Reinforcement Learning China Summer School

RLChina 2020

Learning with Sparse Rewards

Jianye Hao
Noah's Ark Lab/Tianjin University

Aug 3th, 2020

Learning with Sparse Rewards

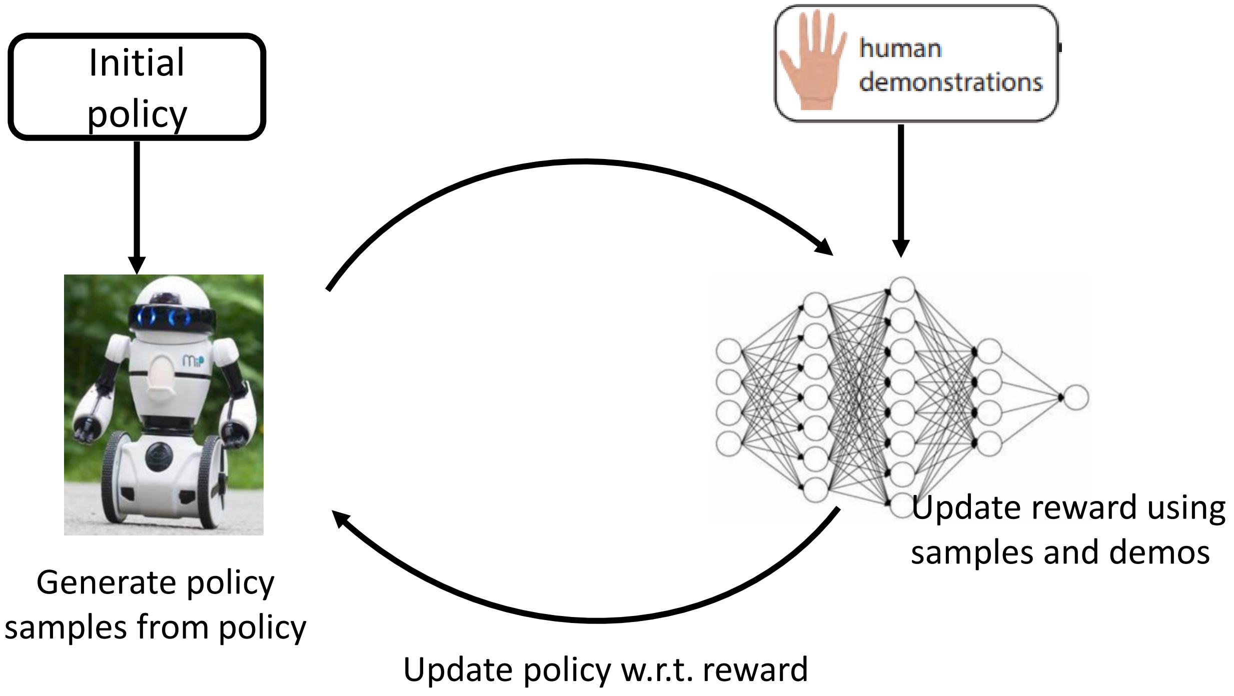
- From Sparse to Dense
  - Reward Learning/Shaping
    - leveraging expert/good trajectory to learn optimal reward signals (SGAIL/Multiagent GSAIL)
    - generate intrinsic rewards to encourage better explorations (exploration-oriented intrinsic rewards)
  - Temporal/spatial credit assignment (single-agent/multiagent settings)
    - Decompose sparse termination reward into previous time steps (single-agent credit assignment)
    - Decompose global rewards into individual agents (multiagent credit assignment)
  - Task hierarchical decomposition (hierarchical RL)
    - Decompose original tasks into discrete/continuous subtasks to provide dense rewards
    - High-level: MDP->Semi-MDP; Low-level: receive reward feedbacks from subgoals

Learning with Sparse Rewards

- From Sparse to Dense
  - **Reward Learning/Shaping** (SGAIL/Multiagent SGAIL, exploration-oriented intrinsic rewards)
  - Temporal/spatial credit assignment (single-agent/multiagent settings)
  - Task hierarchical decomposition (hierarchical RL)

Generative Adversarial Imitation learning

- Policy: generator
- Reward Function: Discriminator



Generative Adversarial Imitation learning

- The objective of GAIL is defined as：

$$\underset{\theta}{\operatorname{argmin}}\,\underset{\phi}{\operatorname{arg}max}\,\mathcal{L}_{\mathrm{GASIL}}(\theta,\phi)\text{=}E_{\pi_\theta}[\log D_\phi(s,a)]+E_{\pi_E}\left[\log\left(1-D_\phi(s,a)\right)\right]-\lambda H(\pi_\theta)$$

- The discriminator and the policy plays an adversarial game by maximizing or minimizing the above objective function

$$\nabla_\phi\mathcal{L}_{GAIL}=\mathbb{E}_{\tau_\pi}\left[\nabla_\phi\log D_\phi(s,a)\right]+\mathbb{E}_{\tau_E}\left[\nabla_\phi\log\left(1-D_\phi(s,a)\right)\right]$$

$$\begin{aligned}\nabla_\theta\mathcal{L}_{GAIL}&=\mathbb{E}_{\tau_\pi}\left[\nabla_\theta\log D_\phi(s,a)\right]-\lambda\nabla_\theta\mathcal{H}(\pi_\theta)\\&=\mathbb{E}_{\tau_\pi}[\nabla_\theta lo\,g\,\pi_\theta\,(\,a\mid s\,)Q(s,a)]-\lambda\nabla_\theta\mathcal{H}(\pi_\theta)\end{aligned}$$

$$Q(s,a)=\mathbb{E}_{\tau_\pi}[log\,D_\phi(s,a)\mid s_0=s,a_0=a]$$

Generative adversarial imitation learning, NeurIPS 2016

Generative Adversarial Imitation learning

Algorithm 1 Generative adversarial imitation learning

1: **Input:** Expert trajectories $\tau_E \sim \pi_E$, initial policy and discriminator parameters θ_0, w_0
2: **for** $i = 0, 1, 2, \ldots$ **do**
3: Sample trajectories $\tau_i \sim \pi_{\theta_i}$
4: Update the discriminator parameters from w_i to w_{i+1} with the gradient

$$\hat{\mathbb{E}}_{\tau_i}[\nabla_w \log(D_w(s,a))] + \hat{\mathbb{E}}_{\tau_E}[\nabla_w \log(1 - D_w(s,a))] \tag{17}$$

5: Take a policy step from θ_i to θ_{i+1}, using the TRPO rule with cost function $\log(D_{w_{i+1}}(s,a))$. Specifically, take a KL-constrained natural gradient step with

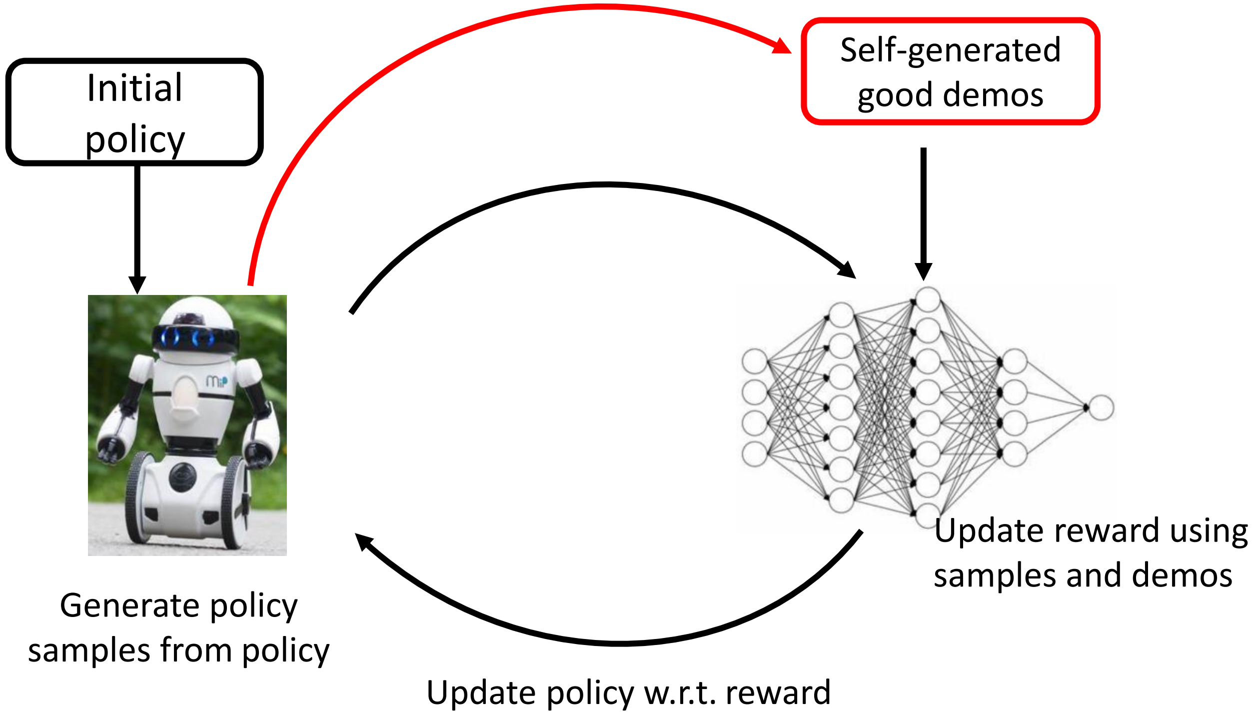
$$\begin{aligned} &\hat{\mathbb{E}}_{\tau_i}\left[\nabla_\theta \log \pi_\theta(a|s) Q(s,a)\right] - \lambda \nabla_\theta H(\pi_\theta), \\ &\text{where } Q(\bar{s},\bar{a}) = \hat{\mathbb{E}}_{\tau_i}[\log(D_{w_{i+1}}(s,a)) \mid s_0 = \bar{s}, a_0 = \bar{a}] \end{aligned} \tag{18}$$

6: **end for**

Generative adversarial imitation learning, NeurIPS 2016

Generative Adversarial Self-Imitation Learning

- Generative Adversarial Self-Imitation Learning (GSAIL)
  - Imitate past good trajectories that the agent has generated using generative adversarial imitation learning framework
  - Solves the temporal credit assignment problem- make long-term temporal credit assignment easier when reward signal is delayed and sparse
  - Discriminator – reward shaping function: providing dense internal rewards for the agent to reproduce relatively better trajectories



Generative Adversarial Self-Imitation learning

- Update good trajectory buffer
  - Maintain a good trajectory buffer with high-reward trajectories in the past
  - High-reward trajectories: rewards higher than that of the current policy (top-K episodes according to the return)
  - Update discriminator and policy

$$\underset{\theta}{\operatorname{argmin}}\, \underset{\phi}{\operatorname{arg}max}\, \mathcal{L}_{\mathrm{GASIL}}(\theta, \phi) = \mathbb{E}_{\tau_\pi}[log D_\phi(s, a)] + \mathbb{E}_{\tau_E \sim \mathcal{B}}[log\ (1 - D_\phi(s, a))] - \lambda \mathcal{H}(\pi_\theta)$$

Generative Adversarial Self-Imitation Learning, NeurIPS 2019

Generative Adversarial Self-Imitation learning

Algorithm 1 Generative Adversarial Self-Imitation Learning

Initialize policy parameter θ
Initialize discriminator parameter ϕ
Initialize good trajectory buffer $\mathcal{B} \leftarrow \emptyset$
for each iteration **do**
  Sample policy trajectories $\tau_\pi \sim \pi_\theta$
  Update good trajectory buffer $\mathcal{B}$ using τ_π
  Sample good trajectories $\tau_E \sim \mathcal{B}$
  Update the discriminator parameter ϕ via gradient ascent with:

$$\nabla_\phi \mathcal{L}_{\text{GASIL}} = \mathbb{E}_{\tau_\pi}\left[\nabla_\phi \log D_\phi(s,a)\right] + \mathbb{E}_{\tau_E}\left[\nabla_\phi \log(1 - D_\phi(s,a))\right] \tag{8}$$

  Update the policy parameter θ via gradient descent with:

$$\begin{aligned} &\nabla_\theta \mathcal{L}_{\text{GASIL}} = \mathbb{E}_{\tau_\pi}\left[\nabla_\theta \log \pi_\theta(a|s) Q(s,a)\right] - \lambda \nabla_\theta \mathcal{H}(\pi_\theta), \\ &\text{where } Q(s,a) = \mathbb{E}_{\tau_\pi}\left[\log D_\phi(s,a) | s_0 = s, a_0 = a\right] \end{aligned} \tag{9}$$

end for

Generative Adversarial Self-Imitation Learning, NeurIPS 2019

Generative Adversarial Self-Imitation learning

- Connection to reward learning
  - The discriminator serves as the reward function that the policy optimize over $-logD_{\phi}(s,a)$
  - The policy is updated to maximize the sum of rewards provided by the discriminator
  - It can be viewed as an instance of optimal reward learning algorithm since D is also learning
  - Provide dense reward to the policy when the environment reward is spares or delayed.

Generative Adversarial Self-Imitation Learning, NeurIPS 2019

Generative Adversarial Self-Imitation learning

- Connection to reward shaping
  - GSAIL can be combined with policy gradient

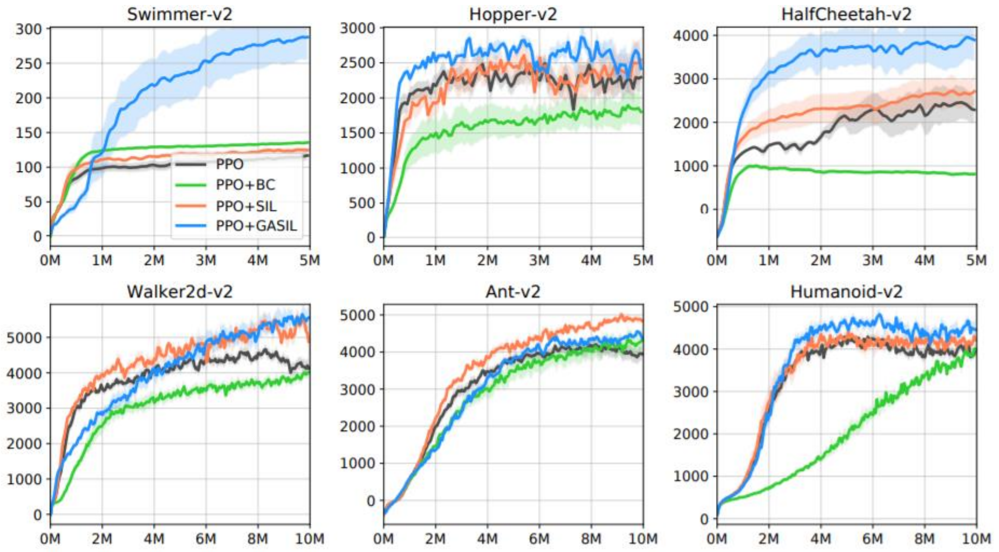
$$\nabla_\theta J_{\mathrm{PG}} - \alpha \nabla_\theta \mathcal{L}_{\mathrm{GASIL}} = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a \mid s) \hat{A}_t^\alpha]$$

  where $\hat{A}_t^\alpha$ is an advantage estimation using a modified reward function

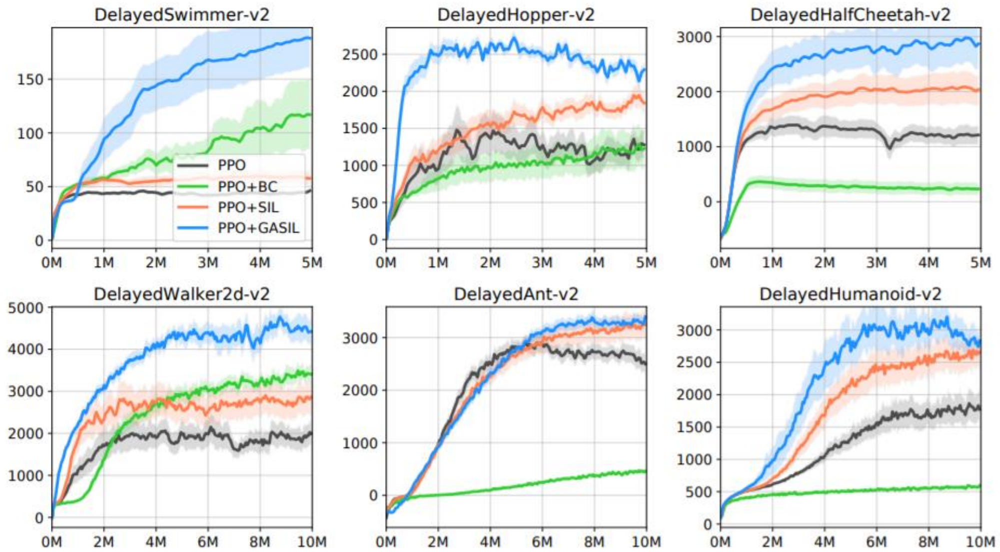
$$r^\alpha(s,a) \triangleq r(s,a) - \alpha log D_\phi(s,a)$$

  - Intuitively D is used to shape the reward function to encourage the policy to imitate good trajectories.

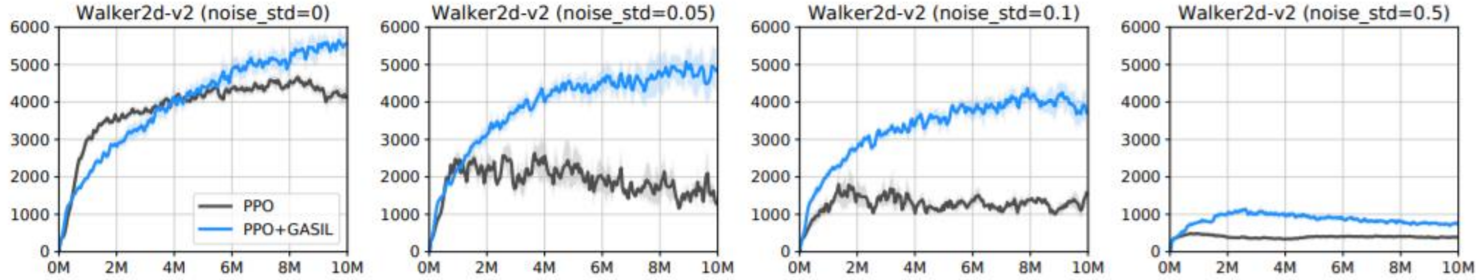
Generative Adversarial Self-Imitation learning



OpenAI Gym MuJoCo tasks



Delayed versions of OpenAI Gym MuJoCo tasks



Generative Adversarial Self-Imitation Learning, NeurIPS 2019

Multiagent Generative Adversarial Self-Imitation learning

- Independent Multiagent Learning Challenges
  - sparse and delayed rewards
  - Each agent only has local observations during both both learning and execution (Independent Training and Decentralized execution)
  - The environment is highly dynamic and non-stationary (exploration of others), and can easily converge to sub-optimal stable solutions (shadowed equilibrium)

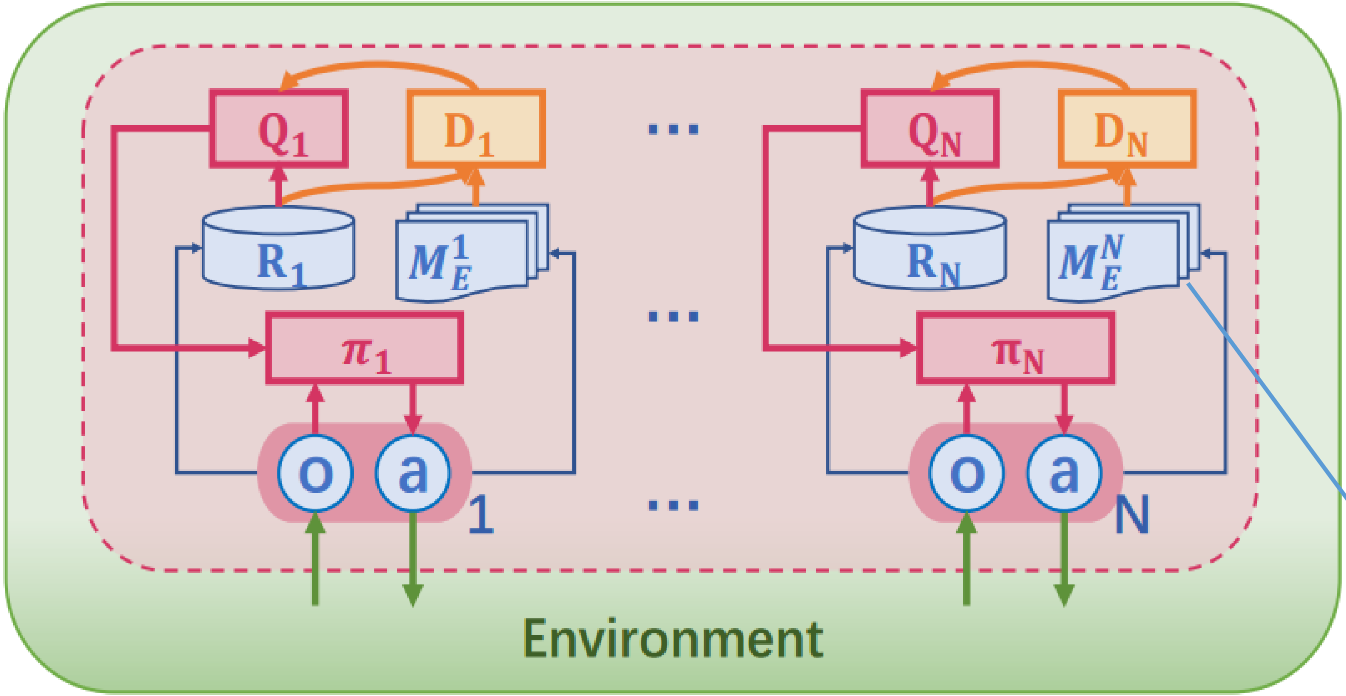
(a) Climbing game

		Agent 2		
		a	b	c
Agent 1	a	11	-30	0
	b	-30	7	6
	c	0	0	5

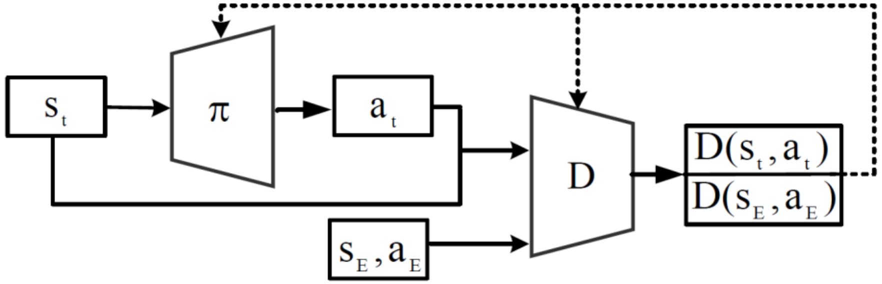
(b) Penalty game

		Agent 2		
		a	b	c
Agent 1	a	10	0	k
	b	0	2	0
	c	k	0	10

Multiagent Generative Adversarial Self-Imitation learning



Independent Generative Adversarial Self-Imitation Learning Framework



GAIL

M_E^i: Sub-Curriculum Experience Replay Buffer

Independent Generative Adversarial Self-Imitation Learning, AAMAS'19

Multiagent Generative Adversarial Self-Imitation learning

- **Sub-curriculum experience replay**
  - Help the independent agents to collect the past useful experiences/skills.
  - Good trajectory: maintain a good trajectory buffer with high-reward trajectories in the past (top-k; **difficult in practice**)
  - an agent's learning process: easier task → harder task

An Example:

[0, 0, 0, 0, … , -15]

[0, 5, 0, 0, … , -20]

Two trajectories with same return

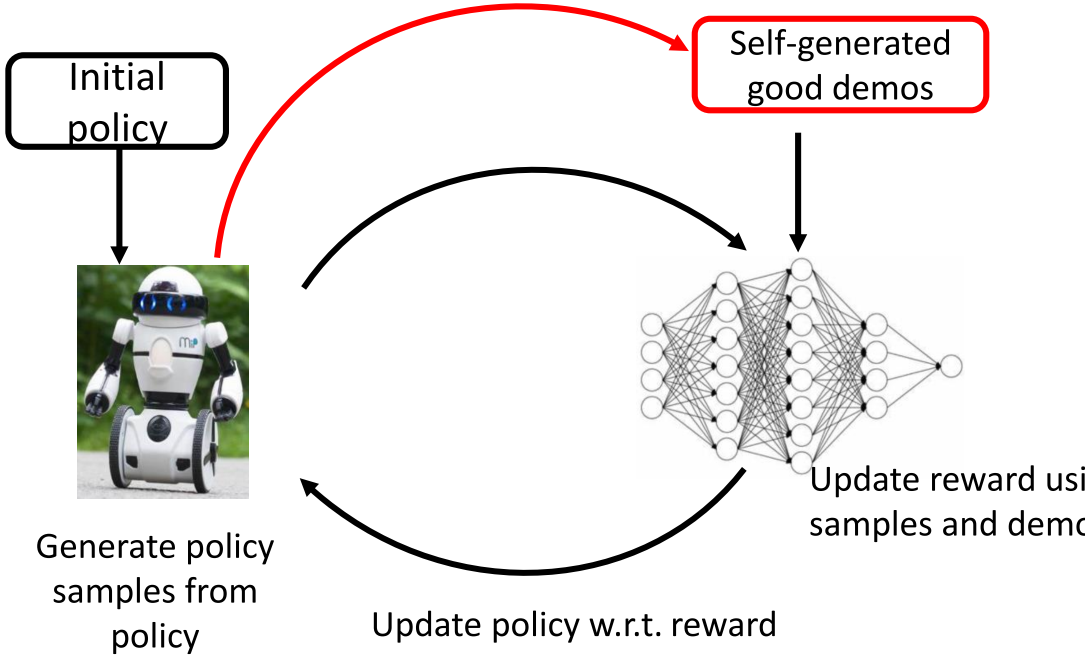
Kill one enemy → Kill all enemies

The sub-trajectory with a total reward +5 still demonstrates some useful behaviors.

Multiagent Generative Adversarial Self-Imitation learning

- **Sample Inefficiency in GAIL**
  - MuJoCo: GAIL requires ~200 expert frame transitions and millions of policy frame transitions sampled from the environment
  - For each agent i, instead of sampling trajectories from the current policy directly, we sample transitions from the replay buffer R_i collected while performing off-policy training:

$$\min_{\theta}\max_{w} \hat{\mathbb{E}}_{(s,a)\sim\pi_E^i}[log(D_{w_i}(s,a))] + \hat{\mathbb{E}}_{(s,a)\sim R_i}[log(1-D_{w_i}(s,a))] - \lambda_H H(\pi_\theta^i)$$



Multiagent Generative Adversarial Self-Imitation learning

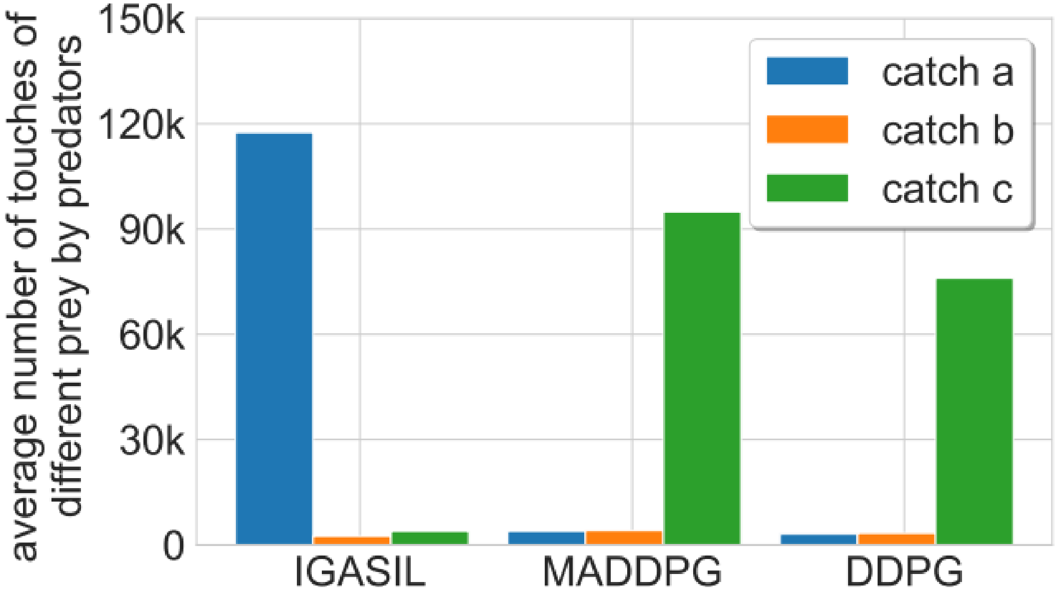
- **Imitation reward function design**
  - $\log(D(s,a))$: is always negative and provides a per step penalty which drives the agent to exit from the environment earlier.
  - $-\log(1-D(s,a))$: is always positive and potentially provides a survival bonus which drives the agent to survive longer in the environment to collect more rewards.

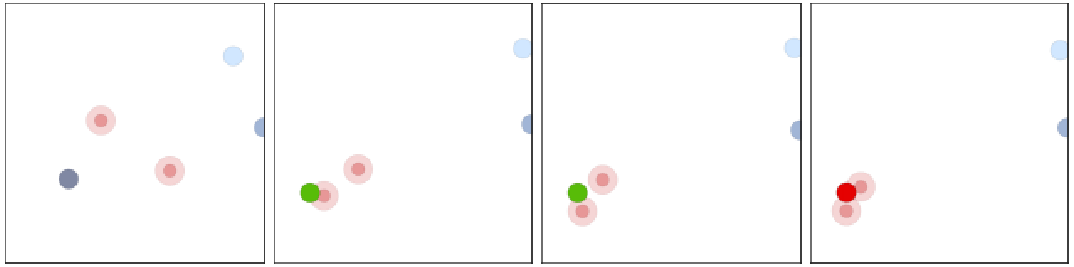
$$r_{imit}(s,a) = log(D(s,a)) - log(1 - D(s,a))$$
$$r'(s,a) = r + \lambda_{imit} * r_{imit}(s,a)$$

Env reward Imitation reward

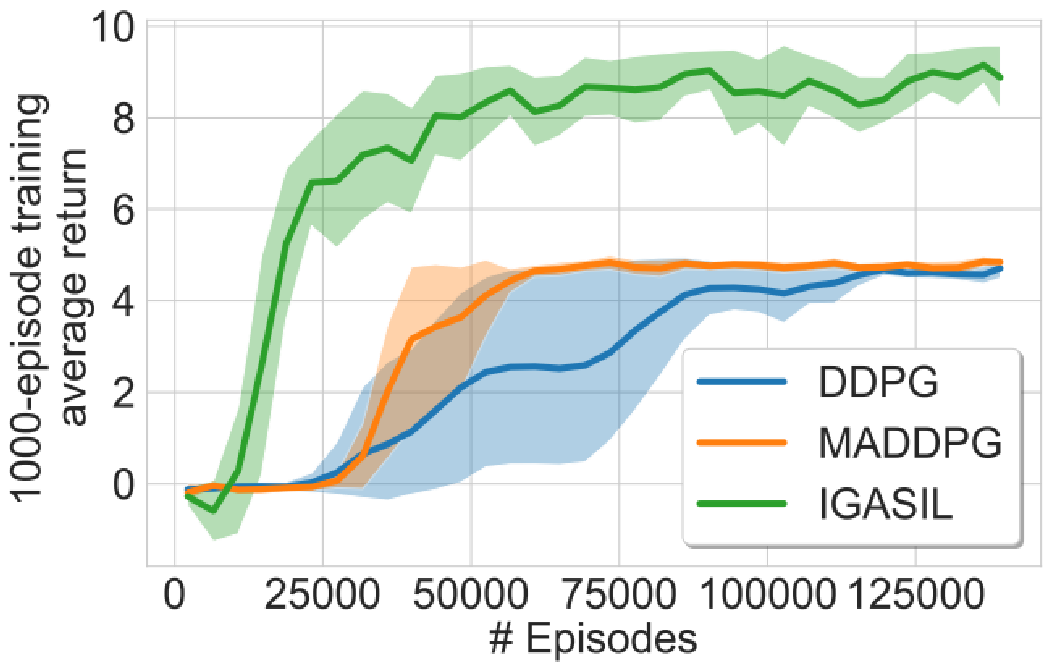
Multiagent Generative Adversarial Self-Imitation learning

- **Predator-prey game**





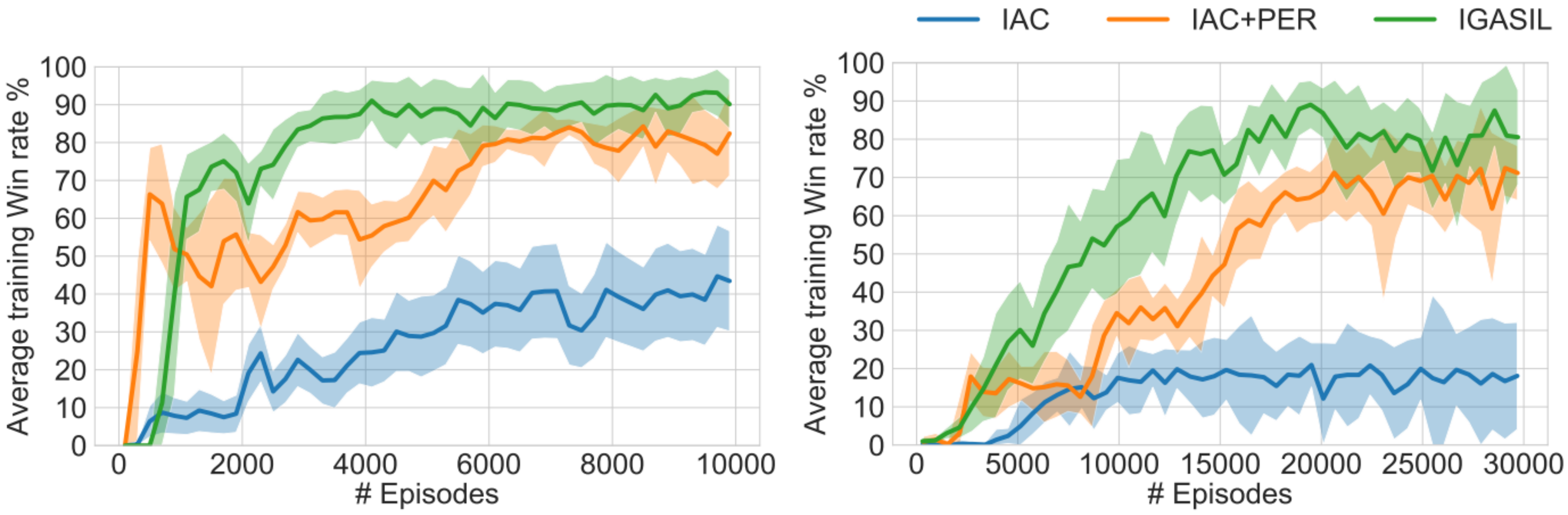
(a) (b) (c) (d)



Multiagent Generative Adversarial Self-Imitation learning

- StarCraft Game

Map	Heur.	IAC	IAC+PER	COMA	IGASIL
5 M	66	45	85	81	**96**
2d3z	63	23	76	47	**87**

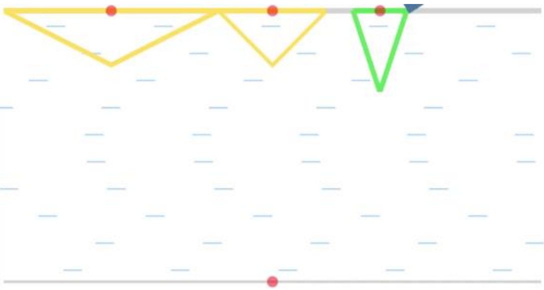


(a) 5m

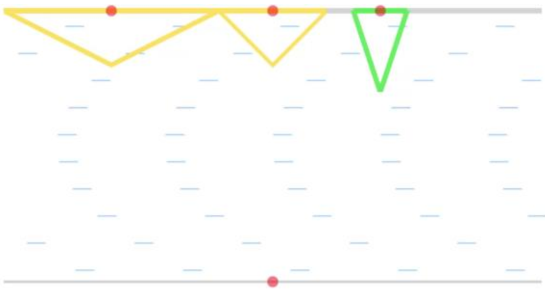
(b) 2d3z

Multiagent Generative Adversarial Self-Imitation learning

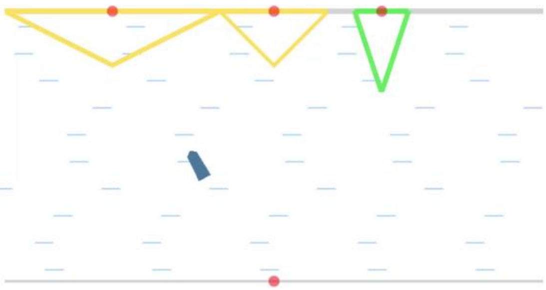
- Cooperative Rowing



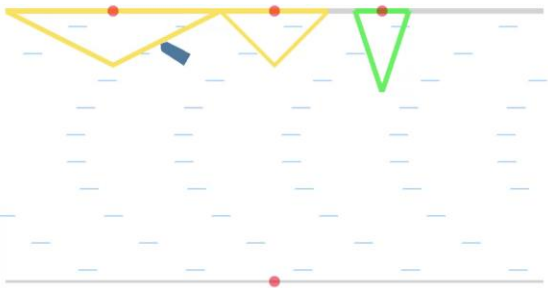
IGASIL



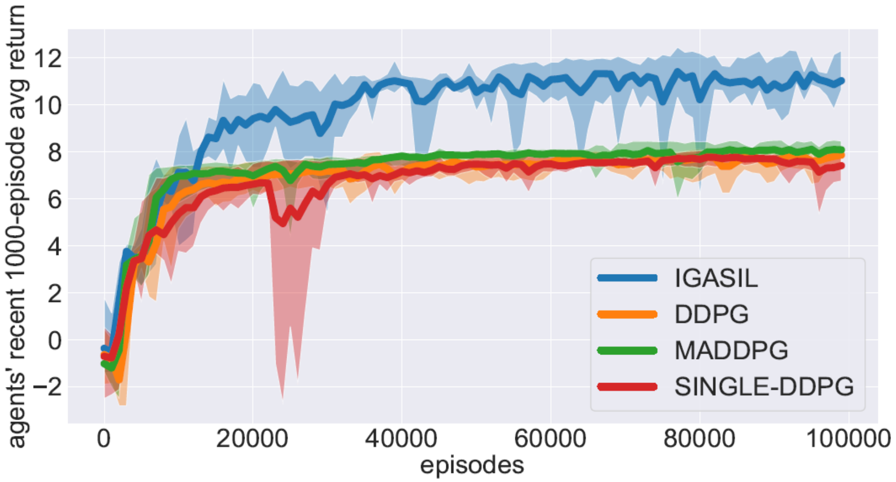
Single-DDPG



MADDPG

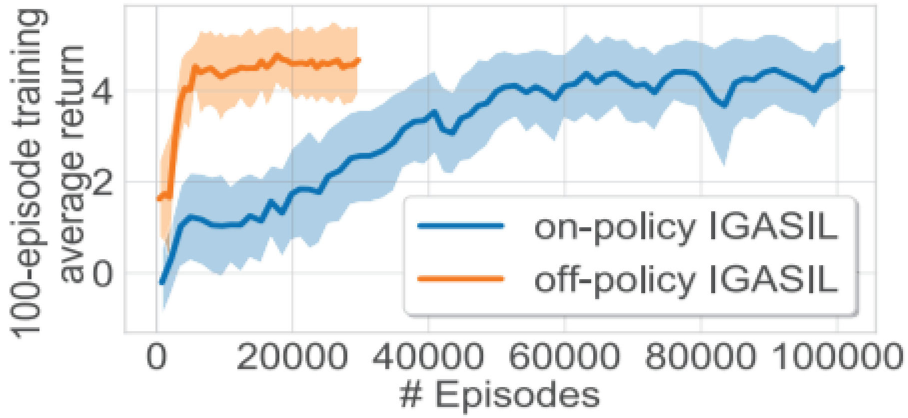


Independent DDPG

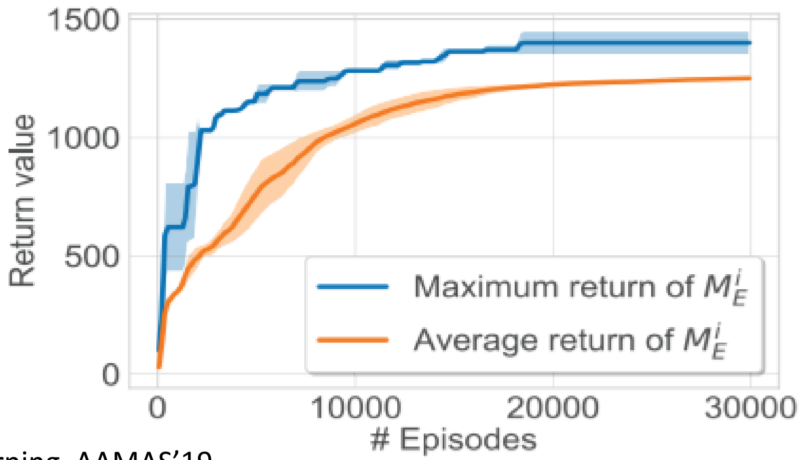


Multiagent Generative Adversarial Self-Imitation learning

- On-policy vs. off-policy



- Sub-trajectory replay buffer



Independent Generative Adversarial Self-Imitation Learning, AAMAS'19

Learning with Sparse Rewards

- From Sparse to Dense
  - Reward Learning/shaping (SGAIL/Multiagent SGAIL, exploration-oriented intrinsic rewards)
  - Temporal/spatial credit assignment (single-agent/multiagent settings)
  - Task hierarchical decomposition (hierarchical RL)

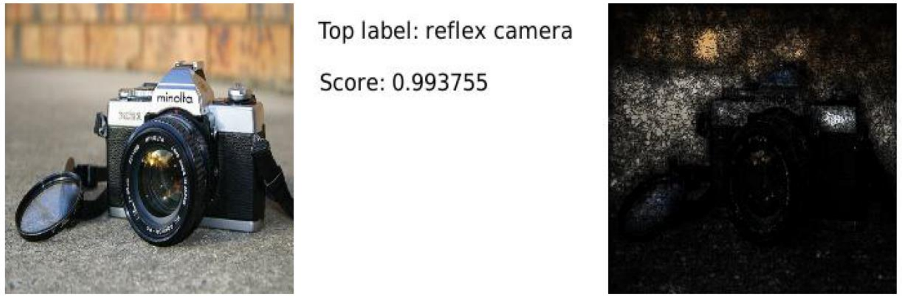
Credit Assignment

- Temporal credit assignment (single-agent settings)
  - Decompose the return of an episode backpropagated into earlier time steps
- Spatial credit assignment (multiagent settings)
  - Decompose the global reward into individual agents according to their contributions.

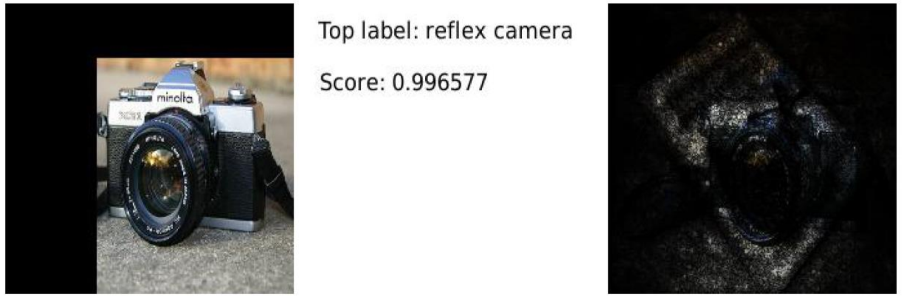
Attribution Method

- How to evaluate feature importance
  - Gradients do not reflect feature importance
  - $\forall i, j: \mathcal{P}^{L}_{i,j}(img) ::= \Sigma_{c \in \{R,G,B\}} |\nabla lncp^{L}_{i,j,c}(img)|$

    where $\nabla\ lncp^{L}_{i,j,c}(img)$ stands for the gradient of a specific pixel(i, j) and color channel c $\in \{R, G, B\}$



(a) Original image.



(b) Ablated image.

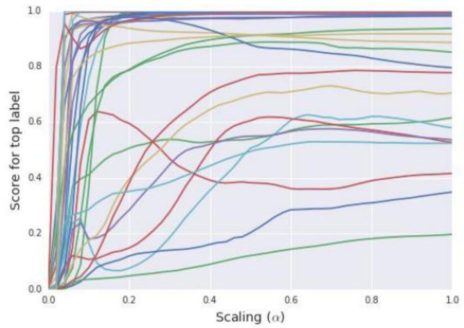
Attribution Method

- We create counterfactual images as follows:

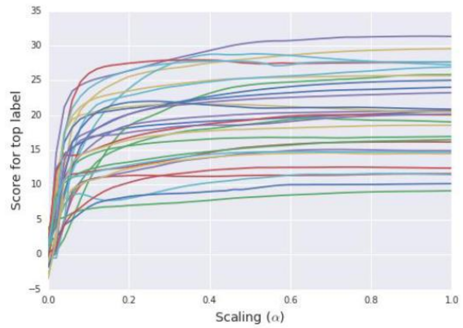
$$\alpha Img := \{\alpha \mid \text{img} \mid 0 \leq \alpha \leq 1\}$$

- We compute the interior gradients of those counterfactual images:

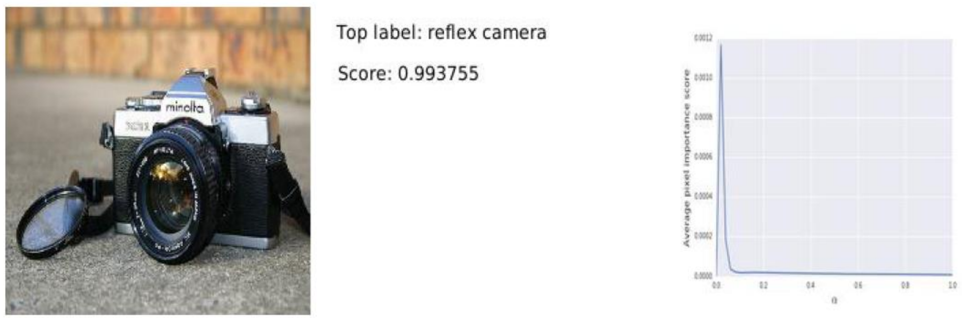
$$InteriorGrads\,(img) ::= \{\nabla ln\, cp(\alpha \mathrm{I}mg) \mid 0 \leq \alpha \leq 1\}$$



(a) Softmax score for top label



(b) Pre-softmax score for top label

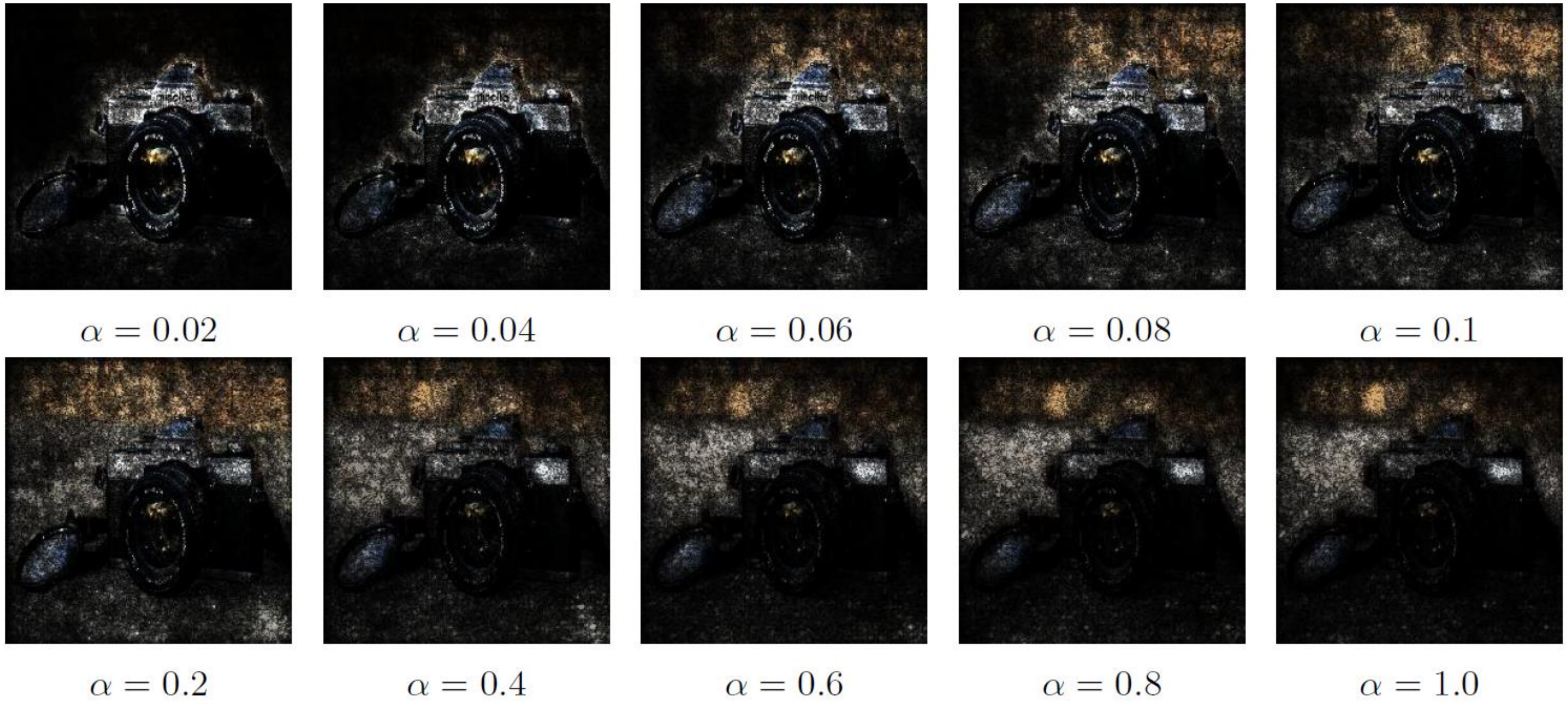


Input image and trend of the pixel importance scores obtained from interior gradients.

Attribution Method

- The interior gradients along the color dimension are aggregated:

$$InteriorPixelImportance\,(img) ::= \{P(\alpha img) \mid 0 \le \alpha \le 1\}$$



Integrated Gradients

- A smooth function specifying the set of counterfactuals

$$\tau = (\tau_1, \dots, \tau_n): [0,1] \to R^n$$

- The integrated gradient along the i^{th} dimension for input $x \in R^n$ is defined as follow.

$$c_j = PathIG_j^{\tau}(\vec{x})::= \int_{\alpha=0}^{1} \frac{\partial F(\tau(\alpha))}{\partial \tau_i(\alpha)} \frac{\partial \tau_i(\alpha)}{\partial \alpha} d\alpha$$

where $\frac{\partial F(x)}{\partial x_i}$ is the gradient of F along the i^{th} dimension at x.

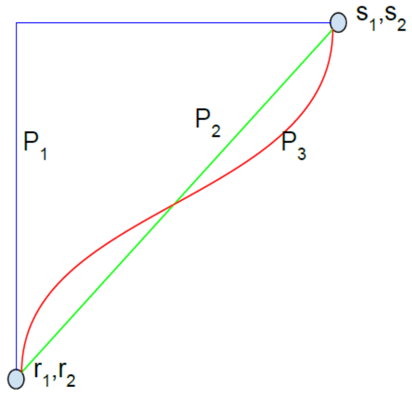
- Additivity property
  - If $F : R^n$ is differentiable almost everywhere, and $\tau : [0,1] \to R^n$ is smooth

$$\Sigma_{i=1}^{n} PathIG_i^{\tau}(\vec{x})::= F(\tau(1)) - F(\tau(0))$$

Integrated Gradients

- An attribution method to understand the influence of each input feature to the network output values

$$c_j = PathIG_j^{\tau}(\vec{x}) ::= \int_{\alpha=0}^{1} \frac{\partial F(\tau(\alpha))}{\partial \tau_i(\alpha)} \frac{\partial \tau_i(\alpha)}{\partial \alpha} d\alpha$$



where $\frac{\partial F(x)}{\partial x_i}$ is the gradient of F along the i^{th} dimension at x.

- In CV or NLP areas, zero embedding vector is usually used as baseline and a straight line is used as the path

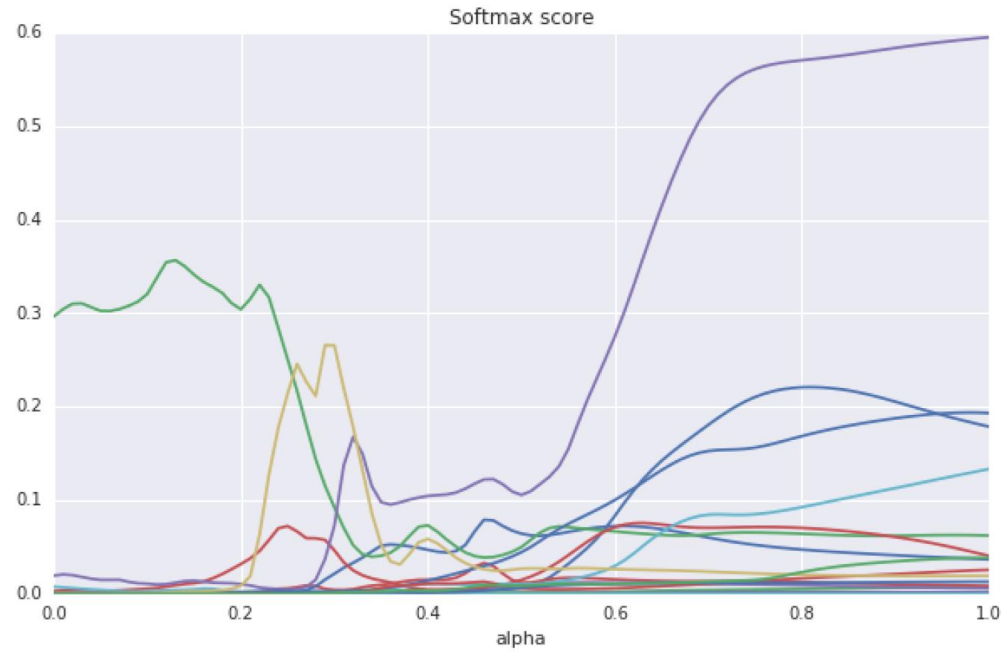
$$c_j = IG_j^{\tau}(\vec{x}) ::= (\vec{x}_j - \vec{b}_j) \int_{\alpha=0}^{1} \frac{\partial F(\tau(\alpha))}{\partial \tau_j(\alpha)} d\alpha \; ; \quad \tau(\alpha) = \vec{b} + \alpha(\vec{x} - \vec{b})$$

Integrated Gradients

- Apply integrated gradients to sequential models
  - NLP example: given a sequence of input words, and the softmax prediction for the next word, we want to identify the importance of the preceding words for the score.

  - Saturation phenomenon also exists in LSTM models

  - Can be easily applied to temporal reward assignment in single-agent RL



Saturation phenomenon

Multiagent Credit Assignment

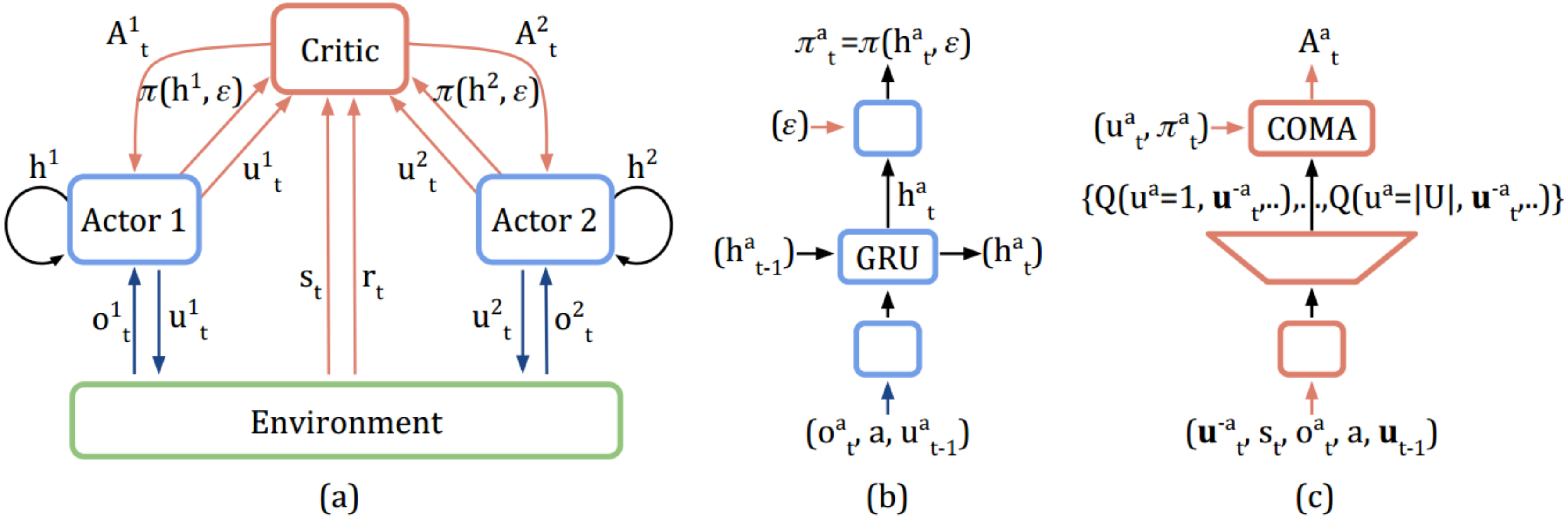
- Multiagent counterfactual baseline
  - Difference rewards： $D^a = r(s,\mathbf{u}) - r(s,(\mathbf{u}^{-a},c^a))$
  - typically require access to a simulator in order to estimate $r(s,(\mathbf{u}^{-a},c^a))$
  - can use functional approximators to estimate instead

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

Counterfactual Multi-Agent Policy Gradients, Jakob et al., AAAI 2018

Multiagent Credit Assignment

Centralized training and decentralized execution

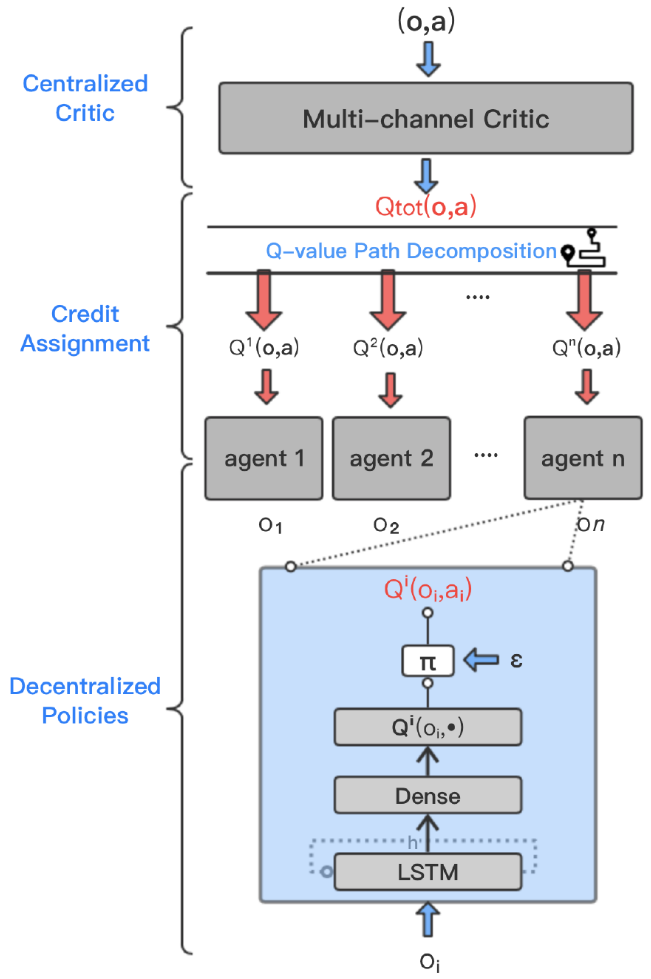


Difference rewards idea naturally fits DRL framework, but performs poorly in complex multiagent learning scenarios!

Counterfactual Multi-Agent Policy Gradients, Jakob et al., AAAI 2018

QPD: Multiagent Q-value Decomposition

- **top block**: the centralized critic with a multi-channel modular design
- **middle block**: applying the Q-value path decomposition technique to achieve credit assignments on the agent level
- **bottom block**: the individual-agent network architecture implemented by the recurrent deep Q-network.

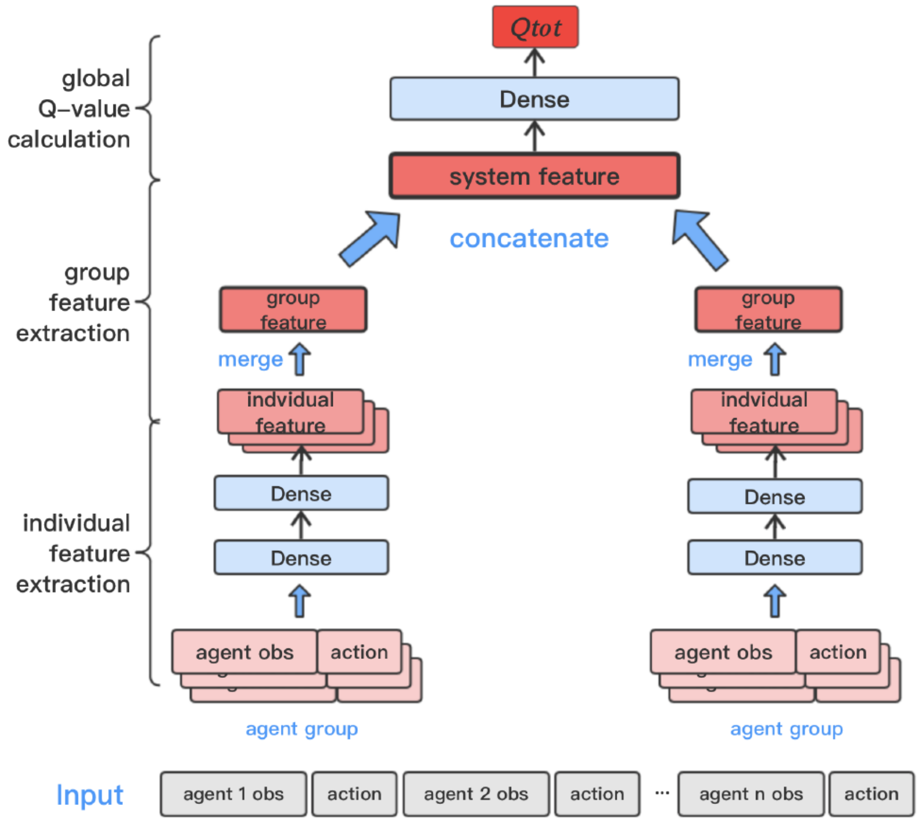


Overall framework of QPD

Q-value Path Decomposition for Deep Multiagent Reinforcement Learning, ICML 2020

QPD: Multiagent Q-value Decomposition

- Agents can be organized into different groups considering the heterogeneous features of the system (agents in the same group are homogeneous).
- Multiple channel within each group to capture different aspect of features
- parameter sharing for homogeneous agents in the same group



Multi-channel global critic framework

Q-value Path Decomposition for Deep Multiagent Reinforcement Learning, ICML 2020

QPD: Multiagent Q-value Decomposition

- The global Q-value can be decomposed into the following

$$Q_{tot}(\vec{o}_t, \vec{a}_t) = \sum_{x_j \in \mathbb{X}_1} PathIG_j^{\tau_i^T}(\vec{o}_t, \vec{a}_t) + \cdots + \sum_{x_j \in \mathbb{X}_1} PathIG_j^{\tau_i^T}(\vec{o}_t, \vec{a}_t)$$

- And each component can be attributed as individual Q for each agent

$$Q^i(\vec{o}_t, \vec{a}_t) \approx \sum_{x_j \in \mathbb{X}_i} PathIG_j^{\tau_i^T}(\vec{o}_t, \vec{a}_t)$$

QPD: Multiagent Q-value Decomposition

- **Additivity Proposition 1**. *If* $F: R^d \rightarrow R$ *is differentiable almost everywhere*

$$\sum_{j=1}^{|\vec{x}|} IG_j^{\tau}(\vec{x}) = F(\vec{x}) - F(\vec{b})$$

- **Theorem 1**. *Let* τ_t^T *represents the joint observation and action trajectory from t to the termination step T, then*

$$Q_{tot}\left(\vec{o}_t, \vec{a}_t\right) = \sum_{i=1}^{n} \sum_{x_j \in \mathbb{X}_i} PathIG_j^{\tau_t^T}(\vec{o}, \vec{a})$$

Proof. *Let* $\overrightarrow{x_t}$ *represents the feature vector* $(\vec{o}_t, \vec{a}_t)$ concisely. τ_t^T is composed of $(\tau_t^{t+1}, \tau_{t+1}^{t+2}, \dots, \tau_{T-1}^{T})$,
where τ_t^{t+1} is the straight line path from $(\vec{o}_t, \vec{a}_t)$ to $(\vec{o}_{t+1}, \vec{a}_{t+1})$

$$Q_{tot}\left(\vec{o}_t, \vec{a}_t\right) = Q_{tot}\left(\vec{x}_t\right) = Q_{tot}\left(\vec{x}_t\right) - Q_{tot}\left(\vec{x}_T\right)$$
$$= Q_{tot}\left(\vec{x}_t\right) - Q_{tot}\left(\vec{x}_{t+1}\right) + Q_{tot}\left(\vec{x}_{t+1}\right) - Q_{tot}\left(\vec{x}_{t+2}\right) + \cdots + Q_{tot}(\vec{x}_{T-1}) - Q_{tot}(\vec{x}_T)$$
$$= \sum_{j=1}^{|\vec{x}_t|} IG_j^{\tau_t^{t+1}}(\vec{x}) + \sum_{j=1}^{|\vec{x}_t|} IG_j^{\tau_{t+1}^{t+2}}(\vec{x}) + \cdots + \sum_{j=1}^{|\vec{x}_t|} IG_j^{\tau_{T-1}^{T}(\vec{x})}$$
$$= PathIG_{j=1}^{\tau_t^T}\left(\vec{x}\right) + PathIG_{j=2}^{\tau_t^T}\left(\vec{x}\right) + \cdots + PathIG_{j=|\vec{x}_t|}^{\tau_t^T}(\vec{x})$$
$$= \sum_{x_j \in X_1} PathIG_j^{\tau_t^T}(\vec{x}) + \sum_{x_j \in X_2} PathIG_j^{\tau_t^T}(\vec{x}) + \cdots + \sum_{x_j \in X_n} PathIG_j^{\tau_t^T}(\vec{x})$$
$$= \sum_{i=1}^{n} \sum_{x_j \in X_i} PathIG_j^{\tau_t^T}(\vec{x}) = \sum_{i=1}^{n} \sum_{x_j \in X_i} PathIG_j^{\tau_t^T}(\vec{o}, \vec{a})$$

Q-value Path Decomposition for Deep Multiagent Reinforcement Learning, ICML 2020

QPD: Multiagent Q-value Decomposition

Table 1. Median and mean performance of the test win percentage.

Map	IQL		COMA		QMIX		QTRAN		QPD	
	$\widetilde{m}$	$\overline{m}$	$\widetilde{m}$	$\overline{m}$	$\widetilde{m}$	$\overline{m}$	$\widetilde{m}$	$\overline{m}$	$\widetilde{m}$	$\overline{m}$
3m	**100**	97	91	92	**100**	99	**100**	**100**	95	92
8m	91	90	95	94	**100**	96	**100**	**97**	94	93
2s3z	39	42	66	64	**100**	**97**	77	80	95	94
3s5z	0	3	0	0	16	25	0	4	**85**	**81**
1c3s5z	7	8	30	30	89	89	31	33	**92**	**92**
3s5z_vs_3s6z	0	0	0	0	0	0	0	0	**8**	**10**

Q-value Path Decomposition for Deep Multiagent Reinforcement Learning, ICML 2020

Learning with Sparse Rewards

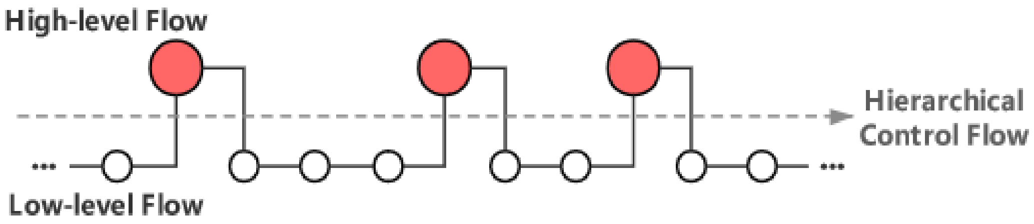
- From Sparse to Dense
  - Reward Learning/Shaping (SGAIL/Multiagent SGAIL, exploration-oriented intrinsic rewards)
  - Temporal/spatial credit assignment (single-agent/multiagent settings)
  - **Task hierarchical decomposition** (hierarchical RL)

Temporal Abstracted Hierarchical Execution

- Basic model—Two levels of hierarchy：
  - High level:
    - High level policy: $g_t \sim \pi^{\mathrm{hi}}(s_t)$
    - The high level policy receives state s_t then chooses an abstracted action $g_t \in G$, where G denotes the set of all possible current abstracted actions (e.g., skills/sub-policies/options/goals).
    - The high level aims to maximize the rewards from environment directly, i.e., extrinsic rewards.
  - Low level:
    - Low level policy: $a_t \sim \pi^{\mathrm{lo}}(s_t, g_t)$
    - The low level policy receives state s_t and g_t then takes a primitive action a_t, while results in a new state s_{t+1}.
    - The low level is expected to accomplish subtasks or achieve goals from high level.
  - Both high level and low level can use RL algorithms to realize (e.g. DQN, PPO, DDPG, TD3)
  - The hierarchy can be *deeper* (i.e., more than 2 levels)

$g_t \sim \pi^{\mathrm{hi}}(s_t)$

$a_t \sim \pi^{\mathrm{lo}}(s_t, g_t)$



Low-level Policy Acquisition in HRL

- Learning from Intrinsic Reward:
  - Intrinsic reward are designed to guide the low-level policy to accomplish specific subtasks or achieve the give goals
  - Common designs:
    - **Termination predicates**: rewards are obtained only when success
      - 1 for accomplishment and 0 for otherwise
    - **Goal-distance intrinsic reward**: penalize the low-level policy according to the distance to the given goal (in continuous space)
      - $\mathrm{r}(s_t, g_t, a_t, s_{t+1}) = -||s_t + g_t - s_{t+1}||_2$

- Skill/Option discovery:
  - Discover effective and diverse skills (i.e., sub-policies/options) in unsupervised learning manner (Campos et al. 2020)

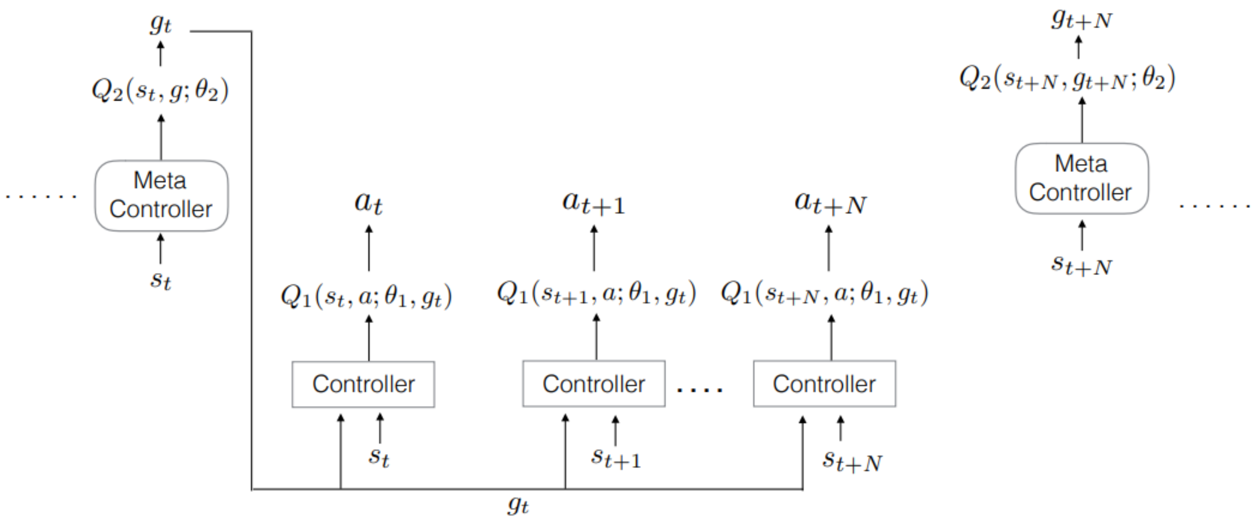
- Others
  - E.g., reuse from other tasks (transfer), manually design

Hierarchical Deep Reinforcement Learning

- **HDRL Categories:**
  - Discrete temporal abstraction/Option architecture
    - Learn/design discrete abstracted actions (e.g., skills/sub-policies/options) from lower-level actions (e.g., primitive actions)
    - Learn higher-level controllers that manipulate among abstracted actions (e.g., skills, options)
  - Continuous goal-oriented architecture
    - Design/learn continuous goal (representation) space that represents target states (in latent space) for lower-level policies
    - Learn higher-level policies among goal (representation) space that direct lower-level execution
  - Hierarchical MARL
    - Incorporate temporal abstraction in MARL
    - Learn to cooperate through hierarchical policies of multiple agents

H-DQN

- Motivation:
  - Solve tasks with sparse and delayed feedback from complex environments
- Key ideas:
  - Temporal abstraction
    - Two levels of DQN controllers executed at different time scales
  - Intrinsic motivation
    - Termination predicates are used as intrinsic reward function for low-level learning

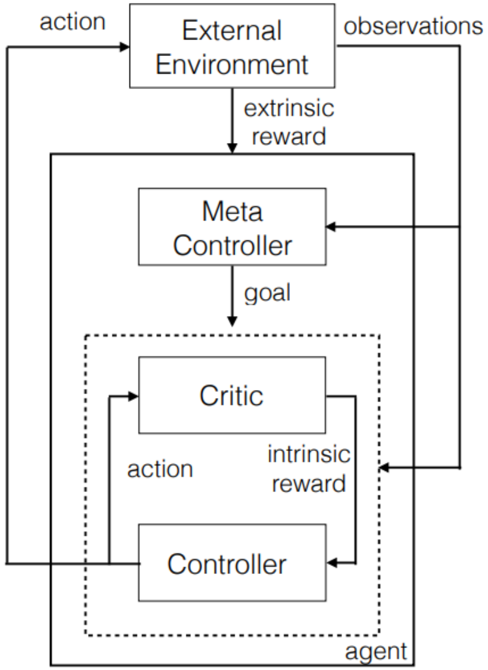


Kulkarni, et al., Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation. NeurIPS 2016

H-DQN

- Meta-Controller Learning (high level):
  - $Q_2^*(s,g) = \max_{\pi_g} \mathrm{E}[\sum_{t'=t}^{t+N} f_{t'} + \gamma \max_{g'} Q_2^*(s_{t+N}, g') \mid s_t = s, g_t = g, \pi_g]$
  - $\nabla_{\theta_{2,i}} L_2(\theta_{2,i}) = \mathrm{E}_{(s_t, g_t, f_t, s_{t+N} \sim D_2)}\left[(\sum_{t'=t}^{t+N} f_{t'} + \gamma \max_{g'} Q_2(s_{t+N}, g'; \theta_{2,i-1}) - Q_2(s_t, g_t; \theta_{2,i})) \nabla_{\theta_{2,i}} Q_2(s_t, g_t; \theta_{2,i}) \right]$

- Controller Learning (low level):
  - $Q_1^*(s,a;g) = \max_{\pi_{ag}} \mathrm{E}[\sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t'} \mid s_t = s, a_t = a, g_t = g, \pi_{ag}]$
    $= \max_{\pi_{ag}} \mathrm{E}[r_t + \gamma \max_{a_{t+1}} Q_1^*(s_{t+1}, a_{t+1}; g) \mid s_t = s, a_t = a, g_t = g, \pi_{ag}]$
  - $\nabla_{\theta_{1,i}} L_1(\theta_{1,i}) = \mathrm{E}_{(s,a,r,s' \sim D_1)}[(r + \gamma \max_{a'} Q_1(s', a'; \theta_{1,i-1}, g) - Q_1(s, a; \theta_{1,i}, g)) \nabla_{\theta_{1,i}} Q_1(s, a; \theta_{1,i}, g)]$

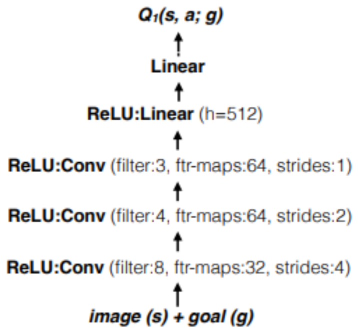


Kulkarni, et al., Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation. NeurIPS 2016

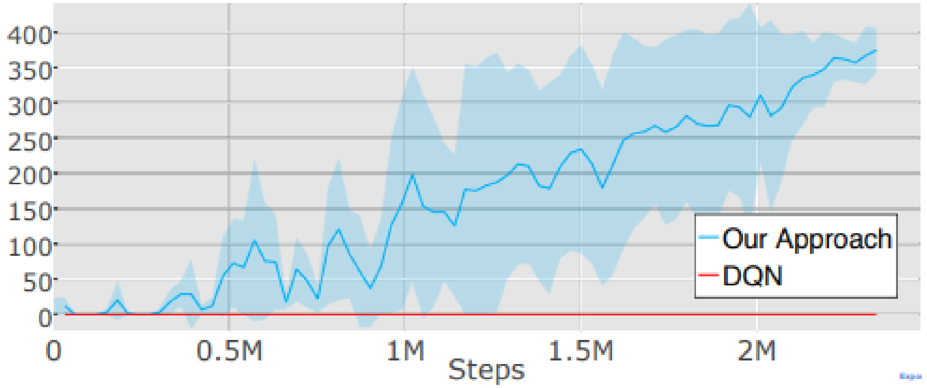
D-DQN

- Montezuma's Revenge

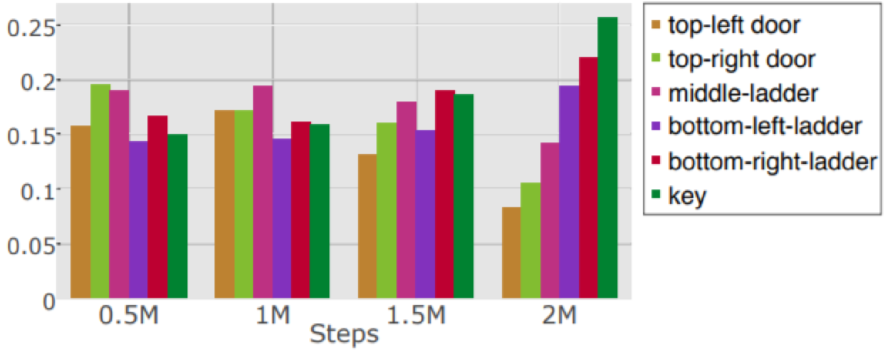
(a)



(b)



(a) Total extrinsic reward

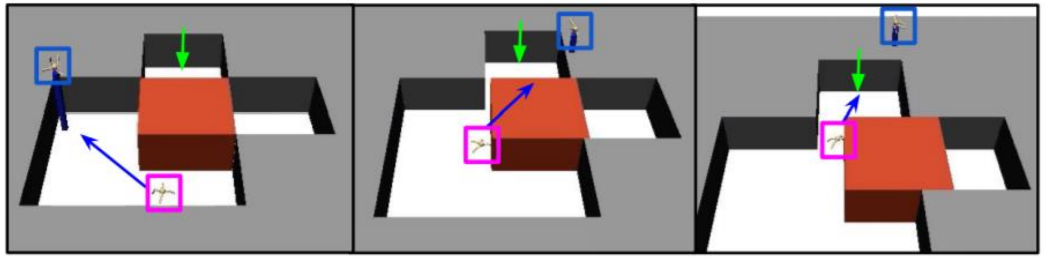


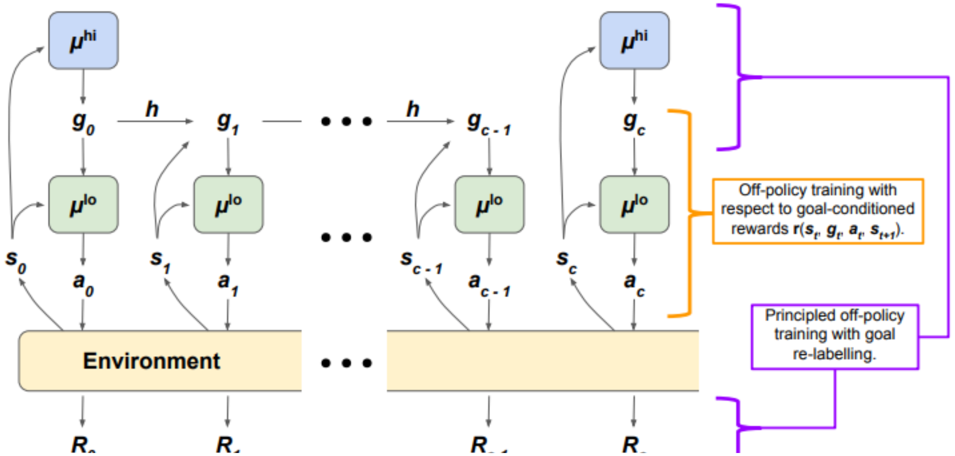
Success % of different goals over time

Kulkarni, et al., Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation. NeurIPS 2016

HIRO

- Motivation:
  - Reduce the dependence on careful task-specific design to improve generality and scalability
  - Use off-policy learning for both higher and lower-level training for higher sample efficiency
- Structure:
  - A high-level policy as a goal sender among pre-defined continuous goal space
  - A goal-conditioned low-level policy as goal reacher





1. Collect experience $s_t, g_t, a_t, R_t, \ldots$.
2. Train μ^{lo} with experience transitions $(s_t, g_t, a_t, r_t, s_{t+1}, g_{t+1})$ using g_t as additional state observation and reward given by goal-conditioned function $r_t = r(s_t, g_t, a_t, s_{t+1}) = -||s_t + g_t - s_{t+1}||_2$.
3. Train μ^{hi} on temporally-extended experience $(s_t, \tilde{g}_t, \sum R_{t:t+c-1}, s_{t+c})$, where $\tilde{g}_t$ is re-labelled high-level action to maximize probability of past low-level actions $a_{t:t+c-1}$.
4. Repeat.

Data-Efficient Hierarchical Reinforcement Learning. NeurIPS 2018

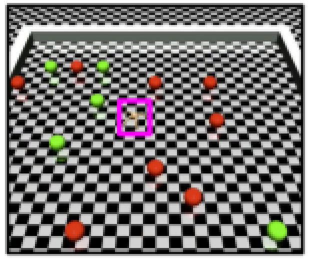
HIRO - Method

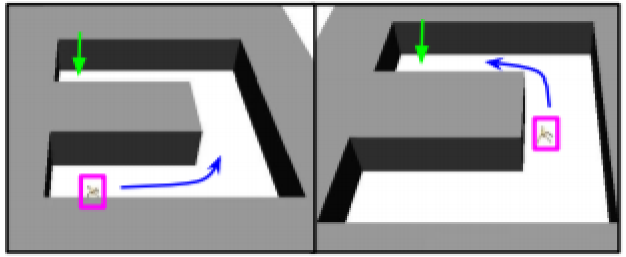
- High-level Off-policy Correction:
  - Instability issues: the changing behavior of the lower-level policy creates a non-stationary problem for the higher-level policy in the hierarchy: **the transition function of higher-level changes as the lower-level policy updates**
  - The instability issue makes it difficult to jointly learn multiple levels of policies
  - The high-level action g_t which in the past induced a low-level behavior may be re-labeled to a goal $\widetilde{g}_t$ which is mostly likely to induce the same low-level behavior with the current instantiation of the lower-level policy.

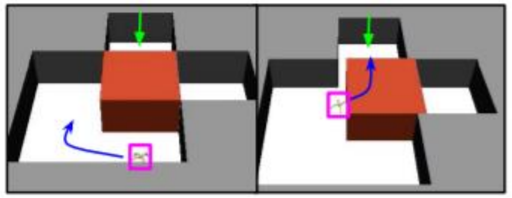
$$log\ \pi^{\mathrm{lo}\prime}(a_{t:t+c-1}|s_{t:t+c-1}, \widetilde{g_{t:t+c-1}}) \propto -\frac{1}{2}\sum_{i=t}^{t+c-1} ||\, a_i - \pi^{\mathrm{lo}\prime}(s_i, \widetilde{g}_i)||_2^2 + const$$

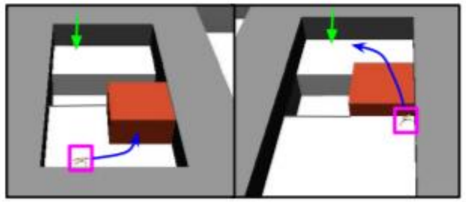
HIRO

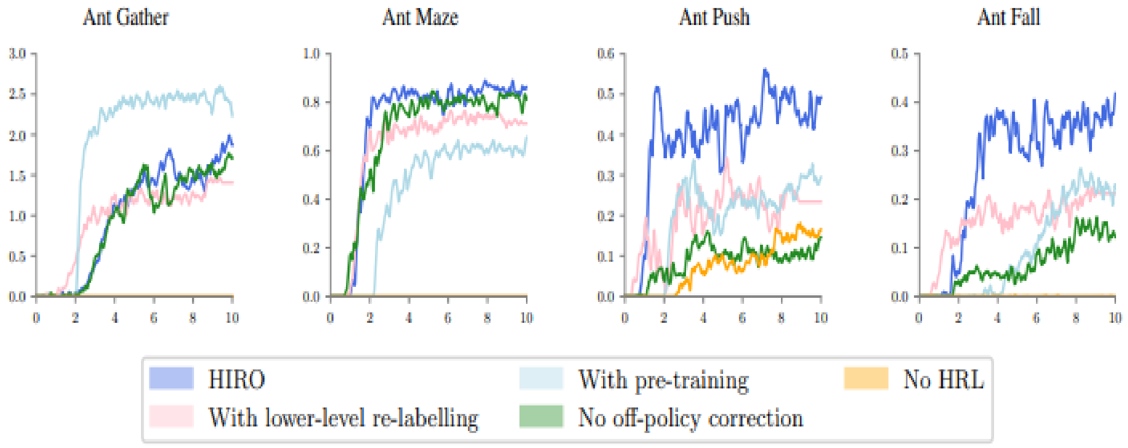
- Empirical Insights
  - Pre-training is beneficial for simpler and homogeneous tasks (ant gather), but is harmful for complex tasks since these tasks requires specialization in different low-level behaviors for different stages of the navigation.
  - Off-policy correction is significant for harder tasks as well.


Ant Gather


Ant Maze


Ant Push

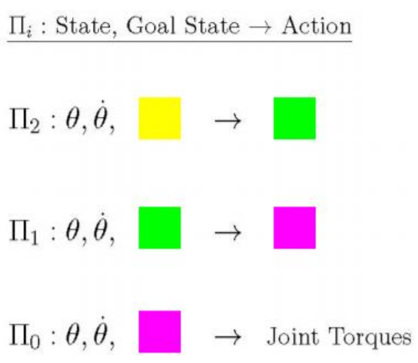

Ant Fall

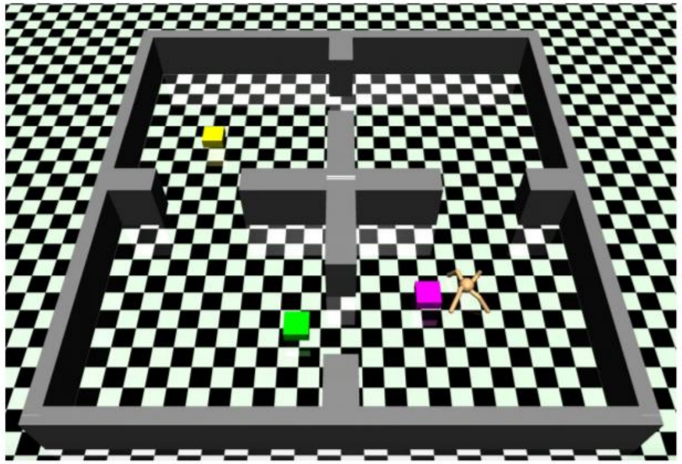


Nachum et al., Data-Efficient Hierarchical Reinforcement Learning. NeurIPS 2018

Hindsight AC

- Motivation:
  - Overcome the instability issues of jointly **learning multiple levels (>2) of hierarchical policies**
- Structure:
  - The controller input at the **top level** issues a action (i.e., goal for lower level) to achieve the **environment goal**
  - The **middle level** then chooses actions (i.e., middle level goals) to reach **the goal issued by higher level**
  - Finally, the **bottom level** chooses a series of actions **(i.e., primitive actions)** according to middle level goals

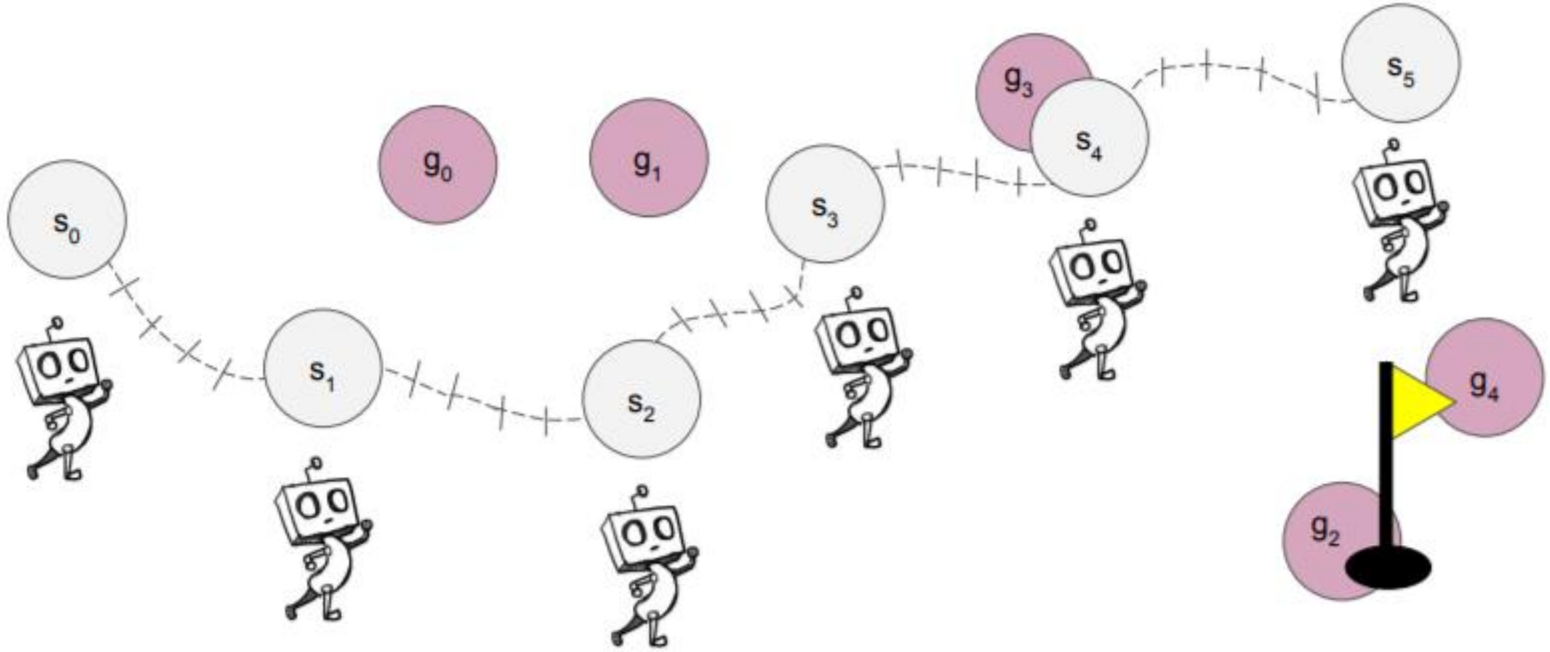




Andrew Levy et al., Learning Multi-Level Hierarchies with Hindsight. ICLR 2019

Hindsight AC

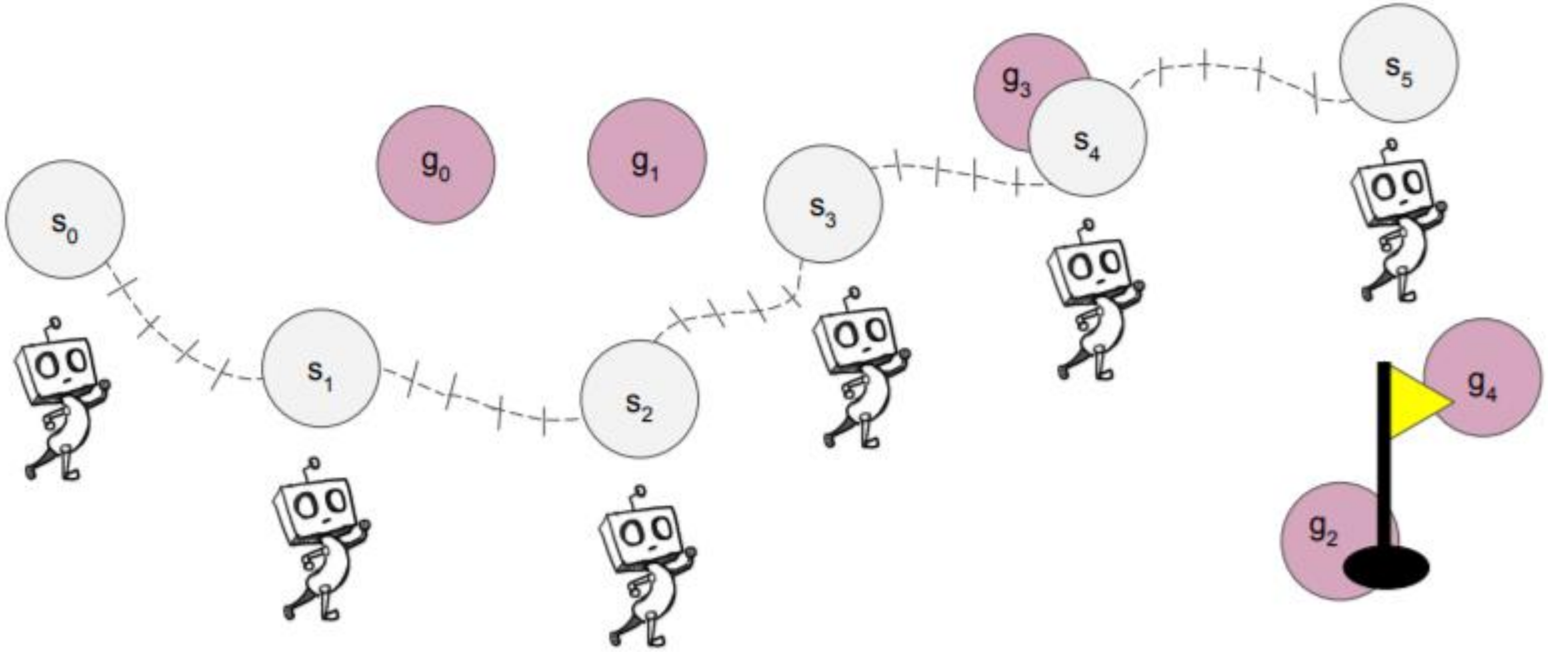
- Hindsight action transition
  - the high level of the robot would receive the hindsight action transition [initial state = s0, action = s1, reward = -1, next state = s1, goal = yellow flag, discount rate = γ]



Andrew Levy et al., Learning Multi-Level Hierarchies with Hindsight. ICLR 2019

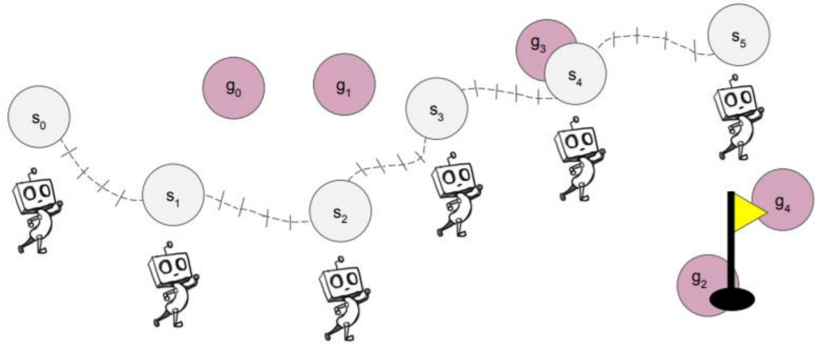
Hindsight AC

- Hindsight action transition
  - the hindsight action transition created for the second action by π_1 would be [initial state = s1, action = s2, reward = -1, next state = s2, goal = yellow flag, discount rate = γ]



Andrew Levy et al., Learning Multi-Level Hierarchies with Hindsight. ICLR 2019

Hindsight AC



- Hindsight goal transition
  - Guarantee that after every sequence of actions by each level in the hierarchy, that level receives a transition containing the sparse reward.
  - An Example (H = 5 primitive actions)
  - H = 0: [initial state = s0, action = joint torques, reward = -1, next state = first tick mark, goal = g0, discount rate = γ]
  - H' = 0: [initial state = s0, action = joint torques, reward = TBD, next state = first tick mark, goal = TBD, discount rate = γ]

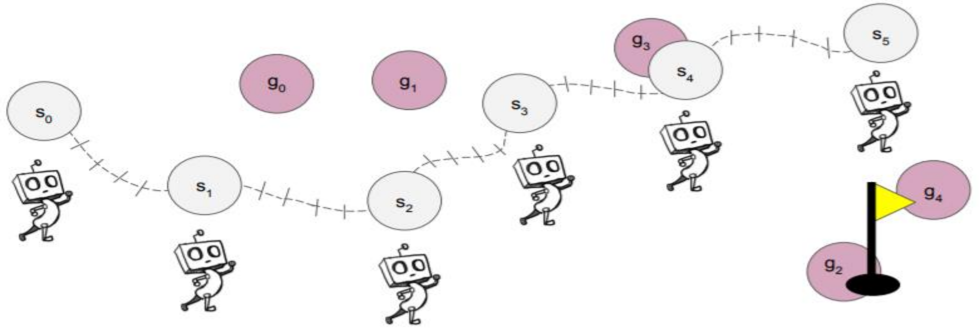
    ……
  - H = 5: [initial state = 4th tick mark, action = joint torques, reward = -1, next state = s1, goal = g0, discount rate = 0]
  - H' = 5: [initial state = 4th tick mark, action = joint torques, reward = TBD, next state = s1, goal = TBD, discount rate = 0]

  - After applying hindsight goal transition: H = 5: [initial state = 4th tick mark, action = joint torques, reward = 0, next state = s1, goal = s1, discount rate = 0]

Andrew Levy et al., Learning Multi-Level Hierarchies with Hindsight. ICLR 2019

Hindsight AC

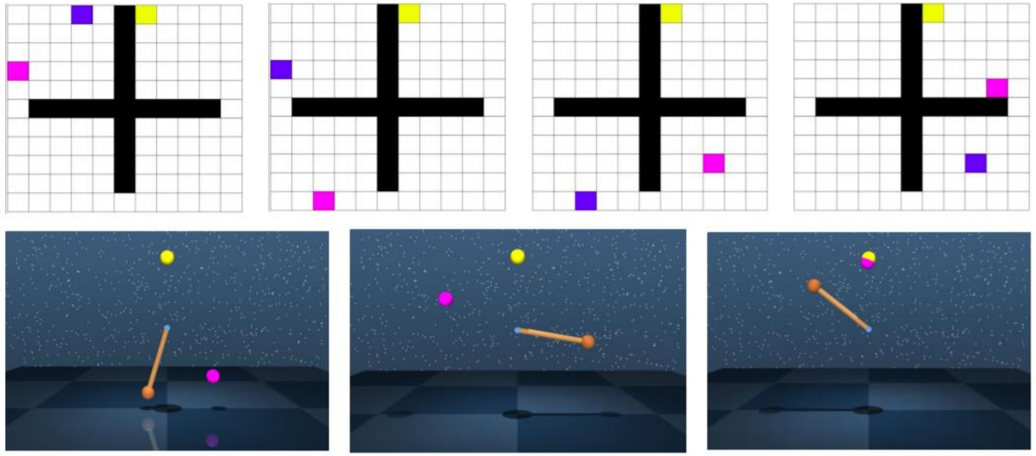
- Hindsight goal transition
  - For high level: based on copies of hindsight action transitions.
  - [initial state = s0, action = s1, reward = -1, next state = s1, goal = yellow flag, discount rate = γ] → [initial state = s0, action = s1, reward = -1, next state = s1, goal = s5, discount rate = γ]
  - [initial state = s4, action = s5, reward = -1, next state = s5, goal = yellow flag, discount rate = γ] →[initial state = s4, action = s5, reward = 0, next state = s5, goal = s5, discount rate = 0]



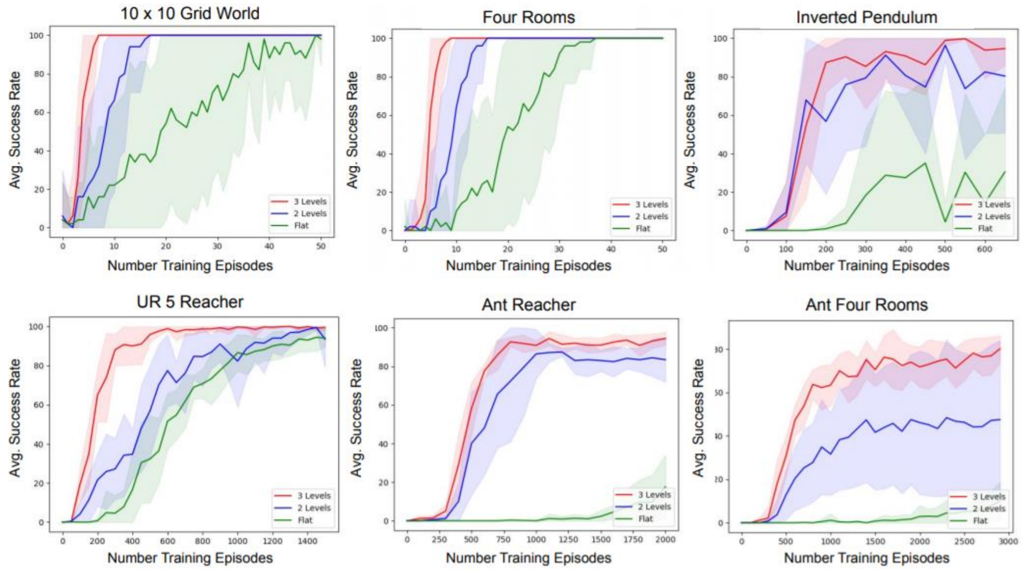
Andrew Levy et al., Learning Multi-Level Hierarchies with Hindsight. ICLR 2019

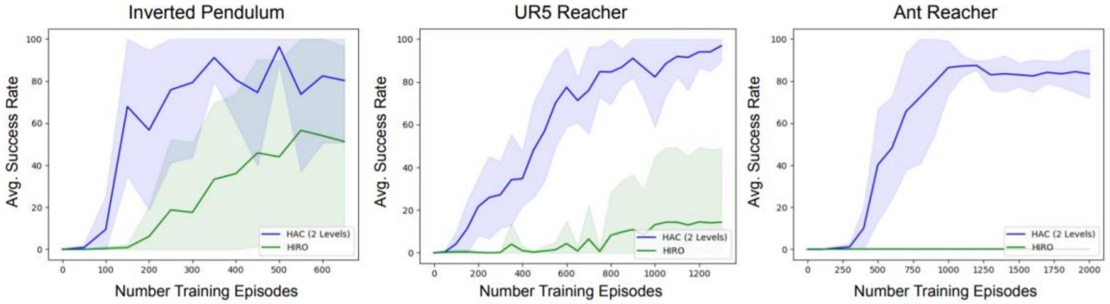
Hindsight AC

- Empirical Insights
  - Benefit from learning multiple hierarchy in parallel and increase levels can boost learning performance
  - HAC performs better than HIRO due to the use of different off-policy correction method (delay in waiting for the lower level policy to learn)



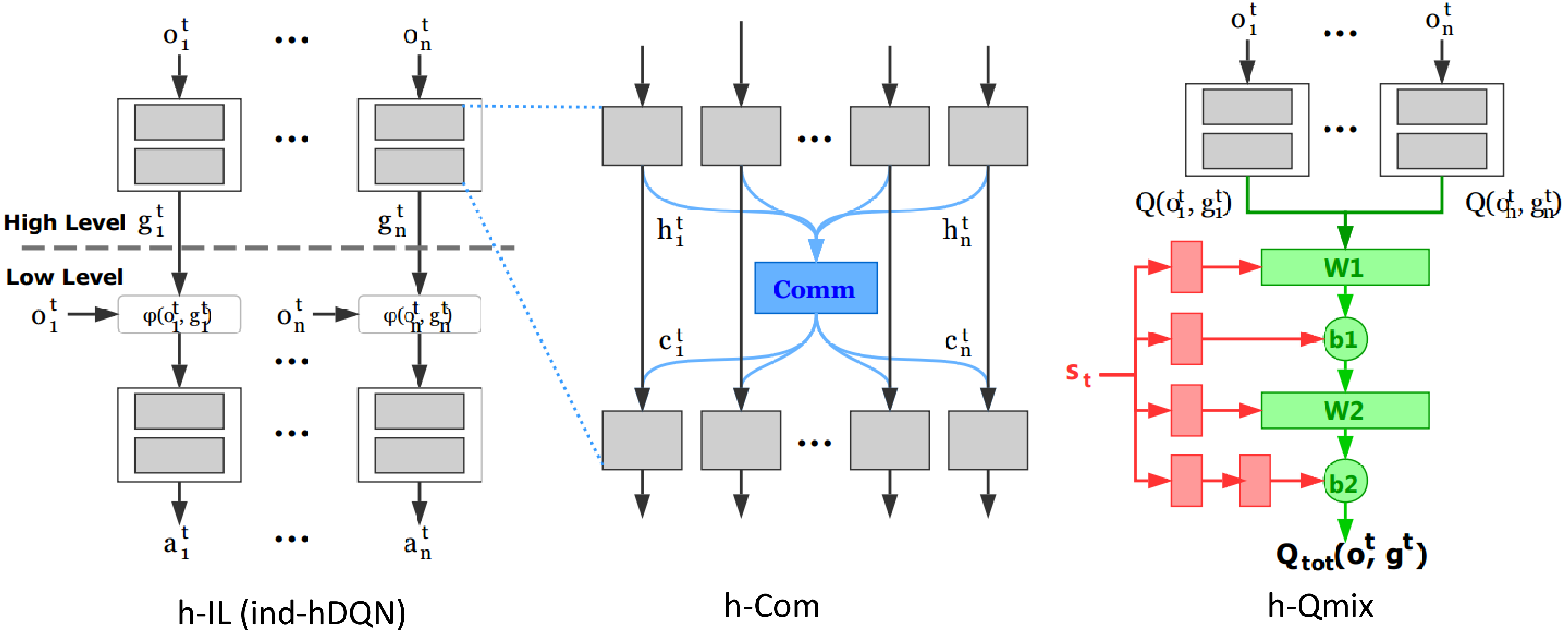
Discrete tasks: grid world environments
Continuous tasks: robotics control in MuJoCo





Andrew Levy et al., Learning Multi-Level Hierarchies with Hindsight. ICLR 2019

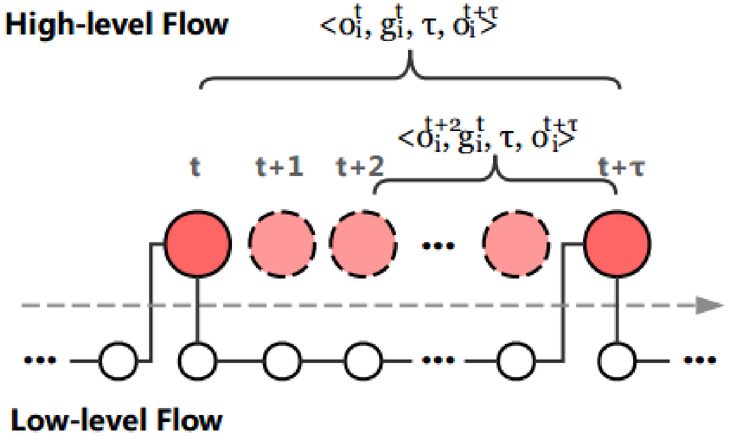
Hierarchical Deep Multiagent Learning



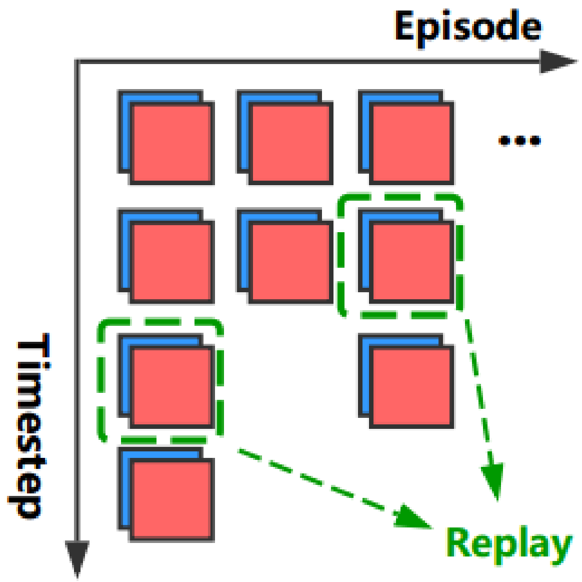
Hierarchical Deep Multiagent Reinforcement Learning, Arxiv 2018

Hierarchical Deep Multiagent Learning

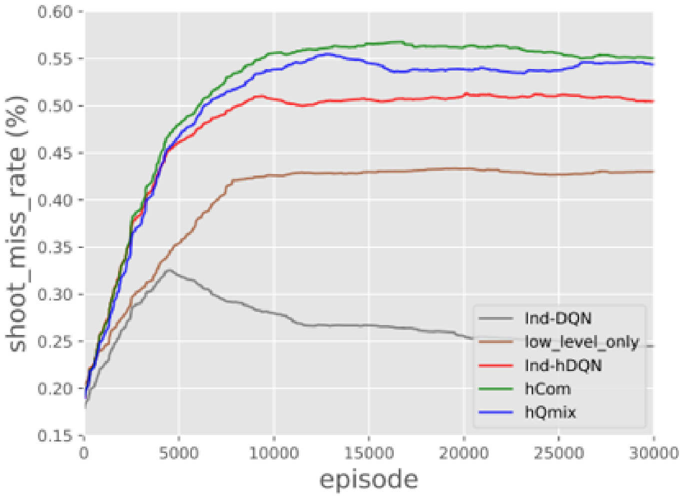
- Augmented Concurrent Experience Replay
  - Coordinate agents policy update
  - Improve high-level sparse experience

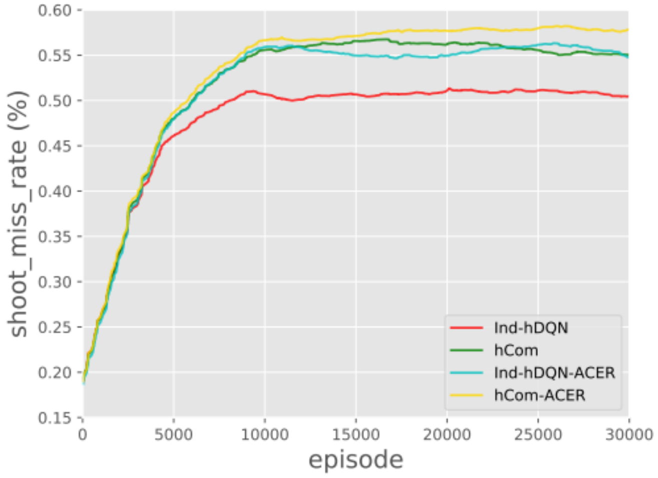


- Low-level Parameter Sharing
  - Support the learning of specialized skills
  - Improve sample efficiency and facilitate training process



Hierarchical Deep Multiagent Learning





Other Works of HRL with Discrete Temporal Abstraction

- Bacon et al., The Option-Critic Architecture, AAAI 2018
  - **Learn option architecture autonomously in an end-to-end fashion** through *Intra-Option Policy Gradients* and *Termination Gradients*
- Harb et al., When Waiting Is Not an Option: Learning Options With a Deliberation Cost, AAAI 2018
  - Leverage the *bounded rationality framework* to **improve the learning** of *the Option-Critic architecture*
- Rafati et al., Learning Representations in Model-Free Hierarchical Reinforcement Learning, AAAI 2019
  - Automatically **discover effective goals** based on *Anomaly Detection* and *K-Means Clustering*
- Wu et al., Model Primitive Hierarchical Lifelong Reinforcement Learning, AAMAS 2019
  - Use diverse **suboptimal world models** to **decompose** complex task into sub-policies and learn to **reuse** sub-policies in *life-long learning*
- Nachum et al., Near-optimal Representation Learning For Hierarchical Reinforcement Learning, ICLR 2019
  - Theoretically prove and propose a approach to **learn near-optimal goal representation**
- Li et al., Hierarchical Reinforcement Learning with Advantage-Based Auxiliary Rewards, NeurlPS 2019
  - Set **auxiliary rewards for low-level learning** based on **the advantages of the high-level policy**
- Minh Le et al., Hierarchical Imitation and Reinforcement Learning, ICML 2018
  - Incorporate *imitation learning* in different levels of hierarchy (e.g., h-DQN) to further **improve sample efficiency**
- Yang et al., Hierarchical Cooperative Multi-Agent Reinforcement Learning with Skill Discovery, AAMAS 2020
  - **Learn cooperative policies at the high level** with centralized training over **independent learned skills/sub-policies** at the low level among multiple agents
- Han et al., Multi-Agent Hierarchical Reinforcement Learning with Dynamic Termination, AAMAS 2020
  - Learn **dynamic termination** to **alleviate the inconsistency** of multiagent learning hierarchical policies

Summary

- Learning with Sparse Rewards: From Sparse to Dense
  - Reward Learning/Shaping
    - leveraging expert/good trajectory to learn optimal reward signals (SGAIL/Multiagent GSAIL)
    - generate intrinsic rewards to encourage better explorations (exploration-oriented intrinsic rewards)
    - Optimality of the reward function?
  - Temporal/spatial credit assignment (single-agent/multiagent settings)
    - Integrated Gradient credit assignment (single-agent credit assignment)
    - Difference reward and path integrated gradient (multiagent credit assignment)
    - Correctness of the credit assignment?
  - Task hierarchical decomposition (hierarchical RL)
    - Discrete vs. continuous subtasks
    - High-level: different ways of performing off-policy correction; requires multiagent coordinations;
    - Low-level: receive reward feedbacks from subgoals; introduce intrinsic rewards
    - More efficient way of automatically learning task hierarchy and decomposition?

Thank you

Q&A