Sampling and deep learning in CFR: MCCFR and Deep CFR

Brian Zhang

Recall: Self-play in regret matching/CFR

for $t = 1, \dots, T$:

- for all J: $\pi^t(\cdot \mid J) \leftarrow$ next behavior strategy from regret minimizer at J

  O(# sequences)

- for all J: $u^t(\cdot \mid J) \leftarrow$ counterfactual values at J

  O(# terminal nodes)

- for all J: regret minimizer at J observes $u^t(\cdot \mid J)$

  O(# sequences)

Can we do better?

Idea: **Estimate** the utilities

Recall: Self-play in regret matching/CFR

for $t = 1, \dots, T$:

- for all J: $\pi^t(\cdot\,|J) \leftarrow$ next behavior strategy from regret minimizer at J

  O(# sequences)

- for all J: $\tilde{u}^t(\cdot\,|J) \leftarrow$ fast randomized estimate of counterfactual values at J

  faster than O(# terminal nodes)?

- for all J: regret minimizer at J observes $\tilde{u}^t(\cdot\,|J)$

  O(# sequences)

Can we do better?

Idea: **Estimate** the utilities

Regret minimization with estimated utility vectors

$$R(T) = \max_{\hat{\boldsymbol{x}} \in \Delta^n} \sum_{t \le T} \langle \boldsymbol{u}^t, \hat{\boldsymbol{x}} - \boldsymbol{x}^t \rangle$$

$$= \max_{\hat{\boldsymbol{x}} \in \Delta^n} \sum_{t \le T} \langle \boldsymbol{u}^t + \tilde{\boldsymbol{u}}^t - \tilde{\boldsymbol{u}}^t, \hat{\boldsymbol{x}} - \boldsymbol{x}^t \rangle$$

$$= \underbrace{\max_{\hat{\boldsymbol{x}} \in \Delta^n} \sum_{t \le T} \langle \tilde{\boldsymbol{u}}^t, \hat{\boldsymbol{x}} - \boldsymbol{x}^t \rangle}_{\text{regret against estimated utilities}} + \underbrace{\sum_{t \le T} \langle \boldsymbol{u}^t - \tilde{\boldsymbol{u}}^t, \hat{\boldsymbol{x}} - \boldsymbol{x}^t \rangle}_{\text{estimation error}}$$

martingale!

bounded by $\mathcal{O}\left(M\sqrt{nT}\right)$
(with RM, where $M = \max_t \|\boldsymbol{u}^t\|_\infty$)

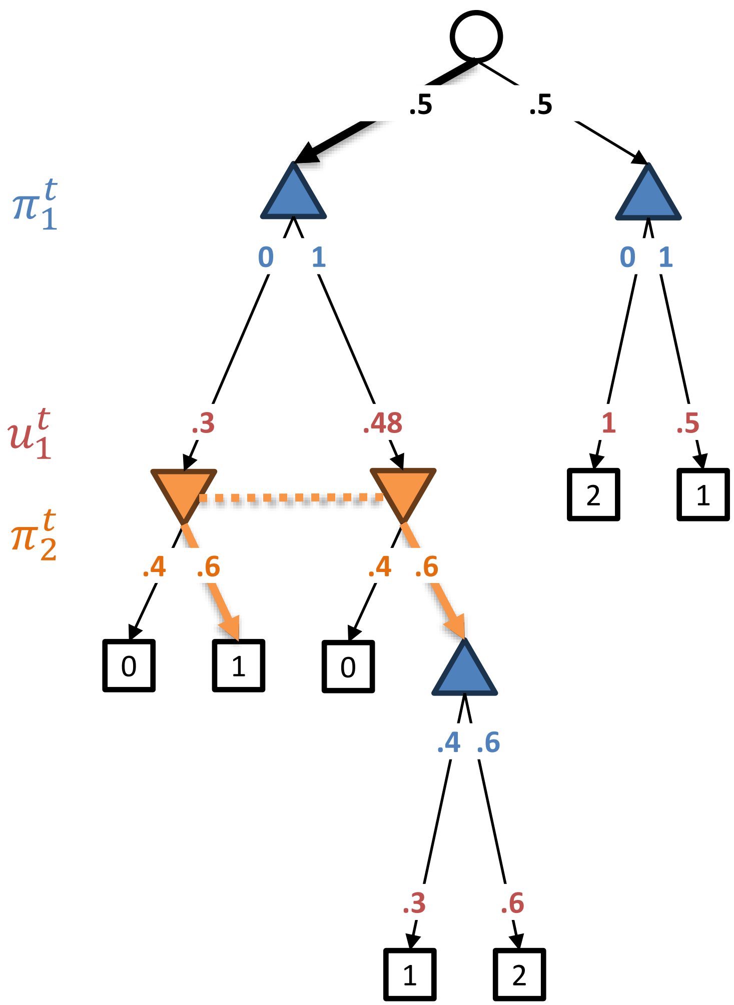
bounded by $\mathcal{O}\left(M\sqrt{nT \log \frac{n}{\delta}}\right)$ w.p. $1-\delta$

$\boxed{\text{if } \mathbb{E}[\tilde{\boldsymbol{u}}^t] = \boldsymbol{u}^t}$

CFR: $R_X(T) \le \sum_J [R_J(T)]^+ \le \tilde{\mathcal{O}}\left(|\Sigma| M \sqrt{T}\right)$
where $\tilde{\mathcal{O}}$ hides a poly $\log(|\Sigma|, 1/\delta)$ term

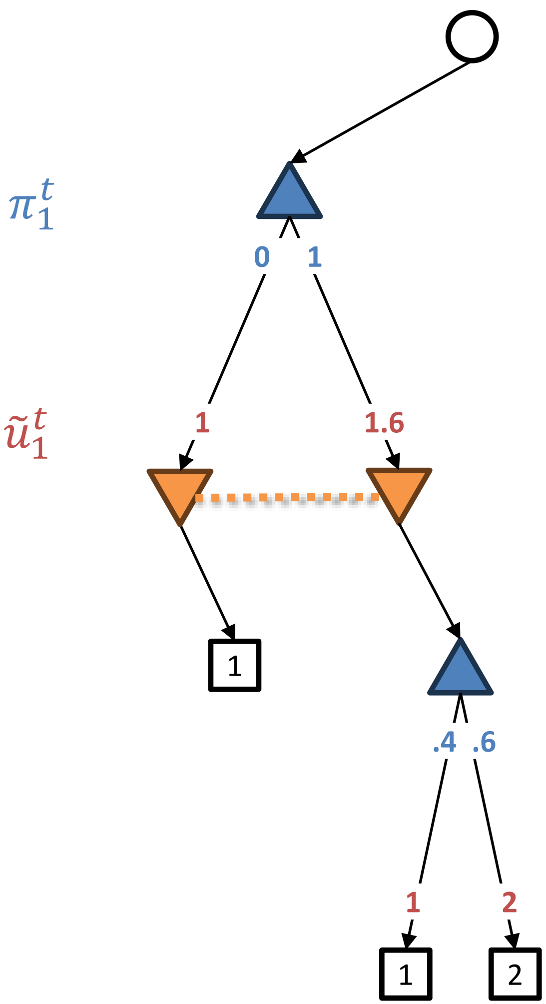
How do we get an **unbiased estimate** of $\boldsymbol{u}^t$?

How do we estimate utility vectors?



Idea 1: **Sample** opponent and chance actions

How do we estimate utility vectors?



Idea 1: **Sample** opponent and chance actions

Claim: $\mathbb{E}[\tilde{u}^t(a|I)] = u^t(a|I)$
$\Rightarrow$ Regret minimization still works!

"External sampling Monte Carlo CFR"

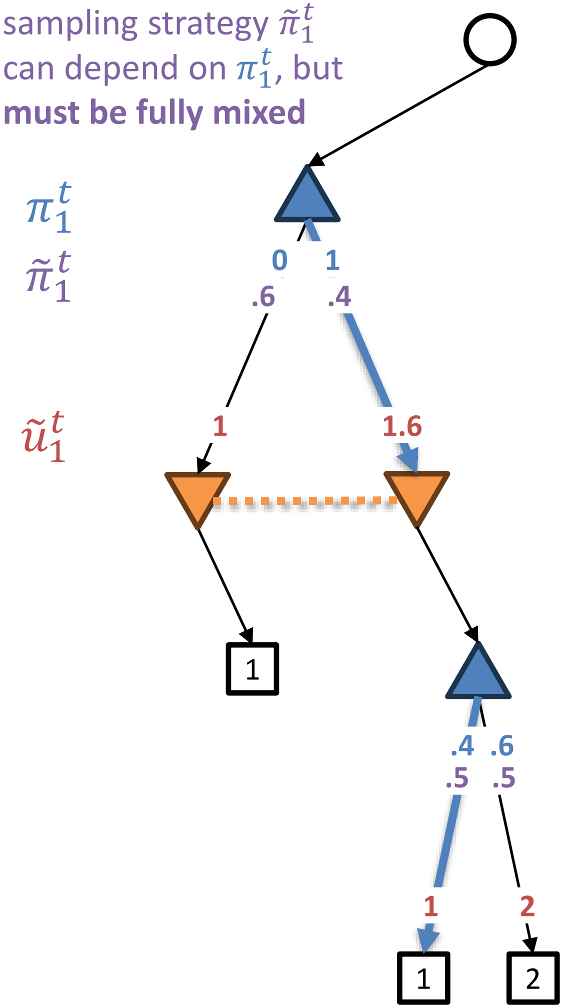
Time complexity: $\mathcal{O}(|\Sigma|)$ per iteration worst-case

Often better! (only need to update infosets if the opponent/chance plays to reach them in the *sampled* strategy)

Can we do even better?

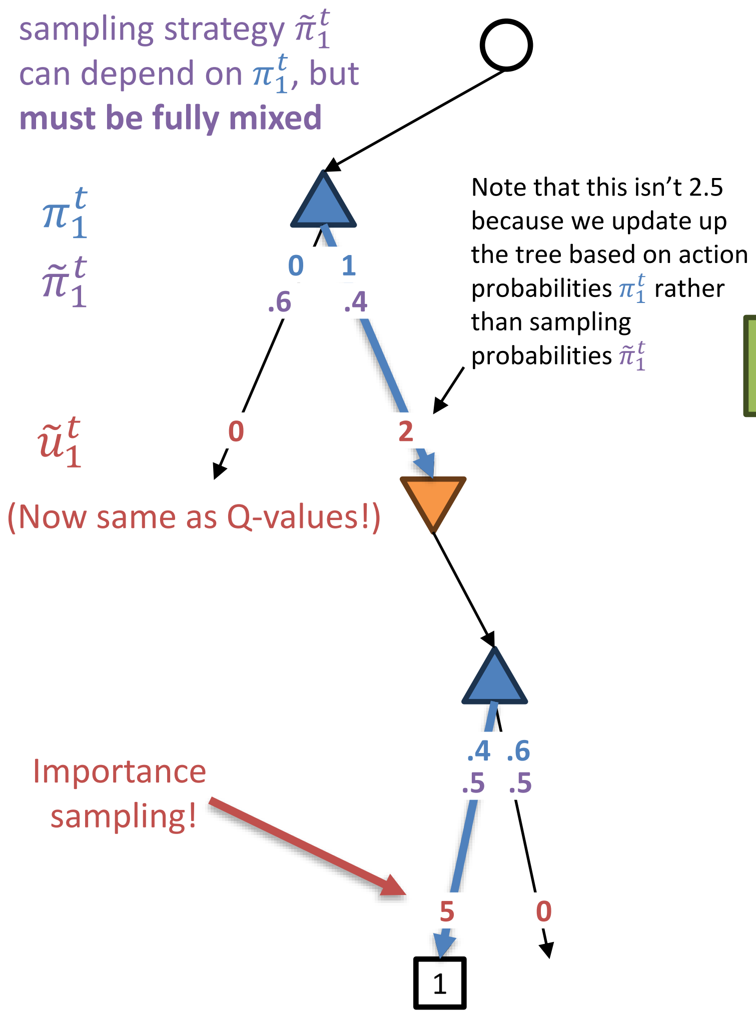
How do we estimate utility vectors?

sampling strategy $\tilde{\pi}_1^t$ can depend on π_1^t, but **must be fully mixed**



Idea 2: Sample **our actions** too?

How do we estimate utility vectors?



Idea 2: Sample **our actions** too?

Claim: $\mathbb{E}[\tilde{u}^t(a|I)] = u^t(a|I)$

$\Rightarrow$ Regret minimization still works!

"Outcome sampling Monte Carlo CFR"

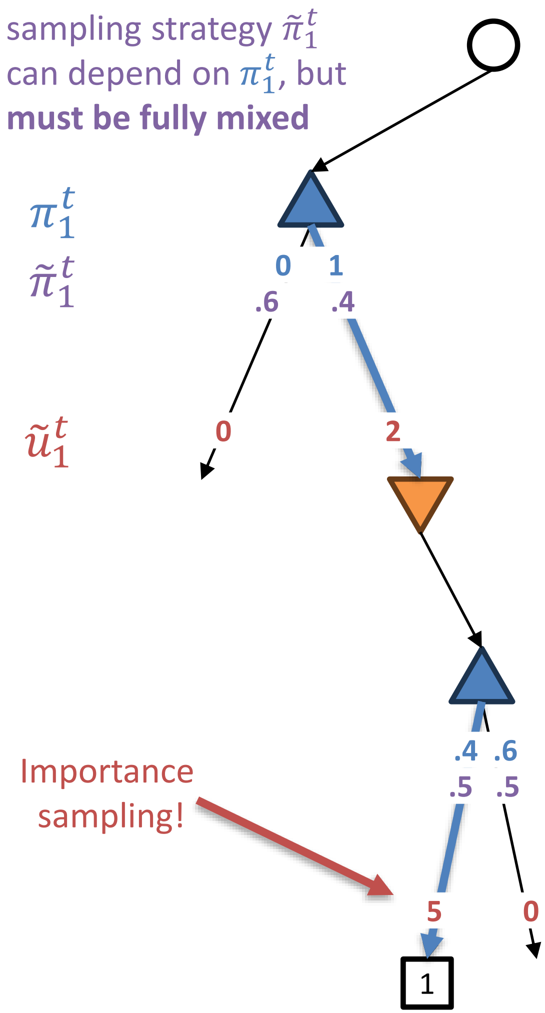
Time complexity: $\mathcal{O}(d|A|)$ per iteration

(d = depth, $|A|$ = action set)

(update all infosets along sampled trajectory)

Problem: Extremely high variance due to importance sampling: $M = \max_z \frac{1}{\tilde{\pi}_1^t(z)}$

Importance sampling



Utility vector for P1, estimated with outcome sampling:

- Sample terminal node $z \sim \tilde{\pi}^{t,1} := (\tilde{\pi}_1^t, \pi_2^t)$
- Set counterfactual utilities for all $J \preccurlyeq z,\ a \in A$

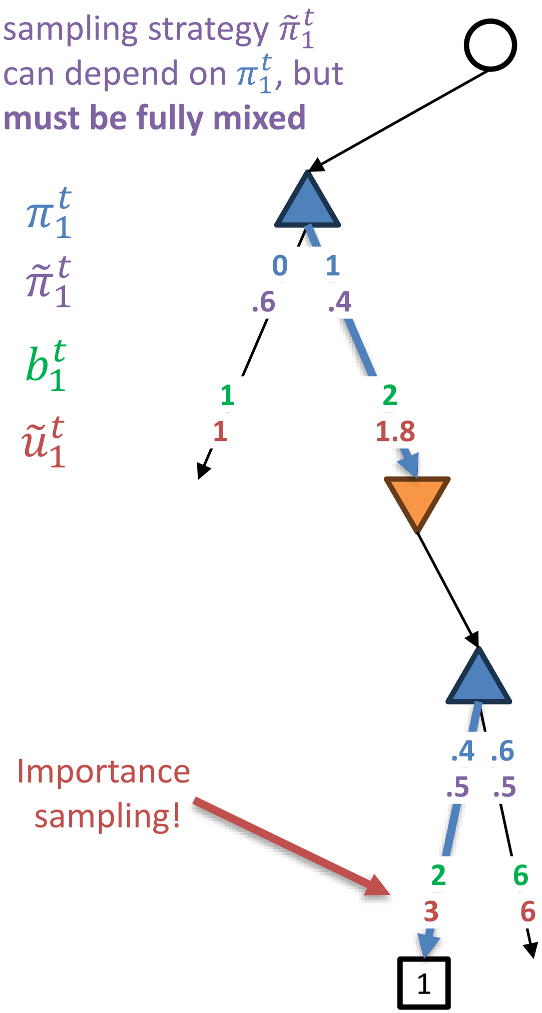
$$\tilde{u}_1^t(a|J) = \begin{cases} \dfrac{u_1(z)}{\tilde{\pi}_1^t(z)} & \text{if } \sigma_1(z) = Ja \\ \displaystyle\sum_{\substack{J' \in N(Ja) \\ a' \in A}} \pi_1(a'|J')\tilde{u}_1^t(a'|J') & \text{if } \sigma_1(z) \succ Ja \\ 0 & \text{if } \sigma_1(z) \nsucceq Ja \end{cases}$$

$$\mathbb{E}\,\tilde{u}_1^t(a|J) = \sum_{z:\sigma_1(z)=Ja} \cancelto{\pi_{-1}^t(z)}{\frac{u_1(z)}{\tilde{\pi}_1^t(z)} \cdot \tilde{\pi}^{t,1}(z)} + \sum_{\substack{J' \in N(Ja) \\ a' \in J'}} \pi_1(a'|J')\,\mathbb{E}\,\tilde{u}_1^t(a'|J')$$

$$= u_1^t(a|J) \quad ✓$$

…but really high variance…

Variance reduction using baselines



Utility vector for P1, estimated with outcome sampling:

- Sample terminal node $z \sim \tilde{\pi}^{t,1} := (\tilde{\pi}_1^t, \pi_2^t)$
- Set counterfactual utilities for all $J \preccurlyeq z,\ a \in A$

$$\hat{u}_1^t(a|J) = \begin{cases} \dfrac{u_1(z)}{\tilde{\pi}_1^t(z)} & \text{if } \sigma_1(z) = Ja \\ \displaystyle\sum_{\substack{J' \in N(Ja) \\ a' \in A}} \pi_1(a'|J')\tilde{u}_1^t(a'|J') & \text{if } \sigma_1(z) \succ Ja \\ 0 & \text{if } \sigma_1(z) \nsucceq Ja \end{cases}$$

$$\tilde{u}_1^t(a|J) = b_1^t(a|J) + \mathbf{1}\{\sigma_1(z) \succcurlyeq Ja\} \cdot \left(\hat{u}_1^t(a|J) - \frac{b_1^t(a|J)}{\tilde{\pi}_1^t(a|J)} \right)$$

$$\mathbb{E}\, \tilde{u}_1^t(a|J) = \mathbb{E}\, \hat{u}_1^t(a|J) = u_1^t(a|J) \quad ✓$$

"Optimal" baseline (if you had an oracle) would be:

$$b_1^t(a|J) = \frac{u_1^t(a|J)}{\tilde{\pi}_1^t(J)}$$

CFR vs MCCFR

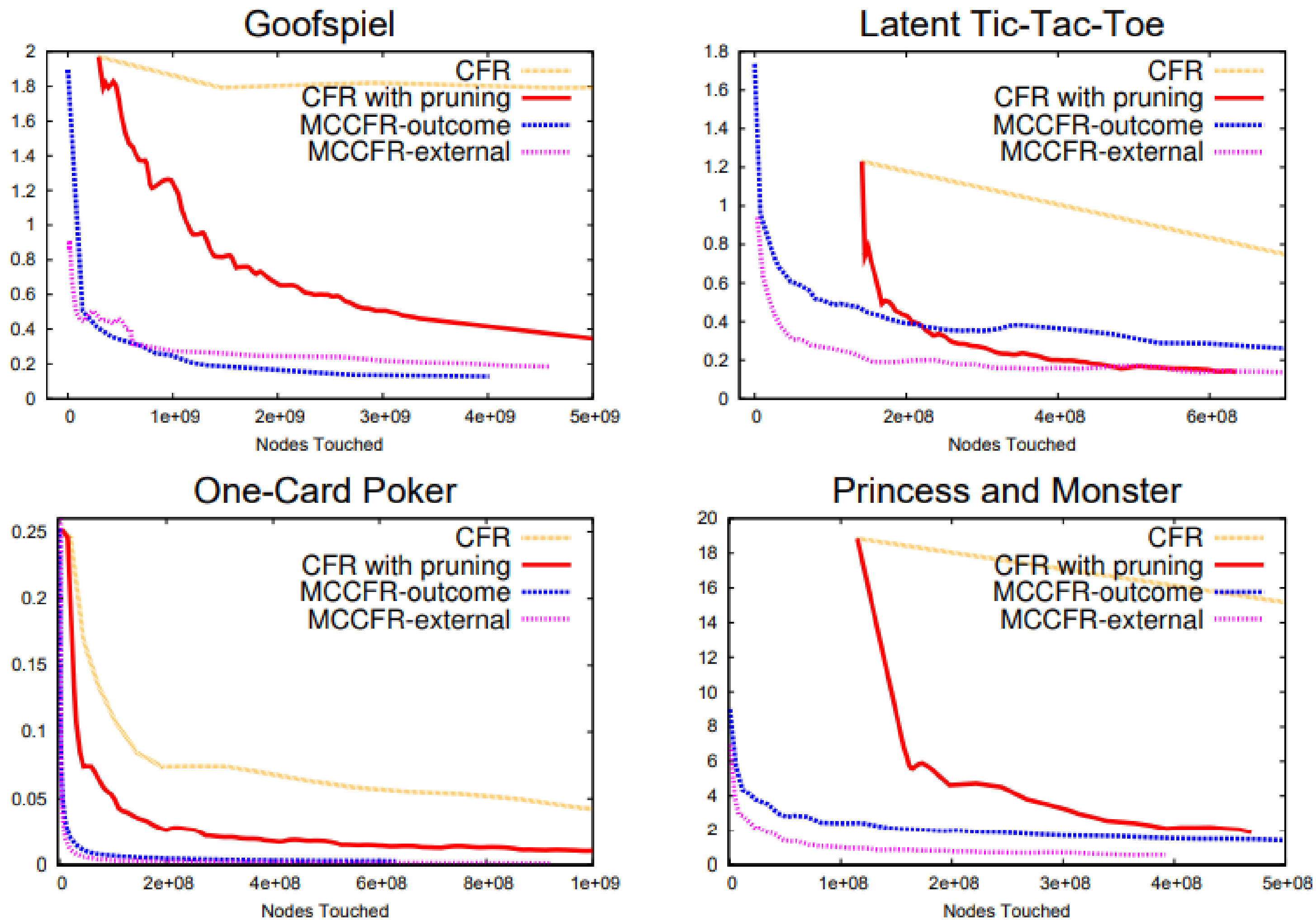
Recall: general stochastic regret minimization regret bound w/ unbiased utility estimates:

$$R_X(T) \leq |\Sigma| M \sqrt{T \log \frac{1}{\delta}}$$

	Per-iteration complexity	Regret bound
CFR	$\mathcal{O}(\#\text{histories})$	$\mathcal{O}(\|\Sigma\|\sqrt{T})$ (often faster in practice, esp. with PCFR+)
External-sampling MCCFR	$\mathcal{O}(\|\Sigma\|)$ (often even faster)	$\tilde{\mathcal{O}}(\|\Sigma\|\sqrt{T})$
Outcome-sampling MCCFR	$\mathcal{O}(d\|A\|)$	$\tilde{\mathcal{O}}(\|\Sigma\|^2\sqrt{T})$ (using balanced sampling strategy)

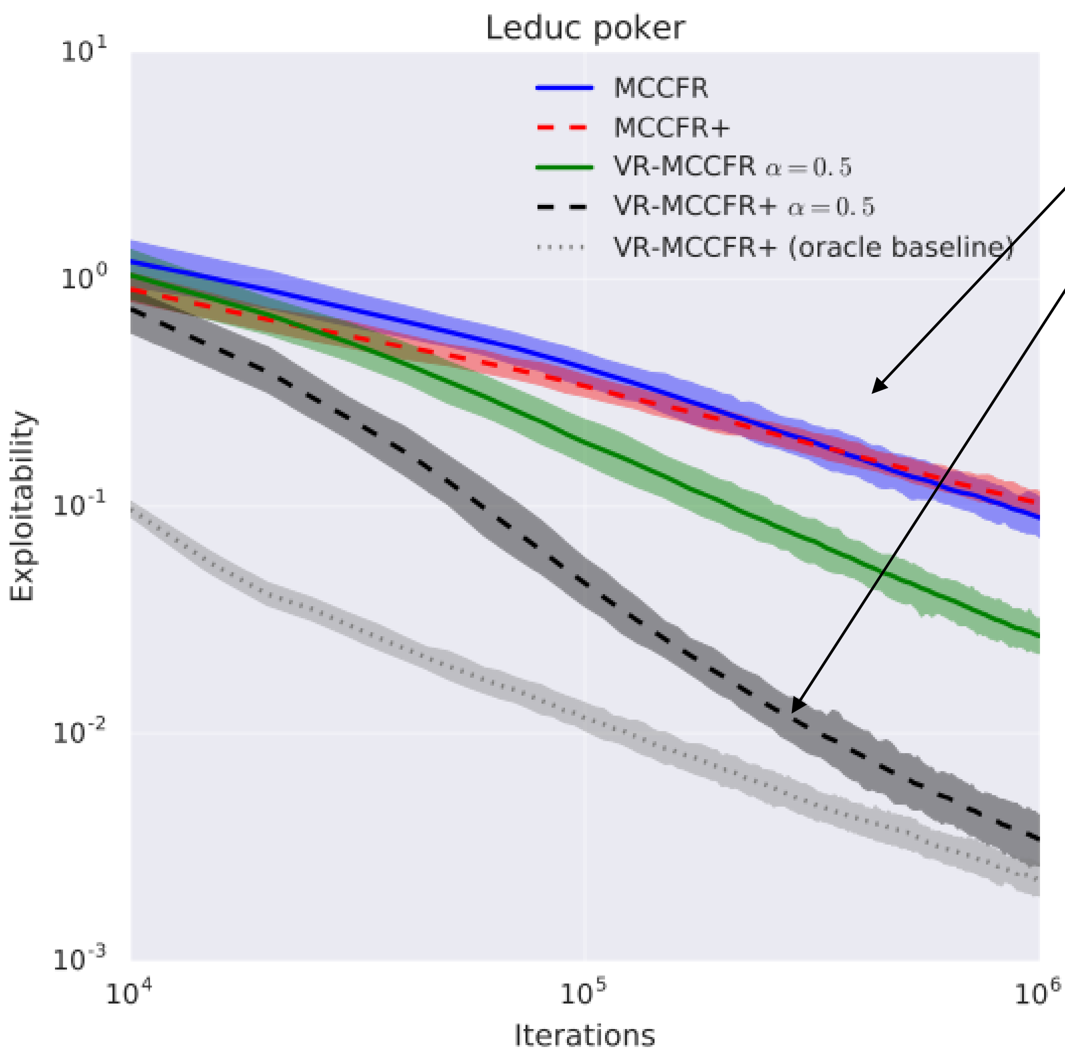
$\exists \tilde{\pi}_i$ with $\tilde{\pi}_i(\sigma) \geq \frac{1}{|\Sigma|}$ $\forall \sigma$, thus $M = |\Sigma|$

Experiments: CFR vs MCCFR



MCCFR reaches reasonable exploitability before CFR even finishes one iteration!

Experiments: MCCFR vs MCCFR+, with and without Baselines for VR



MCCFR+ doesn't beat MCCFR…

unless variance reduction is used!

Practical advice:

- Use PCFR+ (try both $\gamma = 2$ and last-iterate) if doing full game tree traversals is feasible—it will probably win.
- If that isn't feasible, use external sampling.
- If *that* isn't feasible, use outcome sampling with baselines for VR.

Deep Learning for Games
(as an alternative to abstraction)

Deep (MC)CFR

Maintain: Training set S, regret networks $\tilde{r}_i^t : \Sigma \to \mathbb{R}$

On each iteration $t = 1, \dots, T$, for each player i:

1. Iterate through the game tree using MCCFR and (behavioral) strategy profile $\pi^t \propto [\tilde{r}^t]^+$ (and some sampling profile $\tilde{\pi}_i^t$ if using outcome sampling) to compute **sampled counterfactual values** $\tilde{u}_i^t(a|J)$
2. Compute immediate counterfactual regrets
$$\tilde{g}_i^t(a|J) \coloneqq \tilde{u}_i^t(a|J) - \sum_{a'} \pi_i^t(a'|J) \cdot \tilde{u}_i^t(a'|J)$$
for each sampled infoset using $\tilde{u}_i^t$ and π^t, $\tilde{\pi}_i^t$
3. For each sequence (J, a) encountered in Step 1, add $\left((J, a), \tilde{g}_i^t\ (a|J)\right)$ to training set S
4. Train network $\tilde{r}_i^{t+1} : \Sigma \to \mathbb{R}$ on S: $\tilde{r}_i^t(a|J) \approx \frac{1}{N_{t,J,a}} \cdot \underbrace{\sum_{\tau \le t} \tilde{g}_i^\tau(a|J)}_{\text{regret}}$

doesn't matter: RM is scale-invariant!

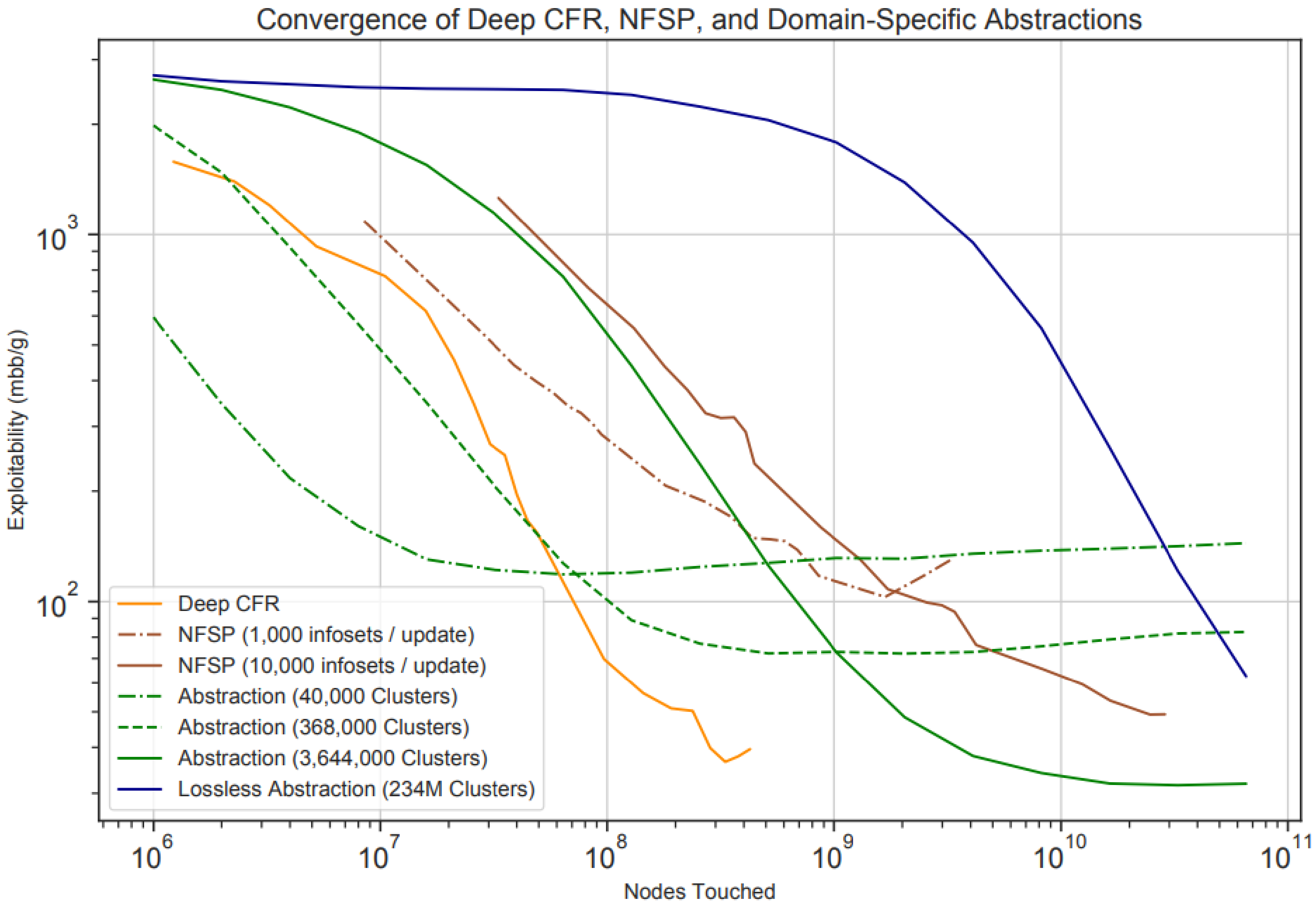
regret matching!
(as usual, if no regrets are positive then play arbitrarily)

DREAM
[Steinberger, Lerer, Brown arXiv 2020]
=
outcome-sampling deep MCCFR with baselines for variance reduction

When S grows too big, use reservoir sampling

If networks $\tilde{r}^t$ are **perfect** then this is just MCCFR

If networks are **imperfect**… we may get to take advantage of **generalization**!


Convergence of Deep CFR, NFSP, and Domain-Specific Abstractions
Exploitability (mbb/g)
10^3
10^2
10^6
10^7
10^8
10^9
10^{10}
10^{11}
Nodes Touched
Deep CFR
NFSP (1,000 infosets / update)
NFSP (10,000 infosets / update)
Abstraction (40,000 Clusters)
Abstraction (368,000 Clusters)
Abstraction (3,644,000 Clusters)
Lossless Abstraction (234M Clusters)

ESCHER: Can we get rid of importance sampling?

Motivation: Deep networks have a hard time with outputs of different magnitudes (*e.g.,* with importance sampling)

Idea: For each player i, at each timestep t:

1. Train network $Q : \Sigma \to \mathbb{R}$ **directly to estimate the conditional** (not counterfactual) **values** $Q(a|J) := \frac{u^t(a|J)}{\pi^t_{-i}(J)}$
2. Use **fixed** (i.e., time-independent) sampling strategy $\tilde{\pi}^*_i$ for each player i
3. Sample trajectory $z \sim \left(\tilde{\pi}^*_i, \pi^t_{-i}\right)$
4. Update regret minimizer at each player i infoset $J \preccurlyeq z$ using Q-values $\tilde{u}^t_i(\cdot\, |J) = Q(\cdot\, |J)$

$$\mathbb{E}\, \tilde{u}^t_i(a|J) = \tilde{\pi}^*_i(J) \cdot \pi^t_{-i}(J) \cdot Q(a|J) = \tilde{\pi}^*_i(J) \cdot u^t(a|J)$$

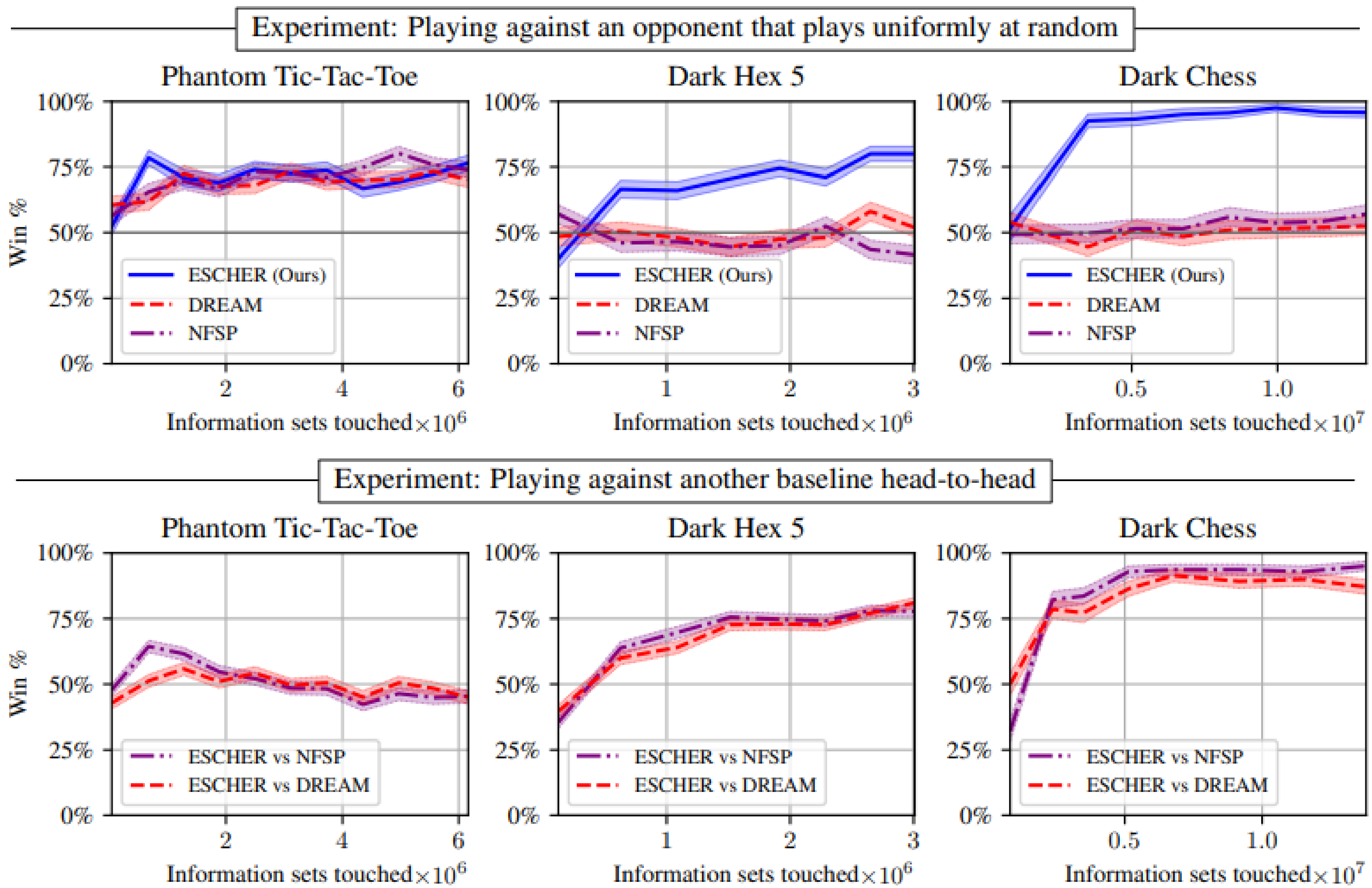
doesn't matter—RM is scale-invariant!

plug this idea into deep CFR ⇒ ESCHER

CFR vs MCCFR vs ESCHER

	Per-iteration complexity	**Regret bound**	**Importance sampling?**
CFR	$\mathcal{O}(\#\text{histories})$	$\mathcal{O}\left(\lvert\Sigma\rvert\sqrt{T}\right)$ (often faster in practice, esp. with PCFR+)	No (deterministic algorithm)
External-sampling MCCFR	$\mathcal{O}(\lvert\Sigma\rvert)$ (often even faster)	$\tilde{\mathcal{O}}\left(\lvert\Sigma\rvert\sqrt{T}\right)$	No
Outcome-sampling MCCFR	$\mathcal{O}(d\lvert A\rvert)$	$\tilde{\mathcal{O}}\left(\lvert\Sigma\rvert^2\sqrt{T}\right)$ using balanced sampling strategy	Yes
ESCHER (Tabular, oracle Q-values)			No

Experiments: Head-to-head against other algorithms



Experiments: ablations

divides through by $\tilde{\pi}_i^*(J)$ to make $\mathbb{E}\,\tilde{u}_i^t(a|J) = u^t(a|J)$ (needlessly adds variance, so we should expect this to be worse)

uses $\tilde{u}_i^t(\cdot\,|J)$ from Deep VR-MCCFR (DREAM), but multiplied by $\tilde{\pi}_i^*(J)$

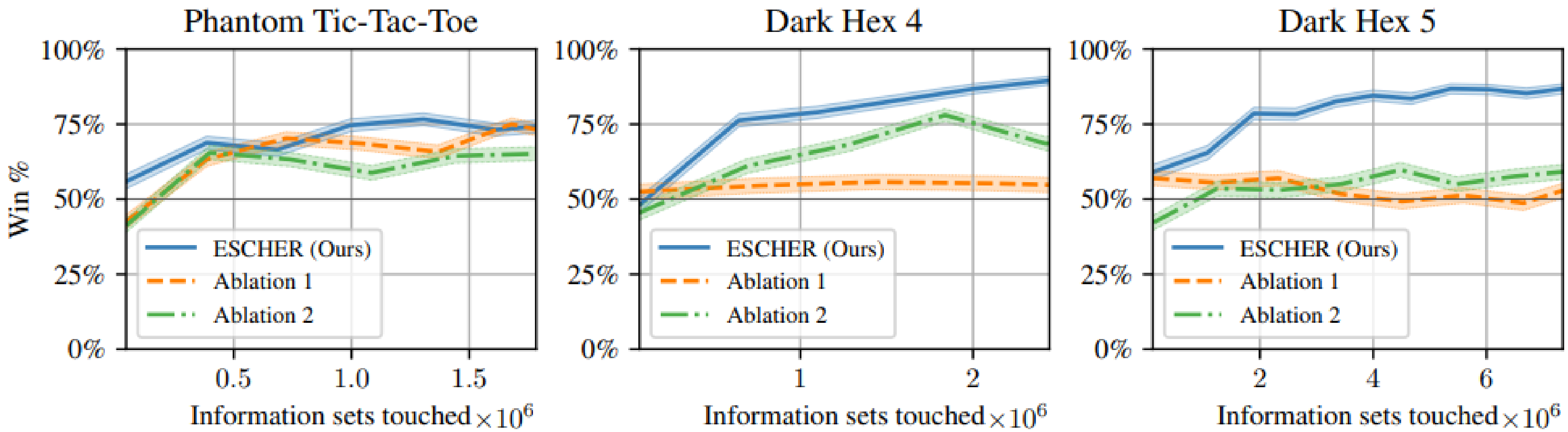
Algorithm	**History value function**	**Boostrapped baseline**	**Importance sampling**
ESCHER (Ours)	✓	✗	✗
Ablation 1	✓	✓	✗
Ablation 2	✓	✗	✓
DREAM / VR-MCCFR	✓	✓	✓
Deep CFR / OS-MCCFR	✗	✗	✓

Experiments: variance in counterfactual utility estimates

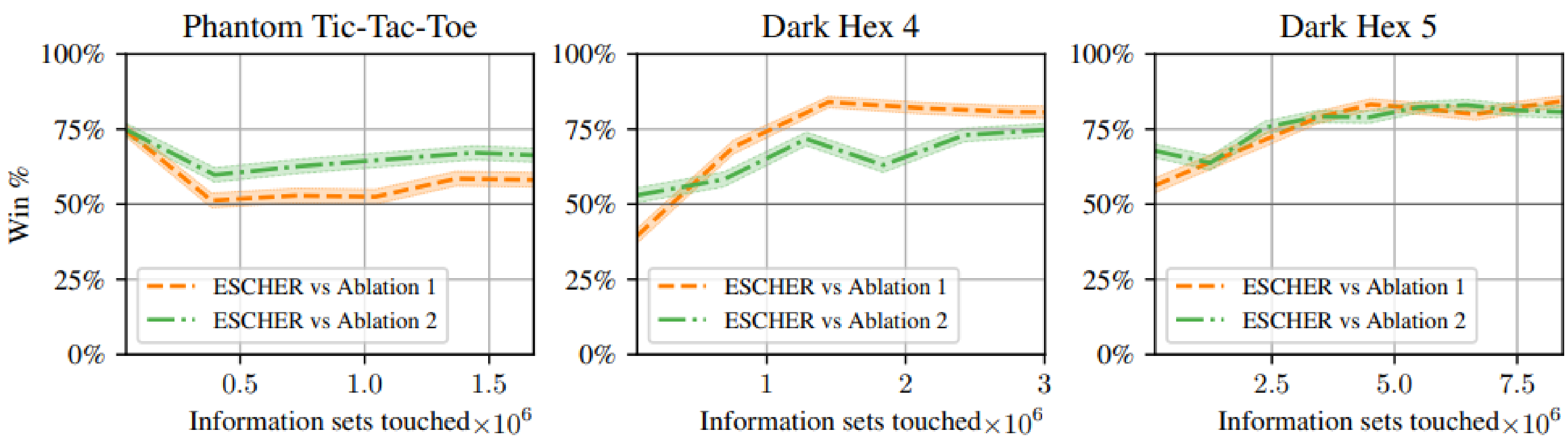
Game	ESCHER (Ours)	Ablation 1	Ablation 2	DREAM
Phantom Tic-Tac-Toe	$(2.6 \pm 0.1) \times 10^{-1}$	$(4.1 \pm 0.7) \times 10^{1}$	$(1.4 \pm 0.4) \times 10^{7}$	$(4.6 \pm 1.0) \times 10^{7}$
Dark Hex 4	$(1.8 \pm 0.1) \times 10^{-1}$	$(1.3 \pm 0.9) \times 10^{2}$	$(3.1 \pm 1.7) \times 10^{8}$	$(2.8 \pm 2.0) \times 10^{8}$
Dark Hex 5	$(1.3 \pm 0.1) \times 10^{-1}$	$(3.3 \pm 1.6) \times 10^{2}$	$(2.0 \pm 0.6) \times 10^{5}$	$(5.3 \pm 3.9) \times 10^{8}$

Game	ESCHER (Ours)	Ablation 2	DREAM	OS-MCCFR
Leduc	$(5.3 \pm 0.0) \times 10^{0}$	$(3.3 \pm 0.7) \times 10^{2}$	$(2.8 \pm 0.0) \times 10^{2}$	$(2.2 \pm 0.0) \times 10^{3}$
Battleship	$(1.4 \pm 0.0) \times 10^{0}$	$(7.1 \pm 0.3) \times 10^{2}$	$(1.2 \pm 0.0) \times 10^{3}$	$(2.4 \pm 0.0) \times 10^{3}$
Liar's Dice	$(9.0 \pm 0.1) \times 10^{-1}$	$(7.8 \pm 0.9) \times 10^{1}$	$(4.0 \pm 0.8) \times 10^{2}$	$(1.2 \pm 0.1) \times 10^{3}$

Ablation experiment: Playing against an opponent that plays uniformly at random



Ablation experiment: Playing against another ablation head-to-head



Bonus slides on single-agent RL

Single-agent reinforcement learning

Single-agent learning:

- Set of states S (information sets) with fixed "start state" $s_1 \in S$ (root infoset)
- Set of actions A
- We'll assume finite horizon: $S = S_1 \sqcup S_2 \sqcup \cdots S_H$, where $H =$ time horizon (depth of game tree), and $S_1 = \{s_1\}$
- Fixed environment (opponent/nature) given by transition functions $P : S_h \times A \to \Delta(S_{h+1})$ for each $h < H$. Playing action a in state s results in random next state s' w.p. $P(s'|s, a)$.
- Trajectory (history): $\tau = (s_1, a_1, s_2, a_2, \dots, a_{H-1}, s_H)$
- Policy (strategy): $\pi : S \to \Delta(A)$
- Reward (utility): $R : S_H \to \mathbb{R}$
  (assume for simplicity that reward is only received at the end)

Q-values, state values, and advantages

Define recursively:

In extensive form, when multiplied by environment reach probability of infoset s, these are:

$$Q^\pi(s,a) = \mathop{\mathbb{E}}_{s'\sim P(\cdot|s,a)} V^\pi(s')$$

"state-action value" counterfactual value $u(a|s)$

$$V^\pi(s) = \begin{cases} R(s) & \text{if } s \in S_H \\ \mathop{\mathbb{E}}_{a\sim\pi(\cdot|s)} Q^\pi(s,a) & \text{otherwise} \end{cases}$$

"state value" counterfactual value $u(s)$

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$$

"advantage" immediate regret $g(a|s)$

Goal: find π maximizing expected reward

$$V^\pi(s_1) = \mathop{\mathbb{E}}_{\tau\sim\pi} R(s_H, a_H).$$

S, A small enough to iterate over
$\Rightarrow$ easy! (backwards induction)

S, A large $\Rightarrow$???

Policy gradient theorem

$$\nabla V^{\pi}(s_1) = \sum_{\tau} R(s_H)\nabla P(\tau|\pi)$$

$$= \sum_{\tau} R(s_H)\, P(\tau|\pi)\nabla \log P(\tau|\pi)$$

using $\nabla \log f(x) = \frac{\nabla f(x)}{f(x)}$

$$= \mathop{\mathbb{E}}_{\tau\sim\pi} R(s_H)\nabla \log P(\tau|\pi)$$

$$= \mathop{\mathbb{E}}_{\tau\sim\pi} \sum_{h=1}^{H-1} R(s_H)\nabla \log \pi(a_h|s_h)$$

$$= \mathop{\mathbb{E}}_{\tau\sim\pi} \sum_{h=1}^{H-1} A(s_h, a_h)\nabla \log \pi(a_h|s_h)$$

(won't show—same idea as "baselines")

Advantage actor-critic (A2C) (very roughly)

initialize policy π^1 to be uniform random

for $t = 1, \dots, T$:

of course, $\hat{V}(s_H) \coloneqq R(s_H)$

- train value function estimate $\tilde{V}^t \approx V^\pi$ using MSE:

$$\tilde{V}^t \coloneqq \arg\min_{\hat{V}} \mathbb{E}_{\tau\sim\pi^t} \sum_{h=1}^{H-1} \left[\hat{V}(s_h) - \hat{V}(s_{h+1})\right]^2$$

- train policy π^{t+1} by taking gradient steps according to the policy gradient theorem:

$$\nabla V^\pi(s_1) = \mathbb{E}_{\tau\sim\pi^t} \sum_{h=1}^{H-1} \hat{A}(s_h, a_h) \nabla \log \pi(a_h | s_h)$$

Problem: Variance in gradients can be very large,
so π can change very fast $\Rightarrow$ training can be unstable

where

$$\tilde{A}^t(s,a) \coloneqq \mathbb{E}_{s'\sim P(\cdot|s,a)} \tilde{V}^t(s') - \tilde{V}^t(s)$$

is an advantage function estimate

Proximal policy optimization (PPO) (very roughly)

initialize policy π^1 to be uniform random

for $t = 1, \ldots, T$:

of course, $\hat{V}(s_H) \coloneqq R(s_H)$

- train value function estimate $\tilde{V}^t \approx V^\pi$ using MSE:

$$\tilde{V}^t \coloneqq \arg\min_{\hat{V}} \mathop{\mathbb{E}}_{\tau\sim\pi^t} \sum_{h=1}^{H-1} \left[\hat{V}(s_h) - \hat{V}(s_{h+1})\right]^2$$

- train policy π^{t+1} according to:

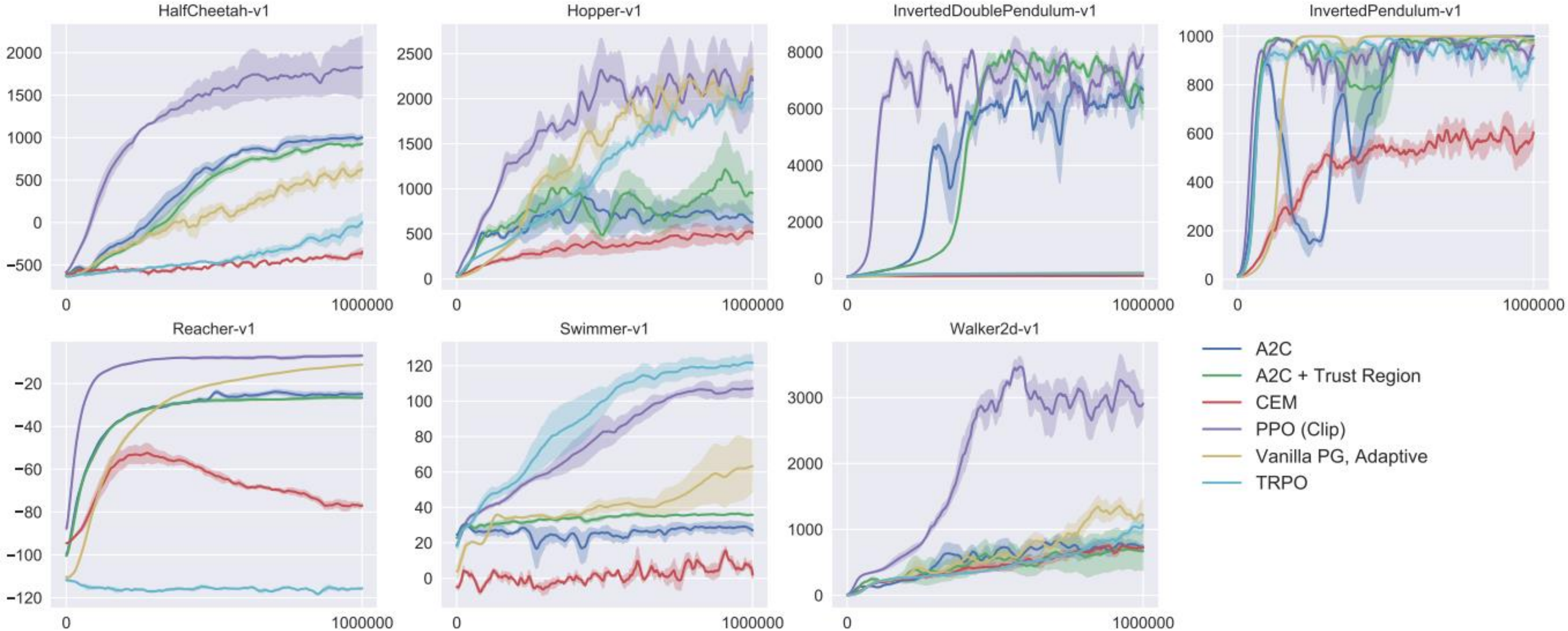
$$\pi^{t+1} \coloneqq \arg\max_{\hat{\pi}} \mathop{\mathbb{E}}_{\tau\sim\pi^t} \sum_{h=1}^{H-1} \begin{cases} \min\left\{\dfrac{\hat{\pi}(a_h|s_h)}{\pi^t(a_h|s_h)}, 1+\epsilon\right\} \hat{A}^t(s_h,a_h) & \text{if } \hat{A}^t(s_h,a_h) > 0 \\ \max\left\{\dfrac{\hat{\pi}(a_h|s_h)}{\pi^t(a_h|s_h)}, 1-\epsilon\right\} \hat{A}^t(s_h,a_h) & \text{if } \hat{A}^t(s_h,a_h) < 0 \end{cases}$$

where

$$\tilde{A}^t(s,a) \coloneqq \mathop{\mathbb{E}}_{s'\sim P(\cdot|s,a)} \tilde{V}^t(s') - \tilde{V}^t(s)$$

is an advantage function estimate

PPO is great*



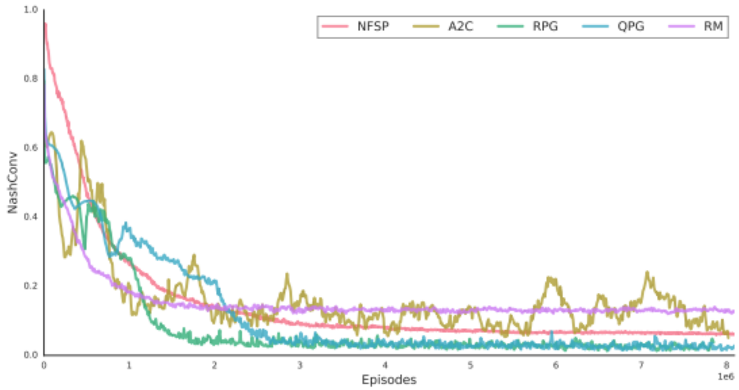
Best large-scale single-agent RL algorithm right now!

***Very very sensitive** to hyperparameters… hard to use in practice…

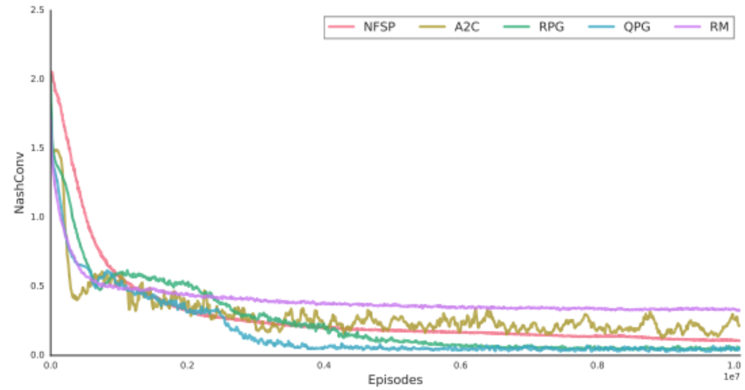
Do these algorithms work for games?

Certainly not in theory. In practice… kind of, at small scale?

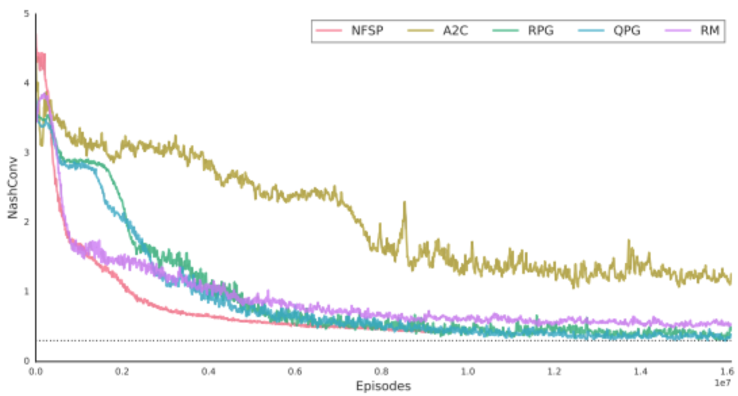
(but probably at this scale you should just use PCFR+ instead…)



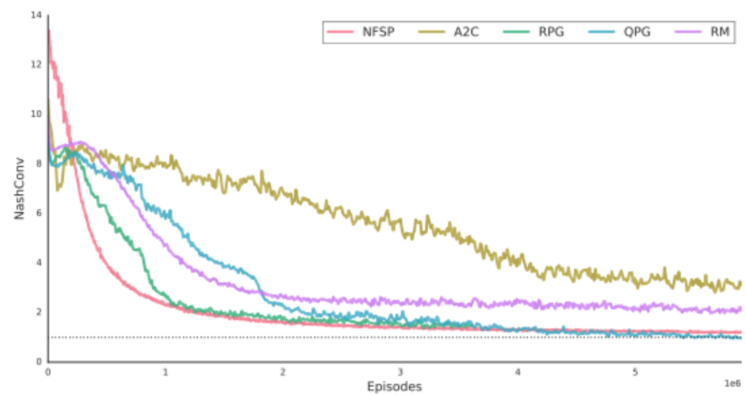
NASHCONV in 2-player Kuhn



NASHCONV in 3-player Kuhn



NASHCONV in 2-player Leduc



NASHCONV in 3-player Leduc

References

MCCFR: Marc Lanctot, Kevin Waugh, Martin Zinkevich, Michael Bowling (NeurIPS 2009) “Monte Carlo sampling for regret minimization in extensive games”

Simplified martingale-based presentation and improved bound in this lecture due to Gabriele Farina, Christian Kroer, Tuomas Sandholm (ICML 2020) “Stochastic regret minimization in extensive-form games”

Martin Schmid, Neil Burch, Marc Lanctot, Matej Moravcik, Rudolf Kadlec, Michael Bowling (AAAI 2019) “Variance Reduction in Monte Carlo Counterfactual Regret Minimization (VR-MCCFR) for Extensive Form Games using Baselines”

Noam Brown, Adam Lerer, Sam Gross, Tuomas Sandholm (ICML 2019) “Deep Counterfactual Regret Minimization”

Deep CFR with variance reduction: Eric Steinberger, Adam Lerer, Noam Brown (arXiv 2020) “DREAM: Deep regret minimization with advantage baselines and model-free learning”

Stephen McAleer, Gabriele Farina, Marc Lanctot, Tuomas Sandholm (ICLR 2023) “Eschewing Importance Sampling in Games by Computing a History Value Function to Estimate Regret”

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov (arXiv 2017) “Proximal Policy Optimization Algorithms”

Sriram Srinivasan, Marc Lanctot, Vinicius Zambaldi, Julien Perolat, Karl Tuyls, Remi Munos, Michael Bowling (NeurIPS 2018) “Actor-Critic Policy Optimization in Partially Observable Multiagent Environments”