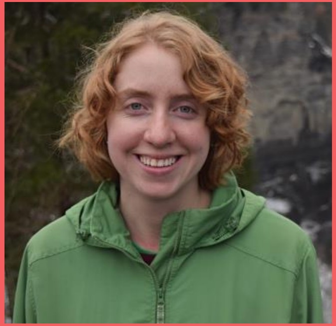
Impact of Decentralized Learning on Player Utilities in Stackelberg Games

Meena Jagadeesan (UC Berkeley)

Joint work with Kate Donahue (Cornell), Nicole Immorlica, Brendan Lucier, Alex Slivkins (MSR)




Published at ICML 2024
Presented at ESIF-AIML 2024

High-level overview of this work

We study **Stackelberg games with decentralized learning**.

Captures systems of two sequential, misaligned agents that learn over time

How do the learning dynamics behave?

Outline for the talk

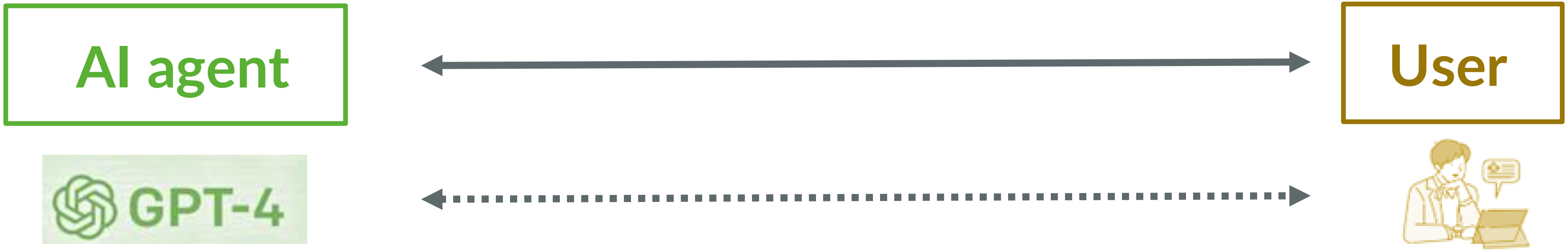
1. Motivation and conceptual overview

2. Model of Stackelberg games with decentralized learning

3. Our analysis of learning dynamics

Motivation: User-AI Interactions

Motivation: User-AI Interactions



Motivation: User-AI Interactions

AI agent

User

GPT-4

YouTube

Motivation: User-AI Interactions

AI agent

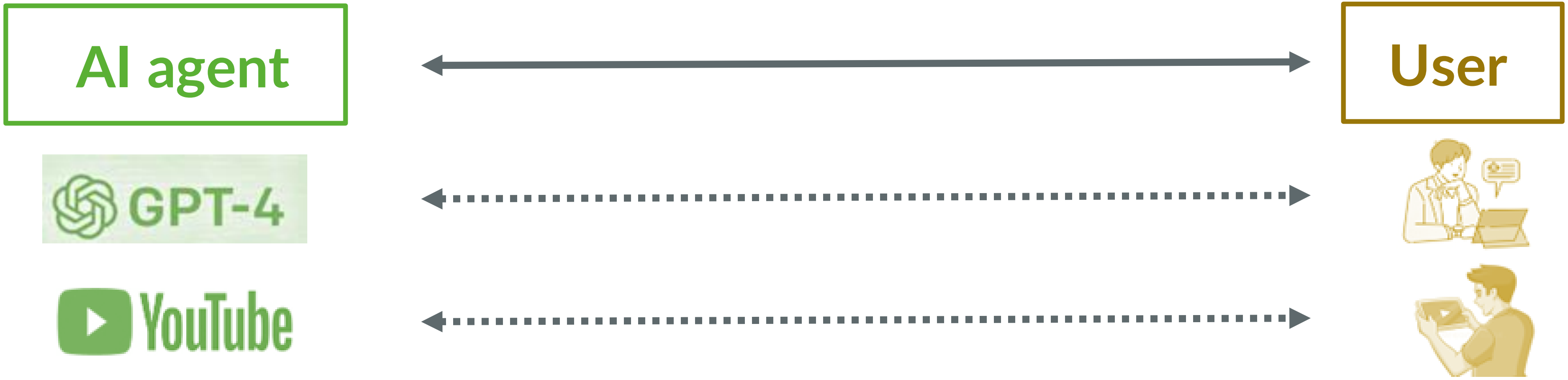
User

GPT-4

YouTube

AI agent learns from repeatedly interacting with a user.

Motivation: User-AI Interactions

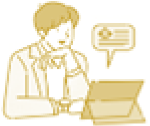


AI agent learns from repeatedly interacting with a user.

User learns from repeatedly interacting with the AI agent.

Motivation: User-AI Interactions

AI agent ⟷ User

GPT-4 

YouTube 

AI agent learns from repeatedly interacting with a user.

User learns from repeatedly interacting with the AI agent.

User-AI interactions are a form of multi-agent learning.

Our focus: Sequential, Misaligned, Decentralized Systems

User **(leader)**

GPT-4 Chatbot **(follower)**

Our focus: Sequential, Misaligned, Decentralized Systems

*User **(leader)***

GPT-4 *Chatbot **(follower)***

M **You**

Prompt

Sequential: User leads (w/ prompt)

Our focus: Sequential, Misaligned, Decentralized Systems

*User **(leader)*** ⟷ GPT-4 *Chatbot **(follower)***

You — Prompt

ChatGPT — Output

Sequential: User leads (w/ prompt); chatbot follows (w/ output).

Our focus: Sequential, Misaligned, Decentralized Systems

*User **(leader)*** ⟷ GPT-4 *Chatbot **(follower)***

You — Prompt

ChatGPT — Output

Sequential: User leads (w/ prompt); chatbot follows (w/ output).

Misaligned: User utility (individual preferences) vs. Chatbot utility (e.g., societal preferences, implicit objective learned during training)

Our focus: Sequential, Misaligned, Decentralized Systems

*User **(leader)*** ⟷ GPT-4 *Chatbot **(follower)***

You Prompt — **ChatGPT** Output — **You** Prompt — **ChatGPT** Output

Sequential: User leads (w/ prompt); chatbot follows (w/ output).

Misaligned: User utility (individual preferences) vs. Chatbot utility (e.g., societal preferences, implicit objective learned during training)

Decentralized: User + chatbot learn separately over a chat session.

Main question

We study **Stackelberg games with decentralized learning**, which are:

- **Sequential**: One agent leads, and the other agent follows.
- **Misaligned:** Agents have different utility functions.
- **Decentralized:** Agents learn separately from repeated interactions.

Main question

We study **Stackelberg games with decentralized learning**, which are:

- **Sequential**: One agent leads, and the other agent follows.
- **Misaligned:** Agents have different utility functions.
- **Decentralized:** Agents learn separately from repeated interactions.

How do the learning dynamics behave?

Main question

We study **Stackelberg games with decentralized learning**, which are:

- **Sequential**: One agent leads, and the other agent follows.
- **Misaligned:** Agents have different utility functions.
- **Decentralized:** Agents learn separately from repeated interactions.

How do the learning dynamics behave?

- Implications for **each agent's cumulative utility over time?**

Main question

We study **Stackelberg games with decentralized learning**, which are:

- **Sequential**: One agent leads, and the other agent follows.
- **Misaligned:** Agents have different utility functions.
- **Decentralized:** Agents learn separately from repeated interactions.

How do the learning dynamics behave?

- Implications for **each agent's cumulative utility over time?**
- Implications for **learning algorithm design**?

Overview of our contributions

Misalignment in agent utilities distorts the learning dynamics.

- When agents can be arbitrarily misaligned, we show the (full-information) Stackelberg equilibrium utilities are **unachievable**.

- We develop **relaxed benchmarks** for each agent's utility, and construct algorithms that perform well for both agents w.r.t. these benchmarks.

- When agents are partially aligned, we show the Stackelberg equilibrium utilities can be achieved.

Outline for the talk

1. Motivation and conceptual overview
2. **Model of Stackelberg games with decentralized learning**
3. Our analysis of learning dynamics

Overview of our model

- ❑ **Sequential** (one agent goes first)
- ❑ **Misaligned** (agent utilities are not equal)
- ❑ **Decentralized** (agents learn separately)

Overview of our model

- ✓ **Sequential** (one agent goes first)
- ✓ **Misaligned** (agent utilities are not equal)
- X **Decentralized** (agents learn separately)

Stackelberg games: sequential, misaligned, **static** environments

Overview of our model

✓ **Sequential** (one agent goes first)
✓ **Misaligned** (agent utilities are not equal)
✓ **Decentralized** (agents learn separately)

Stackelberg games: sequential, misaligned, **static** environments

Our model: Stackelberg games with decentralized learning

Overview of our model

- ✓ **Sequential** (one agent goes first)
- ✓ **Misaligned** (agent utilities are not equal)
- ✓ **Decentralized** (agents learn separately)

Stackelberg games: sequential, misaligned, **static** environments

Our model: Stackelberg games with decentralized learning

- Agents interact over T rounds.
- At every round, the leader goes first, and the follower goes second.
- Each agent observes their own **stochastic rewards**.

Recap of static Stackelberg games

	Action	Utility
Leader	$a \in A$	$u_1(a, b)$
Follower	$b \in B$	$u_2(a, b)$

Follower best-responds to leader: $b^*(a) = argmax_{b \in B} u_2(a, b)$

Leader **anticipates follower's actions** and best-responds:

$$a^* = argmax_{a \in A} u_1(a, b^*(a))$$

Recap of static Stackelberg games

	Action	Utility
Leader	$a \in A$	$u_1(a, b)$
Follower	$b \in B$	$u_2(a, b)$

Follower best-responds to leader: $b^*(a) = argmax_{b \in B} u_2(a, b)$

Requires full knowledge of u_2

Leader **anticipates follower's actions** and best-responds:

$$a^* = argmax_{a \in A} u_1(a, b^*(a))$$

Requires full knowledge of u_1 and b^*

Stackelberg games with decentralized learning

Each agent **learns** how to select actions over T rounds.

Stackelberg games with decentralized learning

Each agent **learns** how to select actions over T rounds.

We cast agent learning within the **stochastic multi-armed bandit framework**:

Leader

A = arms, u_1 = mean reward function

$\boldsymbol{ALG_1}$ **= bandit algorithm**

Follower

B = arms, u_2 = mean reward function

$\boldsymbol{ALG_2}$ **= bandit algorithm**

Stackelberg games with decentralized learning

Each agent **learns** how to select actions over T rounds.

We cast agent learning within the **stochastic multi-armed bandit framework**:

Leader

A = arms, u_1 = mean reward function

$\boldsymbol{ALG_1}$ **= bandit algorithm**

Follower

B = arms, u_2 = mean reward function

$\boldsymbol{ALG_2}$ **= bandit algorithm**

At each time step t:

Stackelberg games with decentralized learning

Each agent **learns** how to select actions over T rounds.

We cast agent learning within the **stochastic multi-armed bandit framework**:

Leader

A = arms, u_1 = mean reward function

$\boldsymbol{ALG_1}$ **= bandit algorithm**

Follower

B = arms, u_2 = mean reward function

$\boldsymbol{ALG_2}$ **= bandit algorithm**

At each time step t:

Chooses action a_t using $\boldsymbol{ALG_1}$

Stackelberg games with decentralized learning

Each agent **learns** how to select actions over T rounds.

We cast agent learning within the **stochastic multi-armed bandit framework**:

Leader

A = arms, u_1 = mean reward function

$\boldsymbol{ALG_1}$ **= bandit algorithm**

Follower

B = arms, u_2 = mean reward function

$\boldsymbol{ALG_2}$ **= bandit algorithm**

At each time step t:

Chooses action a_t using $\boldsymbol{ALG_1}$

Observes a_t & chooses b_t using $\boldsymbol{ALG_2}$

Stackelberg games with decentralized learning

Each agent **learns** how to select actions over T rounds.

We cast agent learning within the **stochastic multi-armed bandit framework**:

Leader

A = arms, u_1 = mean reward function

$\boldsymbol{ALG_1}$ **= bandit algorithm**

Follower

B = arms, u_2 = mean reward function

$\boldsymbol{ALG_2}$ **= bandit algorithm**

At each time step t:

Leader:

Chooses action a_t using $\boldsymbol{ALG_1}$

Observe **stochastic reward** $u_1(a_t, b_t) + \eta_{1,t}$

Follower:

Observes a_t & chooses b_t using $\boldsymbol{ALG_2}$

Observe **stochastic reward** $u_2(a_t, b_t) + \eta_{2,t}$

Stackelberg games with decentralized learning

Each agent **learns** how to select actions over T rounds.

We cast agent learning within the **stochastic multi-armed bandit framework**:

Leader

A = arms, u_1 = mean reward function

$\boldsymbol{ALG_1}$ **= bandit algorithm**

Follower

B = arms, u_2 = mean reward function

$\boldsymbol{ALG_2}$ **= bandit algorithm**

At each time step t:

Chooses action a_t using $\boldsymbol{ALG_1}$

Observe **stochastic reward** $u_1(a_t, b_t) + \eta_{1,t}$

Observes a_t & chooses b_t using $\boldsymbol{ALG_2}$

Observe **stochastic reward** $u_2(a_t, b_t) + \eta_{2,t}$

Cumulative utility: $\sum_{t=1}^{T} u_1(a_t, b_t)$

Cumulative utility: $\sum_{t=1}^{T} u_2(a_t, b_t)$

Related work

Learning in Stackelberg games with unknown utilities for both agents:

e.g., Camara, Hartline, Johnsen (2020), Bai, Jin, Wang, Xiong (2021), Gan, Han, Wu, Xu (2023), Haghtalab, Podimata, Yang (2023), Collina, Roth, Shao (2023), etc.

Broader literature on learning in Stackelberg games:

e.g., Letchford, Conitzer, Munagala (2009), Balcan, Blum, Haghtalab, Procaccia (2015), Braverman, Mao, Schneider, Weinberg (2018), Fiez, Chasnov, Ratliff (2019), Deng, Schneider, Sivan (2019), Zrnic, Mazumdar, Sastry, Jordan (2021), Kao, Wei, Subramanian (2022), Goktas, Zhao, Greenwald (2022), Haghtalab, Lykouris, Nietert, Wei (2022), Zhao, Zhu, Jiao, Jordan (2023), Brown, Schneider, Vodrahalli (2023), Guruganesh, Kolumbus, Schneider, Talgam-Cohen, Vasileios-Vlatakis-Gkaragkounis (2024), etc.

Interacting learners:

*e.g. Chayes, Immorlica, Jain, Etesami, Mahdian (2007), Chan, Hadfield-Menell, Srinivasa, Dragan (2010), Daskalakis, Deckelbaum, Kim (2011), Borgs, Agarwal, Luo, Neyshabur, Schapire (2017), Aridor, Mansour, Slivkins, Wu (2020), Zhuang, Hadfield-Menell (2020), **J.**, Jordan, Haghtalab (2023), etc.*

Our model: stochastic rewards, utility of **both** agents, **arbitrary** misalignment, decentralized learning

Outline for the talk

1. Motivation and conceptual overview

2. Model of Stackelberg games with decentralized learning

3. **Our analysis of learning dynamics**

Measuring each agent's regret

We study each agent i's expected pseudo-regret: $\mathrm{E}[\sum_{t=1}^{T} u_i(a_t, b_t)] - \alpha_i \cdot T$

Agent i's cumulative utility

Agent i's benchmark

Measuring each agent's regret

We study each agent i's expected pseudo-regret: $\mathrm{E}[\sum_{t=1}^{T} u_i(a_t, b_t)] - \alpha_i \cdot T$

Agent i's cumulative utility

Agent i's benchmark

What benchmarks are appropriate for this environment?

Stackelberg benchmark is unachievable

Naïve benchmark: $\alpha_i^{orig} := u_i(a^*, b^*(a^*))$ (the Stackelberg equilibrium utility)

Stackelberg benchmark is unachievable

Naïve benchmark: $\alpha_i^{orig} := u_i(a^*, b^*(a^*))$ (the Stackelberg equilibrium utility)

Theorem: For any ALG_1 and ALG_2, some agent *i* incurs linear regret w.r.t. α_i^{orig}.

Stackelberg benchmark is unachievable

Naïve benchmark: $\alpha_i^{orig} := u_i(a^*, b^*(a^*))$ (the Stackelberg equilibrium utility)

Theorem: For any ALG_1 and ALG_2, some agent *i* incurs linear regret w.r.t. α_i^{orig}.

	b_1	. b_2
a_1	$(0.6, \delta)$	$(0.2, \mathbf{0})$
a_2	$(0.5, 0.6)$	$(0.4, 0.4)$

	b_1	. b_2
a_1	$(0.6, \delta)$	$(0.2, \mathbf{2\ \delta})$
a_2	$(0.5, 0.6)$	$(0.4, 0.4)$

Pair = $(u_1(a,b), u_2(a,b))$

Stackelberg benchmark is unachievable

Naïve benchmark: $\alpha_i^{orig} := u_i(a^*, b^*(a^*))$ (the Stackelberg equilibrium utility)

Theorem: For any ALG_1 and ALG_2, some agent *i* incurs linear regret w.r.t. α_i^{orig}.

	b_1	. b_2
a_1	$(0.6, \delta)$	$(0.2, \mathbf{0})$
a_2	$(0.5, 0.6)$	$(0.4, 0.4)$

	b_1	. b_2
a_1	$(0.6, \delta)$	$(0.2, \mathbf{2\ \delta})$
a_2	$(0.5, 0.6)$	$(0.4, 0.4)$

Pair = $(u_1(a,b), u_2(a,b))$, Gold = follower's best-response

Stackelberg benchmark is unachievable

Naïve benchmark: $\alpha_i^{orig} := u_i(a^*, b^*(a^*))$ (the Stackelberg equilibrium utility)

Theorem: For any ALG_1 and ALG_2, some agent *i* incurs linear regret w.r.t. α_i^{orig}.

	b_1	. b_2
a_1	$(0.6, \delta)$	$(0.2, \mathbf{0})$
a_2	$(0.5, 0.6)$	$(0.4, 0.4)$

	b_1	. b_2
a_1	$(0.6, \delta)$	$(0.2, \mathbf{2\ \delta})$
a_2	$(0.5, 0.6)$	$(0.4, 0.4)$

Pair = $(u_1(a, b), u_2(a, b))$, Gold = follower's best-response Purple = leader's best-response

Stackelberg benchmark is unachievable

Naïve benchmark: $\alpha_i^{orig} := u_i(a^*, b^*(a^*))$ (the Stackelberg equilibrium utility)

Theorem: For any ALG_1 and ALG_2, some agent *i* incurs linear regret w.r.t. α_i^{orig}.

	b_1	. b_2
a_1	$(0.6, \delta)$	$(0.2, \mathbf{0})$
a_2	$(0.5, 0.6)$	$(0.4, 0.4)$

	b_1	. b_2
a_1	$(0.6, \delta)$	$(0.2, \mathbf{2\ \delta})$
a_2	$(0.5, 0.6)$	$(0.4, 0.4)$

Pair = $(u_1(a,b), u_2(a,b))$, Gold = follower's best-response Purple = leader's best-response

$(\boldsymbol{\alpha_1^{orig}}, \boldsymbol{\alpha_2^{orig}}) = (\mathbf{0.6}, \boldsymbol{\delta})$

$(\boldsymbol{\alpha_1^{orig}}, \boldsymbol{\alpha_2^{orig}}) = (\mathbf{0.5}, \mathbf{0.6})$

Stackelberg benchmark is unachievable

Naïve benchmark: $\alpha_i^{orig} := u_i(a^*, b^*(a^*))$ (the Stackelberg equilibrium utility)

Theorem: For any ALG_1 and ALG_2, some agent *i* incurs linear regret w.r.t. α_i^{orig}.

	b_1	. b_2
a_1	$(0.6, \delta)$	$(0.2, \mathbf{0})$
a_2	$(0.5, 0.6)$	$(0.4, 0.4)$

	b_1	. b_2
a_1	$(0.6, \delta)$	$(0.2, \mathbf{2\ \delta})$
a_2	$(0.5, 0.6)$	$(0.4, 0.4)$

Pair = $(u_1(a,b), u_2(a,b))$, Gold = follower's best-response Purple = leader's best-response

$(\boldsymbol{\alpha_1^{orig}}, \boldsymbol{\alpha_2^{orig}}) = (\mathbf{0.6}, \boldsymbol{\delta})$

$(\boldsymbol{\alpha_1^{orig}}, \boldsymbol{\alpha_2^{orig}}) = (\mathbf{0.5}, \mathbf{0.6})$

Due to misalignment, small errors by one agent can distort the other agent's utility.

Our error-tolerant benchmarks

Account for agent errors via **ϵ-approximate best-response sets**:

<u>**Definition (Error-tolerant benchmarks):**</u>

$\alpha_1^{tol} := \inf_{\epsilon \leq \gamma} \left(\max_{a \in A} \min_{b \in B_\epsilon(a)} u_1(a,b) + \epsilon \right)$ and $\alpha_2^{tol} := \inf_{\epsilon \leq \gamma} \left(\min_{a \in A_\epsilon} \max_{b \in B} u_2(a,b) + \epsilon \right)$

$$B_\epsilon(a) := \left\{ b \in B \mid u_2(a,b) \geq \max_{b' \in B} u_2(a,b') - \epsilon \right\}$$

$$A_\epsilon := \left\{ a \in A \mid \max_{b \in B_\epsilon(a)} u_1(a,b) \geq \max_{a' \in A} \min_{b' \in B_\epsilon(a')} u_1(a',b') - \epsilon \right\}$$

Our error-tolerant benchmarks

Account for agent errors via **ϵ-approximate best-response sets**:

Definition (Error-tolerant benchmarks):

$$\alpha_1^{tol} := \inf_{\epsilon \le \gamma} \left(\underbrace{\max_{a \in A} \min_{b \in B_\epsilon(a)} u_1(a,b)} + \epsilon \right) \text{ and } \alpha_2^{tol} := \inf_{\epsilon \le \gamma} \left(\underbrace{\min_{a \in A_\epsilon} \max_{b \in B} u_2(a,b)} + \epsilon \right)$$

ϵ-relaxed Stackelberg utility

$$B_\epsilon(a) := \left\{ b \in B \mid u_2(a,b) \ge \max_{b' \in B} u_2(a,b') - \epsilon \right\}$$

$$A_\epsilon := \left\{ a \in A \mid \max_{b \in B_\epsilon(a)} u_1(a,b) \ge \max_{a' \in A} \min_{b' \in B_\epsilon(a')} u_1(a',b') - \epsilon \right\}$$

Our error-tolerant benchmarks

Account for agent errors via **ϵ-approximate best-response sets**:

Definition (Error-tolerant benchmarks):

$$\alpha_1^{tol} := \inf_{\epsilon \le \gamma} \left(\max_{a \in A} \min_{b \in B_\epsilon(a)} u_1(a,b) + \epsilon \right) \text{ and } \alpha_2^{tol} := \inf_{\epsilon \le \gamma} \left(\min_{a \in A_\epsilon} \max_{b \in B} u_2(a,b) + \epsilon \right)$$

ϵ-relaxed Stackelberg utility

ϵ-regularizer

$$B_\epsilon(a) := \left\{ b \in B \mid u_2(a,b) \ge \max_{b' \in B} u_2(a,b') - \epsilon \right\}$$

$$A_\epsilon := \left\{ a \in A \mid \max_{b \in B_\epsilon(a)} u_1(a,b) \ge \max_{a' \in A} \min_{b' \in B_\epsilon(a')} u_1(a',b') - \epsilon \right\}$$

Our error-tolerant benchmarks

Account for agent errors via **ϵ-approximate best-response sets**:

Definition (Error-tolerant benchmarks):

$$\alpha_1^{tol} := \underbrace{\inf_{\epsilon \le \gamma}}_{\text{worst-case error level}} \Big(\underbrace{\max_{a \in A} \min_{b \in B_\epsilon(a)} u_1(a,b)}_{\epsilon\text{-relaxed Stackelberg utility}} + \underbrace{\epsilon}_{\epsilon\text{-regularizer}} \Big) \quad \text{and} \quad \alpha_2^{tol} := \underbrace{\inf_{\epsilon \le \gamma}}_{\text{worst-case error level}} \Big(\underbrace{\min_{a \in A_\epsilon} \max_{b \in B} u_2(a,b)}_{\epsilon\text{-relaxed Stackelberg utility}} + \underbrace{\epsilon}_{\epsilon\text{-regularizer}} \Big)$$

worst-case error level

ϵ-relaxed Stackelberg utility

ϵ-regularizer

$$B_\epsilon(a) := \left\{ b \in B \mid u_2(a,b) \ge \max_{b' \in B} u_2(a,b') - \epsilon \right\}$$

$$A_\epsilon := \left\{ a \in A \mid \max_{b \in B_\epsilon(a)} u_1(a,b) \ge \max_{a' \in A} \min_{b' \in B_\epsilon(a')} u_1(a',b') - \epsilon \right\}$$

Example of error-tolerant benchmarks

$\alpha_1^{tol} := \inf_{\epsilon \leq \gamma} \left(\max_{a \in A} \min_{b \in B_\epsilon(a)} u_1(a,b) + \epsilon \right)$ and $\alpha_2^{tol} := \inf_{\epsilon \leq \gamma} \left(\min_{a \in A_\epsilon} \max_{b \in B} u_2(a,b) + \epsilon \right)$

Example of error-tolerant benchmarks

$\alpha_1^{tol} := \inf_{\epsilon \leq \gamma} \left(\max_{a \in A} \min_{b \in B_\epsilon(a)} u_1(a,b) + \epsilon \right)$ and $\alpha_2^{tol} := \inf_{\epsilon \leq \gamma} \left(\min_{a \in A_\epsilon} \max_{b \in B} u_2(a,b) + \epsilon \right)$

	b_1	. b_2
a_1	$(0.6, \delta)$	$(0.2, \mathbf{0})$
a_2	$(0.5, 0.6)$	$(0.4, 0.4)$

	b_1	. b_2
a_1	$(0.6, \delta)$	$(0.2, \mathbf{2\ \delta})$
a_2	$(0.5, 0.6)$	$(0.4, 0.4)$

We take $\delta < \gamma = 0.05$ in this example.

Example of error-tolerant benchmarks

$$\alpha_1^{tol} := \inf_{\epsilon \le \gamma} \left(\max_{a \in A} \min_{b \in B_\epsilon(a)} u_1(a,b) + \epsilon \right) \text{ and } \alpha_2^{tol} := \inf_{\epsilon \le \gamma} \left(\min_{a \in A_\epsilon} \max_{b \in B} u_2(a,b) + \epsilon \right)$$

	b_1	. b_2
a_1	$(0.6, \delta)$	$(0.2, \mathbf{0})$
a_2	$(0.5, 0.6)$	$(0.4, 0.4)$

	b_1	. b_2
a_1	$(0.6, \delta)$	$(0.2, \mathbf{2\ \delta})$
a_2	$(0.5, 0.6)$	$(0.4, 0.4)$

Green is $B_\delta(a)$ Pink is A_δ

$$\boldsymbol{(\alpha_1^{tol}, \alpha_2^{tol}) = (0.5 + \delta, \delta)}$$

$$\boldsymbol{(\alpha_1^{tol}, \alpha_2^{tol}) = (0.5, 3\delta)}$$

We take $\delta < \gamma = 0.05$ in this example.

Our algorithmic results

Goal: design algorithms that achieve sublinear regret for both agents

- Standard algorithms can incur **linear regret** w.r.t the error-tolerant benchmarks.
- We construct algorithms achieving $\tilde{O}\left(T^{\frac{2}{3}}\right)$ regret w.r.t error-tolerant benchmarks.
- Any algorithms incur $\Omega\left(T^{\frac{2}{3}}\right)$ regret for some agent w.r.t error-tolerant benchmarks.

- When agent utilities are partially aligned, we construct algorithms which achieve $\tilde{O}\left(T^{\frac{1}{2}}\right)$ regret w.r.t the **original Stackelberg benchmarks.**

Conclusion

Standard algorithms can incur linear regret

Proposition (Informal): Suppose both agents run $\mathrm{ExploreThenCommit}$. Then **both agents** can **incur linear regret** w.r.t. the error-tolerant benchmarks.

Standard algorithms can incur linear regret

Proposition (Informal): Suppose both agents run $\mathrm{ExploreThenCommit}$. Then **both agents** can **incur linear regret** w.r.t. the error-tolerant benchmarks.

	b_1	b_2
a_1	$(0.6, 0.4)$	$(0.2, 0)$
a_2	$(0.5, 0.3)$	$(0.4, 0.2)$

$$(\boldsymbol{\alpha_1^{tol}}, \boldsymbol{\alpha_2^{tol}}) = (\mathbf{0.6}, \mathbf{0.4})$$

Standard algorithms can incur linear regret

Proposition (Informal): Suppose both agents run $\mathrm{ExploreThenCommit}$. Then **both agents** can **incur linear regret** w.r.t. the error-tolerant benchmarks.

	b_1	b_2
a_1	$(0.6, 0.4)$	$(0.2, 0)$
a_2	$(0.5, 0.3)$	$(0.4, 0.2)$

$$(\boldsymbol{\alpha_1^{tol}}, \boldsymbol{\alpha_2^{tol}}) = (\mathbf{0.6}, \mathbf{0.4})$$

Leader's estimated rewards during exploration:

- $0.5\left(u_1(a_1, b_1) + u_1(a_1, b_2)\right) = \mathbf{0.4}$ on a_1

Standard algorithms can incur linear regret

Proposition (Informal): Suppose both agents run ExploreThenCommit. Then **both agents** can **incur linear regret** w.r.t. the error-tolerant benchmarks.

	b_1	b_2
a_1	$(0.6, 0.4)$	$(0.2, 0)$
a_2	$(0.5, 0.3)$	$(0.4, 0.2)$

$$(\boldsymbol{\alpha_1^{tol}}, \boldsymbol{\alpha_2^{tol}}) = (\mathbf{0.6, 0.4})$$

Leader's estimated rewards during exploration:

- $0.5\left(u_1(a_1, b_1) + u_1(a_1, b_2)\right) = \mathbf{0.4}$ on a_1
- $0.5\left(u_1(a_2, b_1) + u_1(a_2, b_2)\right) = \mathbf{0.45}$ on a_2

Standard algorithms can incur linear regret

Proposition (Informal): Suppose both agents run ExploreThenCommit. Then **both agents** can **incur linear regret** w.r.t. the error-tolerant benchmarks.

	b_1	b_2
a_1	$(0.6, 0.4)$	$(0.2, 0)$
a_2	$(0.5, 0.3)$	$(0.4, 0.2)$

$(\boldsymbol{\alpha_1^{tol}}, \boldsymbol{\alpha_2^{tol}}) = (\mathbf{0.6}, \mathbf{0.4})$

Leader's estimated rewards during exploration:

- $0.5\left(u_1(a_1, b_1) + u_1(a_1, b_2)\right) = \mathbf{0.4}$ on a_1
- $0.5\left(u_1(a_2, b_1) + u_1(a_2, b_2)\right) = \mathbf{0.45}$ on a_2

Leader would commit to a_2

$\Rightarrow$ Both players incur linear regret

Standard algorithms can incur linear regret

Proposition (Informal): Suppose both agents run $\texttt{ExploreThenCommit}$. Then **both agents** can **incur linear regret** w.r.t. the error-tolerant benchmarks.

	b_1	b_2
a_1	$(0.6, 0.4)$	$(0.2, 0)$
a_2	$(0.5, 0.3)$	$(0.4, 0.2)$

$(\alpha_1^{tol}, \alpha_2^{tol}) = (\mathbf{0.6}, \mathbf{0.4})$

Leader's estimated rewards during exploration:

- $0.5\left(u_1(a_1, b_1) + u_1(a_1, b_2)\right) = \mathbf{0.4}$ on a_1
- $0.5\left(u_1(a_2, b_1) + u_1(a_2, b_2)\right) = \mathbf{0.45}$ on a_2

Leader would commit to a_2
$\Rightarrow$ Both players incur linear regret

Key issue: the follower's exploration phase distorts the leader's learning

Warm-up: Modified ETC yields sublinear regret

Key algorithmic idea: leader waits for the follower to finish exploring before learning

Warm-up: Modified ETC yields sublinear regret

Key algorithmic idea: leader waits for the follower to finish exploring before learning

Proposition (Informal): Suppose that:

- The follower runs ExploreThenCommit.
- The leader runs a modified ExploreThenCommit **where they discard observations from the follower's exploration phase** when computing the empirical means.

=> Both agents achieve $\tilde{\boldsymbol{O}}\left(\boldsymbol{T}^{\frac{2}{3}}\left(|A||B|\log T\right)^{\frac{1}{3}}\right)$ regret w.r.t. error-tolerant benchmarks.

Warm-up: Modified ETC yields sublinear regret

Key algorithmic idea: leader waits for the follower to finish exploring before learning

Proposition (Informal): Suppose that:

- The follower runs $\mathrm{ExploreThenCommit}$.
- The leader runs a modified $\mathrm{ExploreThenCommit}$ **where they discard observations from the follower's exploration phase** when computing the empirical means.

=> Both agents achieve $\tilde{O}\left(T^{\frac{2}{3}}\left(|A||B|\log T\right)^{\frac{1}{3}}\right)$ regret w.r.t. error-tolerant benchmarks.

Regret is sublinear for both players!

Main Result: Flexibility in Follower's Algorithm

ExploreThenUCB enables flexibility for the follower while maintaining the **same regret.**

Main Result: Flexibility in Follower's Algorithm

ExploreThenUCB enables flexibility for the follower while maintaining the **same regret.**

Theorem (Informal): Suppose that:

- The leader runs **ExploreThenUCB** where **they discard observations from an initial phase and then run a variant of UCB**.
- The follower runs any algorithm with **sufficiently low instantaneous regret**.

=> Both agents achieve $\tilde{\boldsymbol{O}}\left(\boldsymbol{T}^{\frac{2}{3}}\left(|A||B| \log T\right)^{\frac{1}{3}}\right)$ regret w.r.t. error-tolerant benchmarks.

Main Result: Flexibility in Follower's Algorithm

ExploreThenUCB enables flexibility for the follower while maintaining the **same regret.**

Theorem (Informal): Suppose that:

- The leader runs **ExploreThenUCB** where **they discard observations from an initial phase and then run a variant of UCB**.
- The follower runs any algorithm with **sufficiently low instantaneous regret**.

=> Both agents achieve $\widetilde{\boldsymbol{O}}\left(\boldsymbol{T}^{\frac{2}{3}}\,(|A||B|\log T)^{\frac{1}{3}}\right)$ regret w.r.t. error-tolerant benchmarks.

Leader waits for follower to **sufficiently converge** and then runs UCB

Main Result: Flexibility in Follower's Algorithm

ExploreThenUCB enables flexibility for the follower while maintaining the **same regret.**

Theorem (Informal): Suppose that:

- The leader runs **ExploreThenUCB** where **they discard observations from an initial phase and then run a variant of UCB**.
- The follower runs any algorithm with **sufficiently low instantaneous regret**.

=> Both agents achieve $\tilde{\boldsymbol{O}}\left(\boldsymbol{T}^{\frac{2}{3}}\left(|A||B|\log T\right)^{\frac{1}{3}}\right)$ regret w.r.t. error-tolerant benchmarks.

Leader waits for follower to **sufficiently converge** and then runs UCB

Follower must gracefully learn (satisfied by AAE, ExploreThenCommit, etc)

Main Result: Flexibility in Follower's Algorithm

ExploreThenUCB enables flexibility for the follower while maintaining the **same regret.**

Theorem (Informal): Suppose that:

- The leader runs **ExploreThenUCB** where **they discard observations from an initial phase and then run a variant of UCB**.
- The follower runs any algorithm with **sufficiently low instantaneous regret**.

=> Both agents achieve $\tilde{\boldsymbol{O}}\left(\boldsymbol{T}^{\frac{2}{3}}\,(|A||B|\log T)^{\frac{1}{3}}\right)$ regret w.r.t. error-tolerant benchmarks.

Leader waits for follower to **sufficiently converge** and then runs UCB

Follower must gracefully learn (satisfied by AAE, ExploreThenCommit, etc)

Regret scaling with $T^{2/3}$ rate is <u>unavoidable</u>

Theorem (Informal): For any ALG_1 and ALG_2, some agent incurs $\Omega\left(\boldsymbol{T}^{\frac{2}{3}}\,|B|^{\frac{1}{3}}\right)$ regret.

Conclusion

Regret scaling with $T^{2/3}$ rate is unavoidable

Theorem (Informal): For any ALG_1 and ALG_2, some agent incurs $\Omega\left(T^{\frac{2}{3}}\,|B|^{\frac{1}{3}}\right)$ regret.

	b_1	. b_2
a_1	$(0.5+\delta,\delta)$	$(0,\mathbf{0})$
a_2	$(0.5,3\delta)$	$(0.5,3\delta)$

$(\alpha_1^{tol},\alpha_2^{tol}) = (0.5+\delta,\delta)$

	b_1	. b_2
a_1	$(0.5+\delta,\delta)$	$(0,\mathbf{2\,\delta})$
a_2	$(0.5,3\,\delta)$	$(0.5,3\delta)$

$(\alpha_1^{tol},\alpha_2^{tol}) = (0.5,3\delta)$

Regret scaling with $T^{2/3}$ rate is unavoidable

Theorem (Informal): For any ALG_1 and ALG_2, some agent incurs $\Omega\left(T^{\frac{2}{3}}|B|^{\frac{1}{3}}\right)$ regret.

	b_1	. b_2
a_1	$(0.5+\delta, \delta)$	$(0, \mathbf{0})$
a_2	$(0.5, 3\delta)$	$(0.5, 3\delta)$

$(\alpha_1^{tol}, \alpha_2^{tol}) = (0.5+\delta, \delta)$

	b_1	. b_2
a_1	$(0.5+\delta, \delta)$	$(0, \mathbf{2\ \delta})$
a_2	$(0.5, 3\ \delta)$	$(0.5, 3\delta)$

$(\alpha_1^{tol}, \alpha_2^{tol}) = (0.5, 3\delta)$

To distinguish instances, need to explore a **very suboptimal arm** (a_1, b_2) **for the leader**

What if agents are partially aligned?

Suppose that the two agents agree over which outcomes are different.

$$L = \sup_{a,a',b,b'} \left(\frac{u_1(a,b) - u_1(a',b')}{u_2(a,b) - u_2(a',b')}, \frac{u_2(a,b) - u_2(a',b')}{u_1(a,b) - u_1(a',b')} \right)$$

We show that L is bounded => the original Stackelberg benchmarks are achievable.

Theorem (Informal): There exist algorithms such that both players achieve $\tilde{O}\left(L^2\sqrt{T|A||B|}\right)$ regret w.r.t. the **original Stackelberg benchmarks**.

Takeaway: Partial alignment makes learning easier and faster.

Conclusion

We study Stackelberg games with decentralized learning.

Main finding: misalignment in agent utilities distorts learning dynamics

- We showed that the Stackelberg equilibrium utilities are unachievable.
- We designed error-tolerant benchmarks to better capture learning dynamics.
- We constructed algorithms which achieve optimal $\widetilde{\Theta}\left(T^{\frac{2}{3}}\right)$ regret.
- We showed that partial alignment makes learning easier and faster.

Future directions: allow for greater flexibility in leader algorithm, study application-specific learning algorithms, characterize equilibria in the meta-game between agents, etc.