Interactive Actor-Critic for RL in Cooperative-Competitive Environments

Prashant Doshi
THINC Lab[1]
University of Georgia

Keyang He
THINC Lab
University of Georgia

[1]http://thinc.cs.uga.edu

Outline

- Reinforcement learning introduction
- Organization domain
- Multi-agent reinforcement learning
- Interactive advantage actor-critic (IA2C)
- Many-agent reinforcement learning
- IA2C with Dirichlet-multinomial model

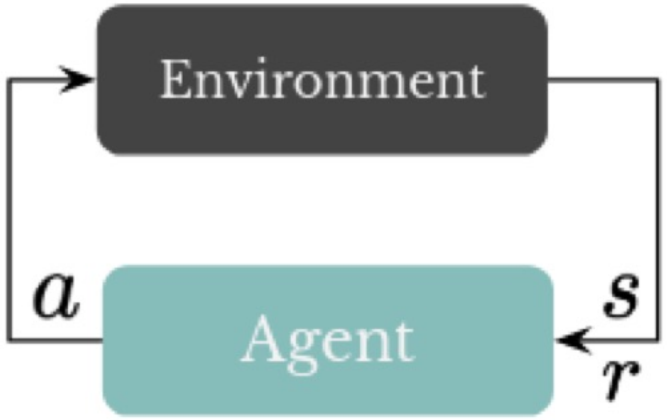
Outline

- Reinforcement learning introduction
- Organization domain
- Multi-agent reinforcement learning
- Interactive advantage actor-critic (IA2C)
- Many-agent reinforcement learning
- IA2C with Dirichlet-multinomial model

Reinforcement Learning

A learning agent interacts with an environment to solve a sequential decision-making problem. Fully observable environments are modeled as Markov Decision Processes (MDPs) as:

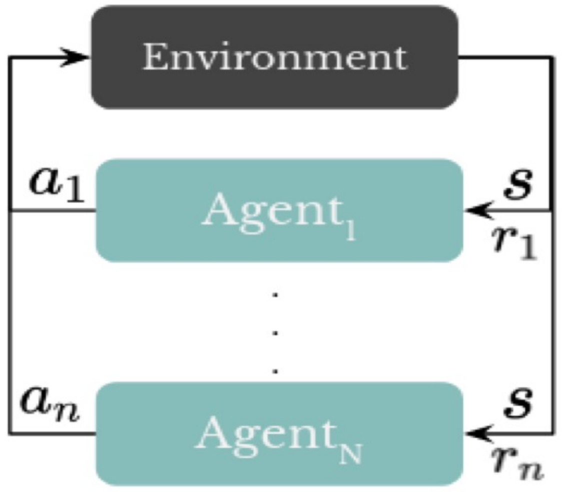
S	State Space
A	Action Space
T	Transition function $T: S \times A \times S \mapsto [0,1]$
R	Reward function $R: S \times A \mapsto \mathbb{R}$



The agent aims to find an optimal policy π^*, a mapping from the environment states to actions, that maximizes the expected return.

Multi-Agent Reinforcement Learning

In multi-agent systems, state transition and reward depend on the joint action of all the agents.

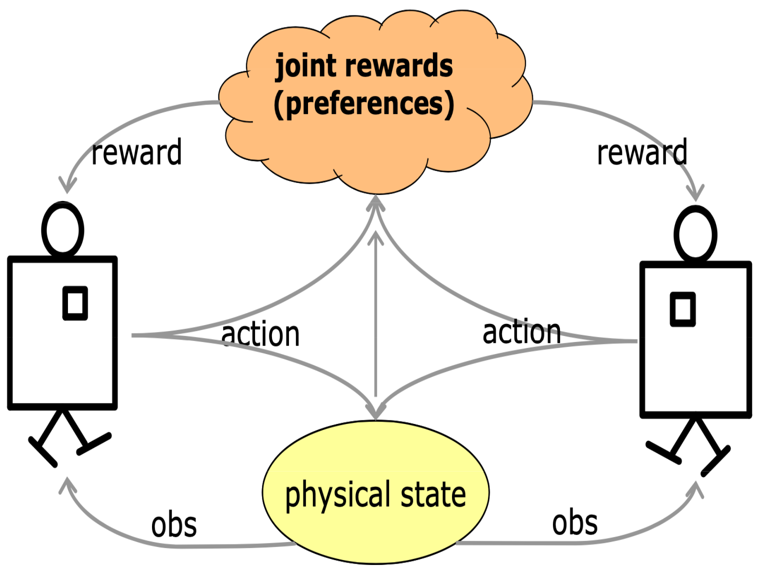


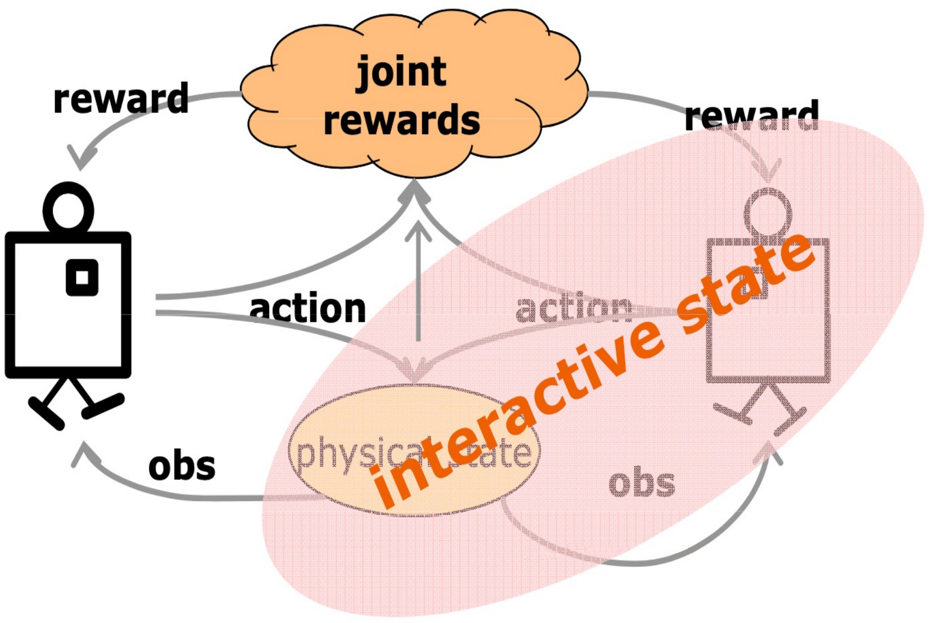
The complexity of multi-agent system arises many challenges such as:

- Curse of dimensionality: exponential growth of the joint action space
- Learning goal: agent returns are correlated and cannot be maximized independently
- Nonstationarity: all agents learning simultaneously

Interactive Partially Observable Markov Decision Process (I-POMDP)[1]

Partially observable multi-agent environments can be modeled as I-POMDPs





[1] Piotr Gmytrasiewicz and Prashant Doshi, *JAIR* 2005

Outline

- Reinforcement learning introduction
- Organization domain
- Multi-agent reinforcement learning
- Interactive advantage actor-critic (IA2C)
- Many-agent reinforcement learning
- IA2C with Dirichlet-multinomial model

Organization Domain: Overview

Models a business organization featuring a mixed cooperative-competitive setting

- Compete for individual rewards
- Cooperate for group rewards
- A proportion of past rewards is added to current reward as a bonus

Organization Domain: Joint Action

Individual actions: *self*, *balance*, *group*

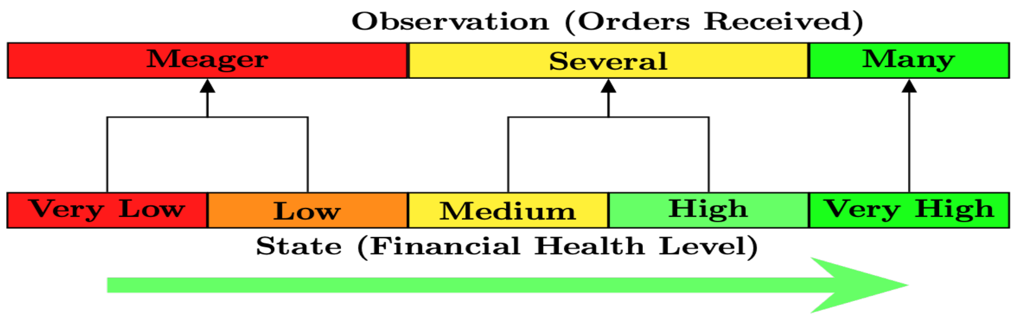
The joint action is determined as:

Joint Action	Individual Actions
Self	# of agents picking self > # of agents picking group
Balance	# of agents picking self = # of agents picking group
Group	# of agents picking self < # of agents picking group

If all agents pick *balance* action, the joint action is also *balance*.

Organization Domain: State Transition

States represent the organization's financial health level:



The state transition is determined by:

Joint Action	State Transition
Self	State decrease by 1 level. State remain unchanged if it is already at the 'Very Low' level.
Balance	State remain unchanged.
Group	State increase by 1 level. State remain unchanged if it is already at the 'Very High' level.
Group (all)	State increase by 2 level. State remain unchanged if it is already at the 'Very High' level. State increase by 1 level if it is at level 'High'.

Organization Domain: Reward

Each agent receives rewards from three sources:

Reward Type	Reward Function
Individual	$R_i^t \leftarrow R_i(s^t, a_i^t)$
Group	$R_0^t \leftarrow R(s^t, a^t)$
History-dependent	$R_{-1}^t = \phi(\sum_i R_i^{t-1} + R_0^{t-1})$

The goal for each agent i is to optimize

$$\mathbb{E}_{trajectories}\left(\sum_t \gamma^t (R_0^t + R_i^t + R_{-1}^t)\right)$$

To obtain optimal action, each agent needs to consider cooperation and competition simultaneously.

Outline

- Reinforcement Learning
- Organization domain
- Multi-agent reinforcement learning
- Interactive advantage actor-critic (IA2C)
- Many-agent reinforcement learning
- IA2C with Dirichlet-multinomial model

Modeling Organization Domain as I-POMDP[1]

Integrate the non-Markovian reward

- History-dependent reward is included as an extra state feature s_r, while s_f represents the underlying physical state features

$$T_i(\langle s_f, s_r\rangle, a_i, a_j, \langle s'_f, s'_r\rangle) = \begin{cases} T(s_f, a_i, a_j, s'_f), & \text{if } s'_r = R(s_f, a_i, a_j) + \phi \cdot s_r \\ 0 & \text{otherwise} \end{cases}$$

- o_f is the noisy observation of s_f, o_r is set equal to s_r (i.e., agents have perfect information about previous reward).

$$Z_i(a_i, a_j, \langle s_f, s_r\rangle, \langle s'_f, s'_r\rangle, \langle o'_f, o'_r\rangle) = \begin{cases} Z(a_i, a_j, s_f, s'_f, o'_f), & \text{if } (s'_r = R(s_f, a_i, a_j) + \phi \cdot s_r) \wedge (o'_r = s'_r) \\ 0 & \text{otherwise} \end{cases}$$

- The reward function has an extra term $\phi \cdot s_r$ representing the history-dependent reward.

$$R_i(\langle s_f, s_r\rangle, a_i, a_j) = R(s_f, a_i, a_j) + \phi \cdot s_r$$

[1] Gmytrasiewicz and Doshi, *JAIR* 2005

Modeling Organization Domain as I-POMDP

Interactive state: IS_i

- Include s_f, s_r, and M_j

Private observation: ω_i noised observation of other agents' action

The belief update for the new I-POMDP formulation is:

$$b'_i(is'|b_i, a_i, o'_i, \omega'_i) = \underbrace{b'_i(\langle s'_f, s'_r\rangle|b_i, a_i, o'_i, \omega'_i)}_{\text{belief over states}} \times \underbrace{b'_i(m'_j|\langle s'_f, s'_r\rangle, b_i, a_i, o'_i, \omega'_i)}_{\text{belief over models}}$$

The Bellman equation for the new I-POMDP formulation is:

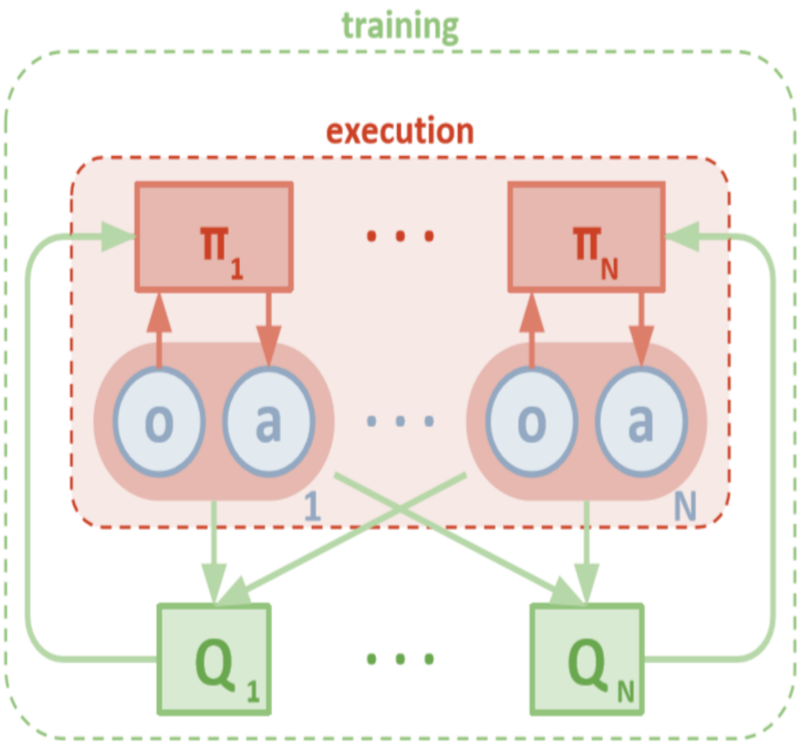
$$V(b_i) = \max_{a_i}[\sum_{s_f, s_r}\sum_{a_j} R_i(\langle s_f, s_r\rangle, a_i, a_j)\Pr(a_j|m_j)b_i(\langle s_f, s_r\rangle) +$$

reward from current belief state

$$\gamma\sum_{a_j}\sum_{s'_f, s'_r = R(s_f, a_i, a_j)+\phi\cdot s_r}\Pr(a_j|m_j)\sum_{o'_i, \omega'_i} T(s_f, a_i, a_j, s'_f)\times b_i(\langle s_f, s_r\rangle)Z(a_i, a_j, s_f, s'_f, o'_f)W_i(a_i, a_j, \omega'_i)V(\tau(b_i, a_i, o'_i, \omega'_i, b'_i))$$

discounted future reward

Related Work: MADDPG[1]



Multi-agent deep deterministic policy gradient (MADDPG) adopts a centralized critic and decentralized actor network structure:

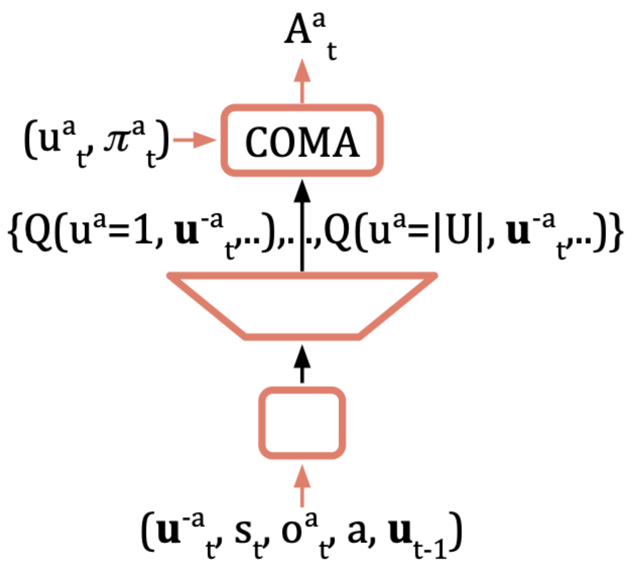
- Actor networks use local observations for deterministic actions
- Critic network uses joint state-action pairs to estimate Q-values.
- Policy inference: maximize the log probability of other agent's actions

$$\mathcal{L}(\phi_i^j) = -\mathbb{E}_{o_j,a_j}[\log \hat{\boldsymbol{\mu}}_i^j(a_j|o_j) + \lambda H(\hat{\boldsymbol{\mu}}_i^j)]$$

[1] Lowe et al., *NIPS* 2018

Related Work: COMA[1]

Counterfactual multi-agent policy gradient addresses the credit assignment in multi-agent reinforcement learning by quantifying contributions of individual agents


A^a_t
(u^a_t, π^a_t)
COMA
$\{Q(u^a=1, \mathbf{u}^{-a}_t..),..,Q(u^a=|U|, \mathbf{u}^{-a}_t..)\}$
$(\mathbf{u}^{-a}_t, s_t, o^a_t, a, \mathbf{u}_{t-1})$

- Unlike MADDPG, COMA trains a probabilistic policy.
- COMA calculates an expected value over all actions that an agent can take while keeping the actions of all other agents fixed.

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

[1] Foerster et al., *AAAI* 2018

Related Work: LOLA

Learning with opponent learning awareness takes account of the learning of other agent when updating its own policy

- LOLA include an extra term in its update rule:

$$\boxed{\left(\frac{\partial V^1(\theta_i^1,\theta_i^2)}{\partial \theta_i^2}\right)^T}\boxed{\frac{\partial^2 V^2(\theta_i^1,\theta_i^2)}{\partial \theta_i^1 \partial \theta_i^2} \cdot \delta\eta}$$

- LOLA agents try to optimize their return after one anticipated learning step of the opponent.

[1] Foerster et al., *AAMAS* 2018

Outline

- Reinforcement learning introduction
- Organization domain
- Multi-agent reinforcement learning
- Interactive advantage actor-critic (IA2C)
- Many-agent reinforcement learning
- IA2C with Dirichlet-multinomial model

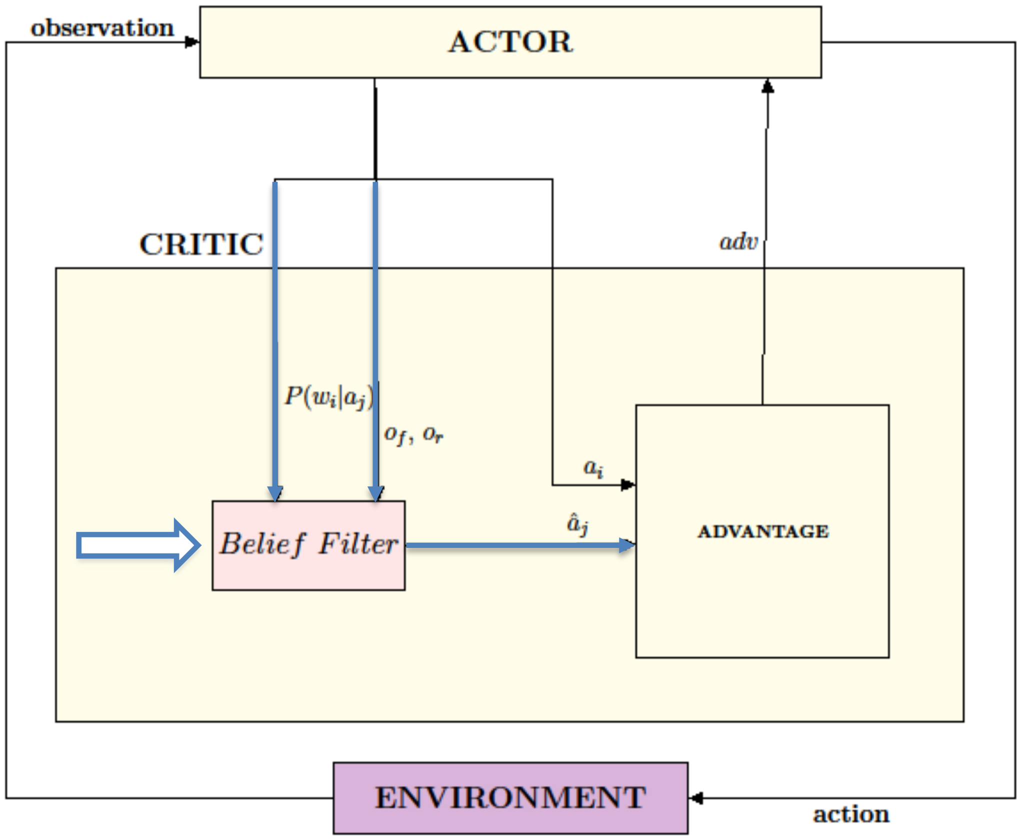
Interactive Advantage Actor-Critic (IA2C)[1]

Overview

- IA2C extends advantage actor-critic by predicting other agents' actions based on maintaining beliefs over models.

- A belief filter is added to the critic network for predicting other agents' action.

- The belief filter uses ω_i and o_f, o_r to predict other agents' action $\hat{a}_j$ for next timestep used in advantage computation.



Interactive Advantage Actor-Critic (IA2C)

Advantage function is modified to accommodate I-POMDP and history-dependent reward state feature.

- $A(\langle o_f, o_r\rangle, a_i, \hat{a}_j) = avg[r + \gamma Q(\langle o_f', o_r'\rangle, a_i', \hat{a}_j') - Q(\langle o_f, o_r\rangle, a_i, \hat{a}j)]$

The actor's gradient is:

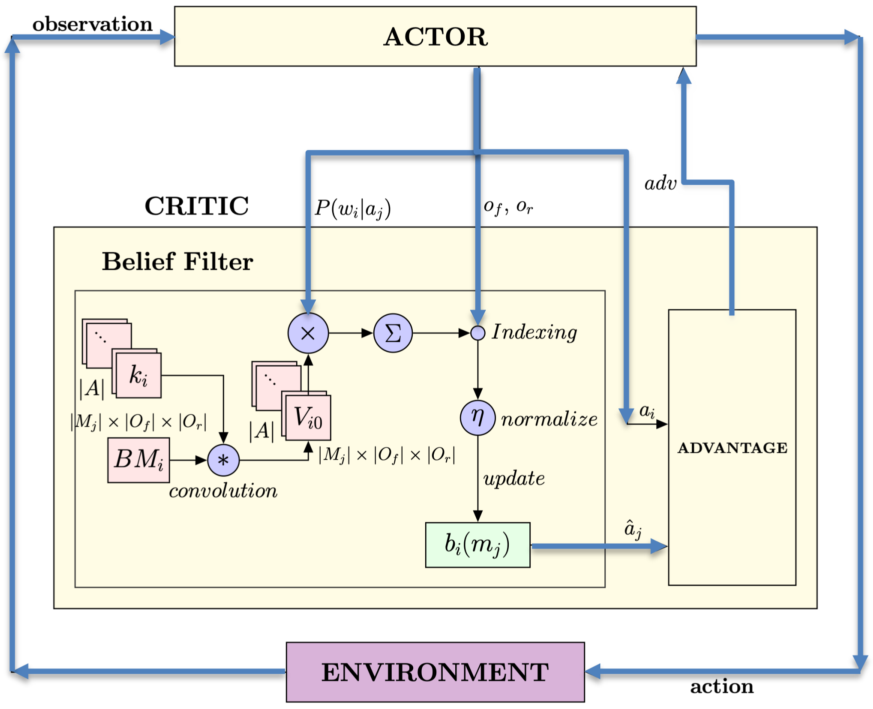
- $avg[\nabla_\theta \log \pi_\theta(a_i | \langle o_f, o_r\rangle) A(\langle o_f, o_r\rangle, a_i, \hat{a}_j)]$

$r, \langle o_f', o_r'\rangle$, and a_i' are samples, $\hat{a}_j$ and $\hat{a}_j'$ are predicted actions. The avg is taken over sampled trajectories.

Interactive Advantage Actor-Critic (IA2C)

IA2C workflow

- Actor interacts with environment, receives observations. At the same time, the actor also receives private observations (ω_i).
- Actor sends o_f, o_r and ω_i to the belief filter in critic network for action prediction.
- The critic network use the predicted action $\hat{a}_j$ from belief filter to compute advantage.
- Critic sends advantage value to actor.
- Actor updates network parameter based on the advantage value.

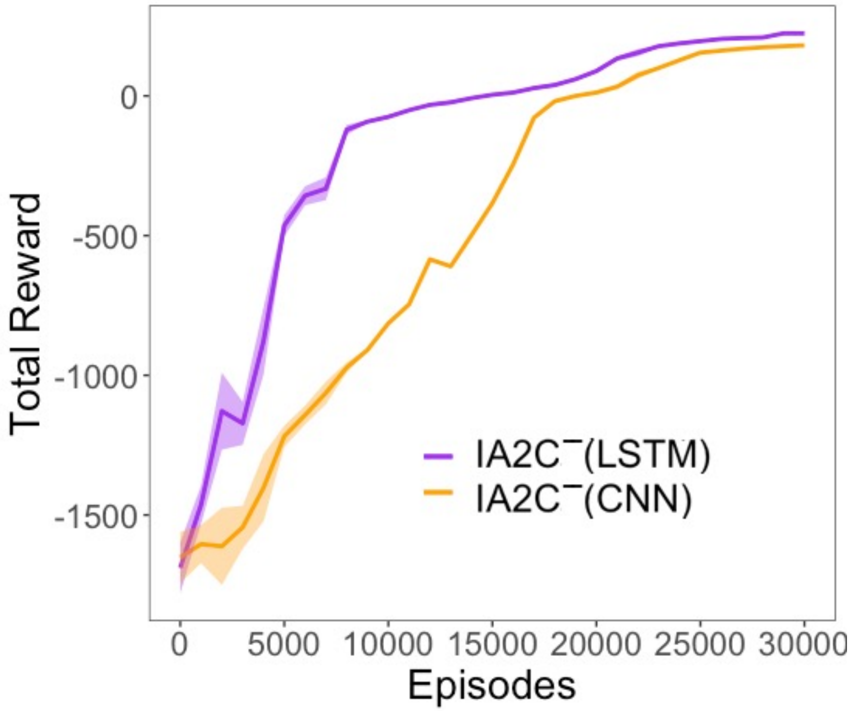


Experiments: History-Dependent Rewards

IA2C$^-$ only utilizes s_f and o_f , omits s_r and o_r.

- IA2C$^-$(LSTM): converged to optimal policy.
- IA2C$^-$(CNN): converged to sub-optimal policy.
- Figure is plotted by averaged data from 5 independent runs.

Experiment shows that our approach of accommodating history in the I-POMDP formulation is not only sufficient, but also necessary.



Experiments: Cooperation in Organization

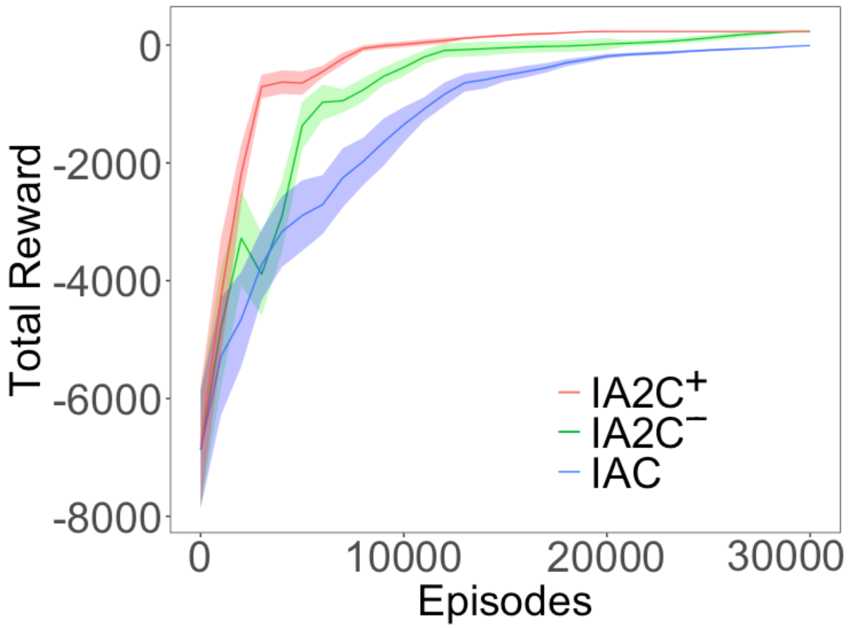
We compare Independent actor-critic (IAC) with IA2C.

- Independently learning agent cannot learn optimal policy.

- **IA2C$^+$ utilizes s_r, o_r.**

- Both IA2C$^+$ and IA2C$^-$ learn optimal policy, while IA2C$^+$ converges faster.

- Figure is plotted by averaged data from 5 independent runs.



Experiment shows that cooperation is needed to reach optimality in Organization domain.

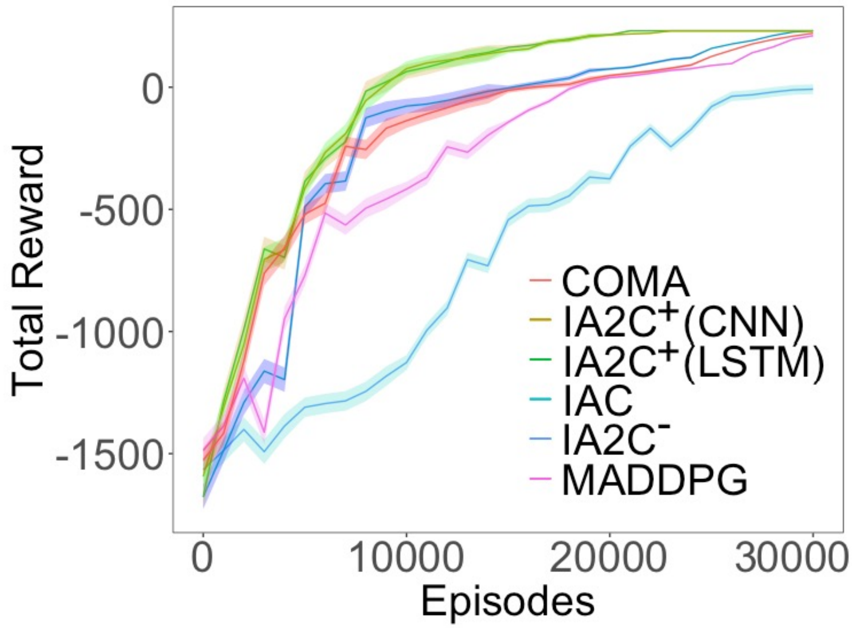
Experiments: Comparison with MARL techniques

We compare the performance of IA2C with COMA and MADDPG.

- Both COMA and MADDPG can converge to optimal policy but requires almost twice as many episodes as IA2C$^+$.

- Figure is plotted by averaged data from 5 independent runs.

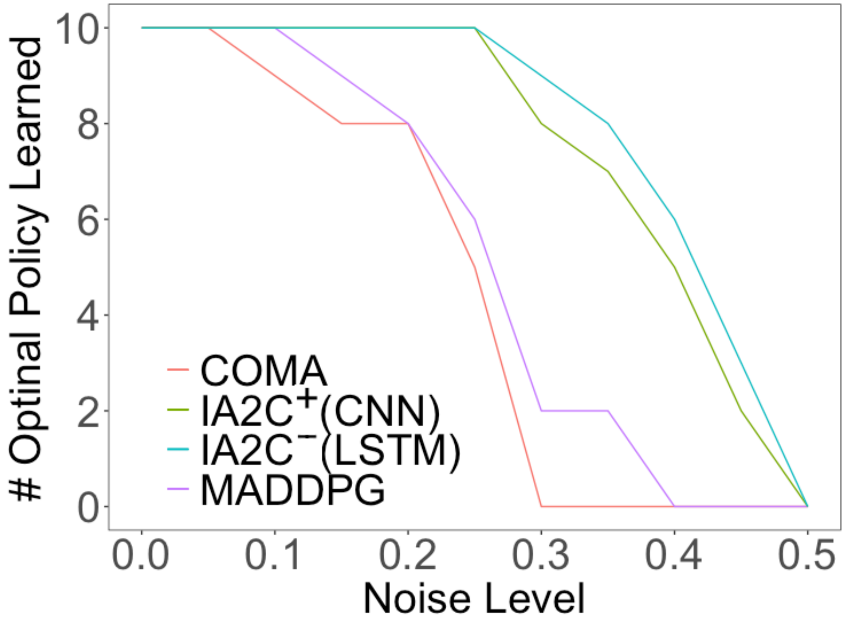
Experiment shows that IA2C$^+$ demonstrates better performance compared to existing MARL techniques.



Experiments: Varying private observation noise

We gradually increase the noise level in private observation to test the robustness of each method.

- The performance of COMA and MADDPG drop drastically when the private observation noise level becomes greater than 0.9.
- IA2C^{+} still managed to learn optimal policy when private observation only reveals other agents' true actions at 0.6 probability.
- Figure is plotted by the # of optimal policy learned from 10 experiment runs.



Experiment shows that IA2C demonstrates consistent learning and robustness to higher levels of noise.

IA2C Summary

- IA2C$^+$ combines decentralized actor-critic based learning with belief filter that maintains beliefs over a finite set of models of the other agents.
- IA2C$^+$ doesn’t require policy exchanging among agents or perfect observation of other agents’ actions. It converges faster and is less prone to noise from observing other agents’ actions compared to existing MARL techniques.
- IA2C$^+$ still suffers from the curse of dimensionality:
  - The joint action space grows exponentially.
  - Belief updates needs to be done for each other agents.

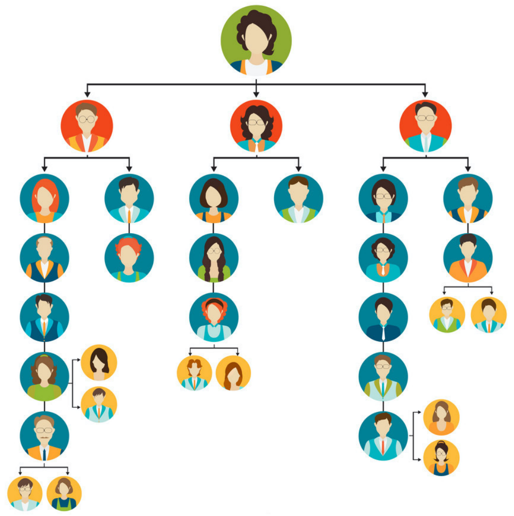
Outline

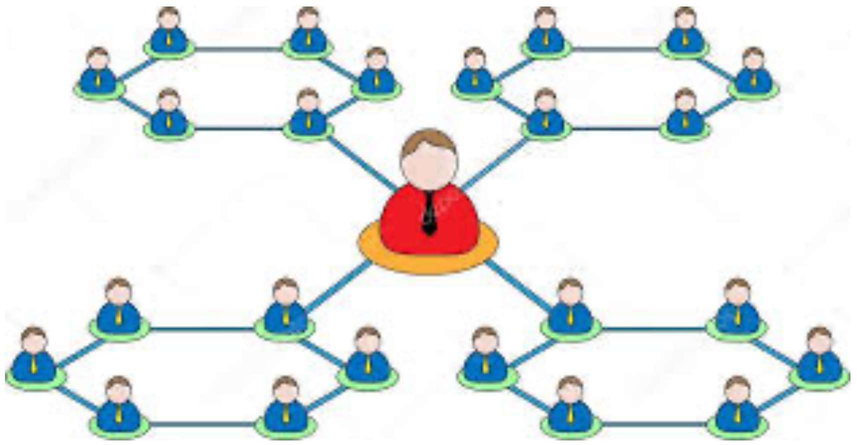
- Reinforcement learning introduction
- Organization domain
- Multi-agent reinforcement learning
- Interactive advantage actor-critic (IA2C)
- Many-agent reinforcement learning
- IA2C with Dirichlet-multinomial model

Many-Agent Domain Features

Many-agent domains often exhibit the following features:

- Population homogeneity
    - All agents have the same action space
- Action anonymity
    - State transition and reward only depend on the count distribution of actions in the population





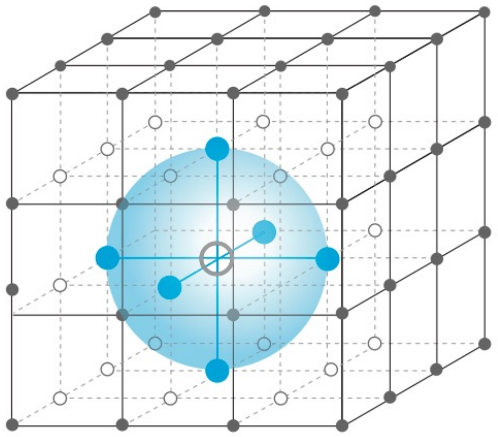
Related Work: Mean-Field Reinforcement Learning[1]

To address the exponentially increased joint action space, mean-field reinforcement learning factorize Q-function using only pairwise local interactions:

$$Q^j(s,\boldsymbol{a}) = \frac{1}{N^j}\sum_k Q^j(s,a^j,a^k) \approx Q^j(s,a^j,\bar{a}^j), \quad \bar{a}^j = \frac{1}{N^j}\sum_k a^k$$

- $\bar{a}^j = [\bar{a}^j_1, \ldots, \bar{a}^j_{|A|}]$ can be interpreted as the empirical distribution of the actions taken by agent j's neighbors.

The pairwise interactions $Q^j(s,a^j,a^k)$ between agent j and each neighboring agent k are simplified as that between the central agent and the virtual mean agent. Many-agent interactions are converted into two-agent interactions.



[1] Yang et al., *ICML* 2018

Representing Joint Action as Action Configuration

Action configuration is a vector of the distinct actions performed by the agent population, denoted as:

$$C^{\boldsymbol{a}} = \langle \#a^1, \#a^2, \dots, \#a^{|A|} \rangle$$

Joint actions are mapped to configurations by a projection function δ.

- For example, $\delta(\langle self, self, group, group \rangle) = \langle 2{,}2{,}0 \rangle$.
- δ is a many-to-one mapping. The original joint action cannot be decided given its projected action configuration.

Action Configuration in POMDP

Let $\dot{\boldsymbol{a}}_{-0}$ denotes any permutation of $\boldsymbol{a}_{-0}$. For any $\mathrm{s}, \mathrm{a}_0, \mathrm{s}', \dot{\boldsymbol{a}}_{-0}$, we have:

- $T_0(s, a_0, \dot{\boldsymbol{a}}_{-0}, s') = T_0(s, a_0, C^{\boldsymbol{a}}_{-0}, s')$
- $Z_0(a_0, \dot{\boldsymbol{a}}_{-0}, s, s', o') = Z_0(a_0, C^{\boldsymbol{a}}_{-0}, s, s', o')$
- $W_0(a_0, \dot{\boldsymbol{a}}_{-0}, \omega_0') = W_0(a_0, C^{\boldsymbol{a}}_{-0}, {\omega_0}')$
- $R_0(s, a_0, \dot{\boldsymbol{a}}_{-0}) = R_0(s, a_0, C^{\boldsymbol{a}}_{-0})$

The above equivalences naturally lead to the following property of the Q-function:

- $Q_0(o, a_0, \dot{\boldsymbol{a}}_{-0}) = Q_0(o, a_0, C^{\boldsymbol{a}}_{-0})$

Action Configuration Belief Update

$$b_0'(m_j'|b_0, a_0, o_0', \omega_0') \propto \sum_{m_j \in M_j} b_0(m_j) \sum_{a_j} Pr(a_j|m_j) \sum_{C \in \boldsymbol{C}^{a-0}} \boxed{\Pr(C|b_0(M_1), \ldots, b_0(M_N))}$$
$$W_0(a_0, C, \omega_0')\delta_K(\pi_j, \pi_j')\delta_K(APPEND(h_j, \langle a_j, o' \rangle), h_j')$$

The probability of an action configuration in the distribution over the set of configurations is obtained using a dynamic programming procedure introduced by Jiang et al.[1]

1 Jiang et al., *Games and Economic Behavior* 2011

Dynamic Programming for Obtaining Action Configuration Distribution

Algorithm 1 Computing configuration distribution $Pr(\mathcal{C}|b_0(M_1), b_0(M_2), \ldots, b_0(M_N))$

```
Require: $\langle b_0(M_1), b_0(M_2), \ldots, b_0(M_N) \rangle$
Ensure: $P_N$, which is the distribution $Pr(\mathcal{C}^{\mathbf{a}-o})$ represented as
  a trie.
  Initialize $c_0^{a_i} \leftarrow (0, \ldots, 0)$, $P_0[c_0^{a_i}] \leftarrow 1.0$
  for $k = 1$ to $N$ do
    Initialize $P_k$ to be an empty trie
    for $c_{k-1}^{a_i}$ from $P_{k-1}$ do
      for $a_k^{a_i} \in A_k^{a_i}$ such that $\pi_k^{a_i}(a_k^{a_i}) > 0$ do
        $c_k^{a_i} \leftarrow c_{k-1}^{a_i}$
        if $a_k^{a_i} \neq \emptyset$ then
          $c_k^{a_i}(a_k^{a_i}) \overset{+}{\leftarrow} 1$
        end if
        if $P_k[c_k^{a_i}]$ does not exist then
          $P_k[c_k^{a_i}] \leftarrow 0$
        end if
        $P_k[c_k^{a_i}] \overset{+}{\leftarrow} P_{k-1}[c_{k-1}^{a_i}] \times \pi_k^{a_i}(a_k^{a_i})$
      end for
    end for
  end for
  return $P_N$
```

Check if the new agent's action distribution introduces new configurations

Compute probabilities of all possible configurations

Limitations of Belief Update

The belief update procedure requires models of other agents or a pre-defined model set

- Beliefs are over other agent's models
- Models are required for obtaining other agent's action distributions

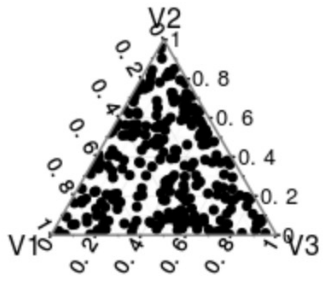
The belief update procedure needs to be done for each other agents

Outline

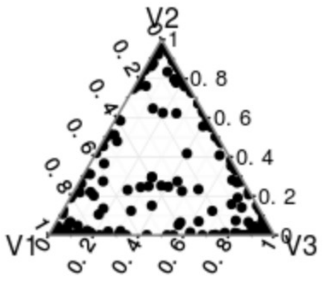
- Reinforcement learning introduction
- Organization domain
- Multi-agent reinforcement learning
- Interactive advantage actor-critic (IA2C)
- Many-agent reinforcement learning
- IA2C with Dirichlet-multinomial model

Modeling Agent Population

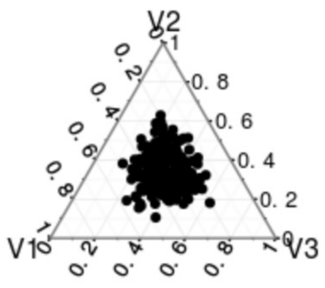
The Dirichlet-multinomial distribution models categorical variables.



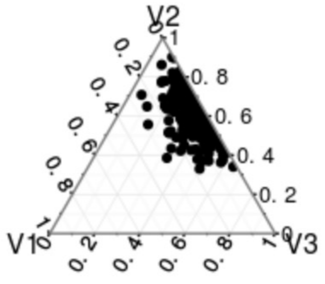
(1,1,1)



(0.2,0.2,0.2)

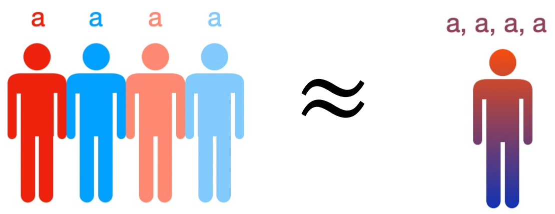


(10,10,10)



(1,10,5)

- Action configuration can be treated as a set of samples drawn from the Dirichlet distribution.



- The Dirichlet distribution is updated using private observation at every time step.
- The accuracy benefits from large agent population.

Dirichlet-Multinomial for Action Configuration

- Suppose the action space is $\{a_1, \dots, a_{|A|}\}$ for each agent.
- $\boldsymbol{\theta} = (\theta_1, \dots, \theta_{|A|})$, θ_n is the probability for action a_n
- $\boldsymbol{\theta}$ has a Dirichlet-multinomial distribution with parameter $\boldsymbol{\alpha}$ if:

$$Pr(\boldsymbol{\theta}|\boldsymbol{\alpha}) = \frac{\Gamma(\sum_n \alpha_n)}{\Pi_n \Gamma(\alpha_n)} \Pi_n \theta_n^{\alpha_n - 1}$$

$\alpha_n > 0$ for all n, $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_N)$, and $\sum_n \theta_n = 1$.

- The probability of an action configuration C can be expressed as:

$$\Pr(C|\boldsymbol{\theta}) = \Pr\left(\#a_1^C, \dots, \#a_{|A|}^C \middle| \boldsymbol{\theta}\right) = \Pi_{n=1}^{|A|} \theta_n^{\#a_n^C}$$

Dirichlet Distribution Update

After receiving private observation ω_0' at each time step, the Dirichlet-multinomial distribution can be updated by:

$$\Pr(\theta|a_0, \omega_0') \propto \sum_C \Pr(C, \omega_0'|a_0, \boldsymbol{\theta})\ Dirichlet(\boldsymbol{\alpha})$$

$$\propto \boxed{\sum_C W_0(a_0, C, \omega_0') \Pr(C|\boldsymbol{\theta})\ Dirichlet(\boldsymbol{\alpha})}$$

$$\propto \boxed{\sum_C W_0(a_0, C, \omega_0') Dirichlet(\boldsymbol{\alpha} + C)} \approx \boxed{Dirichlet(\boldsymbol{\alpha} + C_{max})}$$

Many-Agent IA2C

Action configuration in actor-critic network:

- Actor network gradient:

$$avg[\nabla_{\theta} \log \pi_{0,\boldsymbol{\theta}}(a_0|o)\; A_0(o, a_0, C^{\boldsymbol{a}_{-0}})]$$

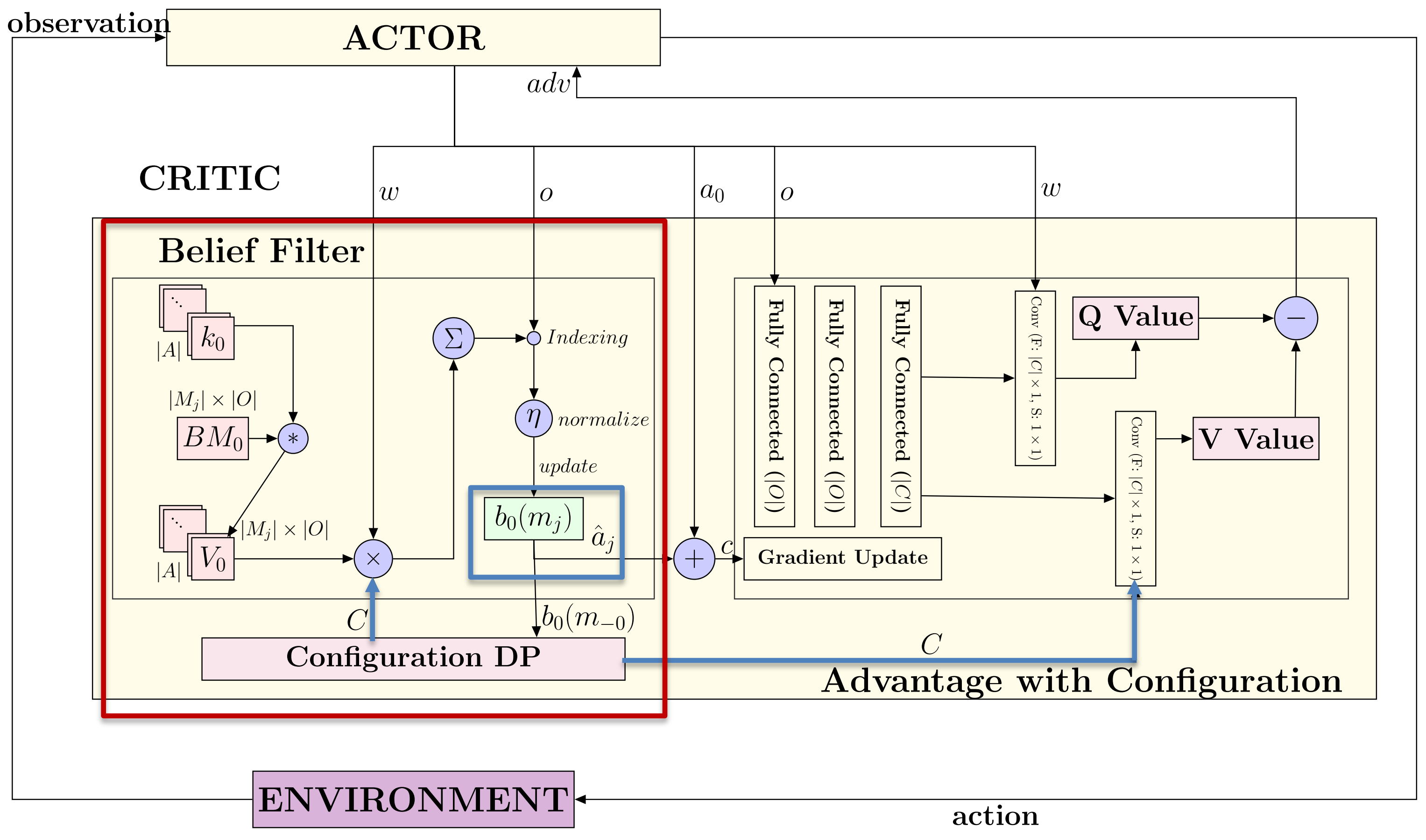
- Advantage function:

$$A_0(o, a_0, C^{\boldsymbol{a}_{-0}}) = avg[r + \gamma Q_0\left(o', a_0', C^{\boldsymbol{a}'_{-0}}\right) - Q_0(o, a_0, C^{\boldsymbol{a}_{-0}})]$$

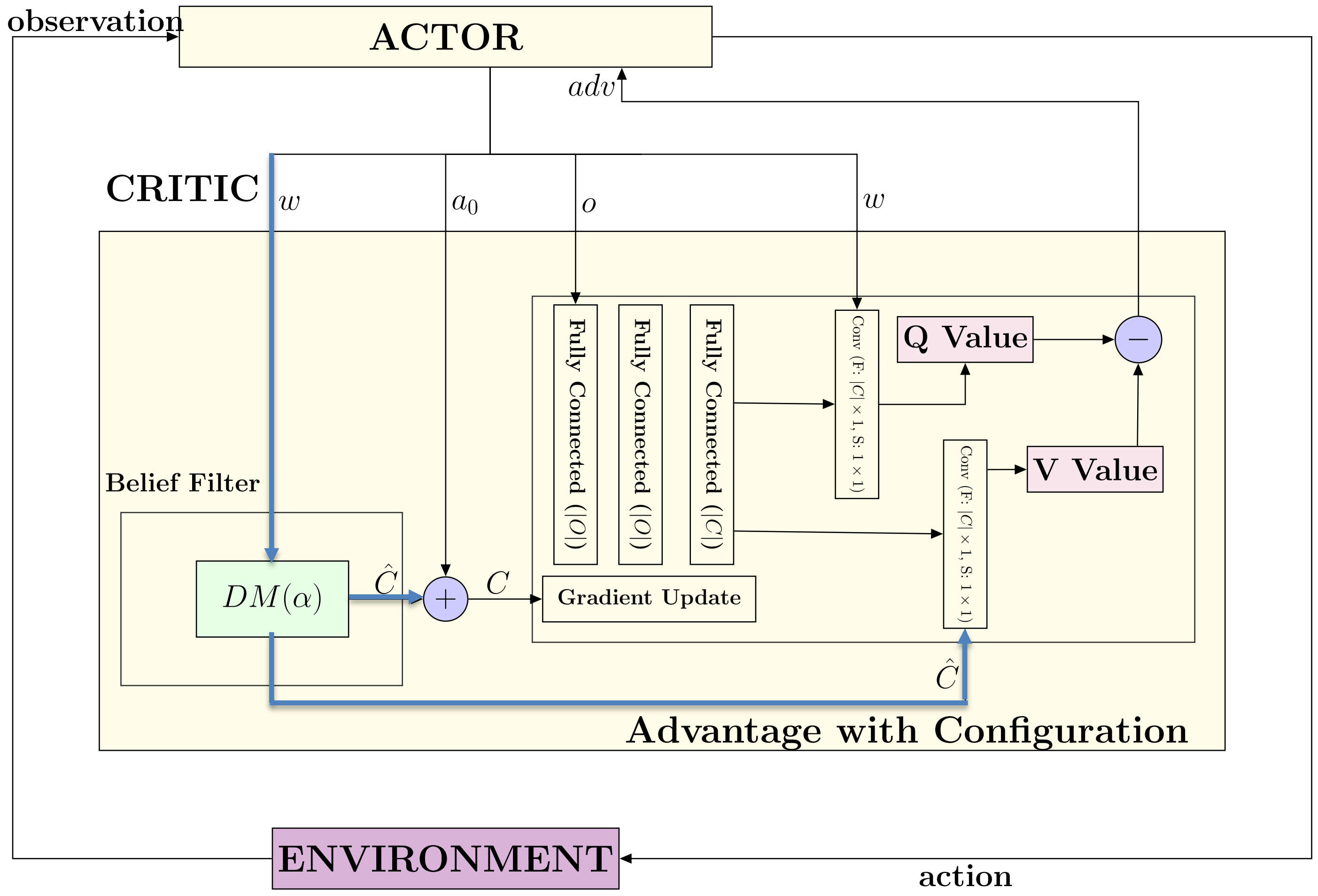
r, o', a_0' are samples, $\boldsymbol{a}_{-0}, \boldsymbol{a}'_{-0}$ are predicted actions.

- Joint actions in IA2C advantage function are replaced by action configurations.

Many-Agent IA2C Network



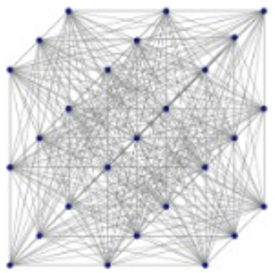
Many-Agent IA2C Network

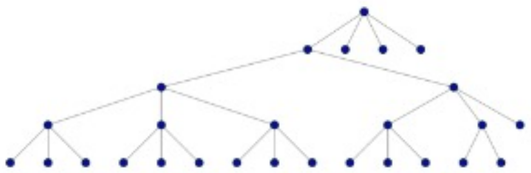


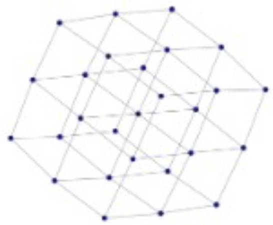
Experiment Domain: Organization Structures

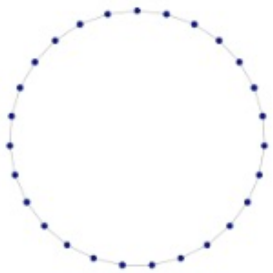
We select five Organization structures that differ in the number of neighborhoods and the number of agents in each neighborhood.

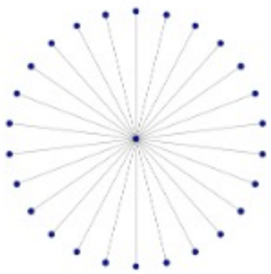
- Group reward is only shared within the neighborhood.


(a) Connected

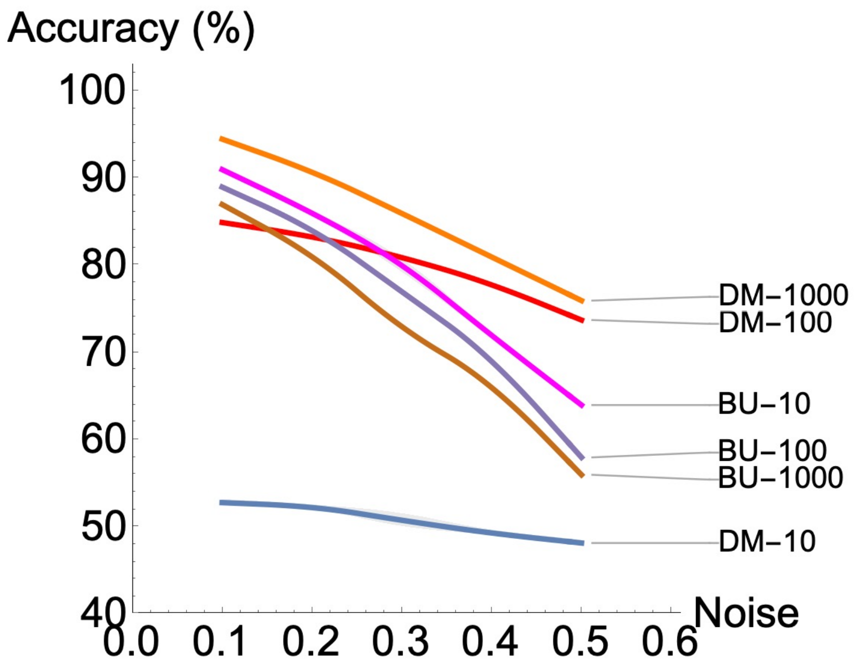

(b) Tree


(c) Lattice


(d) Circle


(e) Star

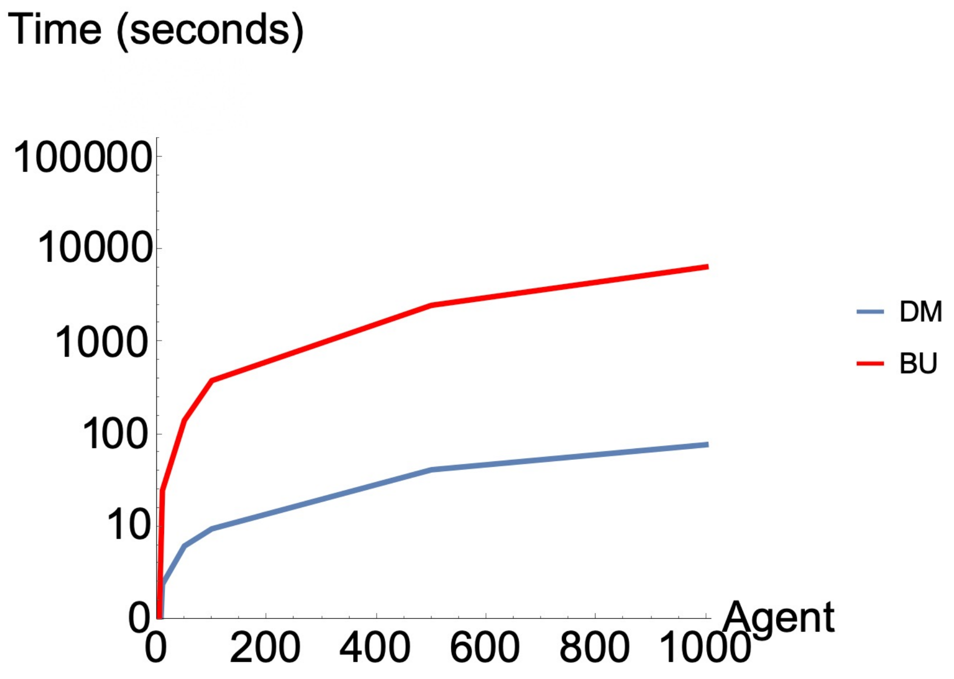
Experiment: Prediction Accuracy



- Dirichlet-multinomial achieves higher accuracy given large population size.
- Belief update has higher accuracy for small agent population.

Dirichlet-multinomial model is more robust than belief update in noisy environments with relatively large population size.

Experiment: Prediction Time



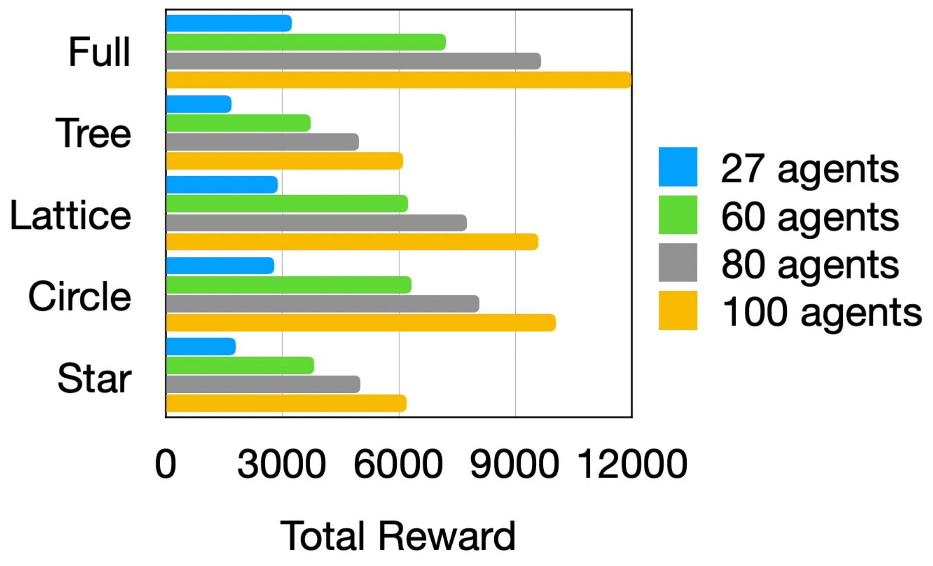
- Dirichlet-multinomial has a near constant run time.

  Belief update run time increase polynomially.

Dirichlet-multinomial model saves time by directly modeling the whole population instead of individuals.

Experiment: Action Configuration

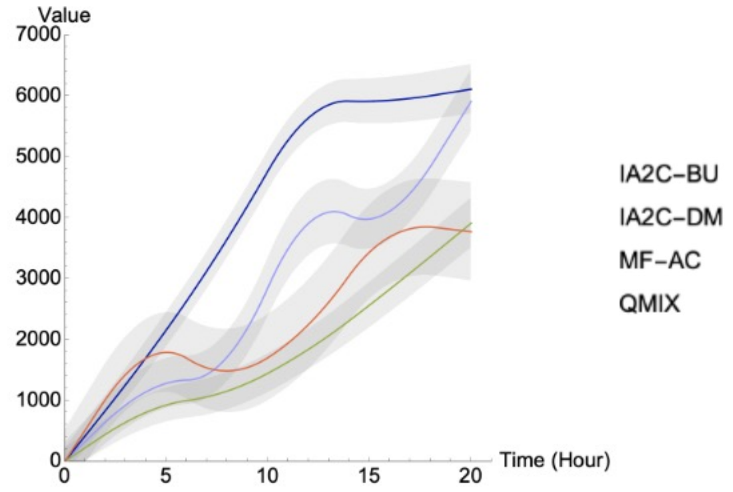
- The cumulative rewards are higher in fully connected structures than other structures.
- It is harder to coordinate across many small neighborhoods.

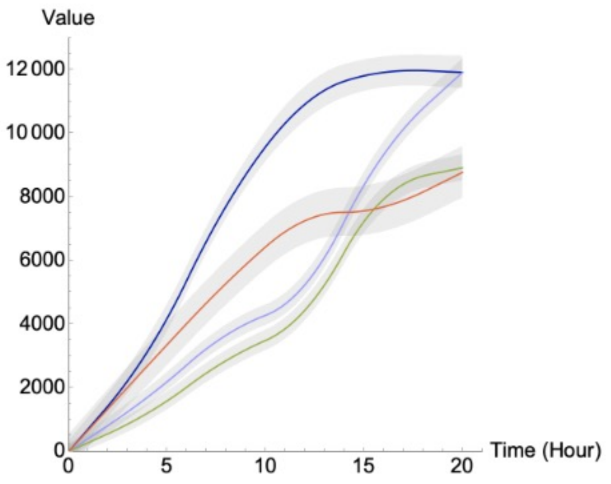


Experiment: Comparison with Many-Agent Methods

- (top) Tree structure Organization and (bottom) Fully connected Organization with 100 agents.
- Both IA2C methods converged to optimal policies. IA2C-DM learns much faster than IA2C-BU.
- MF-AC and QMIX cannot converge within time limit.

IA2C-DM shortens the amount of time IA2C-BU required to converge to optimal policy in Organization by 30-40%.





Many-Agent IA2C Summary

IA2C^{++} replaces joint action with action configuration to reduce the exponentially increased action space.

IA2C^{++} efficiently models agent population.

- IA2C^{++}BU has higher accuracy in small agent populations.
- IA2C^{++}DM does not require pre-defined models and is advantageous in large agent populations.

Overall Conclusion

- Organization domain: a quintessential cooperative-competitive multi-agent domain that features history-dependent reward
- IA2C$^+$ incorporates Bayesian belief update into advantage actor-critic for modeling agent interactions in partially observable settings
- IA2C^{++} with belief update filter that use action configuration scales up in many-agent settings with a relatively small population size. IA2C^{++} that utilizes Dirichlet-multinomial distribution can accurately and efficiently model large agent populations

Thank You

Questions