Deep Learning in Tree-Based Game Solving 2

Stephen McAleer

Outline of the next few lectures

- Deep learning in tree-based game solving 1
  - Deep learning recap
  - NFSP
  - Deep CFR
  - Policy gradient methods
- Deep learning in tree-based game solving 2
  - MCCFR
  - DREAM
  - ESCHER
  - NeuRD
- Deep learning in tree-based game solving 3
  - DeepNash for expert-level Stratego
- Deep learning in tree-based game solving 4
  - AlphaStar and OpenAI 5 for SOTA in video games
  - Double Oracle brief intro
- SOTA in double oracle algorithms
  - PSRO
  - XDO
  - SP-PSRO

A Taxonomy of Game-Theoretic RL

Lecture 1

- Counterfactual Regret Minimization (Zinkevich et al. 2007)
  - CFR: Zinkevich et al. 2007
  - MC-CFR: Lanctot et al. 2009
  - **Deep CFR: Brown et al. 2019**
  - DREAM: Steinberger et al. 2020
  - ESCHER: McAleer et al. 2022
- Policy Gradients
  - **Regret Policy Gradient (Srinivasan et al. 2018)**
  - OpenAI Five (OpenAI 2019)
  - Neural Replicator Dynamics (Hennes, Morrill, and Omidshafiei et al. 2020)
  - Actor Critic Hedge (Fu et al. 2022)
  - DeepNash for expert-level Stratego (Perolat, de Vylder, and Tuyls et al. 2022)
  - Magnetic Mirror Descent (Sokota et al. 2022)
- PSRO (McMahan et. al. 2003, Lanctot et al. 2017)
  - AlphaStar for expert-level Starcraft (Vinyals et al. 2019)
  - Pipeline PSRO (McAleer and Lanier et al. 2020)
  - α-PSRO (Muller et al. 2020)
  - XDO (McAleer et al. 2021)
  - Joint-PSRO (Marris et al. 2021)
  - Anytime PSRO (McAleer et al. 2022)
  - Self-Play PSRO (McAleer et al. 2022)
- **Neural Fictitious Self Play (Heinrich and Silver 2016)**

A Taxonomy of Game-Theoretic RL

Lecture 2 (This Lecture)

- Counterfactual Regret Minimization (Zinkevich et al. 2007)
  - CFR: Zinkevich et al. 2007
  - **MC-CFR: Lanctot et al. 2009**
  - Deep CFR: Brown et al. 2019
  - **DREAM: Steinberger et al. 2020**
  - **ESCHER: McAleer et al. 2022**
- Policy Gradients
  - Regret Policy Gradient (Srinivasan et al. 2018)
  - OpenAI Five (OpenAI 2019)
  - **Neural Replicator Dynamics (Hennes, Morrill, and Omidshafiei et al. 2020)**
  - Actor Critic Hedge (Fu et al. 2022)
  - DeepNash for expert-level Stratego (Perolat, de Vylder, and Tuyls et al. 2022)
  - Magnetic Mirror Descent (Sokota et al. 2022)
- PSRO (McMahan et. al. 2003, Lanctot et al. 2017)
  - AlphaStar for expert-level Starcraft (Vinyals et al. 2019)
  - Pipeline PSRO (McAleer and Lanier et al. 2020)
  - α-PSRO (Muller et al. 2020)
  - XDO (McAleer et al. 2021)
  - Joint-PSRO (Marris et al. 2021)
  - Anytime PSRO (McAleer et al. 2022)
  - Self-Play PSRO (McAleer et al. 2022)
- Neural Fictitious Self Play (Heinrich and Silver 2016)

A Taxonomy of Game-Theoretic RL

Lecture 3

- Counterfactual Regret Minimization (Zinkevich et al. 2007)
  - CFR: Zinkevich et al. 2007
  - MC-CFR: Lanctot et al. 2009
  - Deep CFR: Brown et al. 2019
  - DREAM: Steinberger et al. 2020
  - ESCHER: McAleer et al. 2022
- Policy Gradients
  - Regret Policy Gradient (Srinivasan et al. 2018)
  - OpenAI Five (OpenAI 2019)
  - Neural Replicator Dynamics (Hennes, Morrill, and Omidshafiei et al. 2020)
  - Actor Critic Hedge (Fu et al. 2022)
  - **DeepNash for expert-level Stratego (Perolat, de Vylder, and Tuyls et al. 2022)**
    - **From Poincaré Recurrence to Convergence in Imperfect Information Games: Finding Equilibrium via Regularization (Perolat et al. 2021)**
  - **Magnetic Mirror Descent (Sokota et al. 2022)**
- PSRO (McMahan et. al. 2003, Lanctot et al. 2017)
  - AlphaStar for expert-level Starcraft (Vinyals et al. 2019)
  - Pipeline PSRO (McAleer and Lanier et al. 2020)
  - α-PSRO (Muller et al. 2020)
  - XDO (McAleer et al. 2021)
  - Joint-PSRO (Marris et al. 2021)
  - Anytime PSRO (McAleer et al. 2022)
  - Self-Play PSRO (McAleer et al. 2022)
- Neural Fictitious Self Play (Heinrich and Silver 2016)

A Taxonomy of Game-Theoretic RL

Lecture 4

- Counterfactual Regret Minimization (Zinkevich et al. 2007)
  - CFR: Zinkevich et al. 2007
  - MC-CFR: Lanctot et al. 2009
  - Deep CFR: Brown et al. 2019
  - DREAM: Steinberger et al. 2020
  - ESCHER: McAleer et al. 2022
- Policy Gradients
  - Regret Policy Gradient (Srinivasan et al. 2018)
  - **OpenAI Five (OpenAI 2019)**
  - Neural Replicator Dynamics (Hennes, Morrill, and Omidshafiei et al. 2020)
  - Actor Critic Hedge (Fu et al. 2022)
  - DeepNash for expert-level Stratego (Perolat, de Vylder, and Tuyls et al. 2022)
  - Magnetic Mirror Descent (Sokota et al. 2022)
- PSRO (McMahan et. al. 2003, Lanctot et al. 2017)
  - **AlphaStar for expert-level Starcraft (Vinyals et al. 2019)**
  - Pipeline PSRO (McAleer and Lanier et al. 2020)
  - α-PSRO (Muller et al. 2020)
  - XDO (McAleer et al. 2021)
  - Joint-PSRO (Marris et al. 2021)
  - Anytime PSRO (McAleer et al. 2022)
  - Self-Play PSRO (McAleer et al. 2022)
- Neural Fictitious Self Play (Heinrich and Silver 2016)

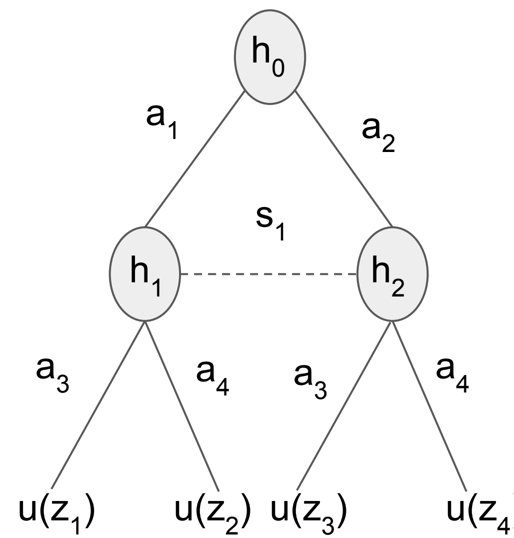
A Taxonomy of Game-Theoretic RL

- Counterfactual Regret Minimization (Zinkevich et al. 2007)
    - CFR: Zinkevich et al. 2007
    - MC-CFR: Lanctot et al. 2009
    - Deep CFR: Brown et al. 2019
    - DREAM: Steinberger et al. 2020
    - ESCHER: McAleer et al. 2022
- Policy Gradients
    - Regret Policy Gradient (Srinivasan et al. 2018)
    - OpenAI Five (OpenAI 2019)
    - Neural Replicator Dynamics (Hennes, Morrill, and Omidshafiei et al. 2020)
    - Actor Critic Hedge (Fu et al. 2022)
    - DeepNash for expert-level Stratego (Perolat, de Vylder, and Tuyls et al. 2022)
    - Magnetic Mirror Descent (Sokota et al. 2022)
- **PSRO (McMahan et. al. 2003, Lanctot et al. 2017)**
    - AlphaStar for expert-level Starcraft (Vinyals et al. 2019)
    - **Pipeline PSRO (McAleer and Lanier et al. 2020)**
    - **α-PSRO (Muller et al. 2020)**
    - **XDO (McAleer et al. 2021)**
    - **Joint-PSRO (Marris et al. 2021)**
    - **Anytime PSRO (McAleer et al. 2022)**
    - **Self-Play PSRO (McAleer et al. 2022)**
- Neural Fictitious Self Play (Heinrich and Silver 2016)

Lecture 5

Extensive-Form Games

- ***History h*** is ground truth state of the game
  - All cards for all players
- ***Information set s*** is observation for one player
  - Set of histories consistent with observation
  - The hand for one player
- ***Policy $\pi_i(a \mid s)$*** gives distribution over actions at information set s
- ***Reach probability $\eta^{\pi}(h)$*** is joint probability of reaching history h under π
- ***Terminal history z*** is history at end of game
- ***Utility $u_i(z)$*** is utility for player i



CFR Recap

- Independently minimize counterfactual regret at every information set

$$v_i(\pi, h) = \sum_{z \sqsupset h} \eta^{\pi}(h, z) u_i(z)$$

CFR Recap

- Independently minimize counterfactual regret at every information set

$$v_i(\pi, h) = \sum_{z \sqsupset h} \eta^{\pi}(h, z) u_i(z)$$

$$v_i^c(\pi, s) = \sum_{h \in s} \eta_{-i}^{\pi}(h) v_i(\pi, h)$$

CFR Recap

- Independently minimize counterfactual regret at every information set
- Tabular CFR traverses entire tree and updates policy via no-regret at every information set

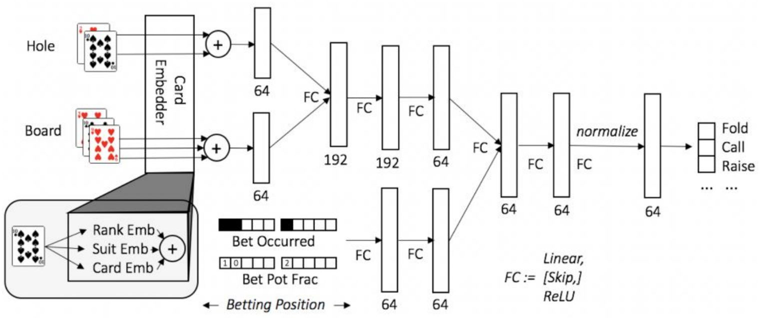
$$v_i(\pi, h) = \sum_{z \sqsupset h} \eta^\pi(h, z) u_i(z)$$

$$v_i^c(\pi, s) = \sum_{h \in s} \eta_{-i}^\pi(h) v_i(\pi, h)$$

$$R_s^T := \max_{\hat{a} \in A_s} \sum_{t=1}^{T} r_i^c(\pi^t, s, a) = \max_{\hat{a} \in A_s} \sum_{t=1}^{T} q_i^c(\pi^t, s, a) - v_i^c(\pi^t, s)$$

Deep CFR

- Estimate counterfactual regret
- Regrets are updated only for the traverser on an iteration.
- At infosets where the traverser acts, all actions are explored. At other infosets and chance nodes, only a single action is explored.
- Add counterfactual regret estimates to replay buffer
- Train neural network to estimate cumulative regret conditioned on information set



Brown, Lerer, Gross, and Sandholm, *Deep Counterfactual Regret Minimization,* ICML 2019

External Sampling Traversal

Algorithm 2 CFR Traversal with External Sampling

function TRAVERSE($h, p, \theta_1, \theta_2, \mathcal{M}_V, \mathcal{M}_\Pi$, t)

Input: History h, traverser player p, regret network parameters θ for each player, advantage memory $\mathcal{M}_V$ for player p, strategy memory $\mathcal{M}_\Pi$, CFR iteration t.

if h is terminal **then**
  return the payoff to player p
else if h is a chance node **then**
  $a \sim \sigma(h)$
  return TRAVERSE($h \cdot a, p, \theta_1, \theta_2, \mathcal{M}_V, \mathcal{M}_\Pi$, t)
else if $P(h) = p$ **then** ▷ If it's the traverser's turn to act
  Compute strategy $\sigma^t(I)$ from predicted advantages $V(I(h), a|\theta_p)$ using regret matching.
  for $a \in A(h)$ **do**
    $v(a) \leftarrow$ TRAVERSE($h \cdot a, p, \theta_1, \theta_2, \mathcal{M}_V, \mathcal{M}_\Pi$, t) ▷ Traverse each action
  for $a \in A(h)$ **do**
    $\tilde{r}(I, a) \leftarrow v(a) - \sum_{a' \in A(h)} \sigma(I, a') \cdot v(a')$ ▷ Compute advantages
  Insert the infoset and its action advantages $(I, t, \tilde{r}^t(I))$ into the advantage memory $\mathcal{M}_V$
else ▷ If it's the opponent's turn to act
  Compute strategy $\sigma^t(I)$ from predicted advantages $V(I(h), a|\theta_{3-p})$ using regret matching.
  Insert the infoset and its action probabilities $(I, t, \sigma^t(I))$ into the strategy memory $\mathcal{M}_\Pi$
  Sample an action a from the probability distribution $\sigma^t(I)$.
  return TRAVERSE($h \cdot a, p, \theta_1, \theta_2, \mathcal{M}_V, \mathcal{M}_\Pi$, t)

Deep CFR Pseudocode

Algorithm 1 Deep Counterfactual Regret Minimization

function DEEPCFR
    Initialize each player's advantage network $V(I, a|\theta_p)$ with parameters θ_p so that it returns 0 for all inputs.
    Initialize reservoir-sampled advantage memories $\mathcal{M}_{V,1}, \mathcal{M}_{V,2}$ and strategy memory $\mathcal{M}_{\Pi}$.
    for CFR iteration $t = 1$ to T **do**
        for each player p **do**
            for traversal $k = 1$ to K **do**
                TRAVERSE($\emptyset, p, \theta_1, \theta_2, \mathcal{M}_{V,p}, \mathcal{M}_{\Pi}$)   ▷ Collect data from a game traversal with external sampling
            Train θ_p from scratch on loss $\mathcal{L}(\theta_p) = \mathbb{E}_{(I,t',\tilde{r}^{t'})\sim\mathcal{M}_{V,p}}\left[t' \sum_a \left(\tilde{r}^{t'}(a) - V(I, a|\theta_p)\right)^2\right]$
    Train θ_{Π} on loss $\mathcal{L}(\theta_{\Pi}) = \mathbb{E}_{(I,t',\sigma^{t'})\sim\mathcal{M}_{\Pi}}\left[t' \sum_a \left(\sigma^{t'}(a) - \Pi(I, a|\theta_{\Pi})\right)^2\right]$
    return θ_{Π}

(Outcome-Sampling) Monte-Carlo CFR

- Traversing game tree is too costly
- Instead, sample a trajectory and update information sets along trajectory
- Include importance sampling term to remain unbiased

CFR Term

$$\hat{v}_i(\pi, s|z) = \frac{\boxed{\eta^{\pi_{-i}}(z[s])\eta^{\pi}(z[s], z)u_i(z)}}{\boxed{\eta^{\tilde{\pi}}(z)}}$$

Importance Sampling Term

Lanctot et al. Monte Carlo Sampling for Regret Minimization in Extensive Games. NIPS '09

(Outcome-Sampling) Monte-Carlo CFR

- Traversing game tree is too costly
- Instead, sample a trajectory and update information sets along trajectory
- Include importance sampling term to remain unbiased

$$\hat{v}_i(\pi, s|z) = \frac{\overset{\text{CFR Term}}{\boxed{\eta^{\pi_{-i}}(z[s])\eta^{\pi}(z[s], z)u_i(z)}}}{\underset{\text{Importance Sampling Term}}{\boxed{\eta^{\tilde{\pi}}(z)}}} = \frac{1}{\eta^{\tilde{\pi}_i}(z[s])} \overset{\text{Unbiased Estimator of History Value}}{\boxed{\frac{\eta^{\pi_i}(z[s], z)}{\eta^{\tilde{\pi}_i}(z[s], z)} u_i(z)}}$$

Lanctot et al. Monte Carlo Sampling for Regret Minimization in Extensive Games. NIPS ‘09

DREAM

- Use MC-CFR estimator to estimate counterfactual regret
- Add counterfactual regret estimates to replay buffer
- Train neural network to estimate cumulative regret conditioned on information set
- DREAM reduces variance with history-value baseline

$$\hat{u}_i^b(\sigma, h, a|z) = \begin{cases} b_i(I_i(h), a) + \boxed{\frac{\hat{u}_i^b(\sigma, ha|z) - b_i(I_i(h), a)}{\xi(h,a)}} & \text{if } ha \sqsubseteq z \\ b_i(I_i(h), a) & \text{if } h \sqsubset z, ha \not\sqsubseteq z \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

and

$$\hat{u}_i^b(\sigma, h|z) = \begin{cases} u_i(h) & \text{if } h = z \\ \sum_a \sigma(h,a)\hat{u}_i^b(\sigma, h, a|z) & \text{if } h \sqsubset z \\ 0 & \text{otherwise} \end{cases}. \tag{10}$$

$$\hat{v}_i^b(\sigma, I(h), a|z) = \hat{v}_i^b(\sigma, h, a|z) = \boxed{\frac{\pi_{-i}^{\sigma}(h)}{q(h)}}\hat{u}_i^b(\sigma, h, a|z). \tag{11}$$

Problem: importance-sampling term causes estimator to have high variance, making it very difficult to train neural network in large games

ESCHER

1. Replace estimator of history value with learned value function
2. Remove reach-weight importance sampling term by sampling from fixed distribution

$$\boxed{\frac{1}{\eta^{\tilde{\pi}_i}(z[s])}}\boxed{\frac{\eta^{\pi_i}(z[s], z)}{\eta^{\tilde{\pi}_i}(z[s], z)} u_i(z)}$$

$$\downarrow$$

$$v_i(\pi, z[s])$$

$$
\begin{aligned}
\mathbf{E}_{z\sim\tilde{\pi}^i}[\hat{r}_i(\pi,s,a|z)] &= \sum_{z\in Z}\eta^{\tilde{\pi}^i}(z)[\hat{r}_i(\pi,s,a|z)] \\
&= \sum_{z\in Z(s)}\eta^{\tilde{\pi}^i}(z)[q_i(\pi,z[s],a)-v_i(\pi,z[s])] \\
&= \sum_{h\in s}\sum_{z\sqsupset h}\eta^{\tilde{\pi}^i}(z)[q_i(\pi,z[s],a)-v_i(\pi,z[s])] \\
&= \sum_{h\in s}\eta^{\tilde{\pi}^i}(h)[q_i(\pi,h,a)-v_i(\pi,h)] \\
&= \eta_i^{\tilde{\pi}^i}(s)\sum_{h\in s}\eta_{-i}^{\pi}(h)[q_i(\pi,h,a)-v_i(\pi,h)] \\
&= w(s)[v_i^c(\pi,s,a)-v_i^c(\pi,s)] = w(s)r^c(\pi,s,a)
\end{aligned}
$$

Tabular ESCHER Algorithm

Algorithm 1: Tabular ESCHER with Oracle Value Function

```
for t = 1, ..., T do
    for update player i ∈ {0, 1} do
        Sample trajectory τ using sampling distribution π̃^i (Equation 4)
        for each state s ∈ τ do
            for each action a do
                Estimate immediate regret vector r̂(π, s, a|z) = q_i(π, z[s], a) − v_i(π, z[s])
                Update total estimated regret of action a at infostate s:
                  R̂(s, a) = R̂(s, a) + r̂(π, s, a|z)
                Update π_i(s, a) via regret matching on total estimated regret
return average policy π̄
```

ESCHER

Algorithm 2: ESCHER

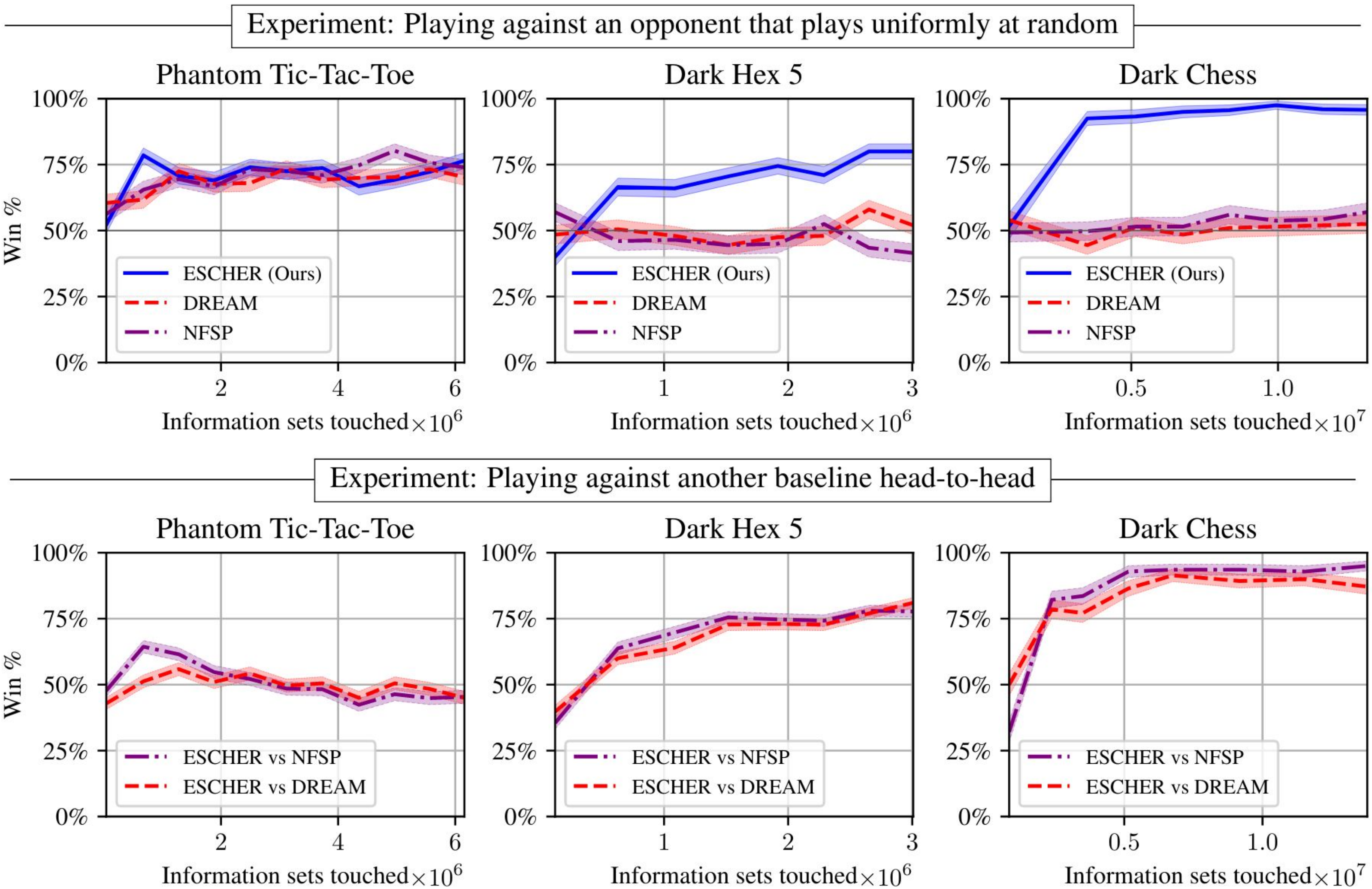
Initialize history value function q
Initialize policy π_i for both players
for $t = 1, ..., T$ **do**
    Retrain history value function q on data from π
    Reinitialize regret networks R_0, R_1
    for *update player* $i \in \{0, 1\}$ **do**
        for P *trajectories* **do**
            Get trajectory τ using sampling distribution (Equation 4)
            for *each state* $s \in \tau$ **do**
                for *each action* a **do**
                    Estimate immediate cf-regret
                    $\hat{r}(\pi, s, a|z) = q_i(\pi, z[s], a|\theta) - \sum_a \pi_i(s, a) q_i(\pi, z[s], a|\theta)$
                Add $(s, \hat{r}(\pi, s))$ to cumulative regret buffer
                Add (s, a') to average policy buffer where a' is action taken at state s in trajectory τ
        Train regret network R_i on cumulative regret buffer
Train average policy network $\bar{\pi}_\phi$ on average policy buffer
return average policy network $\bar{\pi}_\phi$

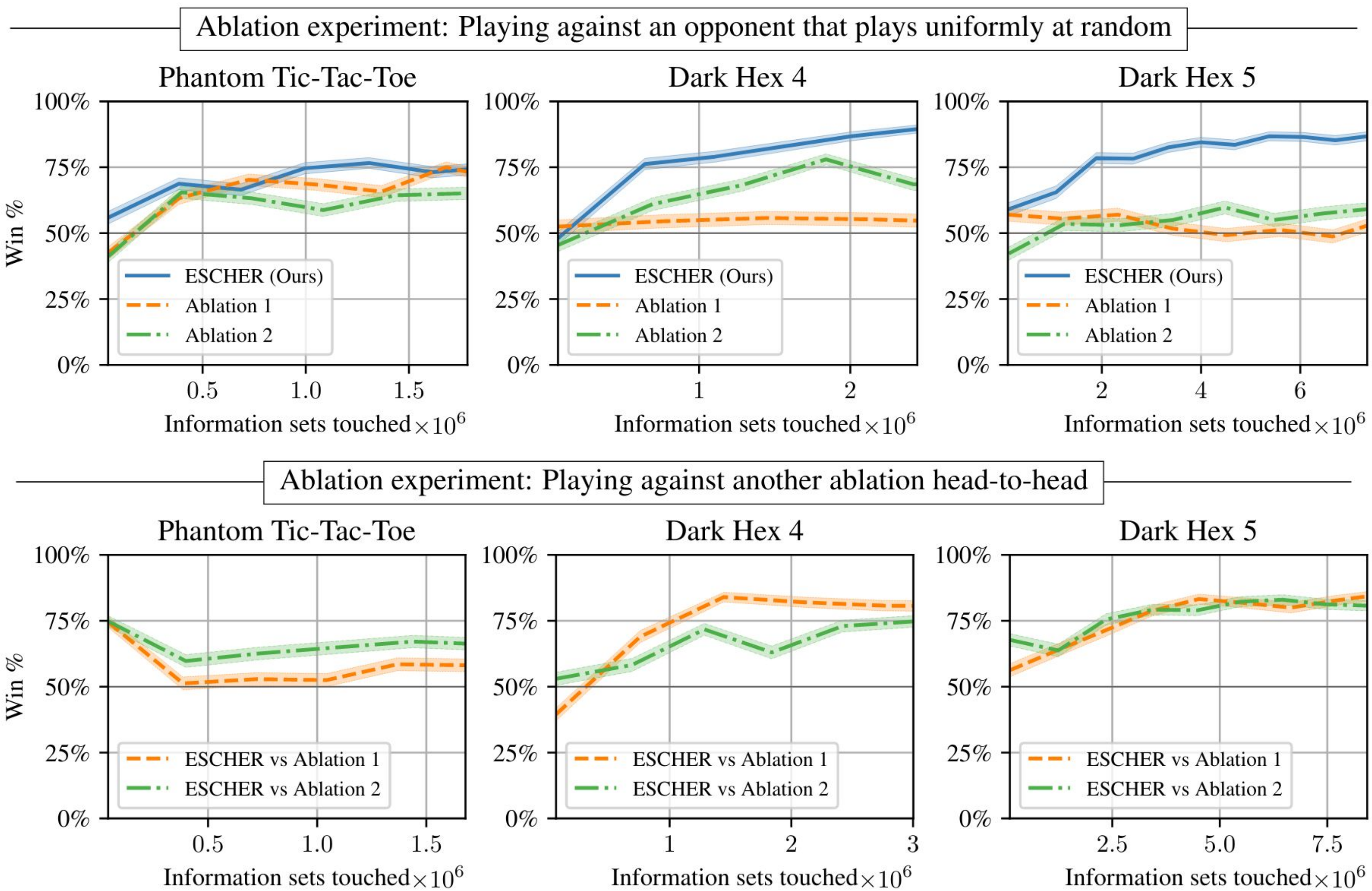
Algorithm	**History value function**	**Boostrapped baseline**	**Importance sampling**
ESCHER (Ours)	✓	✗	✗
Ablation 1	✓	✓	✗
Ablation 2	✓	✗	✓
DREAM / VR-MCCFR	✓	✓	✓
Deep CFR / OS-MCCFR	✗	✗	✓

Variance

Game	ESCHER (Ours)	Ablation 1	Ablation 2	DREAM
Phantom Tic-Tac-Toe	$(2.6 \pm 0.1)\times10^{-1}$	$(4.1 \pm 0.7)\times10^{1}$	$(1.4 \pm 0.4)\times10^{7}$	$(4.6 \pm 1.0)\times10^{7}$
Dark Hex 4	$(1.8 \pm 0.1)\times10^{-1}$	$(1.3 \pm 0.9)\times10^{2}$	$(3.1 \pm 1.7)\times10^{8}$	$(2.8 \pm 2.0)\times10^{8}$
Dark Hex 5	$(1.3 \pm 0.1)\times10^{-1}$	$(3.3 \pm 1.6)\times10^{2}$	$(2.0 \pm 0.6)\times10^{5}$	$(5.3 \pm 3.9)\times10^{8}$

Game	ESCHER (Ours)	Ablation 2	DREAM	OS-MCCFR
Leduc	$(5.3 \pm 0.0)\times10^{0}$	$(3.3 \pm 0.7)\times10^{2}$	$(2.8 \pm 0.0)\times10^{2}$	$(2.2 \pm 0.0)\times10^{3}$
Battleship	$(1.4 \pm 0.0)\times10^{0}$	$(7.1 \pm 0.3)\times10^{2}$	$(1.2 \pm 0.0)\times10^{3}$	$(2.4 \pm 0.0)\times10^{3}$
Liar’s Dice	$(9.0 \pm 0.1)\times10^{-1}$	$(7.8 \pm 0.9)\times10^{1}$	$(4.0 \pm 0.8)\times10^{2}$	$(1.2 \pm 0.1)\times10^{3}$


Experiment: Playing against an opponent that plays uniformly at random
Phantom Tic-Tac-Toe
Dark Hex 5
Dark Chess
Win %
100%
75%
50%
25%
0%
ESCHER (Ours)
DREAM
NFSP
2
4
6
1
3
0.5
1.0
Information sets touched $\times 10^6$
Information sets touched $\times 10^7$
Experiment: Playing against another baseline head-to-head
Phantom Tic-Tac-Toe
Dark Hex 5
Dark Chess
ESCHER vs NFSP
ESCHER vs DREAM
Information sets touched $\times 10^6$
Information sets touched $\times 10^7$


Ablation experiment: Playing against an opponent that plays uniformly at random
Phantom Tic-Tac-Toe
Dark Hex 4
Dark Hex 5
Win %
100%
75%
50%
25%
0%
ESCHER (Ours)
Ablation 1
Ablation 2
0.5
1.0
1.5
1
2
4
6
Information sets touched $\times 10^6$
Ablation experiment: Playing against another ablation head-to-head
Phantom Tic-Tac-Toe
Dark Hex 4
Dark Hex 5
ESCHER vs Ablation 1
ESCHER vs Ablation 2
3
2.5
5.0
7.5

NeuRD

- Recall: All-actions policy gradient

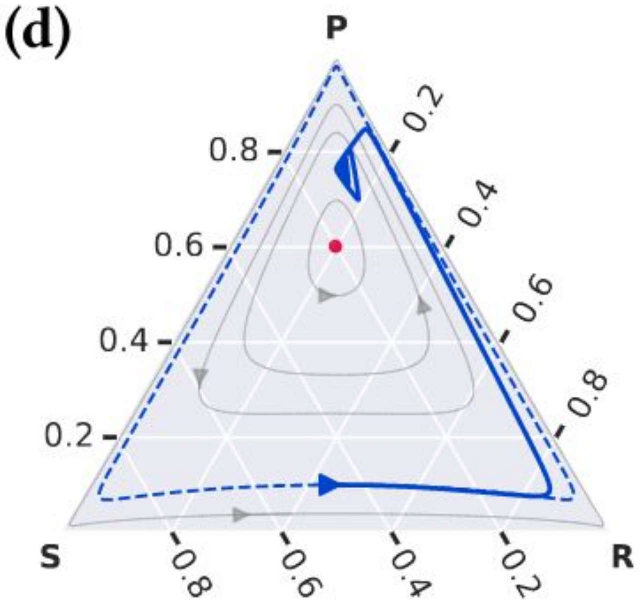
$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + \eta_t \sum_a \nabla_{\boldsymbol{\theta}} \pi(a|s;\boldsymbol{\theta}_{t-1}) \big[q(s,a;\boldsymbol{w}) - v(s;\boldsymbol{w}) \big]$$

NeuRD

- Recall: All-actions policy gradient (where π is a softmax over logits y)

$$\theta_t = \theta_{t-1} + \eta_t \sum_a \nabla_{\theta} \pi(a|s;\theta_{t-1}) \left[q(s,a;\mathbf{w}) - v(s;\mathbf{w}) \right]$$

- This doesn't converge to a NE



NeuRD

- Instead of taking gradient of softmax, take gradient of logits

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + \eta_t \sum_a \nabla_{\boldsymbol{\theta}} \pi(a|s;\boldsymbol{\theta}_{t-1})\left[q(s,a;\boldsymbol{w}) - v(s;\boldsymbol{w})\right]$$

$$\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} + \eta_t \sum_{a'} \nabla_{\boldsymbol{\theta}} y(s,a';\boldsymbol{\theta}_{t-1})\left(q_t(s,a';\boldsymbol{w}_t) - v(s;\boldsymbol{w}_t)\right)$$

NeuRD

- In normal form, gradient of the logit is just the identity, so equivalent to hedge
- As a result, could put it in as the local learning rule in CFR to get CFR-Hedge

$$\theta_t \leftarrow \theta_{t-1} + \eta_t \sum_{a'} \nabla_\theta y(s, a'; \theta_{t-1}) \big(q_t(s, a'; \mathbf{w}_t) - v(s; \mathbf{w}_t) \big)$$

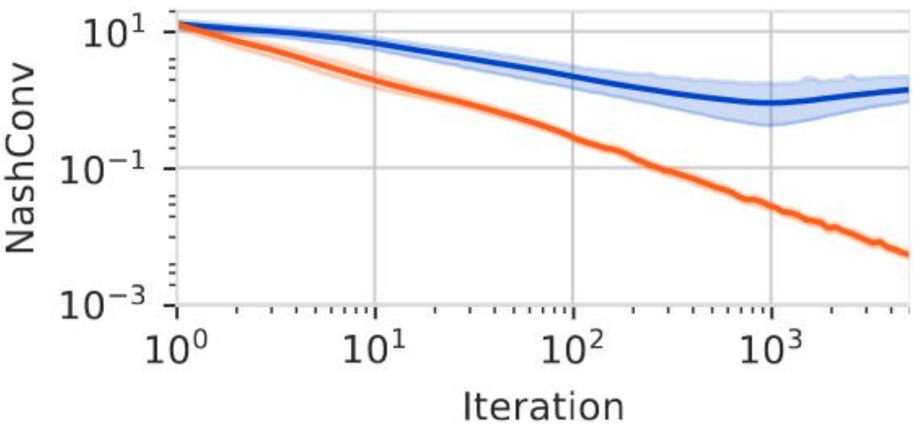
NeuRD

- Practical algorithm: don't do reach weighting but do policy gradient with NeuRD objective **[not guaranteed to converge]**

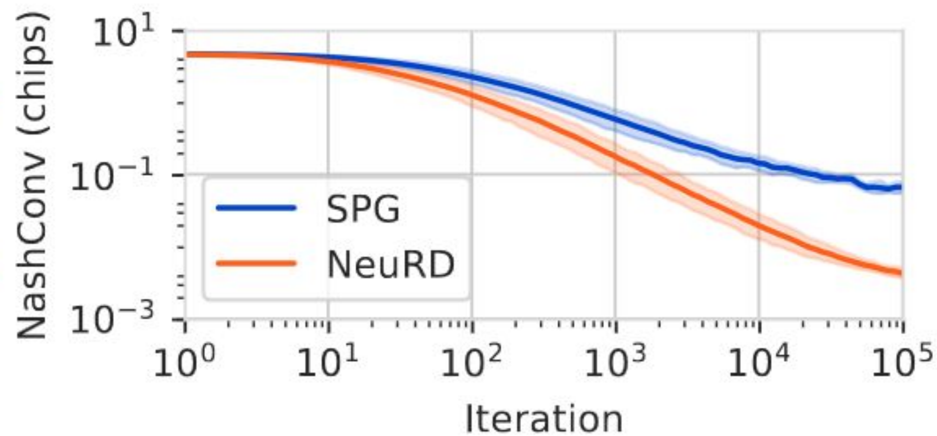
Algorithm 1 Neural Replicator Dynamics (NeuRD)

```
Initialize policy weights θ_0 and critic weights w_0.
for t ∈ {1, 2, ...} do
    π_{t−1}(θ_{t−1}) ← Π(y(θ_{t−1}))
    for all τ ∈ SampleTrajectories(π_{t−1}) do
        for s, a ∈ τ do                                          ▷ POLICY EVALUATION
            R ← Return(s, τ, γ)
            w_t ← UpdateCritic(w_{t−1}, s, a, R)
        for s ∈ τ do                                             ▷ POLICY IMPROVEMENT
            v(s; w_t) ← Σ_{a'} π(s,a';θ_{t−1}) q_t(s, a';w_t)
            θ_t ← θ_{t−1} + η_t Σ_{a'} ∇_θ y(s, a';θ_{t−1})(q_t(s, a';w_t) − v(s; w_t))
```

NeuRD



(a) Biased RPS



(b) Leduc Poker