to the given quantity
- Price of good is determined by the joint choice of quantities $\boldsymbol{s} = (s_1, \ldots, s_n)$

Example: Cournot Competition

- Games played among n firms (players)
- Every firm i decides the quantity s_i of a good to produce from an interval
- A cost function c_i assigns a production cost to the given quantity
- Price of good is determined by the joint choice of quantities $\boldsymbol{s} = (s_1, \ldots, s_n)$
- Utility of firm i is defined as $u_i(\boldsymbol{s}) = s_i p(\boldsymbol{s}) - c_i(\boldsymbol{s})$

Example: Cournot Competition

- Games played among n firms (players)
- Every firm i decides the quantity s_i of a good to produce from an interval
- A cost function c_i assigns a production cost to the given quantity
- Price of good is determined by the joint choice of quantities $\boldsymbol{s} = (s_1, \ldots, s_n)$
- Utility of firm i is defined as $u_i(\boldsymbol{s}) = s_i p(\boldsymbol{s}) - c_i(\boldsymbol{s})$
- Important case: concave and smooth u_i [Even-Dar et al., 2009]

Learning Setup in Convex Games

- Repeated interaction

- Repeated interaction
- At all times t, each player outputs their strateg $\boldsymbol{x}_i^{(t)} \in \mathcal{X}_i$

- Repeated interaction
- At all times t, each player outputs their strateg $\boldsymbol{x}_i^{(t)} \in \mathcal{X}_i$
- Then, everyone observes the gradient of their own utility

$$\boldsymbol{u}_i^{(t)} := \nabla_{\boldsymbol{x}_i} u_i(\boldsymbol{x}_1^{(t)}, \ldots, \boldsymbol{x}_n^{(t)})$$

- Repeated interaction
- At all times t, each player outputs their strateg $\boldsymbol{x}_i^{(t)} \in \mathcal{X}_i$
- Then, everyone observes the gradient of their own utility

$$\boldsymbol{u}_i^{(t)} := \nabla_{\boldsymbol{x}_i} u_i(\boldsymbol{x}_1^{(t)}, \dots, \boldsymbol{x}_n^{(t)})$$

The canonical measure of performance of each player is **regret**

$$\mathrm{Reg}_i^{(T)} := \max_{\boldsymbol{x}^* \in \mathcal{X}_i} \sum_{t=1}^{T} \langle \boldsymbol{u}^{(t)}, \boldsymbol{x}^* - \boldsymbol{x}_i^{(t)} \rangle$$

Our Technique — Main Insights

Outline

1. What RVU bounds enable and what they don't
   - $O(1)$ social regret, but no guarantees[???] on individual regret

Outline

1. What RVU bounds enable and what they don't
   - $O(1)$ social regret, but no guarantees$^{???}$ on individual regret
2. What would be enough to enable $O(1)$ social $\rightarrow$ individual regret guarantee?

Outline

1. What RVU bounds enable and what they don't
   - $O(1)$ social regret, but no guarantees$^{???}$ on individual regret
2. What would be enough to enable $O(1)$ social $\rightarrow$ individual regret guarantee?
3. That will give intuition as to how we got to our dynamics

RVU bounds and *social* regret

- Optimistic FTRL / OMD guarantee RVU bounds:[1]

$$\mathrm{Reg}_i^{(T)} \lessapprox \frac{1}{\eta} + \eta \sum_{t=1}^{T} \left\| \boldsymbol{u}_i^{(t)} - \boldsymbol{u}_i^{(t-1)} \right\|_*^2 - \frac{1}{\eta} \sum_{t=1}^{T} \left\| \boldsymbol{x}_i^{(t)} - \boldsymbol{x}_i^{(t-1)} \right\|^2$$

[1] Stepsize- and time-independent factors are omitted

RVU bounds and *social* regret

- Optimistic FTRL / OMD guarantee RVU bounds:[1]

$$\text{Reg}_i^{(T)} \lessapprox \frac{1}{\eta} + \eta \sum_{t=1}^{T} \left\| \boldsymbol{u}_i^{(t)} - \boldsymbol{u}_i^{(t-1)} \right\|_*^2 - \frac{1}{\eta} \sum_{t=1}^{T} \left\| \boldsymbol{x}_i^{(t)} - \boldsymbol{x}_i^{(t-1)} \right\|^2$$

RVU bounds are powerful

This fact alone implies that the **social regret** (sum of regrets of all players) is at most a T-independent constant

[1] Stepsize- and time-independent factors are omitted

RVU bounds and *social* regret

- Optimistic FTRL / OMD guarantee RVU bounds:[1]

$$\mathrm{Reg}_i^{(T)} \lessapprox \frac{1}{\eta} + \eta \sum_{t=1}^{T} \left\| \boldsymbol{u}_i^{(t)} - \boldsymbol{u}_i^{(t-1)} \right\|_*^2 - \frac{1}{\eta} \sum_{t=1}^{T} \left\| \boldsymbol{x}_i^{(t)} - \boldsymbol{x}_i^{(t-1)} \right\|^2$$

- Using the smoothness of the utilities, the middle sum can be bounded as

$$\sum_{t=1}^{T} \left\| \boldsymbol{u}_i^{(t)} - \boldsymbol{u}_i^{(t-1)} \right\|_*^2 \leq L^2 \sum_{t=1}^{T} \sum_{j=1}^{n} \left\| \boldsymbol{x}_j^{(t)} - \boldsymbol{x}_j^{(t-1)} \right\|^2$$

[1] Stepsize- and time-independent factors are omitted

RVU bounds and *social* regret

- Optimistic FTRL / OMD guarantee RVU bounds:[1]

$$\mathrm{Reg}_i^{(T)} \lessapprox \frac{1}{\eta} + \eta \sum_{t=1}^{T} \left\| \boldsymbol{u}_i^{(t)} - \boldsymbol{u}_i^{(t-1)} \right\|_*^2 - \frac{1}{\eta} \sum_{t=1}^{T} \left\| \boldsymbol{x}_i^{(t)} - \boldsymbol{x}_i^{(t-1)} \right\|^2$$

- Using the smoothness of the utilities, the middle sum can be bounded as

$$\sum_{t=1}^{T} \left\| \boldsymbol{u}_i^{(t)} - \boldsymbol{u}_i^{(t-1)} \right\|_*^2 \le L^2 \sum_{t=1}^{T} \sum_{j=1}^{n} \left\| \boldsymbol{x}_j^{(t)} - \boldsymbol{x}_j^{(t-1)} \right\|^2$$

- So, the **social** regret is bounded as

$$\begin{aligned}\sum_{i=1}^{n} \mathrm{Reg}_i^{(T)} &\lessapprox \frac{n}{\eta} + \left(n\eta L^2 - \frac{1}{\eta} \right) \sum_{t=1}^{T} \sum_{j=1}^{n} \left\| \boldsymbol{x}_j^{(t)} - \boldsymbol{x}_j^{(t-1)} \right\|^2 \\ &\le \frac{n}{\eta} \qquad \left(\text{as long as } \eta \le \frac{1}{L\sqrt{n}} \right)\end{aligned}$$

[1] Stepsize- and time-independent factors are omitted

What about *individual* regret?

- Unfortunately, convergence to coarse-correlated equilibria in multiplayer games is driven by the maximum **individual** regret, and not by the social regret

What about *individual* regret?

- Unfortunately, convergence to coarse-correlated equilibria in multiplayer games is driven by the maximum **individual** regret, and not by the social regret

Natural question

What do RVU bounds tell us about **individual** regret?

What about *individual* regret?

- Unfortunately, convergence to coarse-correlated equilibria in multiplayer games is driven by the maximum **individual** regret, and not by the social regret

Natural question

What do RVU bounds tell us about **individual** regret?

1. Regret is upper bounded by the social squared path length

$$\mathrm{Reg}_i^{(T)} \lessapprox \frac{1}{\eta} + \eta L^2 \sum_{t=1}^{T} \sum_{j=1}^{n} \left\| \boldsymbol{x}_j^{(t)} - \boldsymbol{x}_j^{(t-1)} \right\|^2$$

What about *individual* regret?

- Unfortunately, convergence to coarse-correlated equilibria in multiplayer games is driven by the maximum **individual** regret, and not by the social regret

Natural question

What do RVU bounds tell us about **individual** regret?

1. Regret is upper bounded by the social squared path length

$$\mathrm{Reg}_i^{(T)} \lessapprox \frac{1}{\eta} + \eta L^2 \sum_{t=1}^{T} \sum_{j=1}^{n} \left\| \boldsymbol{x}_j^{(t)} - \boldsymbol{x}_j^{(t-1)} \right\|^2$$

2. So, if the social squared path length was small...

What about *individual* regret?

- Unfortunately, convergence to coarse-correlated equilibria in multiplayer games is driven by the maximum **individual** regret, and not by the social regret

Natural question

What do RVU bounds tell us about **individual** regret?

1. Regret is upper bounded by the social squared path length

$$\text{Reg}_i^{(T)} \lessapprox \frac{1}{\eta} + \eta L^2 \sum_{t=1}^{T} \sum_{j=1}^{n} \left\| \boldsymbol{x}_j^{(t)} - \boldsymbol{x}_j^{(t-1)} \right\|^2$$

2. So, if the social squared path length was small...
3. On the other hand, for $\eta = O(1)$ the social regret is

$$\sum_{i=1}^{n} \text{Reg}_i^{(T)} \lessapprox \frac{n}{\eta} - \frac{1}{\eta} \sum_{t=1}^{T} \sum_{j=1}^{n} \left\| \boldsymbol{x}_j^{(t)} - \boldsymbol{x}_j^{(t-1)} \right\|^2$$

Main insight

1. Regret is upper bounded by the social squared path length

$$\mathrm{Reg}_i^{(T)} \lesssim \frac{1}{\eta} + \eta L^2 \sum_{t=1}^{T} \sum_{j=1}^{n} \left\| \boldsymbol{x}_j^{(t)} - \boldsymbol{x}_j^{(t-1)} \right\|^2$$

2. So, if the social squared path length was small...
3. On the other hand, for $\eta = O(1)$ the social regret is

$$\sum_{i=1}^{n} \mathrm{Reg}_i^{(T)} \lesssim \frac{n}{\eta} - \frac{1}{\eta} \sum_{t=1}^{T} \sum_{j=1}^{n} \left\| \boldsymbol{x}_j^{(t)} - \boldsymbol{x}_j^{(t-1)} \right\|^2$$

Main insight

1. Regret is upper bounded by the social squared path length

$$\mathrm{Reg}_i^{(T)} \lessapprox \frac{1}{\eta} + \eta L^2 \sum_{t=1}^{T} \sum_{j=1}^{n} \left\| \boldsymbol{x}_j^{(t)} - \boldsymbol{x}_j^{(t-1)} \right\|^2$$

2. So, if the social squared path length was small...
3. On the other hand, for $\eta = O(1)$ the social regret is

$$\sum_{i=1}^{n} \mathrm{Reg}_i^{(T)} \lessapprox \frac{n}{\eta} - \frac{1}{\eta} \sum_{t=1}^{T} \sum_{j=1}^{n} \left\| \boldsymbol{x}_j^{(t)} - \boldsymbol{x}_j^{(t-1)} \right\|^2$$

Main insight

If we knew that $\mathrm{Reg}_i^{(T)} \geq 0$ for all player, then:

1. From second inequality: social path length $\lessapprox n$, that is at most **constant** wrt time T!

Main insight

1. Regret is upper bounded by the social squared path length

$$\mathrm{Reg}_i^{(T)} \lessapprox \frac{1}{\eta} + \eta L^2 \sum_{t=1}^{T} \sum_{j=1}^{n} \left\| \boldsymbol{x}_j^{(t)} - \boldsymbol{x}_j^{(t-1)} \right\|^2$$

2. So, if the social squared path length was small...
3. On the other hand, for $\eta = O(1)$ the social regret is

$$\sum_{i=1}^{n} \mathrm{Reg}_i^{(T)} \lessapprox \frac{n}{\eta} - \frac{1}{\eta} \sum_{t=1}^{T} \sum_{j=1}^{n} \left\| \boldsymbol{x}_j^{(t)} - \boldsymbol{x}_j^{(t-1)} \right\|^2$$

Main insight

If we knew that $\mathrm{Reg}_i^{(T)} \geq 0$ for all player, then:

1. From second inequality: social path length $\lessapprox n$, that is at most **constant** wrt time T!
2. Plugging into first inequality: constant **individual** regret

Main question

(How) Can we

♦ **Ensure the nonnegativity of the player regrets**,

While at the same time

♦ **Not losing the RVU bound**?

Our Technique — Technical Details

Overview of our dynamics

Based on Optimistic FTRL, but with **three** important twists:

1. **Lifting**
   - OFTRL operates on a lifted space $\tilde{\mathcal{X}} \subseteq \mathbb{R}^{d+1}$
   - Feedback is lifted to $\tilde{\mathcal{X}}$ before iterates can be produced

Overview of our dynamics

Based on Optimistic FTRL, but with **three** important twists:

1. **Lifting**
   - OFTRL operates on a lifted space $\tilde{\mathcal{X}} \subseteq \mathbb{R}^{d+1}$
   - Feedback is lifted to $\tilde{\mathcal{X}}$ before iterates can be produced
2. **Log regularization**
   - Proximal steps are regularized on $\tilde{\mathcal{X}}$ with a log regularizer that is *not* a barrier for $\tilde{\mathcal{X}}$

Overview of our dynamics

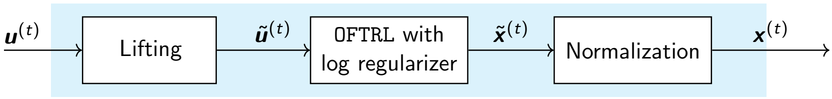
Based on Optimistic FTRL, but with **three** important twists:

1. **Lifting**
   - OFTRL operates on a lifted space $\tilde{\mathcal{X}} \subseteq \mathbb{R}^{d+1}$
   - Feedback is lifted to $\tilde{\mathcal{X}}$ before iterates can be produced
2. **Log regularization**
   - Proximal steps are regularized on $\tilde{\mathcal{X}}$ with a log regularizer that is *not* a barrier for $\tilde{\mathcal{X}}$
3. **Normalization**
   - Iterates are projected back from $\tilde{\mathcal{X}}$ to $\mathcal{X}$

Overview of our dynamics

Based on Optimistic FTRL, but with **three** important twists:

1. **Lifting**
   - OFTRL operates on a lifted space $\tilde{\mathcal{X}} \subseteq \mathbb{R}^{d+1}$
   - Feedback is lifted to $\tilde{\mathcal{X}}$ before iterates can be produced
2. **Log regularization**
   - Proximal steps are regularized on $\tilde{\mathcal{X}}$ with a log regularizer that is *not* a barrier for $\tilde{\mathcal{X}}$
3. **Normalization**
   - Iterates are projected back from $\tilde{\mathcal{X}}$ to $\mathcal{X}$



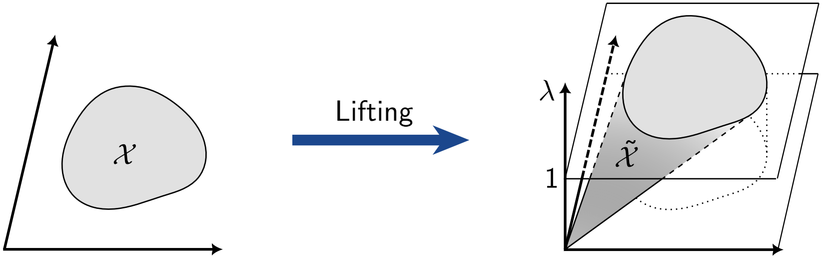
Notation & assumptions

- Let $\mathcal{X}$ be the strategy set of a player
- Without loss of generality, $\mathcal{X} \subseteq [0, +\infty)^d$ (else shift $\mathcal{X}$)
- Given a vector $\boldsymbol{x} \in \mathcal{X}$, denote $\boldsymbol{x}[r]$ its r-th coordinate
- There is no coordinate r s.t. $\boldsymbol{x}[r] = 0 \ \forall \boldsymbol{x} \in \mathcal{X}$ (or drop d)

Lifting

The **lifting** of $\mathcal{X}$ is the $d+1$ dimensional set

$$\tilde{\mathcal{X}} := \left\{ \begin{pmatrix} \lambda \\ \boldsymbol{y} \end{pmatrix} : \lambda \in [0,1], \boldsymbol{y} \in \lambda \mathcal{X} \right\}$$



Lifted utilities

Because we will operate on the lifted strategy space $\tilde{\mathcal{X}}$, we will need a way to **lift utilities** as well!

- Let $\boldsymbol{x}^{(t)} \in \mathcal{X}$ be the last-output strategy
- The lifted utility is defined as

$$\tilde{\boldsymbol{u}}^{(t)} := \begin{bmatrix} -\langle \boldsymbol{u}^{(t)}, \boldsymbol{x}^{(t)} \rangle \\ \boldsymbol{u}^{(t)} \end{bmatrix}$$

Lifted utilities

Because we will operate on the lifted strategy space $\tilde{\mathcal{X}}$, we will need a way to **lift utilities** as well!

- Let $\boldsymbol{x}^{(t)} \in \mathcal{X}$ be the last-output strategy
- The lifted utility is defined as

$$\tilde{\boldsymbol{u}}^{(t)} := \begin{bmatrix} -\langle \boldsymbol{u}^{(t)}, \boldsymbol{x}^{(t)} \rangle \\ \boldsymbol{u}^{(t)} \end{bmatrix}$$

Important observation

$$\left\langle \tilde{\boldsymbol{u}}^{(t)}, \begin{bmatrix} 1 \\ \boldsymbol{x}^{(t)} \end{bmatrix} \right\rangle = 0$$

Log regularization

The **logarithmic regularizer** for $\mathbb{R}^{d+1}$ is

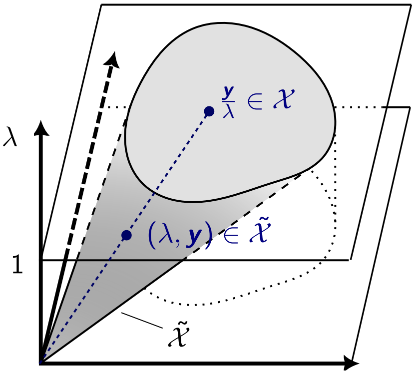
$$\mathcal{R}(\lambda, \boldsymbol{y}) := -\log \lambda - \sum_{r=1}^{d} \log \boldsymbol{y}[r] \qquad (\lambda, \boldsymbol{y}) \in \mathbb{R}^{d+1}_{>0}$$

- **Self-concordant** function, but **not** a barrier for $\tilde{\mathcal{X}}$

Normalization

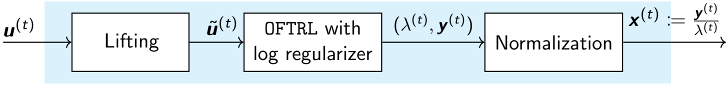
Iterates produced on the lifted space $\tilde{\mathcal{X}}$ are then **renormalized** back to $\mathcal{X}$:

$$\tilde{\mathcal{X}} \ni \begin{bmatrix} \lambda \\ \boldsymbol{y} \end{bmatrix} \mapsto \frac{\boldsymbol{y}}{\lambda} \in \mathcal{X}$$



The complete algorithm

Algorithm: Log-Regularized Lifted Optimistic FTRL (LRL-OFTRL)



Data: Learning rate η

Set $\tilde{\boldsymbol{U}}^{(1)}, \boldsymbol{u}^{(0)} \leftarrow \boldsymbol{0} \in \mathbb{R}^{d+1}$

for $t = 1, 2, \dots, T$ **do**

Set $\begin{bmatrix} \lambda^{(t)} \\ \boldsymbol{y}^{(t)} \end{bmatrix} \leftarrow \arg\max_{(\lambda, \boldsymbol{y}) \in \tilde{\mathcal{X}}} \left\{ \eta \left\langle \tilde{\boldsymbol{U}}^{(t)} + \tilde{\boldsymbol{u}}^{(t-1)}, \begin{bmatrix} \lambda \\ \boldsymbol{y} \end{bmatrix} \right\rangle + \log \lambda + \sum_{r=1}^{d} \log \boldsymbol{y}[r] \right\}$ [▷ OFTRL]

Play strategy $\boldsymbol{x}^{(t)} := \frac{\boldsymbol{y}^{(t)}}{\lambda^{(t)}} \in \mathcal{X}$ [▷ Normalization]

Observe $\boldsymbol{u}^{(t)} \in \mathbb{R}^d$

Set $\tilde{\boldsymbol{u}}^{(t)} \leftarrow \begin{bmatrix} -\langle \boldsymbol{u}^{(t)}, \boldsymbol{x}^{(t)} \rangle \\ \boldsymbol{u}^{(t)} \end{bmatrix}$ [▷ Lifting]

Set $\tilde{\boldsymbol{U}}^{(t+1)} \leftarrow \tilde{\boldsymbol{U}}^{(t)} + \tilde{\boldsymbol{u}}^{(t)}$

Regret Analysis

Lifting makes regret nonnegative

- Regret on the original strategy space:

$$\mathrm{Reg}^{(T)} := \max_{\boldsymbol{x}^* \in \mathcal{X}} \sum_{t=1}^{T} \langle \boldsymbol{u}^{(t)}, \boldsymbol{x}^* - \boldsymbol{x}^{(t)} \rangle$$

- Regret on lifted space:

$$\tilde{\mathrm{Reg}}^{(T)} := \max_{(\lambda^*, \boldsymbol{y}^*) \in \tilde{\mathcal{X}}} \sum_{t=1}^{T} \left\langle \tilde{\boldsymbol{u}}^{(t)}, \begin{bmatrix} \lambda^* \\ \boldsymbol{y}^* \end{bmatrix} - \begin{bmatrix} \lambda^{(t)} \\ \boldsymbol{y}^{(t)} \end{bmatrix} \right\rangle$$

- What is the relationship between the two?

Lifting makes regret nonnegative

- Regret on the original strategy space:

$$\text{Reg}^{(T)} := \max_{\boldsymbol{x}^* \in \mathcal{X}} \sum_{t=1}^{T} \langle \boldsymbol{u}^{(t)}, \boldsymbol{x}^* - \boldsymbol{x}^{(t)} \rangle$$

Lifting of utilities: $\tilde{\boldsymbol{u}}^{(t)} := \begin{bmatrix} -\langle \boldsymbol{u}^{(t)}, \boldsymbol{x}^{(t)} \rangle \\ \boldsymbol{u}^{(t)} \end{bmatrix}$

Normalization: $\boldsymbol{x}^{(t)} := \frac{\boldsymbol{y}^{(t)}}{\lambda^{(t)}}$

$$\tilde{\text{Reg}}^{(T)} := \max_{(\lambda^*, \boldsymbol{y}^*) \in \tilde{\mathcal{X}}} \sum_{t=1}^{T} \left\langle \tilde{\boldsymbol{u}}^{(t)}, \begin{bmatrix} \lambda^* \\ \boldsymbol{y}^* \end{bmatrix} - \begin{bmatrix} \lambda^{(t)} \\ \boldsymbol{y}^{(t)} \end{bmatrix} \right\rangle$$

- What is the relationship between the two?

Lifting makes regret nonnegative

- Regret on the original strategy space:

$$\mathrm{Reg}^{(T)} := \max_{\boldsymbol{x}^* \in \mathcal{X}} \sum_{t=1}^{T} \langle \boldsymbol{u}^{(t)}, \boldsymbol{x}^* - \boldsymbol{x}^{(t)} \rangle$$

Lifting of utilities: $\tilde{\boldsymbol{u}}^{(t)} := \begin{bmatrix} -\langle \boldsymbol{u}^{(t)}, \boldsymbol{x}^{(t)} \rangle \\ \boldsymbol{u}^{(t)} \end{bmatrix}$

Normalization: $\boldsymbol{x}^{(t)} := \frac{\boldsymbol{y}^{(t)}}{\lambda^{(t)}}$

$$\tilde{\mathrm{Reg}}^{(T)} := \max_{(\lambda^*, \boldsymbol{y}^*) \in \tilde{\mathcal{X}}} \sum_{t=1}^{T} \left\langle \tilde{\boldsymbol{u}}^{(t)}, \begin{bmatrix} \lambda^* \\ \boldsymbol{y}^* \end{bmatrix} - \begin{bmatrix} \lambda^{(t)} \\ \boldsymbol{y}^{(t)} \end{bmatrix} \right\rangle$$

- What is the relationship between the two?

Result

$$\tilde{\mathrm{Reg}}^{(T)} = \max\{0, \mathrm{Reg}^{(T)}\}$$

Lifting makes regret nonnegative

Result

$$\tilde{\text{Reg}}^{(T)} = \max\{0, \text{Reg}^{(T)}\}$$

Consequences:

Lifting makes regret nonnegative

Result

$$\tilde{\text{Reg}}^{(T)} = \max\{0, \text{Reg}^{(T)}\}$$

Consequences:

1. $\text{Reg}^{(T)} \leq \tilde{\text{Reg}}^{(T)}$
   - Any algorithm that guarantees small regret on the lifted space $\tilde{\mathcal{X}}$ automatically guarantees small regret on $\mathcal{X}$

Lifting makes regret nonnegative

Result

$$\tilde{\text{Reg}}^{(T)} = \max\{0, \text{Reg}^{(T)}\}$$

Consequences:

1. $\text{Reg}^{(T)} \leq \tilde{\text{Reg}}^{(T)}$
   - Any algorithm that guarantees small regret on the lifted space $\tilde{\mathcal{X}}$ automatically guarantees small regret on $\mathcal{X}$

2. $\tilde{\text{Reg}}^{(T)} \geq 0$
   - The lifted regret is always nonnegative

What do we have at this point?

We are not done

While we have established nonnegative regret in the lifted space, we cannot invoke the result we mentioned earlier

What do we have at this point?

We are not done

While we have established nonnegative regret in the lifted space, we cannot invoke the result we mentioned earlier

Utilities might not be Lipschitz continuous

The utilities are in response of the normalized $\boldsymbol{x}^{(t)} = \boldsymbol{y}^{(t)}/\lambda^{(t)}$, but the iterates produced on the lifted space are $(\lambda^{(t)}, \boldsymbol{y}^{(t)})$.
In other words we:

- **have** $\left\| \tilde{\boldsymbol{u}}^{(t)} - \tilde{\boldsymbol{u}}^{(t-1)} \right\|_* \le L \left\| \dfrac{\boldsymbol{y}^{(t)}}{\lambda^{(t)}} - \dfrac{\boldsymbol{y}^{(t-1)}}{\lambda^{(t-1)}} \right\|$
- **want** $\left\| \tilde{\boldsymbol{u}}^{(t)} - \tilde{\boldsymbol{u}}^{(t-1)} \right\|_* \le L \left\| \begin{bmatrix} \lambda^{(t)} \\ \boldsymbol{y}^{(t)} \end{bmatrix} - \begin{bmatrix} \lambda^{(t-1)} \\ \boldsymbol{y}^{(t-1)} \end{bmatrix} \right\|$

If the λ's are very small, what we have is far from what we want

Log regularization leads to multiplicative stability

This is where the choice of optimistic FTRL with log regularizer comes in!

Log regularization leads to multiplicative stability

Multiplicative stability

Logarithmic regularization guarantees **multiplicative stability**:

$$1 - \eta \lessapprox \frac{\lambda^{(t+1)}}{\lambda^{(t)}} \lessapprox 1 + \eta, \qquad 1 - \eta \lessapprox \frac{\boldsymbol{y}^{(t+1)}[r]}{\boldsymbol{y}^{(t)}[r]} \lessapprox 1 + \eta$$

1. OFTRL dynamics are **locally** stable:

$$\begin{bmatrix} \lambda^{(t+1)} - \lambda^{(t)} \\ \boldsymbol{y}^{(t+1)} - \boldsymbol{y}^{(t)} \end{bmatrix}^\top \nabla^2 \mathcal{R}(\lambda^{(t)}, \boldsymbol{y}^{(t)}) \begin{bmatrix} \lambda^{(t+1)} - \lambda^{(t)} \\ \boldsymbol{y}^{(t+1)} - \boldsymbol{y}^{(t)} \end{bmatrix} \lessapprox \eta^2$$

Log regularization leads to multiplicative stability

Multiplicative stability

Logarithmic regularization guarantees **multiplicative stability**:

$$1 - \eta \lessapprox \frac{\lambda^{(t+1)}}{\lambda^{(t)}} \lessapprox 1 + \eta, \qquad 1 - \eta \lessapprox \frac{\boldsymbol{y}^{(t+1)}[r]}{\boldsymbol{y}^{(t)}[r]} \lessapprox 1 + \eta$$

1. OFTRL dynamics are **locally** stable:

$$\begin{bmatrix} \lambda^{(t+1)} - \lambda^{(t)} \\ \boldsymbol{y}^{(t+1)} - \boldsymbol{y}^{(t)} \end{bmatrix}^\top \nabla^2 \mathcal{R}(\lambda^{(t)}, \boldsymbol{y}^{(t)}) \begin{bmatrix} \lambda^{(t+1)} - \lambda^{(t)} \\ \boldsymbol{y}^{(t+1)} - \boldsymbol{y}^{(t)} \end{bmatrix} \lessapprox \eta^2$$

2. The Hessian of the log regularizer is

$$\nabla^2 \mathcal{R}(\lambda, \boldsymbol{y}) = \mathrm{diag}(\lambda^{-2}, \boldsymbol{y}[1]^{-2}, \ldots, \boldsymbol{y}[d]^{-2})$$

Log regularization leads to multiplicative stability

Multiplicative stability

Logarithmic regularization guarantees **multiplicative stability**:

$$1 - \eta \lessapprox \frac{\lambda^{(t+1)}}{\lambda^{(t)}} \lessapprox 1 + \eta, \qquad 1 - \eta \lessapprox \frac{\boldsymbol{y}^{(t+1)}[r]}{\boldsymbol{y}^{(t)}[r]} \lessapprox 1 + \eta$$

1. OFTRL dynamics are **locally** stable:

$$\begin{bmatrix} \lambda^{(t+1)} - \lambda^{(t)} \\ \boldsymbol{y}^{(t+1)} - \boldsymbol{y}^{(t)} \end{bmatrix}^\top \nabla^2 \mathcal{R}(\lambda^{(t)}, \boldsymbol{y}^{(t)}) \begin{bmatrix} \lambda^{(t+1)} - \lambda^{(t)} \\ \boldsymbol{y}^{(t+1)} - \boldsymbol{y}^{(t)} \end{bmatrix} \lessapprox \eta^2$$

2. The Hessian of the log regularizer is

$$\nabla^2 \mathcal{R}(\lambda, \boldsymbol{y}) = \mathrm{diag}(\lambda^{-2}, \boldsymbol{y}[1]^{-2}, \ldots, \boldsymbol{y}[d]^{-2})$$

3. Combining the two, we find

$$\left(\frac{\lambda^{(t+1)}}{\lambda^{(t)}} - 1 \right)^2 + \sum_{r=1}^{d} \left(\frac{\boldsymbol{y}^{(t+1)}[r]}{\boldsymbol{y}^{(t)}[r]} - 1 \right)^2 \lessapprox \eta^2 \implies \left| \frac{\boldsymbol{y}^{(t+1)}[r]}{\boldsymbol{y}^{(t)}[r]} - 1 \right| \lessapprox \eta$$

Multiplicative stability

Multiplicative stability enables us to **transfer** smoothness guarantees in the lifted space to to original space

Multiplicative stability

Multiplicative stability enables us to **transfer** smoothness guarantees in the lifted space to to original space

In particular, we can establish the following RVU bound

$$0 \leq \tilde{\text{Reg}}^{(T)} \lessapprox \frac{\log T}{\eta} + \eta \sum_{t=1}^{T} \left\| \boldsymbol{u}^{(t+1)} - \boldsymbol{u}^{(t)} \right\|_{\infty}^{2} - \frac{1}{\eta} \sum_{t=1}^{T} \left\| \boldsymbol{x}^{(t+1)} - \boldsymbol{x}^{(t)} \right\|_{1}^{2}$$

and from here conclude that

1. Bounded social square path length

$$\sum_{t=1}^{T} \left\| \boldsymbol{x}^{(t+1)} - \boldsymbol{x}^{(t)} \right\|_{1}^{2} \lessapprox \log T$$

Multiplicative stability

Multiplicative stability enables us to **transfer** smoothness guarantees in the lifted space to to original space

In particular, we can establish the following RVU bound

$$0 \le \tilde{\text{Reg}}^{(T)} \lessapprox \frac{\log T}{\eta} + \eta \sum_{t=1}^{T} \left\| \boldsymbol{u}^{(t+1)} - \boldsymbol{u}^{(t)} \right\|_\infty^2 - \frac{1}{\eta} \sum_{t=1}^{T} \left\| \boldsymbol{x}^{(t+1)} - \boldsymbol{x}^{(t)} \right\|_1^2$$

and from here conclude that

1. Bounded social square path length

$$\sum_{t=1}^{T} \left\| \boldsymbol{x}^{(t+1)} - \boldsymbol{x}^{(t)} \right\|_1^2 \lessapprox \log T$$

2. ... And in turn, bounded individual regret

$$\text{Reg}_i^{(T)} \le \tilde{\text{Reg}}_i^{(T)} \lessapprox \log T$$

Exact regret bound

Regret bound

When player $i \in \{1, \ldots, n\}$ plays on a strategy set $\mathcal{X} \subseteq \mathbb{R}^d$ with L-Lipschitz utilities bounded by B and using learning rate

$$\eta = \min\left\{\frac{1}{256}, \frac{1}{128nL\|\mathcal{X}\|_1^2}\right\}$$

then the following regret bounds holds at any T:

$$\mathrm{Reg}_i^{(T)} \leq c \log T$$

where

$$c := B\|\mathcal{X}\|_1\big(12 + 256(d+1)\max\{nL\|\mathcal{X}\|_1^2, 2\}\big)$$

Comparison table

Method	Applies to	Regret bound	Cost per iteration
OFTRL / OMD [Syrgkanis et al., 2015]	General convex set	$O(\sqrt{n}\mathfrak{R}T^{1/4})$	Regularizer- & oracle- dependent
OMWU [Daskalakis et al., 2021]	Simplex Δ^d	$O(n \log d \log^4 T)$	$O(d)$
Clairvoyant MWU [Piliouras et al., 2022]	Simplex Δ^d	$O(n \log d \log T)$ (subsequence)	$O(d)$
Kernelized OMWU [Farina et al., 2022]	Polytope $\Omega = \mathrm{co}\mathcal{V}$ with $\mathcal{V} \subseteq \{0,1\}^d$	$O(n \log \lvert\mathcal{V}\rvert \log^4 T)$	$d \times$ cost of kernel
`LRL-OFTRL` [**This talk**]	General convex set $\mathcal{X} \subseteq \mathbb{R}^d$	$O(nd\lVert\mathcal{X}\rVert_1^3 \log T)$	Oracle-dependent: • $O(\log\log T)$ proximal oracle calls • $O(\text{poly } T)$ linear opt. oracle calls

where:

- n: number of players
- T: number of iterations/repetitions of the game
- $\mathfrak{R}$: regularizer-dependent parameter
- $\mathrm{co}\mathcal{V}$: convex hull of $\mathcal{V}$
- $\lVert\mathcal{X}\rVert_1$: upper bound on $\max_{\boldsymbol{x}\in\mathcal{X}} \lVert\boldsymbol{x}\rVert_1$

Implementation and Iteration Complexity

The proximal step

Algorithm: Log-Regularized Lifted Optimistic FTRL (LRL-OFTRL)

Data: Learning rate η

Set $\tilde{\boldsymbol{U}}^{(1)}, \boldsymbol{u}^{(0)} \leftarrow \mathbf{0} \in \mathbb{R}^{d+1}$

for $t = 1, 2, \dots, T$ **do**

Set $\begin{bmatrix} \lambda^{(t)} \\ \boldsymbol{y}^{(t)} \end{bmatrix} \leftarrow \underset{(\lambda, \boldsymbol{y}) \in \tilde{\mathcal{X}}}{\arg\max} \left\{ \eta \left\langle \tilde{\boldsymbol{U}}^{(t)} + \tilde{\boldsymbol{u}}^{(t-1)}, \begin{bmatrix} \lambda \\ \boldsymbol{y} \end{bmatrix} \right\rangle + \log \lambda + \sum_{r=1}^{d} \log \boldsymbol{y}[r] \right\}$ [▷ OFTRL]

Play strategy $\boldsymbol{x}^{(t)} := \dfrac{\boldsymbol{y}^{(t)}}{\lambda^{(t)}} \in \mathcal{X}$ [▷ Normalization]

Observe $\boldsymbol{u}^{(t)} \in \mathbb{R}^d$

Set $\tilde{\boldsymbol{u}}^{(t)} \leftarrow \begin{bmatrix} -\langle \boldsymbol{u}^{(t)}, \boldsymbol{x}^{(t)} \rangle \\ \boldsymbol{u}^{(t)} \end{bmatrix}$ [▷ Lifting]

Set $\tilde{\boldsymbol{U}}^{(t+1)} \leftarrow \tilde{\boldsymbol{U}}^{(t)} + \tilde{\boldsymbol{u}}^{(t)}$

The proximal step

Algorithm: Log-Regularized Lifted Optimistic FTRL (`LRL-OFTRL`)

Data: Learning rate η

Set $\tilde{\boldsymbol{U}}^{(1)}, \boldsymbol{u}^{(0)} \leftarrow \boldsymbol{0} \in \mathbb{R}^{d+1}$

for $t = 1, 2, \ldots, T$ **do**

$$\text{Set } \begin{bmatrix} \lambda^{(t)} \\ \boldsymbol{y}^{(t)} \end{bmatrix} \leftarrow \arg\max_{(\lambda, \boldsymbol{y}) \in \tilde{\mathcal{X}}} \left\{ \eta \left\langle \tilde{\boldsymbol{U}}^{(t)} + \tilde{\boldsymbol{u}}^{(t-1)}, \begin{bmatrix} \lambda \\ \boldsymbol{y} \end{bmatrix} \right\rangle + \log \lambda + \sum_{r=1}^{d} \log \boldsymbol{y}[r] \right\}$$

[▷ `OFTRL`]

Strictly concave **nonsmooth** problem

How fast can we compute the proximal step for a generic $\mathcal{X}$?

Complications:

1. Gradients of the log regularizer diverge

The proximal step

Algorithm: Log-Regularized Lifted Optimistic FTRL (LRL-OFTRL)

Data: Learning rate η

Set $\tilde{\boldsymbol{U}}^{(1)}, \boldsymbol{u}^{(0)} \leftarrow \mathbf{0} \in \mathbb{R}^{d+1}$

for $t = 1, 2, \dots, T$ **do**

Set $\begin{bmatrix} \lambda^{(t)} \\ \boldsymbol{y}^{(t)} \end{bmatrix} \leftarrow \arg\max_{(\lambda, \boldsymbol{y}) \in \tilde{\mathcal{X}}} \left\{ \eta \left\langle \tilde{\boldsymbol{U}}^{(t)} + \tilde{\boldsymbol{u}}^{(t-1)}, \begin{bmatrix} \lambda \\ \boldsymbol{y} \end{bmatrix} \right\rangle + \log \lambda + \sum_{r=1}^{d} \log \boldsymbol{y}[r] \right\}$ [▷ OFTRL]

Strictly concave **nonsmooth** problem

How fast can we compute the proximal step for a generic $\mathcal{X}$?

Complications:

1. Gradients of the log regularizer diverge
2. Log regularizer is *not* a barrier function

The proximal step

Algorithm: Log-Regularized Lifted Optimistic FTRL (LRL-OFTRL)

Data: Learning rate η

Set $\tilde{\boldsymbol{U}}^{(1)}, \boldsymbol{u}^{(0)} \leftarrow \boldsymbol{0} \in \mathbb{R}^{d+1}$

for $t = 1, 2, \ldots, T$ **do**

Set $\begin{bmatrix} \lambda^{(t)} \\ \boldsymbol{y}^{(t)} \end{bmatrix} \leftarrow \underset{(\lambda, \boldsymbol{y}) \in \tilde{\mathcal{X}}}{\arg\max} \left\{ \eta \left\langle \tilde{\boldsymbol{U}}^{(t)} + \tilde{\boldsymbol{u}}^{(t-1)}, \begin{bmatrix} \lambda \\ \boldsymbol{y} \end{bmatrix} \right\rangle + \log \lambda + \sum_{r=1}^{d} \log \boldsymbol{y}[r] \right\}$ [▷ OFTRL]

Strictly concave **nonsmooth** problem

How fast can we compute the proximal step for a generic $\mathcal{X}$?

Complications:

1. Gradients of the log regularizer diverge
2. Log regularizer is *not* a barrier function
3. What happens to the guarantees if the solutions are only approximated? ⚠ Additive apx guarantees **not** enough

What we need

- We cannot seek additive error guarantees

What we need

- We cannot seek additive error guarantees
- Instead, we seek **relative** (i.e., multiplicative) error guarantees

$$1 - \epsilon^{(t)} \leq \frac{\lambda^{(t)}}{\lambda_\star^{(t)}} \leq 1 + \epsilon^{(t)}, \qquad 1 - \epsilon^{(t)} \leq \frac{\boldsymbol{y}^{(t)}[r]}{\boldsymbol{y}_\star^{(t)}[r]} \leq 1 + \epsilon^{(t)}$$

where $\epsilon^{(t)}$ is the approximation error

What we need

- We cannot seek additive error guarantees
- Instead, we seek **relative** (i.e., multiplicative) error guarantees

$$1 - \epsilon^{(t)} \leq \frac{\lambda^{(t)}}{\lambda_\star^{(t)}} \leq 1 + \epsilon^{(t)}, \qquad 1 - \epsilon^{(t)} \leq \frac{\boldsymbol{y}^{(t)}[r]}{\boldsymbol{y}_\star^{(t)}[r]} \leq 1 + \epsilon^{(t)}$$

  where $\epsilon^{(t)}$ is the approximation error
- As long as $\epsilon^{(t)} = O(1/T)$, regret degradation is $O(1)$

What we need

- We cannot seek additive error guarantees
- Instead, we seek **relative** (i.e., multiplicative) error guarantees

$$1 - \epsilon^{(t)} \leq \frac{\lambda^{(t)}}{\lambda_\star^{(t)}} \leq 1 + \epsilon^{(t)}, \qquad 1 - \epsilon^{(t)} \leq \frac{\boldsymbol{y}^{(t)}[r]}{\boldsymbol{y}_\star^{(t)}[r]} \leq 1 + \epsilon^{(t)}$$

  where $\epsilon^{(t)}$ is the approximation error
- As long as $\epsilon^{(t)} = O(1/T)$, regret degradation is $O(1)$

What we need

- We cannot seek additive error guarantees
- Instead, we seek **relative** (i.e., multiplicative) error guarantees

$$1 - \epsilon^{(t)} \leq \frac{\lambda^{(t)}}{\lambda_\star^{(t)}} \leq 1 + \epsilon^{(t)}, \qquad 1 - \epsilon^{(t)} \leq \frac{\boldsymbol{y}^{(t)}[r]}{\boldsymbol{y}_\star^{(t)}[r]} \leq 1 + \epsilon^{(t)}$$

  where $\epsilon^{(t)}$ is the approximation error
- As long as $\epsilon^{(t)} = O(1/T)$, regret degradation is $O(1)$

Newton method

We can achieve all these properties efficiently by using a modification of Newton method with **quadratic convergence**, even if $\mathcal{R}(\lambda, \boldsymbol{y})$ is *not* a self-concordant barrier

Proximal Newton method

Requirements

Proximal Newton algorithm requires a **local proximal oracle**

$$\Pi_{\tilde{\boldsymbol{w}}}(\tilde{\boldsymbol{g}}) := \arg\min_{\tilde{\boldsymbol{x}} \in \tilde{\mathcal{X}}} \left\{ \tilde{\boldsymbol{g}}^\top \tilde{\boldsymbol{x}} + \frac{1}{2}(\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{w}})^\top \nabla^2 \mathcal{R}(\tilde{\boldsymbol{w}})(\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{w}}) \right\}$$

$$= \arg\min_{\tilde{\boldsymbol{x}} \in \tilde{\mathcal{X}}} \left\{ \tilde{\boldsymbol{g}}^\top \tilde{\boldsymbol{x}} + \frac{1}{2} \sum_{r=1}^{d+1} \left(\frac{\tilde{\boldsymbol{x}}[r]}{\tilde{\boldsymbol{w}}[r]} - 1 \right)^2 \right\}$$

for arbitrary centers $\tilde{\boldsymbol{w}} \in \mathbb{R}^{d+1}_{>0}$ and gradients $\tilde{\boldsymbol{g}} \in \mathbb{R}^{d+1}$.

Proximal Newton method

Requirements

Proximal Newton algorithm requires a **local proximal oracle**

$$\Pi_{\tilde{\boldsymbol{w}}}(\tilde{\boldsymbol{g}}) := \arg\min_{\tilde{\boldsymbol{x}} \in \tilde{\mathcal{X}}} \left\{ \tilde{\boldsymbol{g}}^\top \tilde{\boldsymbol{x}} + \frac{1}{2}(\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{w}})^\top \nabla^2 \mathcal{R}(\tilde{\boldsymbol{w}})(\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{w}}) \right\}$$
$$= \arg\min_{\tilde{\boldsymbol{x}} \in \tilde{\mathcal{X}}} \left\{ \tilde{\boldsymbol{g}}^\top \tilde{\boldsymbol{x}} + \frac{1}{2} \sum_{r=1}^{d+1} \left(\frac{\tilde{\boldsymbol{x}}[r]}{\tilde{\boldsymbol{w}}[r]} - 1 \right)^2 \right\}$$

for arbitrary centers $\tilde{\boldsymbol{w}} \in \mathbb{R}^{d+1}_{>0}$ and gradients $\tilde{\boldsymbol{g}} \in \mathbb{R}^{d+1}$.

In **normal-form** and **extensive-form** games, $\Pi_{\tilde{\boldsymbol{w}}}(\tilde{\boldsymbol{g}})$ can be implemented *exactly* in poly(d) time for any $\tilde{\boldsymbol{w}} \in \mathbb{R}^{d+1}_{>0}$, $\tilde{\boldsymbol{g}} \in \mathbb{R}^{d+1}$

Guarantees with local proximal oracle [Tran-Dinh et al., 2015]

Given $\epsilon > 0$, it is possible to compute $(\lambda^{(t)}, \boldsymbol{y}^{(t)})$ with relative ϵ approximation in $O(\log\log(1/\epsilon))$ operations and $O(\log\log(1/\epsilon))$ calls to the local proximal oracle

This explains the mentioned $O(\log\log T)$ per-iteration complexity

Guarantees with linear optimization oracle

What if we do **not** know how to construct a local proximal oracle for our set at hand $\mathcal{X}$?

Guarantees with linear optimization oracle

What if we do **not** know how to construct a local proximal oracle for our set at hand $\mathcal{X}$?

Linear optimization oracle

$$\mathcal{L}_{\mathcal{X}}(\boldsymbol{u}) := \arg\max_{\boldsymbol{x} \in \mathcal{X}} \langle \boldsymbol{x}, \boldsymbol{u} \rangle.$$

Guarantees with linear optimization oracle

What if we do **not** know how to construct a local proximal oracle for our set at hand $\mathcal{X}$?

Linear optimization oracle

$$\mathcal{L}_{\mathcal{X}}(\boldsymbol{u}) \coloneqq \arg\max_{\boldsymbol{x}\in\mathcal{X}} \langle \boldsymbol{x}, \boldsymbol{u} \rangle.$$

Frank-Wolfe Newton [Liu et al., 2020]

Given any $\epsilon > 0$, it is possible to compute $(\lambda^{(t)}, \boldsymbol{y}^{(t)})$ with relative ϵ approximation in $O(\text{poly}(1/\epsilon))$ operations and $O(\text{poly}(1/\epsilon))$ calls to the linear optimization oracle

Zooming Out

Important takeaway messages

1. We developed `LRL-OFTRL`, an uncoupled no-regret learning algorithm

Important takeaway messages

1. We developed `LRL-OFTRL`, an uncoupled no-regret learning algorithm
2. When all players in a general convex game employ `LRL-OFTRL`, the regret of each player grows only as $O(\log T)$

Important takeaway messages

1. We developed `LRL-OFTRL`, an uncoupled no-regret learning algorithm
2. When all players in a general convex game employ `LRL-OFTRL`, the regret of each player grows only as $O(\log T)$
3. This significantly extends and strengthens the scope of all prior work

Important takeaway messages

1. We developed LRL-OFTRL, an uncoupled no-regret learning algorithm
2. When all players in a general convex game employ LRL-OFTRL, the regret of each player grows only as $O(\log T)$
3. This significantly extends and strengthens the scope of all prior work
4. Further, our uncoupled no-regret learning dynamics can be efficiently implemented using, for example, a proximal oracle for the underlying feasible set

Some open questions

1. In the special case of normal-form games, `LRL-OFTRL`'s dependence on the dimension is linear as opposed to logarithmic as in Daskalakis et al. [2021]

Some open questions

1. In the special case of normal-form games, `LRL-OFTRL`'s dependence on the dimension is linear as opposed to logarithmic as in Daskalakis et al. [2021]
2. Can entropic regularization (which induces OMWU) be incorporated into our framework?

Some open questions

1. In the special case of normal-form games, LRL-OFTRL's dependence on the dimension is linear as opposed to logarithmic as in Daskalakis et al. [2021]
2. Can entropic regularization (which induces OMWU) be incorporated into our framework?
3. Explore having access to different types of oracles
   - For example, is it possible to use a separation oracle for the underlying set of strategies? If so, the ellipsoid algorithm would be the obvious candidate en route to implementing LRL-OFTRL

Some open questions

1. In the special case of normal-form games, `LRL-OFTRL`'s dependence on the dimension is linear as opposed to logarithmic as in Daskalakis et al. [2021]
2. Can entropic regularization (which induces OMWU) be incorporated into our framework?
3. Explore having access to different types of oracles
   - For example, is it possible to use a separation oracle for the underlying set of strategies? If so, the ellipsoid algorithm would be the obvious candidate en route to implementing `LRL-OFTRL`
4. Is $O(\log T)$ per-player regret tight?

Some open questions

1. In the special case of normal-form games, `LRL-OFTRL`'s dependence on the dimension is linear as opposed to logarithmic as in Daskalakis et al. [2021]
2. Can entropic regularization (which induces OMWU) be incorporated into our framework?
3. Explore having access to different types of oracles
   - For example, is it possible to use a separation oracle for the underlying set of strategies? If so, the ellipsoid algorithm would be the obvious candidate en route to implementing `LRL-OFTRL`
4. Is $O(\log T)$ per-player regret tight?
5. What can be said about swap regret (in normal-form games) and Φ-regret (in extensive-form games)?
   - We are doing some work in that direction

Thank you!

Question? Also, feel free to reach out at

`gfarina@{cs.cmu.edu | meta.com}`