Multiagent Reinforcement Learning

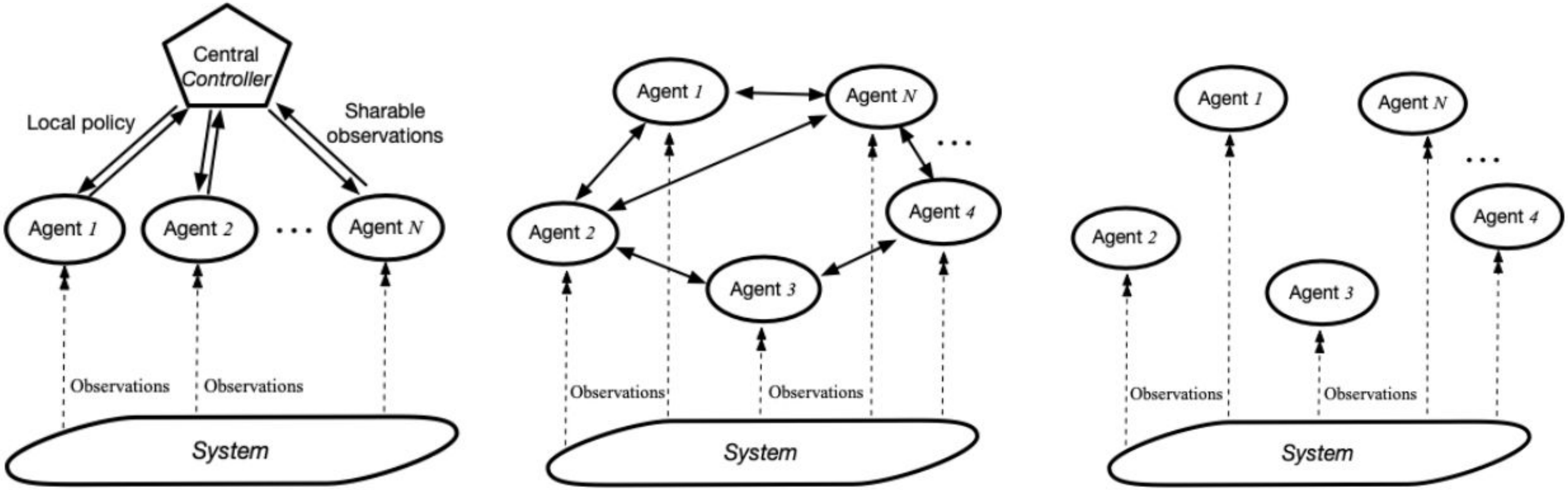
Stephen McAleer

Why Multiagent Reinforcement Learning?

Multiagent RL Landscape

	Cooperative	**Competitive (Zero Sum)**	**General**
2 Player	Independent RL, MADDPG, COMA	Independent RL, NFSP, PSRO, DCFR, NeuRD, ED	???
N Player	Same as above	???	???

Centralized and Decentralized Settings



(a) Centralized setting

(b) Decentralized setting with networked agents

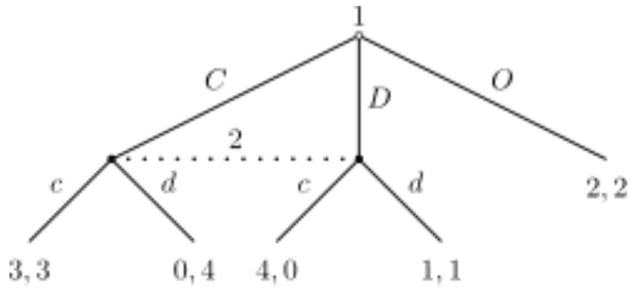
(c) Fully decentralized setting

Game Theory Crash Course

- Game Representation
  - Normal Form Games
  - Extensive Form Games
- Zero-sum
  - One player's gain is the other player's loss
- Best Response
  - Best possible strategy to other player's fixed strategy
- Nash Equilibrium
  - All players are playing a best response to each other
- Mixed Nash always exist
- In zero-sum games, playing a Nash is optimal

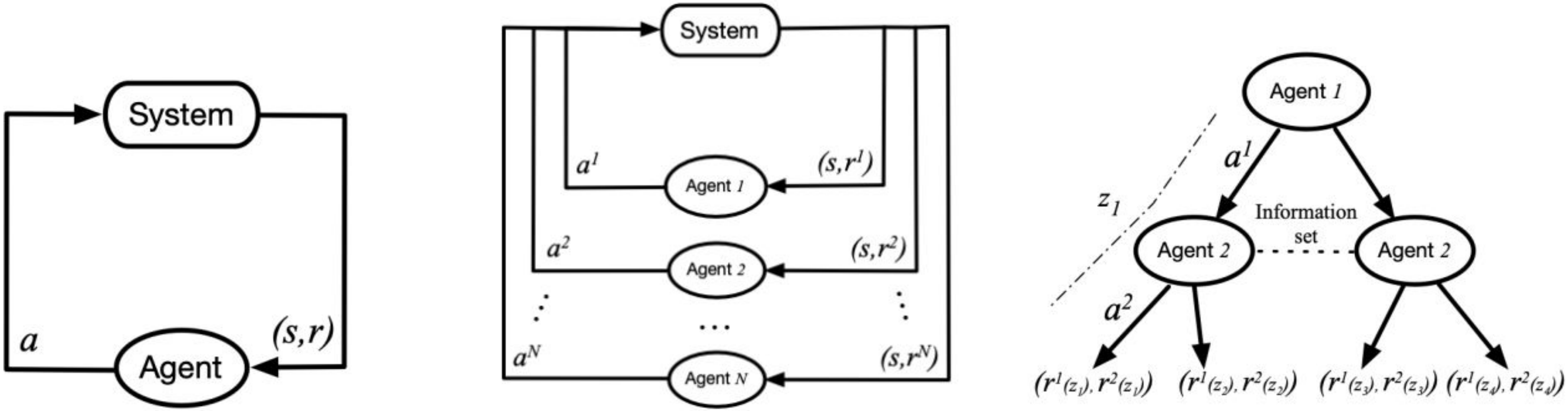
3, 3	1, 4
4, 1	2, 2

Normal Form Game



Extensive Form Game

Multiagent Environments



(a) MDP

(b) Markov game

(c) Extensive-form game

Multiagent RL Landscape

	Cooperative	Competitive (Zero Sum)	General
2 Player	**Independent RL, MADDPG, COMA**	Independent RL, NFSP, PSRO, DCFR, NeuRD, ED	???
N Player	Same as above	???	???

Cooperative RL

- Every agent has same reward function
  - Might be a sum of individual reward functions
- Team of agents work together to accomplish common goal

Difficulties of Cooperative RL

- Nash Equilibria Selection Problem

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

Difficulties of Cooperative RL

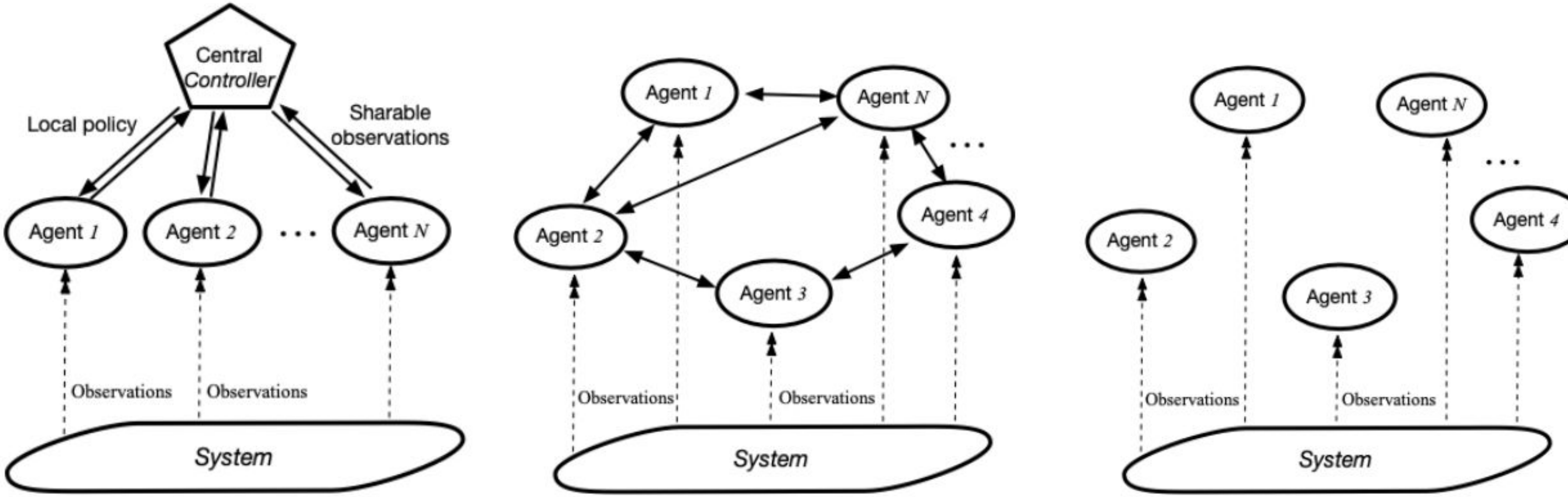
- But in practice it can work surprisingly well

100	20	0	0
20	30	0	0
0	0	1	0
0	0	0	1

Independent RL (Cooperative Setting)

- Simply pretend each agent is in an MDP and optimize using standard RL
- Can be very unstable but can also work well

Centralized and Decentralized Settings



(a) Centralized setting (b) Decentralized setting with networked agents (c) Fully decentralized setting

Multi-Agent Deep Deterministic Policy Gradient (MADDPG)

- Uses centralized training and decentralized execution
- While training, use a centralized critic that takes in all agents' observations

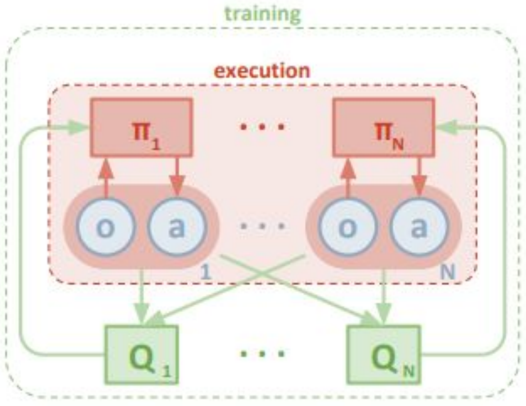


Figure 1: Overview of our multi-agent decentralized actor, centralized critic approach.

$$\nabla_{\theta_i} J(\boldsymbol{\mu}_i) = \mathbb{E}_{\mathbf{x},a\sim\mathcal{D}}[\nabla_{\theta_i}\boldsymbol{\mu}_i(a_i|o_i)\nabla_{a_i}Q_i^{\boldsymbol{\mu}}(\mathbf{x}, a_1, ..., a_N)|_{a_i=\boldsymbol{\mu}_i(o_i)}]$$

Counterfactual Multi-Agent Policy Gradients (COMA)

- Similar idea to MADDPG
- Use centralized critic, “counterfactual values”

$$g = \mathbb{E}_{\boldsymbol{\pi}} \left[\sum_a \nabla_\theta \log \pi^a(u^a|\tau^a) A^a(s, \mathbf{u}) \right], \quad (7)$$

$$A^a(s, \mathbf{u}) = Q(s, \mathbf{u}) - \sum_{u'^a} \pi^a(u'^a|\tau^a) Q(s, (\mathbf{u}^{-a}, u'^a)).$$

Multiagent RL Landscape

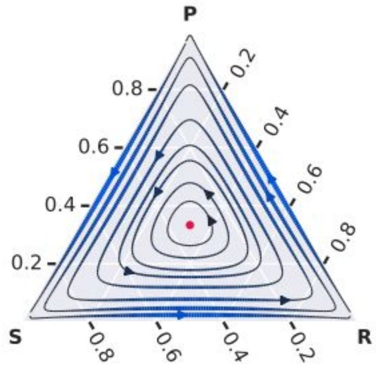
	Cooperative	**Competitive (Zero Sum)**	**General**
2 Player	Independent RL, MADDPG, COMA	**Independent RL, NFSP, PSRO, DCFR, NeuRD, ED**	???
N Player	Same as above	???	???

Two Player Zero-Sum Games

- Most well-understood theoretically
- Algorithms seek to find approximate Nash Equilibria
- Can use reinforcement learning to find approximate best response
- In fully-observable games can use versions of minimax tree search

Independent RL (Zero-Sum games)

- Independent RL fails to converge to Nash for very simple games such as Rock Paper Scissors
- However, in practice it seems to work well on large games
    - Starcraft
    - Dota



Fictitious Play

- Every iteration add best response to population to the population
- Population average strategy converges to Nash

	Rock	Paper	Scissors
Pop Average	.8	.1	.1
BR	0	1	0
Pop Average	.4	.55	.05
BR	0	0	1
Pop Average	.26	.37	.37

Extensive form Fictitious Play (XFP)

$$\hat{\sigma} = \lambda_1 \hat{\pi}_1 + \lambda_2 \hat{\pi}_2,$$

$$\sigma(s,a) \propto \lambda_1 x_{\pi_1}(s) \pi_1(s,a) + \lambda_2 x_{\pi_2}(s) \pi_2(s,a) \quad \forall s, a,$$

Algorithm 1 Full-width extensive-form fictitious play

function FICTITIOUSPLAY(Γ)
  Initialize π_1 arbitrarily
  $j \leftarrow 1$
  while within computational budget **do**
    $\beta_{j+1} \leftarrow$ COMPUTEBRS(π_j)
    $\pi_{j+1} \leftarrow$ UPDATEAVGSTRATEGIES(π_j, β_{j+1})
    $j \leftarrow j + 1$
  end while
  return π_j
end function

function COMPUTEBRS(π)
  Recursively parse the game's state tree to compute a best response strategy profile, $\beta \in b(\pi)$.
  return β
end function

function UPDATEAVGSTRATEGIES(π_j, β_{j+1})
  Compute an updated strategy profile π_{j+1} according to Theorem 7.
  return π_{j+1}
end function

Fictitious Self Play

- For each information state u the probability distribution of player i's behaviour at u induced by sampling from the strategy profile Π defines a behavioural strategy at u and is realization equivalent to Π.

Algorithm 2 General Fictitious Self-Play

function FICTITIOUSSELFPLAY(Γ, n, m)
  Initialize completely mixed π_1
  $\beta_2 \leftarrow \pi_1$
  $j \leftarrow 2$
  while within computational budget **do**
    $\eta_j \leftarrow$ MIXINGPARAMETER(j)
    $\mathcal{D} \leftarrow$ GENERATEDATA($\pi_{j-1}, \beta_j, n, m, \eta_j$)
    for each player $i \in \mathcal{N}$ **do**
      $\mathcal{M}^i_{RL} \leftarrow$ UPDATERLMEMORY($\mathcal{M}^i_{RL}, \mathcal{D}^i$)
      $\mathcal{M}^i_{SL} \leftarrow$ UPDATESLMEMORY($\mathcal{M}^i_{SL}, \mathcal{D}^i$)
      $\beta^i_{j+1} \leftarrow$ REINFORCEMENTLEARNING($\mathcal{M}^i_{RL}$)
      $\pi^i_j \leftarrow$ SUPERVISEDLEARNING($\mathcal{M}^i_{SL}$)
    end for
    $j \leftarrow j + 1$
  end while
  return π_{j-1}
end function

function GENERATEDATA(π, β, n, m, η)
  $\sigma \leftarrow (1-\eta)\pi + \eta\beta$
  $\mathcal{D} \leftarrow n$ episodes $\{t_k\}_{1 \le k \le n}$, sampled from strategy profile σ
  for each player $i \in \mathcal{N}$ **do**
    $\mathcal{D}^i \leftarrow m$ episodes $\{t^i_k\}_{1 \le k \le m}$, sampled from strategy profile (β^i, σ^{-i})
    $\mathcal{D}^i \leftarrow \mathcal{D}^i \cup \mathcal{D}$
  end for
  return $\{\mathcal{D}^k\}_{1 \le k \le N}$
end function

Policy Gradient Fictitious Self Play (unpublished)

Initialize average policy

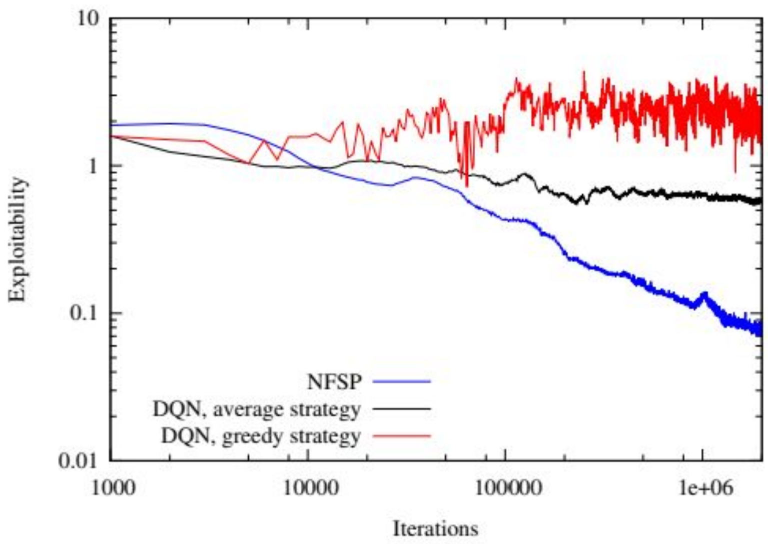
For each episode do:

Train approximate best response to average policy with policy gradient

Update average policy with supervised learning on trajectories

Neural Fictitious Self Play

- Fictitious Self Play with deep Q learning
- Two networks: one learns best response to average strategy with RL, one learns average strategy with supervised learning
- Average strategy converges to Nash

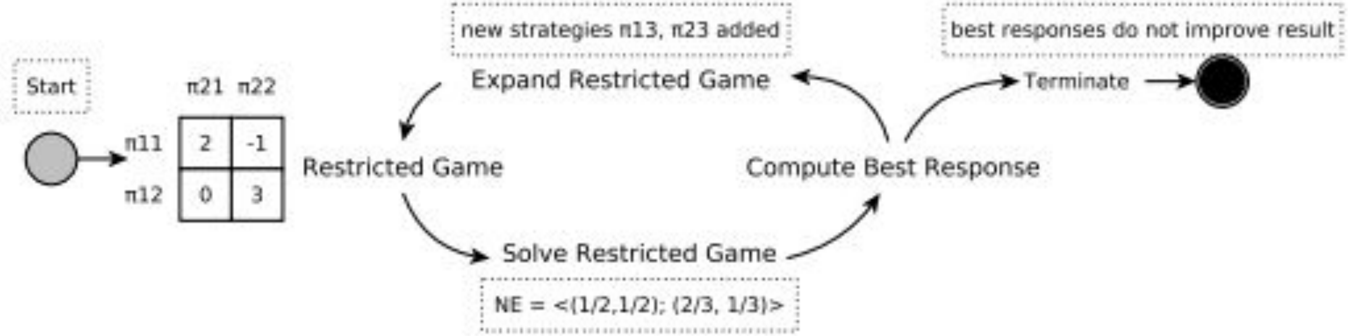


Algorithm 1 Neural Fictitious Self-Play (NFSP) with fitted Q-learning

Initialize game Γ and execute an agent via RunAgent for each player in the game
function RunAgent(Γ)
    Initialize replay memories $\mathcal{M}_{RL}$ (circular buffer) and $\mathcal{M}_{SL}$ (reservoir)
    Initialize average-policy network $\Pi(s, a \mid \theta^{\Pi})$ with random parameters θ^{Π}
    Initialize action-value network $Q(s, a \mid \theta^{Q})$ with random parameters θ^{Q}
    Initialize target network parameters $\theta^{Q'} \leftarrow \theta^{Q}$
    Initialize anticipatory parameter η
    for each episode **do**
        Set policy $\sigma \leftarrow \begin{cases} \epsilon\text{-greedy}\,(Q), & \text{with probability } \eta \\ \Pi, & \text{with probability } 1-\eta \end{cases}$
        Observe initial information state s_1 and reward r_1
        for $t = 1, T$ **do**
            Sample action a_t from policy σ
            Execute action a_t in game and observe reward r_{t+1} and next information state s_{t+1}
            Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in reinforcement learning memory $\mathcal{M}_{RL}$
            if agent follows best response policy $\sigma = \epsilon$-greedy (Q) **then**
                Store behaviour tuple (s_t, a_t) in supervised learning memory $\mathcal{M}_{SL}$
            end if
            Update θ^{Π} with stochastic gradient descent on loss
                $\mathcal{L}(\theta^{\Pi}) = \mathbb{E}_{(s,a)\sim\mathcal{M}_{SL}}\left[-\log \Pi(s, a \mid \theta^{\Pi})\right]$
            Update θ^{Q} with stochastic gradient descent on loss
                $\mathcal{L}\left(\theta^{Q}\right) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{M}_{RL}}\left[\left(r + \max_{a'} Q(s', a' \mid \theta^{Q'}) - Q(s, a \mid \theta^{Q})\right)^2\right]$
            Periodically update target network parameters $\theta^{Q'} \leftarrow \theta^{Q}$
        end for
    end for
end function

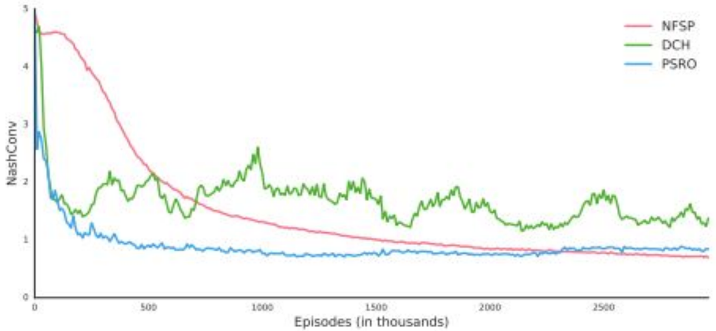
Double Oracle

- Every iteration, add best response to meta-Nash of population



PSRO

- Double Oracle Algorithm but uses RL as approximate best response
- Fictitious Self Play is PSRO but weighted uniformly and with supervised learning for average policy

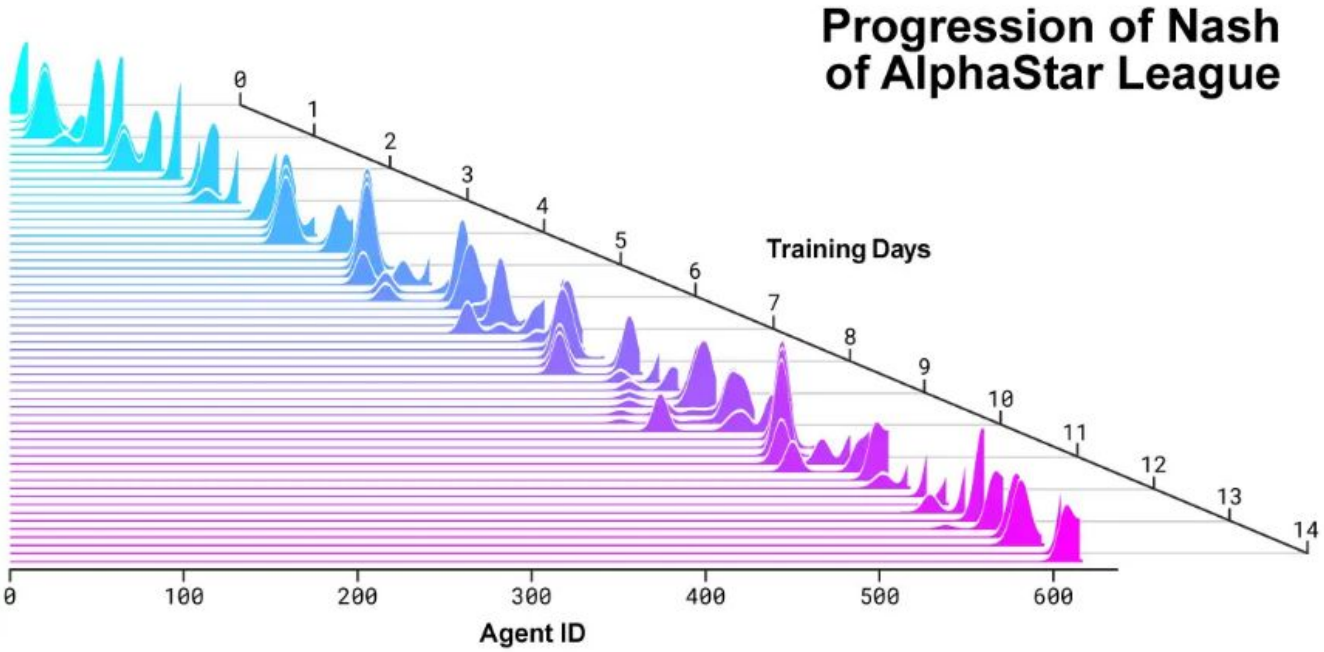


(a) 2 players

Algorithm 1: Policy-Space Response Oracles

---

input : initial policy sets for all players Π
Compute exp. utilities U^{Π} for each joint $\pi \in \Pi$
Initialize meta-strategies $\sigma_i = \textsc{Uniform}(\Pi_i)$
while *epoch* e *in* $\{1, 2, \cdots\}$ **do**
    for *player* $i \in [[n]]$ **do**
        for *many episodes* **do**
            Sample $\pi_{-i} \sim \sigma_{-i}$
            Train oracle π'_i over $\rho \sim (\pi'_i, \pi_{-i})$
        $\Pi_i = \Pi_i \cup \{\pi'_i\}$
    Compute missing entries in U^{Π} from Π
    Compute a meta-strategy σ from U^{Π}
Output current solution strategy σ_i for player i



Progression of Nash
of AlphaStar League
Training Days
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
0
100
200
300
400
500
600
Agent ID

Counterfactual Regret Minimization (CFR)

- Same as regret matching but for extensive form games
- Weights regret by probability that that game node is reached when player always takes actions to get to that game node

$$v_i(\sigma, h) = \sum_{z \in Z, h \sqsubset z} \pi^{\sigma}_{-i}(h)\pi^{\sigma}(h, z)u_i(z).$$

$$r(h, a) = v_i(\sigma_{I \to a}, h) - v_i(\sigma, h).$$

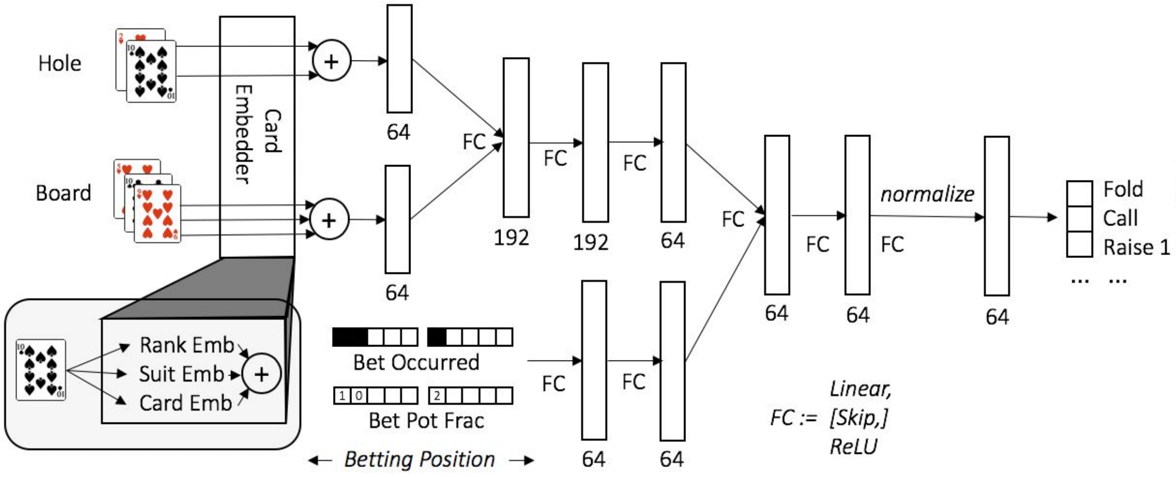
$$r(I, a) = \sum_{h \in I} r(h, a)$$

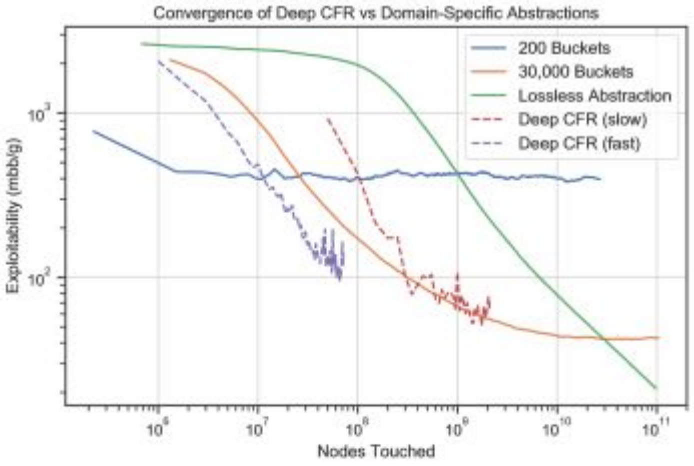
$$R_i^T(I, a) = \sum_{t=1}^{T} r_i^t(I, a)$$

$$\sigma_i^{T+1}(I, a) = \begin{cases} \frac{R_i^{T,+}(I,a)}{\sum_{a \in A(I)} R_i^{T,+}(I,a)} & \text{if } \sum_{a \in A(I)} R_i^{T,+}(I, a) > 0 \\ \frac{1}{|A(I)|} & \text{otherwise.} \end{cases}$$

Deep CFR

- Partially goes through game tree and trains neural network on CFR buffer





Connection Between Policy Gradients and Counterfactual Regret

So, $q_{\pi,i}(s_t, a_t) = \mathbb{E}_{\rho\sim\pi}[G_{t,i} \mid S_t = s_t, A_t = a_t]$

$$= \sum_{h,z\in\mathcal{Z}(s_t,a_t)} \Pr(h \mid s_t)\eta^\pi(ha,z)u_i(z) \qquad \text{where } \eta^\pi(ha,z) = \frac{\eta^\pi(z)}{\eta^\pi(h)\pi(s,a)}$$

$$= \sum_{h,z\in\mathcal{Z}(s_t,a_t)} \frac{\Pr(s_t \mid h)\Pr(h)}{\Pr(s_t)}\eta^\pi(ha,z)u_i(z) \qquad \text{by Bayes' rule}$$

$$= \sum_{h,z\in\mathcal{Z}(s_t,a_t)} \frac{\Pr(h)}{\Pr(s_t)}\eta^\pi(ha,z)u_i(z) \qquad \text{since } h\in s_t,\ h \text{ is unique to } s_t$$

$$= \sum_{h,z\in\mathcal{Z}(s_t,a_t)} \frac{\eta^\pi(h)}{\sum_{h'\in s_t}\eta^\pi(h')}\eta^\pi(ha,z)u_i(z)$$

$$= \sum_{h,z\in\mathcal{Z}(s_t,a_t)} \frac{\eta_i^\pi(h)\eta_{-i}^\pi(h)}{\sum_{h'\in s_t}\eta_i^\pi(h')\eta_{-i}^\pi(h')}\eta^\pi(ha,z)u_i(z)$$

$$= \sum_{h,z\in\mathcal{Z}(s_t,a_t)} \frac{\eta_i^\pi(s)\eta_{-i}^\pi(h)}{\eta_i^\pi(s)\sum_{h'\in s_t}\eta_{-i}^\pi(h')}\eta^\pi(ha,z)u_i(z) \qquad \text{due to def. of } s_t \text{ and perfect recall}$$

$$= \sum_{h,z\in\mathcal{Z}(s_t,a_t)} \frac{\eta_{-i}^\pi(h)}{\sum_{h'\in s_t}\eta_{-i}^\pi(h')}\eta^\pi(ha,z)u_i(z) = \frac{1}{\sum_{h\in s_t}\eta_{-i}^\pi(h)}v_i^c(\pi, s_t, a_t).$$

QPG/RPG/RMPG

$$\nabla_{\boldsymbol{\theta}}^{\text{QPG}}(s) = \sum_{a} [\nabla_{\theta} \pi(s, a; \boldsymbol{\theta})] \left(q(s, a; \mathbf{w}) - \sum_{b} \pi(s, b; \boldsymbol{\theta}) q(s, b, \mathbf{w}) \right)$$

$$\nabla_{\boldsymbol{\theta}}^{\text{RPG}}(s) = -\sum_{a} \nabla_{\theta} \left(q(s, a; \mathbf{w}) - \sum_{b} \pi(s, b; \boldsymbol{\theta}) q(s, b; \mathbf{w}) \right)^{+}$$

$$\nabla_{\boldsymbol{\theta}}^{\text{RMPG}}(s) = \sum_{a} [\nabla_{\theta} \pi(s, a; \boldsymbol{\theta})] \left(q(s, a; \mathbf{w}) - \sum_{b} \pi(s, b; \boldsymbol{\theta}) q(s, b, \mathbf{w}) \right)^{+}$$

Exploitability Descent

- Based off of BR-CFR
- Last iteration converges to approximate Nash
- However, very expensive to do BR calculations

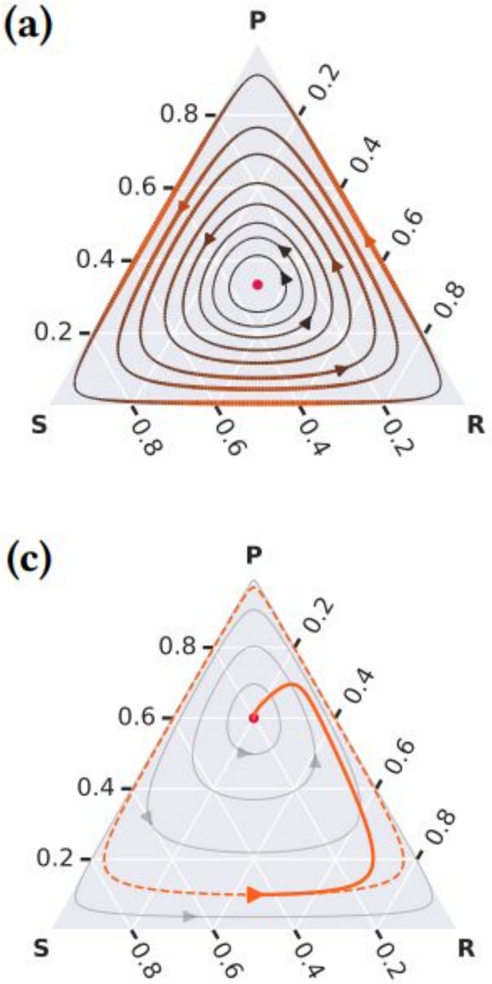
Algorithm 2: Exploitability Descent (ED)

input : π^0 — initial joint policy
for $t \in \{1, 2, \cdots\}$ **do**
  for $i \in \{1, \cdots, n\}$ **do**
    Compute a best response $b_i^t(\pi_{-i}^{t-1})$
  for $i \in \{1, \cdots, n\}, s \in \mathcal{S}_i$ **do**
    Define $b_{-i}^t = \{b_j^t\}_{j \neq i}$
    Let $\mathbf{q}^b(s) = \textsc{ValuesVsBRs}(\pi_i^{t-1}(s), b_{-i}^t)$
    $\pi_i^t(s) = \textsc{GradAscent}(\pi_i^{t-1}(s), \alpha^t, \mathbf{q}^b(s))$

Replicator Dynamics

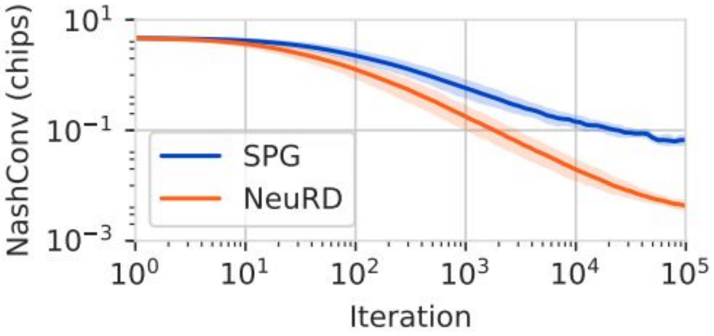
- Members of the population replicate in proportion to their relative fitness
- Average dynamics converges to Nash

$$x_i(t+1) = x_i(t)\frac{f_i(t)}{\bar{f}(t)}$$



Neural Replicator Dynamics (NeuRD)

- Approximates replicator dynamics with neural network
- Turns out to be a policy gradient



(b) Leduc Poker

$$y_t(a) \doteq y(a;\boldsymbol{\theta}_{t-1}) + \eta_t \left(q^{\boldsymbol{\pi}_t}(a) - v^{\boldsymbol{\pi}_t}\right)$$

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \sum_a \nabla_{\boldsymbol{\theta}} d(y_t(a), y(a;\boldsymbol{\theta}_{t-1})).$$

$$\begin{aligned}
\boldsymbol{\theta}_t &= \boldsymbol{\theta}_{t-1} - \sum_a \nabla_{\boldsymbol{\theta}} \frac{1}{2} \| y_t(a) - y(a;\boldsymbol{\theta}_{t-1}) \|^2 \\
&= \boldsymbol{\theta}_{t-1} + \sum_a (y_t(a) - y(a;\boldsymbol{\theta}_{t-1})) \nabla_{\boldsymbol{\theta}} y(a;\boldsymbol{\theta}_{t-1}) \\
&\overset{(9)}{=} \boldsymbol{\theta}_{t-1} + \eta_t \sum_a \nabla_{\boldsymbol{\theta}} y(a;\boldsymbol{\theta}_{t-1}) \left(q^{\boldsymbol{\pi}}(a) - v^{\boldsymbol{\pi}}\right),
\end{aligned}$$

Multiagent RL Landscape

	Cooperative	Competitive (Zero Sum)	General
2 Player	Independent RL, MADDPG, COMA	Independent RL, NFSP, PSRO, DCFR, NeuRD, ED	???
N Player	Same as above	???	???

References

- Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments
- Counterfactual Multi-Agent Policy Gradients
- Grandmaster level in StarCraft II using multi-agent reinforcement learning
- Fictitious Self-Play in Extensive-Form Games
- Deep Reinforcement Learning from Self-Play in Imperfect-Information Games
- A Unified Game-Theoretic Approach to Multiagent Reinforcement Learning
- Actor-Critic Policy Optimization in Partially Observable Multiagent Environments
- Neural Replicator Dynamics
- Deep Counterfactual Regret Minimization
- Computing Approximate Equilibria in Sequential Adversarial Games by Exploitability Descent