Efficient Multi-agent Reinforcement Learning and Applications

Presenter: Yan Li

Georgia Institute of Technology

Outline

- **Introduction**: Lightweight Intro to MARL
- **Evaluation**: Efficient policy evaluation for decentralized learning
- **Scalability**: Scaling up MARL for large number of agents
- **Robustness**: Against environment change and applications to traffic network control

What is MARL?

Multi-Agent Reinforcement Learning (MARL):

⋆ A sequential decision problem

⋆ A group of agents, each maximizing its own long-term reward

⋆ Agents interacts with each other in a common environment.

Applications:

Autonomous driving, robotics control, fleet control, E-sports.

Key Elements MARL

Typical MARL Problem:

- Number of agents N
- State space: $\mathcal{S}^N$ (product of each agent's state space)
- Action space: $\mathcal{A}^N$ (product of each agent's action space)
- Reward for each agent: $\{r_i(s^N, a^N)\}_{i=1}^N$
- Discount factor: $\gamma \in (0, 1)$
- Transition kernel: $\mathbb{P}\left\{(s^N)'|s^N, a^N\right\}$

Three Central Quantities

- ⋆ **Policy**: $\{\pi_i\}_{i=1}^N$: each agent use $\pi_i(a_i|s^N)$ to decide action.
- ⋆ **Value Functions**:

$$V_i^{\overbrace{\pi_1,\ldots,\pi_N}^{\text{joint policy } \pi}}(s^N) = \mathbb{E}\left\{\sum_{t=0}^{\infty} \gamma^t r_i(s_t^N, a_t^N) | s_0^N = s^N\right\}$$

Expected reward given starting state and follow joint policy π

- ⋆ **Q-functions**:

$$Q_i^{\overbrace{\pi_1,\ldots,\pi_N}^{\text{joint policy } \pi}}(s^N, a^N) = \mathbb{E}\left\{\sum_{t=0}^{\infty} \gamma^t r_i(s_t^N, a_t^N) | s_0^N = s^N, a_0^N = a^N\right\}$$

Expected reward given starting state-action pair and follow joint policy π

What's so hard about MARL

Self-interest Agents: Each agent maximizes its long-term discounted reward.

$$V_i^{\overbrace{\pi_1, \ldots, \pi_N}^{\text{joint policy } \pi}}(s^N) = \mathbb{E}\left\{\sum_{t=0}^{\infty} \gamma^t r_i(s_t^N, a_t^N) | s_0^N = s^N\right\}$$

⋆ *A multi-player game instead of single-objective optimization*

⋆ *Cooperative if r_i's are the same*

Interaction Between Agents:

- reward: $r_i(s^N, a_i, a_{-i}) \neq r_i(s^N, a_i, a'_{-i})$.
- transition: $\mathbb{P}(s'_i | s^N, a_i, a_{-i}) \neq \mathbb{P}(s'_i | s^N, a_i, a'_{-i})$.

⋆ *Actions of agents affect rewards/dynamics of each other*

Mainstream Approach

$\star$ Policy Gradients $\star$

$$\frac{\partial}{\partial \theta_i} V_i(s^N) = \mathbb{E}_{s^N \sim \rho^\pi, a^N \sim \pi} \left\{ \frac{\partial}{\partial \theta_i} \log \pi_{\theta_i}(s^N, a^N) Q_i^\pi(s^N, a^N) \right\}$$

Each agent can optimize its policy by evaluating expression above with one stochastic sample

State-action value function Q_i^π is crucial in shaping learned policies!

Our Tasks

Given the central role of Q/V functions, we focus on

Part I:

Efficient evaluation of Q/V, for decentralized learning setting.

Part II:

Efficient learning of Q, for many-agent setting.

Part II:

Robust learning of Q, via adversarial regularization.

Part I: Efficient Evaluation

Cooperative MARL: Agents jointly maximize their team reward $r(s^N, a^N) = \sum_{i=1}^{N} r_i(s^N, a^N)$

Local Reward: Local reward r_i are only directly accessible by agent i.

Motivations: privacy concerns, communication constraints

Evaluation Target: value function (same reasoning applies to Q-function)

$$V^{\overbrace{\pi_1, \ldots, \pi_N}^{\text{joint policy } \pi}}(s^N) = \mathbb{E}\left\{\sum_{t=0}^{\infty} \gamma^t r(s_t^N, a_t^N) | s_0^N = s^N\right\}$$

How can we obtain the value function without sharing the reward ?

Formulation of Evaluation

Linear Function Approximation: given feature mapping $X(\cdot)$,

$$V_\theta(s^N) = \theta^\top X(s^N)$$

Bellman Evaluation Operator:

$$V^\pi(s^N) = \mathbb{E}_{a^N \sim \pi, (s^N)' \sim \mathbb{P}} \left\{ \sum_{i=1}^N r_i(s^N, a^N) + \gamma \mathbb{P}\left\{(s^N)' | s^N, a^N\right\} \right\}$$

Reformulation(centralized):

$$\min_\theta \sum_{i=1}^N \|A\theta - b_i\|^2$$

where

$$A = \mathbb{E}_{s,s' \sim \rho^\pi} X(s)(X(s) - \gamma X(s'))^\top$$

$$b_i = \mathbb{E}_{s \sim \rho^\pi} X(s) r_i(s), \quad \textit{this is only locally accessible}$$

Remark: both A, b_i are unknown except through sample.

Decentralized Formulation

Suppose we are given a communication graph G, where two agents (i, j) are able to communicate with each other if $(i, j) \in \mathcal{E}(G)$.

$$\min_{\theta} \sum_{i=1}^{N} \|A\theta_i - b_i\|^2 , \quad \text{s.t. } \mathbf{L}\Theta = 0$$

where $\mathbf{L} = L_G \otimes I_d$, and L_G is the graph laplacian of G;
$\Theta = (\theta_1, \ldots, \theta_N)$.

$$L_G(i, j) = \begin{cases} 0, & (i, j) \notin \mathcal{E}(G) \\ -1, & (i, j) \in \mathcal{E}(G) \\ deg(i), & i = j \end{cases}$$

Nice Features:

- Linear constraint: equivalent to $\theta_1 = \cdots = \theta_N$
- Objective is both smooth and strongly convex

Decentralized Algorithm

Algorithm 1 Dual Sliding Algorithm

Input: $\{\delta^t\}_{t=0}^{T-1}$
Initialize: $\Lambda^0 = \mathbf{0}$
for $t = 0, \ldots, T-1$ **do**
  Communication: for $i \in [N]$, $z_i^t = \sum_{j \in \mathcal{E}(i)} L_{ij} \lambda_j^t$
  Update primal variables: for $i \in [N]$, $\widehat{\theta}_i^t = \mathrm{GS}(z_i^t, b_i, A, \delta^t)$
  Communication: for $i \in [N]$, $y_i^t = \sum_{j \in \mathcal{E}(i)} L_{ij} \widehat{\theta}_j^t$
  Update dual variables: for $i \in [N]$, $\lambda_i^{t+1} = \lambda_i^t + \eta y_i^t$
end for

where $\mathrm{GS}(\cdot)$ procedure is approximately solving

$$\min_{\theta_i} \left\langle z_i^t, \theta_i \right\rangle + \|A\theta_i - b_i\|^2$$

No reward needs to be shared!

Optimal Efficiency

$\star$ Number of sample needed: $\mathcal{O}\left\{\frac{1}{\epsilon}\log(\frac{1}{\epsilon})\right\}$ - *nearly optimal*

$\star$ Number of communications needed: $\mathcal{O}(\log(\frac{1}{\epsilon}))$ - *optimal*

Communication is crucial - consumes majority computing time

Intuition:

- one-round of communication after one-round of $\mathrm{GS}(\cdot)$ procedure.
- each $\mathrm{GS}(\cdot)$ procedure takes $\mathcal{O}(\frac{1}{\epsilon})$ number of samples.

First algorithm achieving (nearly) both optimal sample complexity and communication complexity

Part II: Efficient Learning

Motivation: policy gradient relies on an *accurate* estimate of Q-function.

$$\frac{\partial}{\partial \theta_i} V_i(s^N) = \mathbb{E}_{s^N \sim \rho^\pi, a^N \sim \pi} \left\{ \frac{\partial}{\partial \theta_i} \log \pi_{\theta_i}(s^N, a^N) Q_i^\pi(s^N, a^N) \right\}$$

♠ *Curse of Many Agents* ♠

s^N and a^N are concatenation of local states and actions. Learning $Q_i^\pi(s^N, a^N)$ is hard, the search space grows *exponentially* with respect to number of agents.

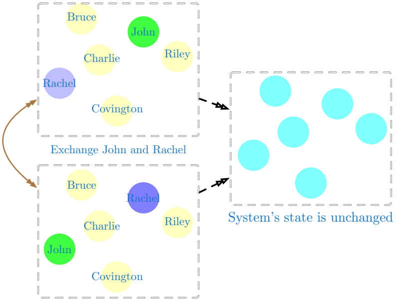
How large a search space exactly?

$$\Theta \left\{ (|\mathcal{S}||\mathcal{A}|)^N \right\}$$

Cooperative MARL with Homogenous System

A MARL system where:

- agents share the same reward
- agents are interchangeable



Examples:
control of unmanned aerial fleet;
tax design on large population.

Formal Description:

$$r(s^N, a^N) = r(\sigma(s^N), \sigma(a^N))$$

$$\mathbb{P}\left\{(s^N)'|s^N, a^N\right\} = \mathbb{P}\left\{\sigma((s^N)')|\sigma(s^N), \sigma(a^N)\right\}$$

In short: *Only the configuration of the system matters.*

Exploiting Homogeneity

Key Observation: *The optimal Q-function and its induced policy is permutation-invariant.*

$$Q^*(s^N, a^N) = Q(\sigma(s^N), \sigma(a^N))$$
$$\pi^*(s^N, a^N) = \pi^*(\sigma(s^N), \sigma(a^N))$$

Direct Implication:

♠ *It suffices to search within the class of permutation-invariant Q-function and policy* ♠

How large is the search space ?

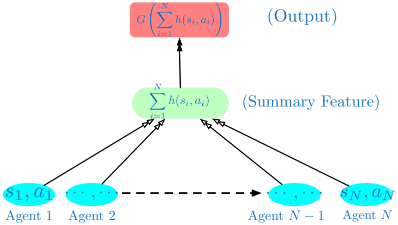
$$\mathcal{O}\left\{\min\{(|\mathcal{S}||\mathcal{A}|)^N, N^{|\mathcal{S}||\mathcal{A}|}\}\right\}$$

Search space can be exponentially smaller!

Permutation-invariant Network

DeepSets: A simple yet effective architecture to induce permutation invariance

$$F(x^N) = G\left(\sum_{i=1}^{N} h(x_i)\right)$$



Interpretation:

- ⋆ Compute local features by $h(\cdot)$
- ⋆ Aggregate local features before feeding into the function $G(\cdot)$ that operates on global information

Experiment Setup

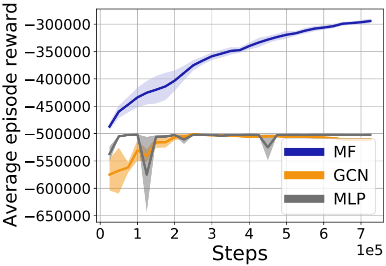
Multi-agent Particle Environment:

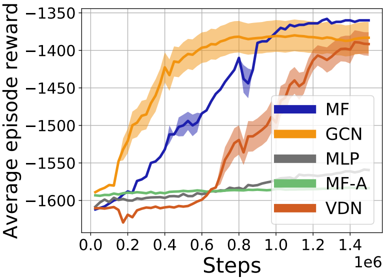
- *Cooperative navigation*: N agents move to cover N fixed landmarks.
- *Cooperative push*: N agents work together to push a ball to a fixed landmark.
- Both tasks reward agents based on distance to landmarks.

Architecture: three-layer Deepsets for to parameterize actor and critic network.

Training Framework: centralized training, decentralized execution (one variant of policy gradient).

Experiment Results





- (Left): Cooperative navigation with $N = 200$ agents.
- (Right): Cooperative push with $N = 15$ agents.

*Proposed method (**MF**) clearly outperforms alternatives!*

Part III: Robust Learning for Traffic Control

- *Task*: let the traffic flows through the network *smoothly*
- *Control Power*: ability to control the traffic light at each intersection

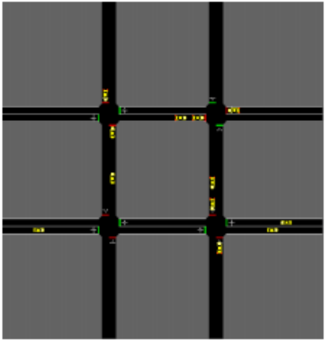


Figure: 2×2-grid traffic network in SUMO simulator

Traffic Control Setup

Key Elements:

- $\star$ *Individual state space $\mathcal{S}$*: incoming queue length, number of vehicles on incoming lanes, average waiting time, average delay, current light phase, duration since last phase change
- $\star$ *Joint state space $\mathcal{S}^N$*: $\prod_{i=1}^{N} \mathcal{S}$
- $\star$ *Individual action space $\mathcal{A}$*: switch traffic light phase
- $\star$ *Individual action space $\mathcal{A}^N$*: $\prod_{i=1}^{N} \mathcal{A}$
- $\star$ *Individual reward $r_i(s_i)$*: linear combination of state features

Cooperation is Necessary

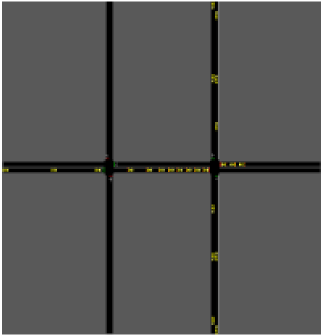


Figure: Two traffic lights with inbalanced flow

♠ *Selfishness leads to bad equilibrium* ♠

♠ *Cooperative MARL* ♠

♢ optimize global reward $r(s)$: $r_n = \sum_{n=1}^{N} k_n r_n$. ♢

Efficient Decomposition of Q-function

Local (individual) Learning:

$$L_n(\theta_n) = \mathbb{E}_\pi \left\{ y_n^t - Q_n(s_n^t, a_n^t) \right\}^2 ,$$
$$y_n^t = r_n^t + \gamma \max_{(a_n^t)'} Q_n(s_n^t, (a_n^t)')$$

Learning Global Q:

$$L_{global}(w) = \sum_{n=1}^{N} \mathbb{E}_\pi \left\{ y^t - Q_w^\pi(s^t, a^t) \right\}^2 ,$$
$$y^t = r^t + \gamma Q_w^\pi(s^t, (a_1^t)', \ldots, (a_N^t)')$$

Consistency Regularization:

$$L_{reg}(\theta, w, k) = \mathbb{E}_\pi \left\{ Q^\pi(s^n, a^n) - \sum_{n=1}^{N} k_n Q(s_n, a_n) \right\}^2$$

♠ *Why do we say efficient?* ♠

Maximization over $\mathcal{A}^N$ reduces to $\mathcal{A}$!

Robust Learning

Why Robust Policy?

Small traffic jam; weather condition; road construction; transfer from simulation to real traffic network

$\diamondsuit$ *Deployed environment might deviate from training* $\diamondsuit$

Intuition for Learning Robust Policy:

Q-function changes smoothly when varying the states s^N

Adversarial Regularization:

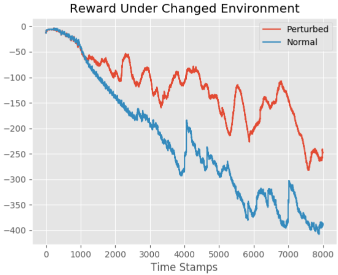
$$R(\theta_n) = \mathbb{E}_\pi \max_{\|\delta_n^t\| \leq \epsilon} \left\{ Q(s_n^t, a_n^t) - Q(s_n^t + \delta_n^t, a_n^t) \right\}^2$$

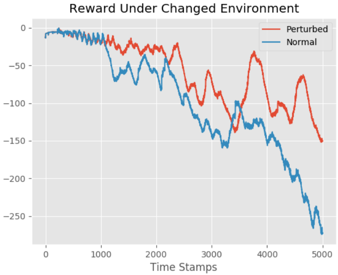
$\spadesuit$ *Finding the worse-case traffic pattern change within a perturbation set* $\spadesuit$

Preliminary Experiments

- ⋆ Environment (data): SUMO simulator, 6-by-6 grid network.
- ⋆ **Optimizing ...**

$$L_{global}(w) + L_{reg}(\theta, w, k) + \sum_{n=1}^{N} L_n(\theta_n) + R(\theta_n)$$





♠ *regularized training shows significant improvement!* ♠

Summary

What we have addressed so far:

- ⋆ *Provably Efficient*: Optimal algorithms for decentralized policy evaluation
- ⋆ *Scaling Up*: Handling curse of many agents via permutation invariance
- ⋆ *Robust Learning*: Learning robust traffic control policy by adversarial regularization

Thank you!