The Evolution of Reinforcement Learning: From Single-Agent to Multi-Agent Paradigms

Networked Intelligence for Comprehensive Efficiency (NICE) Lab
College of Information Science and Electronic Engineering
Zhejiang University
http://nice.rongpeng.info/

Sep. 1, 2025

Content

Reinforcement Learning

Markov Decision Process (MDP)

Figure: State $\mathcal{S}$

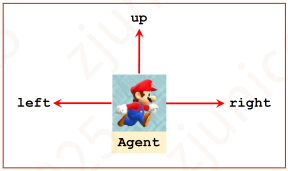


Figure: Action $\mathcal{A}$

- Randomness
  - Action randomness.
  - State Transition randomness.
  - Reward randomness
    $R_t = r(S_t, A_t)$.

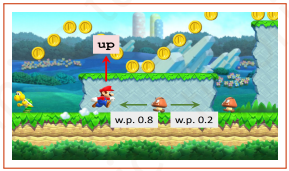


Figure: State Transition $\mathcal{P}$

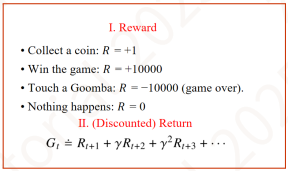
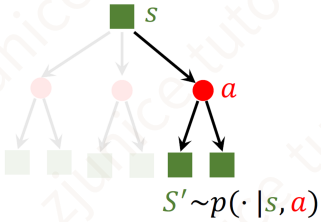
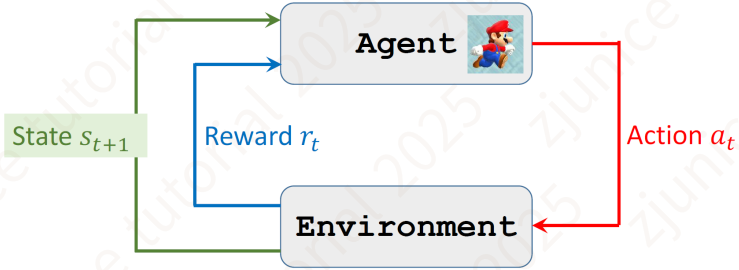


Figure: Reward $\mathcal{R}$



Policy π

- Deterministic Policy $a_t = \mu(s_t)$.
- Stochastic Policy $a_t \sim \pi(\cdot \mid s_t)$.
  - Discrete Action
    - $\pi(\text{left} \mid s_t) = 0.2, \pi(\text{right} \mid s_t) = 0.1, \pi(\text{up} \mid s_t) = 0.7$.
  - Continuous Action
    - $a_t \sim \mathcal{N}(\mu(s_t), \sigma^2)$.
    - $a_t \sim \mathcal{N}(\mu(s_t), \sigma^2(s_t))$.



Value Functions

- Action-Value Function $\mathcal{Q}$
  - $Q_\pi(s_t, a_t) = \mathbb{E}\left[G_t \mid S_t = s_t, A_t = a_t\right]$.
  - Given policy π, $Q_\pi(s_t, a_t)$ evaluates how good it is for an agent to pick action a_t in state s_t.
- Optimal Action-Value Function $\mathcal{Q}*$
  - $Q^\star(s_t, a_t) = \max_\pi Q_\pi(s_t, a_t)$.
  - The optimal policy is the policy that maximizes the Action-Value function.
- State-Value Function $\mathcal{V}$
  -

$$
\begin{aligned}
V_\pi(s_t) &= \mathbb{E}_\pi\left[G_t \mid S_t = s_t\right] \\
&= \sum_{a_t \in \mathcal{A}} \pi(a_t \mid s_t)\, \mathbb{E}_\pi\left[G_t \mid S_t = s_t, A_t = a_t\right] \\
&= \sum_{a_t \in \mathcal{A}} \pi(a_t \mid s_t)\, Q_\pi(s_t, a_t).
\end{aligned}
$$

  - For fixed policy π, $V(s_t)$ evaluates how good the situation is in state s_t.

Bellman Equation

- It defines the relationship between the current and future states, indicating that the value function at the current state can be computed recursively from the value function of the next state.

$$
\begin{aligned}
V(s_t) &= \mathbb{E}[G_t \mid S_t = s_t] \\
&= \mathbb{E}\left[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots \,\middle|\, S_t = s_t\right] \\
&= \mathbb{E}\left[R_t + \gamma\left(R_{t+1} + \gamma R_{t+2} + \cdots\right) \,\middle|\, S_t = s_t\right] \\
&= \mathbb{E}[R_t + \gamma G_{t+1} \mid S_t = s_t] \\
&= \mathbb{E}[R_t \mid S_t = s_t] + \gamma\, \mathbb{E}[G_{t+1} \mid S_t = s_t] \\
&= r(s_t) + \gamma \sum_{s_{t+1} \in \mathcal{S}} P(s_{t+1} \mid s_t)\, \mathbb{E}[G_{t+1} \mid S_{t+1} = s_{t+1}] \\
&= r(s_t) + \gamma \sum_{s_{t+1} \in \mathcal{S}} P(s_{t+1} \mid s_t)\, V(s_{t+1})\,.
\end{aligned}
$$

Classification of Reinforcement Learning Approaches

- Model-Based vs. Model-Free RL
  - Model-Based RL
    - Model the environment (e.g., the state-transition dynamics and the reward function).
  - Model-Free RL
    - No explicit environment model is required; the agent learns an optimal policy through direct interaction with the real environment.
- On-Policy vs. Off-Policy RL
  - On-Policy RL
    - The agent learns from its current policy and updates the policy based on data collected from that same policy.
    - behavior policy = target policy
  - Off-Policy RL
    - The agent can learn from a policy that diverges from its current policy.
    - behavior policy ≠ target policy

Classification of Reinforcement Learning Approaches

- Policy-Based vs. Value-Based RL
  - Policy-Based RL
    - Learning Target : policy function.
    - Directly optimize the policy to maximize the expected cumulative reward.
    - Limitation : **Low sample efficiency**.
  - Value-Based RL
    - Learning Target : value function.
    - Learn the policy implicitly by deriving it from the value function.
    - Limitation : Typically restricted to **discrete action spaces** and plagued by **instability**.

Value-Based RL - Value Function Estimation

- Evaluating the value function requires computing an expectation ($V_\pi(s_t) = \mathbb{E}_\pi\left[G_t \mid S_t = s_t\right]$), which is data-hungry and computationally expensive; therefore we estimate the value function via **sampling-based approximations**.
- Monte Carlo Sampling (MC)
  - $V(s_t) \leftarrow V(s_t) + \alpha\big(G_t - V(s_t)\big)$.
- Temporal-Difference Sampling (TD)
  - $V(s_t) \leftarrow V(s_t) + \alpha\big(r_t + \gamma V(s_{t+1}) - V(s_t)\big)$.

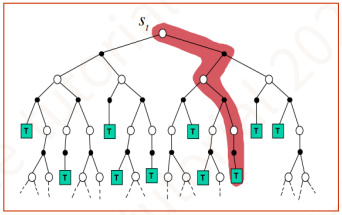


Figure: MC.

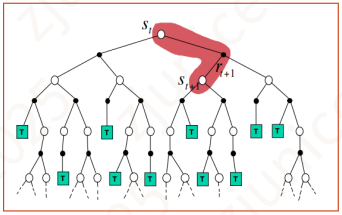


Figure: TD.

Value-Based RL - SARSA[1]

- Step 1. Interact with the environment using an ε-greedy policy to collect trajectories $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$.
  - ε-greedy policy:

$$a_t = \begin{cases} \arg\max_a \tilde{Q}(s_t, a), & \text{with probability } (1-\varepsilon), \\ \text{a uniformly random action from } \mathcal{A}, & \text{with probability } \varepsilon. \end{cases}$$

- Step 2. Update the $Q(s_t, a_t)$ with temporal-difference (TD) learning:
  $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \Big[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \Big]$.
- Step 3. Repeat Steps 1–2 until convergence.
- Target. Action-Value Function $\mathcal{Q}$.

[1] G. A. Rummery and M. Niranjan, *On-line Q-learning using connectionist systems*. University of Cambridge, Department of Engineering Cambridge, UK, 1994, vol. 37.

Value-Based RL - Q-Learning[2]

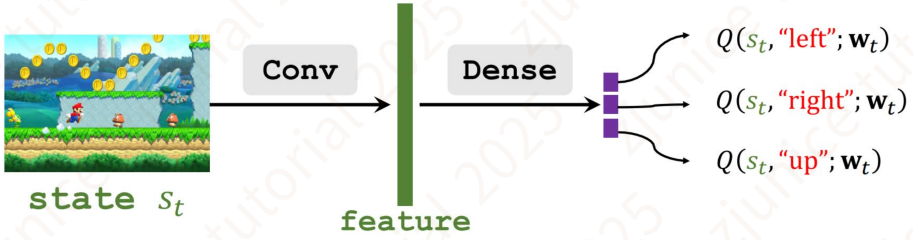
- Step 1. Interact with the environment using an ε-greedy policy to collect trajectories $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$.
- Step 2. Update the $Q(s_t, a_t)$ with temporal-difference (TD) learning:
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right].$$
- Step 3. Repeat Steps 1–2 until convergence.
- Target. Optimal Action-Value Function $\mathcal{Q}*$.

	第 1 种动作	第 2 种动作	第 3 种动作	第 4 种动作
第 1 种状态	380	-95	20	173
第 2 种状态	-7	64	-195	210
第 3 种状态	152	72	413	-80

[2] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3, pp. 279–292, 1992 .

Value-Based RL - Deep Q-Networks[3]

- To approximate the optimal action-value function Q^*, the most effective approach is to employ **a deep neural network**, denoted as $Q(s, a; \mathbf{w})$.
  - Step 1. Observe a transition (s_t, a_t, r_t, s_{t+1}).
  - Step 2. TD target: $y_t = r_t + \gamma \cdot \max_a Q(s_{t+1}, a; \mathbf{w})$.
  - Step 3. TD error: $\delta_t = Q(s_t, a_t; \mathbf{w}) - y_t$.
  - Step 4. Update: $L(\mathbf{w}) \triangleq \frac{1}{2}\Big(Q(s_t, a_t; \mathbf{w}) - y_t\Big)^2$, $\mathbf{w} \leftarrow \mathbf{w} - \alpha \cdot \delta_t \cdot \frac{\partial Q(s_t, a_t; \mathbf{w})}{\partial \mathbf{w}}$.



[3] V. Mnih et al., “Playing atari with deep reinforcement learning,” *arXiv preprint arXiv:1312.5602*, 2013 .

Value-Based RL - Overestimation[4]

- Maximization Bias.

$$y_t = r_t + \gamma \cdot \underbrace{\max_a Q(s_{t+1}, a; \mathbf{w})}_{\text{estimation made by DQN}} .$$

- Error Propagation through Bootstrapping.

$$L(\mathbf{w}) = \frac{1}{2} \underbrace{\left(Q(s_t, a_t; \mathbf{w}) - y_t\right)^2}_{\text{forcing DQN to fit } y_t} .$$

- Target Network.
  - Select Action: $a^- = \arg\max_a Q(s_{t+1}, a; \mathbf{w}^-)$,
  - TD Target: $y_t^- = r_t + Q(s_{t+1}, a^-; \mathbf{w}^-)$.
- Double DQN Network.
  - Select Action: $a^* = \arg\max_a Q(s_{t+1}, a; \mathbf{w})$,
  - TD Target: $\hat{y}_t = r_t + Q(s_{t+1}, a^*; \mathbf{w}^-)$.

	选择	求值	自举造成偏差	最大化造成高估
原始 Q 学习	DQN	DQN	严重	严重
Q 学习 + 目标网络	目标网络	目标网络	不严重	严重
双 Q 学习	DQN	目标网络	不严重	不严重

[4] V. Mnih et al., "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015 .

Policy-Based RL - Policy Gradient

- Policy-learning objective: maximize the value function aggregated over all states.

Objective $J(\boldsymbol{\theta}) = \mathbb{E}_S\big[V_\pi(S)\big]$. **Maximize** $\max_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$. **Gradient ascent** $\boldsymbol{\theta}_{\text{new}} \leftarrow \boldsymbol{\theta}_{\text{now}} + \beta \cdot \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_{\text{now}})$.

Policy Gradient Theorem

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \mathbb{E}_S\left[\mathbb{E}_{A\sim\pi(\cdot|S;\boldsymbol{\theta})}\left[\frac{\partial \ln \pi(A|S;\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \cdot Q_\pi(S,A)\right]\right]$$

Sample → **Unbiased estimate**

$$\boldsymbol{g}(s,a;\boldsymbol{\theta}) \triangleq Q_\pi(s,a) \cdot \nabla_{\boldsymbol{\theta}} \ln \pi(a\,|\,s;\boldsymbol{\theta}).$$

$$\begin{aligned}
\nabla J(\theta) &= \sum_{s\in\mathcal{S}} d(s) \sum_{a\in\mathcal{A}} \nabla\pi(a\mid s;\theta) Q_\pi(s,a) \\
&= \sum_{s\in\mathcal{S}} d(s) \sum_{a\in\mathcal{A}} \pi(a\mid s;\theta) \frac{\nabla\pi(a\mid s;\theta)}{\pi(a\mid s;\theta)} Q_\pi(s,a) \\
&= \sum_{s\in\mathcal{S}} d(s) \sum_{a\in\mathcal{A}} \pi(a\mid s;\theta) \nabla \ln \pi(a\mid s;\theta) Q_\pi(s,a) \\
&= \mathbb{E}_{\pi\theta}\left[\nabla \ln \pi(a\mid s,\theta) Q_\pi(s,a)\right]
\end{aligned}$$

← Apply the derivative identity.

Figure: Derivation of the Policy Gradient Theorem. $d_{\pi_\theta}(s)$ denotes the probability under policy π_θ, of reaching state s from the initial state s_0 in the stationary distribution of the MDP.

Back Link to DPG

Policy-Based RL - REINFORCE[5]

- REINFORCE can be viewed as estimating the value function via Monte Carlo (MC) returns and then improving the policy using the policy gradient (PG).

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \mathbb{E}_S\left[\mathbb{E}_{A\sim\pi(\cdot|S;\boldsymbol{\theta})}\left[\frac{\partial \ln \pi(A|S;\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \cdot Q_\pi(S,A)\right]\right].$$

$$\boldsymbol{g}(s,a;\boldsymbol{\theta}) \triangleq Q_\pi(s,a) \cdot \nabla_{\boldsymbol{\theta}} \ln \pi(a\,|\,s;\boldsymbol{\theta}).$$

$$u_t = \sum_{k=t}^{n} \gamma^{k-t} \cdot r_k$$

$$\tilde{\boldsymbol{g}}(s_t,a_t;\boldsymbol{\theta}) = u_t \cdot \nabla_{\boldsymbol{\theta}} \ln \pi(a_t\,|\,s_t;\boldsymbol{\theta}).$$

Sample

Unbiased estimate

$$Q_\pi(s_t,a_t) = \mathbb{E}\Big[U_t \,\Big|\, S_t = s_t, A_t = a_t\Big].$$

[5] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3, pp. 229–256, 1992.

Policy-Based RL - Actor-Critic[6]

- Update the policy network (actor) by policy gradient.
- Update the value network (critic) by TD learning.
  - $y_t \triangleq r_t + \gamma \cdot q(s_{t+1}, a_{t+1}; \mathbf{w})$
  - $L(\mathbf{w}) \triangleq \frac{1}{2}\Big(q(s_t, a_t; \mathbf{w}) - y_t\Big)^2$
  - $\mathbf{w} \leftarrow \mathbf{w} - \alpha \cdot \nabla_{\mathbf{w}} L(\mathbf{w})$.

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \mathbb{E}_S\left[\mathbb{E}_{A\sim\pi(\cdot|S;\boldsymbol{\theta})}\left[\frac{\partial \ln \pi(A|S;\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \cdot Q_\pi(S, A)\right]\right].$$

$$\boldsymbol{g}(s, a; \boldsymbol{\theta}) \triangleq Q_\pi(s, a) \cdot \nabla_{\boldsymbol{\theta}} \ln \pi(a \,|\, s; \boldsymbol{\theta}).$$

$$\widehat{\boldsymbol{g}}(s, a; \boldsymbol{\theta}) \triangleq q(s, a; \boldsymbol{w}) \cdot \nabla_{\boldsymbol{\theta}} \ln \pi(a \,|\, s; \boldsymbol{\theta}).$$

Directly use the network to approximate the value function.

$$\boldsymbol{\theta}_{\text{new}} \leftarrow \boldsymbol{\theta}_{\text{now}} + \beta \cdot \widehat{q}_t \cdot \nabla_{\boldsymbol{\theta}} \ln \pi(a_t \,|\, s_t; \boldsymbol{\theta}_{\text{now}}).$$

[6] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," *Advances in neural information processing systems*, vol. 12, 1999 .

Policy-Based RL - Sample Efficiency

- Problem. Computing policy gradients requires collecting a large amount of on-policy data for policy evaluation and gradient estimation. After each policy update, previously collected data can no longer be reused, resulting in very **low sample efficiency**.
- Proposed solution. Use data sampled by interacting with the environment under another (behavior) policy to update the target policy π.
- Importance sampling.
  - $\mathbb{E}_{x\sim p}[f(x)] \approx \frac{1}{N}\sum_{i=1}^{N} f(x^i)$
  - $\int f(x)\, p(x)\, dx = \int f(x)\, \frac{p(x)}{q(x)}\, q(x)\, dx = \mathbb{E}_{x\sim q}\left[f(x)\, \frac{p(x)}{q(x)} \right]$
  - $\mathbb{E}_{x\sim p}[f(x)] = \mathbb{E}_{x\sim q}\left[f(x)\, \frac{p(x)}{q(x)} \right].$

Policy-Based RL - Trust Region Policy Optimization(TRPO)[7]

- Problem. When the discrepancy between distributions p and q is large, limited sampling leads to substantially different variances of estimators under the two distributions.
- TRPO principle. The trust region is implemented by a KL-divergence constraint, ensuring the new policy does not deviate too far from the old one.
- Limitation of TRPO. Although theoretically sound, handling the KL-divergence constraint in real systems is computationally cumbersome and often impractical.

$$\begin{aligned} \max_{\theta} \quad & \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right] \\ \text{s.t.} \quad & \hat{\mathbb{E}}_t\left[\mathrm{KL}\left(\pi_{\theta_{\text{old}}}(\cdot \mid s_t) \,\middle\|\, \pi_\theta(\cdot \mid s_t)\right)\right] \le \delta. \end{aligned}$$

or

$$\max_{\theta} \ \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t - \beta\,\mathrm{KL}\left(\pi_{\theta_{\text{old}}}(\cdot \mid s_t) \,\middle\|\, \pi_\theta(\cdot \mid s_t)\right)\right].$$

[7] J. Schulman et al., "Trust region policy optimization," in *International conference on machine learning*, PMLR, 2015, pp. 1889–1897.

Policy-Based RL - Trust Region Policy Optimization(TRPO)[8]

$$\boldsymbol{g}_b(s,a;\boldsymbol{\theta}) \;=\; \Big[\,Q_\pi(s,a) \;-\; b\,\Big] \;\cdot\; \nabla_{\boldsymbol{\theta}} \ln \pi(a \,|\, s;\boldsymbol{\theta}).$$

Figure: Policy Gradient Theorem with Baseline.

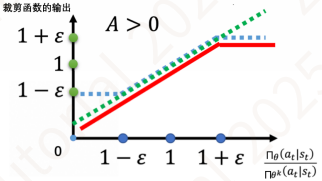
- Loss function.

$$\begin{aligned}
\nabla J(\theta) &= \mathbb{E}_{\pi_\theta}\{\, \nabla_\theta \ln \pi(a \mid s;\theta)\, \big[Q_\theta(s,a) - V(s)\,\big]\,\} \\
&= \mathbb{E}_{\pi_\theta}\big[\, \nabla_\theta \ln \pi(a \mid s;\theta)\, A_\theta(s,a)\,\big] \\
&= \mathbb{E}_{\pi_{\theta_{\text{old}}}}\Big[\, \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \nabla_\theta \ln \pi(a \mid s;\theta)\, A_{\theta_{\text{old}}}(s,a)\,\Big] \\
&= \mathbb{E}_{\pi_{\theta_{\text{old}}}}\Big[\, \frac{1}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \nabla_\theta \pi(a \mid s;\theta)\, A_{\theta_{\text{old}}}(s,a)\,\Big].
\end{aligned}$$

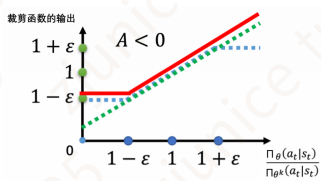
[8] J. Schulman et al., "Trust region policy optimization," in *International conference on machine learning*, PMLR, 2015, pp. 1889–1897.

Policy-Based RL - PPO[9]

- PPO. $\mathcal{J}(\theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} A_{\pi_{\theta_{\text{old}}}}(s,a),\ \text{clip}\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)}, 1-\epsilon, 1+\epsilon\right) A_{\pi_{\theta_{\text{old}}}}(s,a)\right)$.
- Refining the KL constraint. When $A_t > 0$ it prevents importance ratio from increasing too much, and when $A_t < 0$ it prevents importance ratio from decreasing too much.



(a) $A > 0$



(b) $A < 0$

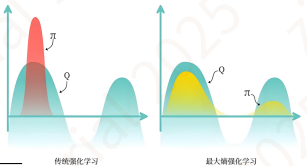
- Advantages. By shaping the objective directly **with clipping**, PPO constrains the deviation between the new and old policies in a simple and efficient manner.

[9] J. Schulman et al., "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017 .

Policy-Based RL - PPO

- Improved advantage estimation. Generalized Advantage Estimation (GAE) aggregates TD-errors over multiple future steps.
  - $$A_t^{\mathrm{GAE}}(\gamma,\lambda) = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}, \qquad \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t).$$
- SAC[10].
  - Maximum Entropy RL.

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\pi} \left[\sum_t r(s_t, a_t) + \alpha H(\pi(\cdot \mid s_t)) \right] \tag{1}$$



[10] T. Haarnoja et al., "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, Pmlr, 2018, pp. 1861–1870.

Policy-Based RL - Deterministic Policy Gradient(DPG)[11]

- **Deterministic Policy.** For a given state s, the policy network μ outputs a deterministic action a. The action is directly the output of the policy, i.e.,$a = \mu(s)$.
  - Step 1. Interact with the environment to collect a batch of transitions (s_t, a_t, r_t, s_{t+1}).
  - Step 2. Update the policy network using the deterministic policy gradient.
    - $\boldsymbol{\theta}^\star = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_S\Big[Q\Big(S, \hat{A}; \mathbf{w}\Big)\Big]$.
    - $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \beta \cdot \nabla_{\boldsymbol{\theta}} \mu(s_t; \boldsymbol{\theta}) \cdot \nabla_a Q(s_t, \hat{a}_t; \mathbf{w})$.
  - Step 3. Update the value network using TD learning:
    - $r_t + \gamma \cdot Q(s_{t+1},\ \mu(s_{t+1};\ \boldsymbol{\theta}^\star);\ \mathbf{w})$.
    - $L(\mathbf{w}) = \frac{1}{2}\Big[Q(s_t, a_t; \mathbf{w}) - \hat{y}_t \Big]^2$.
  - Step 4. Repeat Steps 1–3 until convergence.

▸ Policy Gradient

[11] D. Silver et al., "Deterministic policy gradient algorithms," in *International conference on machine learning*, Pmlr, 2014, pp. 387–395.

Multi-Agent Reinforcement Learning

Setting

Figure: Fully cooperative.

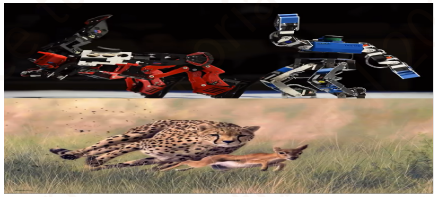


Figure: Fully competitive.

Figure: Mixed cooperative & competitive.

Figure: Self-interested.

Decentralized partially observable Markov Decision Process(Dec-POMDP)

- A Dec-POMDP is a 7-tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, R, \Omega, \mathcal{Z}, \gamma \rangle$:
  - $\mathcal{S}$: Global state space.
  - $\mathcal{A}$: Global action space.
  - $\mathcal{P}(s^{(t+1)} \mid s^{(t)}, a^{(t)})$: State transition probability.
  - R: Global reward.
  - $\Omega(z_i^{(t)} \mid s^{(t)}, i)$: Local state mapping.
  - $\mathcal{Z}$: Local observation space.
  - $\gamma \in [0, 1]$: Discount factor.

IQL[12]

- Step 1. Initialize the Q-function. For each agent i, initialize its action–value function $Q_i(o_t^i, a_t^i)$.
- Step 2. Sample. In each step, every agent independently selects an action a_t^i using an ε-greedy policy, interacts with the environment, and obtains a reward r_t^i.
- Step 3. Q-update. According to the classical Q-learning rule:

$$Q_i(o_t^i, a_t^i) \leftarrow Q_i(o_t^i, a_t^i) + \alpha \left[r_t^i + \gamma \max_{a_{t+1}} Q_i(o_{t+1}^i, a_{t+1}^i) - Q_i(o_t^i, a_t^i) \right].$$

[12] A. Tampuu et al., "Multiagent cooperation and competition with deep reinforcement learning," *PloS one*, vol. 12, no. 4, e0172395, 2017 .

Main Challenges in MARL

- **Partial Observability** ▸ Value Decomposition
  Each agent can only **observe information relevant to itself** rather than the full global state, which increases the complexity of policy learning.
- **Non-stationarity** ▸ Consensus Algorithms
  During joint learning and interaction, updates to one agent's policy **affect the policies of other agents**.
- **Dimensional Explosion** ▸ Decentralized Algorithms
  As the number of agents increases, the complexity of the **global state space and joint action space grows exponentially**, making it harder to find optimal solutions.
- **Credit Assignment**
  In cooperative multi-agent settings where a global return is used, it is necessary to **evaluate each agent's contribution to the joint behavior** and allocate the return appropriately to each agent.

▸ Back to conclusion

Training and Execution Paradigms

- Centralized Training and Centralized Execution (CTCE)
  - Advantage: convenient for solving complex coordination problems.
  - Limitation: the centralized policy must be learned over a joint action space that typically grows **exponentially** with the number of agents, leading to high computational complexity.
- Decentralized Training and Decentralized Execution (DTDE)
  - Advantage: avoids the exponential blow-up of the centralized joint action space, offering scalability and natural applicability to distributed settings.
  - Limitation: training relies only on local views, making **convergence more difficult and prone to local optima**.
- Centralized Training and Decentralized Execution (CTDE)
  - Advantage: combines the strengths of centralized training and decentralized execution.
  - Limitation: the assumption of direct access to global information during training is often **too strong**; for large-scale problems the information is **very high-dimensional**.

VDN[13]

- Step 1. Initialize the Q-function. For each agent i, initialize its action–value function $Q_i(o_t^i, a_t^i)$.
- Step 2. Sample. In each step, every agent independently selects an action a_t^i using an ε-greedy policy, interacts with the environment, and obtains a reward r_t.
- Step 3. Q-update. **Use the joint Q-value as the TD target**:

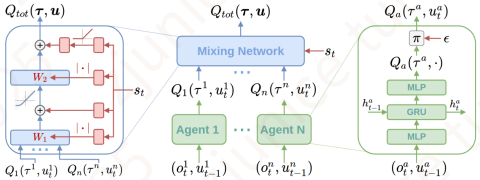
$$y_t = r_t + \gamma \sum_i \max_{a_{t+1}^i} Q_i\big(o_{t+1}^i,\ a_{t+1}^i\big),$$

$$\mathcal{L} = \left(y_t - \sum_i Q_i\big(o_t^i,\ a_t^i\big) \right)^2.$$

- Limitation. Simple linear aggregation of local Q-functions.

[13] P. Sunehag et al., "Value-decomposition networks for cooperative multi-agent learning," *arXiv preprint arXiv:1706.05296*, 2017 .

QMIX[14]



Constraint.

$$\frac{\partial Q_{\text{tot}}}{\partial Q_i} \geq 0, \quad \forall i.$$

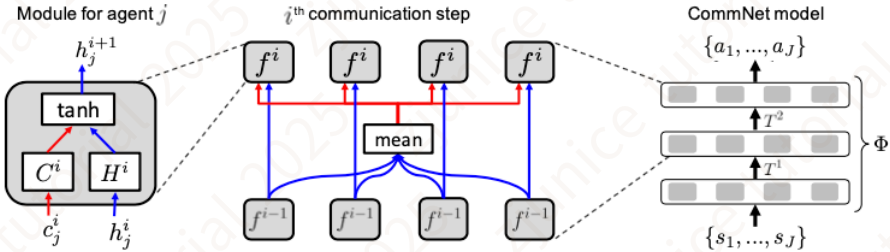
Individual-Global-Max(IGM).

$$\arg\max_{\mathbf{u}} Q_{\text{tot}}(\boldsymbol{\tau}, \mathbf{u}) = \begin{pmatrix} \arg\max_{u_1} Q_1(\tau_1, u_1) \\ \vdots \\ \arg\max_{u_n} Q_n(\tau_n, u_n) \end{pmatrix}.$$

▸ Back to Main Challenges

[14] T. Rashid et al., "Monotonic value function factorisation for deep multi-agent reinforcement learning," *Journal of Machine Learning Research*, vol. 21, no. 178, pp. 1–51, 2020 .

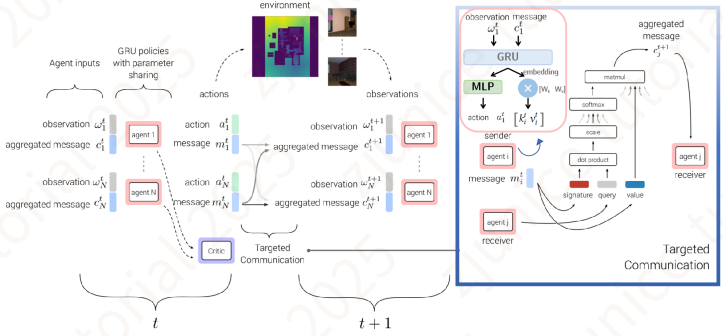
CommNet[15]



- Encoder Function. $h_j^0 = r(s_j)$.
- Decoder Function. $a_j \sim q\left(h_j^K\right)$.
- Model for agent j. $h_j^{i+1} = f^i\left(h_j^i,\ c_j^i\right), c_j^{i+1} = \frac{1}{J-1} \sum_{j' \neq j} h_{j'}^{i+1}$.

[15] S. Sukhbaatar, R. Fergus, et al., "Learning multiagent communication with backpropagation," *Advances in neural information processing systems*, vol. 29, 2016.

TarMAC[16]



- Time t+1: aggregated message $\alpha_j = \text{softmax}\left[\frac{q_j^{t+1\top} k_1^t}{\sqrt{d_k}} \cdots \underbrace{\frac{q_j^{t+1\top} k_i^t}{\sqrt{d_k}}}_{\alpha_{ji}} \cdots \frac{q_j^{t+1\top} k_N^t}{\sqrt{d_k}}\right]$,

$$c_j^{t+1} = \sum_{i=1}^{N} \alpha_{ji}\, v_i^t.$$

[16] A. Das et al., "Tarmac: Targeted multi-agent communication," in *International Conference on machine learning*, PMLR, 2019, pp. 1538–1546.

MASIA[17]

- **Encoder-decoder loss.**

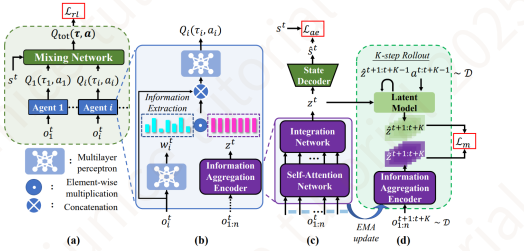
$$\mathcal{L}_{ae}(\theta,\eta)=\sum_{\text{traj}\in\mathcal{B}}\sum_{t=1}^{T}\|g_\eta(z^t)-s^t\|_2^2,\ z^t=f_\theta(\boldsymbol{o}^t).$$

- **Latent model loss.**

$$\mathcal{L}_m(\theta,\psi)=\sum_{\text{traj}\in\mathcal{B}}\sum_{t=1}^{T-K}\sum_{k=1}^{K}\|\hat{z}^{t+k}-z^{t+k}\|_2^2,$$
$$\hat{z}^{k+1}=h_\psi(z^t,\boldsymbol{a}^t),\ \hat{z}^{t+k}=h_\psi(\hat{z}^{t+k-1},\boldsymbol{a}^{t+k-1}),k=2,\dots,K,$$
$$z^{t+k}=f_{\theta^-}(\boldsymbol{o}^{t+k}),k=0,\dots,K,$$

- **RL loss.**

$$\mathcal{L}_{rl}(\theta,\phi)=\sum_{\text{traj}\in\mathcal{B}}\sum_{t=1}^{T-1}\left(r+\gamma\max_{\boldsymbol{a}'}Q_{\text{tot}}(\boldsymbol{\tau}^{t+1},\boldsymbol{a}';\theta^-,\phi^-)-Q_{\text{tot}}(\boldsymbol{\tau}^t,\boldsymbol{a}^t;\theta,\phi)\right)^2.$$



Algorithm 2 Overall Execution Flow

Require: information aggregation encoder with parameters θ, individual Q network with parameters ϕ_i for each Q_i, focusing networks with parameter ω_i for each agent and agent number n.

1: **for** step $=0$ to episode_limit **do**
2: Each agent broadcasts its observational information o_i^t at timestep t, and then each agent feed collected observations $\boldsymbol{o}^t=\{o_i^t\}_n$ into the information aggregation network, obtaining $z^t=f_\theta(\boldsymbol{o}^t)$.
3: **for** $i=1$ to n **do**
4: Agent i calculates $w_i^t=F_{w_i}(o_i^t)$ by using the focusing network, and obtains extracted information $\vec{z}_i^t=w_i^t\cdot z^t$.
5: The o_i^t and $\vec{z}_i^t$ are fed into the individual Q network to calculate $Q_i(\tau_i^t,\cdot)$, and agent i gets action $a_i^t=\arg\max_{a_i}Q_i(\tau_i^t,a)$.
6: **end for**
7: The agent system interacts with the environment by executing actions $\{a_i^t\}_n$.
8: **end for**

Back to Main Challenges

[17] C. Guan et al., "Efficient multi-agent communication via self-supervised information aggregation," *Advances in Neural Information Processing Systems*, vol. 35, pp. 1020–1033, 2022 .

Model-Based Decentralized Policy Optimization[18]

- Network MDP.
  - $\mathcal{G} = (\mathcal{V}, \mathcal{E})$: Undirected graph.
  - $\mathcal{V} = \{1, \ldots, n\}$: Set of agents.
  - $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$: Connectivity of all the agents.
  - $\mathcal{N}_i$: The neighbour of the agent i including itself.
  - $\mathcal{N}_i^K$: The κ-hop neighbours of i.
  - $\mathcal{N}_{-i}^K = \mathcal{V} \setminus \mathcal{N}_i^K$: The set of all nodes except the κ-hop neighbors of i.

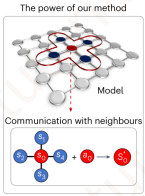


Figure: An illustration of Network MDP.

- Value Function.
  - Use communication within κ-hop neighbours to generate an estimation of the global value function.

$$\tilde{V}_i\left(s_{\mathcal{N}_i^\kappa}^t\right) = \frac{1}{n} \sum_{j \in \mathcal{N}_i^\kappa} V_j\left(s_{\mathcal{N}_j^\kappa}^t\right)$$

Back to Main Challenges

[18] C. Ma et al., "Efficient and scalable reinforcement learning for large-scale network control," *Nature Machine Intelligence*, vol. 6, no. 9, pp. 1006–1020, 2024 .

Conclusion

Conclusion

I have talked about

- RL
  - System model - MDP
  - Value Functions & Bellman Equation
  - Value-based and Policy-Based RL Algorithms
- MARL
  - System model - Dec-POMDP
  - Value Decomposition Algorithms
  - Consensus Algorithms
  - Decentralized Algorithms

Thank you

Networked Intelligence for Comprehensive Efficiency (NICE) Lab
College of Information Science and Electronic Engineering
Zhejiang University
https://nice.rongpeng.info