Multiagent Reinforcement Learning: Rollout and Policy Iteration (Bertsekas, 2020)

Mikalai Korbit

IMT School for Advanced Studies Lucca

February 14, 2021

Outline

MARL Overview

Motivation

Multi-agent systems are ubiquitous

Eg. fleet of drones, factory robots, self-driving cars.

Recent advances in RL applications

Eg. AlphaGo/AlphaZero, playing Starcraft, robotic control.

Utilize modern computer architecture and software frameworks

Eg. cloud computing, stacks of graphics cards, TPUs; PyTorch, OpenAI gyms.

Benefits of modeling a problem as MARL

Scalability, robustness, faster learning through experience sharing, parallel computation.

Multi-Agent Reinforcement Learning Problem

Inherits Reinforcement Learning characteristics:

- Learning how to map situations into actions
- Trial-and-error search
- Delayed feedback
- Trade-off between exploration and exploitation
- Sequential decision making
- Agent's actions affect the subsequent data it receives

Adds multi-agent features:

- Actions of one agent influence other agents' rewards
- Communication problem
- Fully cooperative, fully info sharing (DP) vs. partial info sharing
- Curse of dimensionality (more severe than in RL)

"Bertsekas Dictionary"

Aligns optimal control definitions with the RL-world:

- Maximize value $\rightarrow$ minimize cost
- Agent $\rightarrow$ Decision maker or controller
- Action $\rightarrow$ Decision or control
- Environment $\rightarrow$ Dynamic system
- Learning $\rightarrow$ Solving a DP-related problem using simulation
- Self-learning (self-play) $\rightarrow$ Solving a DP problem using simulation-based policy iteration
- Planning vs Learning distinction $\rightarrow$ Solving a DP problem with model-based vs model-free simulation

Multi-Agent MDP

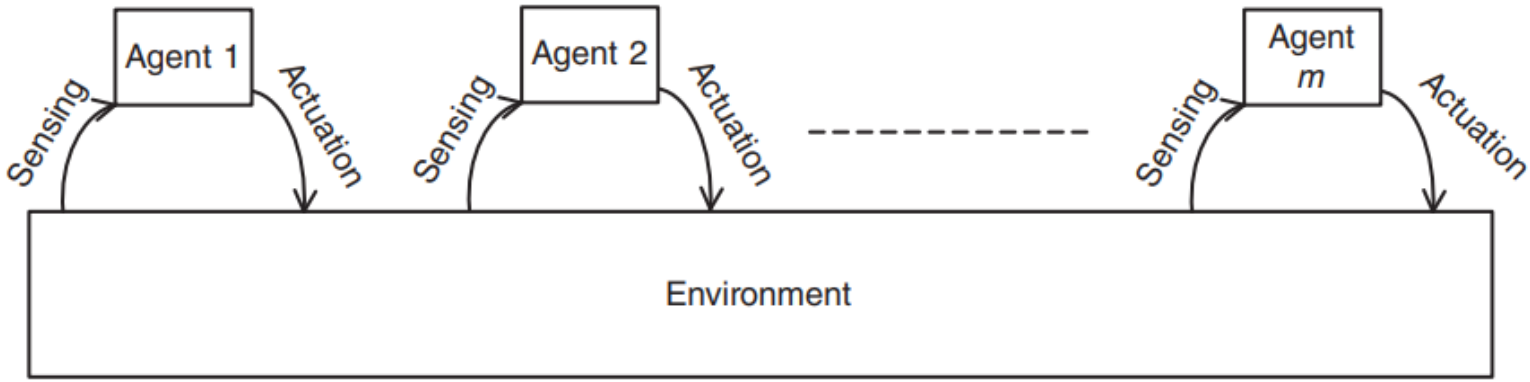

Figure: MARL Problem. Source: Sadhu, Konar (2020)

- All agents see the global state s
- Individual actions: $u^a \in U$
- State transitions: $P(s' \mid s, \mathbf{u}) : S \times \mathbf{U} \times S \to [0, 1]$
- Shared team reward: $S \times \mathbf{U} \to \mathbb{R}$

Taxonomy

Cooperative

- The goal of cooperative agents is to achieve a common objective
- Coordination problem

Competitive

- Zero-sum games (eg. chess, tic-tac-toe)
- Minimax equilibria

Mixed

- General-sum games (win-win, lose-lose scenarios; eg. pollution model, "what movie to watch?")
- Nash equilibria

Algorithms for Cooperative MARL

Static games

- JAL (Joint Action Learners)
- FMQ (Frequency Maximum Q-value)

Dynamic games

- Team-Q
- Distributed-Q
- OAL (Optimal Adaptive Learning)
- SCQL (Sparse Cooperative Q-learning)
- SQL (Sequential Q-learning)
- FMRQ (Frequency of the maximum reward Q-learning)

Multiagent Rollout

Key Ideas

Deal with the exponential increase in the action space

$\rightarrow$ Introduce a form of sequential agent-by-agent one-step lookahead minimization – *multiagent rollout*

Compute the agent actions in parallel

$\rightarrow$ Decouple sequential agent-by-agent computation with *precomputed signaling policy* that embodies agent coordination

The Setting

- $P2_F$ – stochastic discrete-time optimal control problem over a finite horizon, with perfect information on the state
- Fully cooperative
- Tested in Spiders-And-Flies environment

$$x_{k+1} = f_k\left(x_k, u_k, w_k\right), \quad k = 0, 1, \ldots, N-1$$

$$J_\pi\left(x_0\right) = E\left\{g_N\left(x_N\right) + \sum_{k=0}^{N-1} g_k\left(x_k, \mu_k\left(x_k\right), w_k\right)\right\}$$

Policy Iteration and Rollout

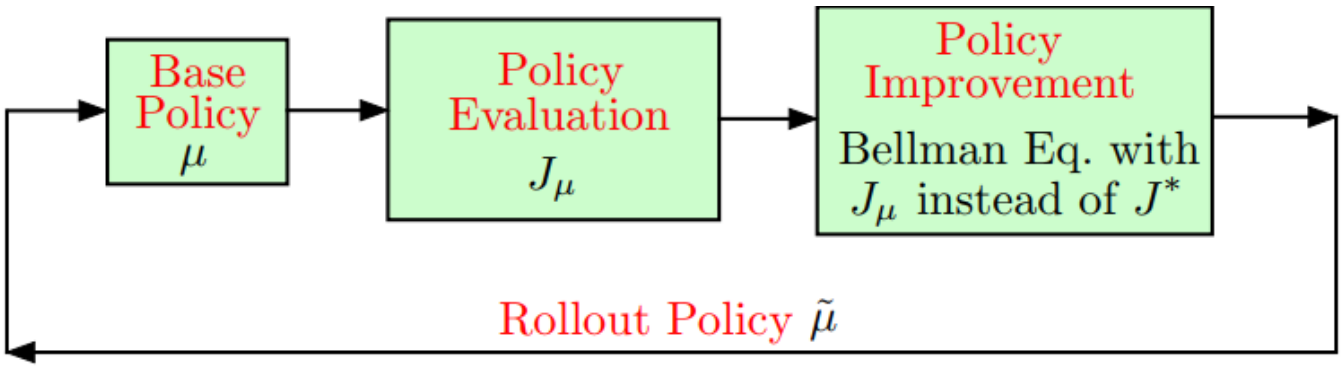

Figure: Policy Iteration Algorithm. Source: Bertsekas (2020)

- Fundamental property: policy improvement

$$J_{k,\tilde{\pi}}(x_k) \leq J_{k,\pi}(x_k), \quad \forall x_k, k$$

All-at-once Rollout

- Rollout is one-time policy iteration

$$\tilde{u}_k \in \arg \min_{u_k \in U_k(x_k)} E\left\{g_k\left(x_k, u_k, w_k\right) + \right.$$
$$\left. + J_{k+1,\pi}\left(f_k\left(x_k, u_k, w_k\right)\right)\right\}$$

- Rollout is an on-line algorithm and possesses robustness property
- Rollout suffers from dimensionality curse problem when the action space is large (eg. exponential in the number of agents)

One-at-a-time Rollout

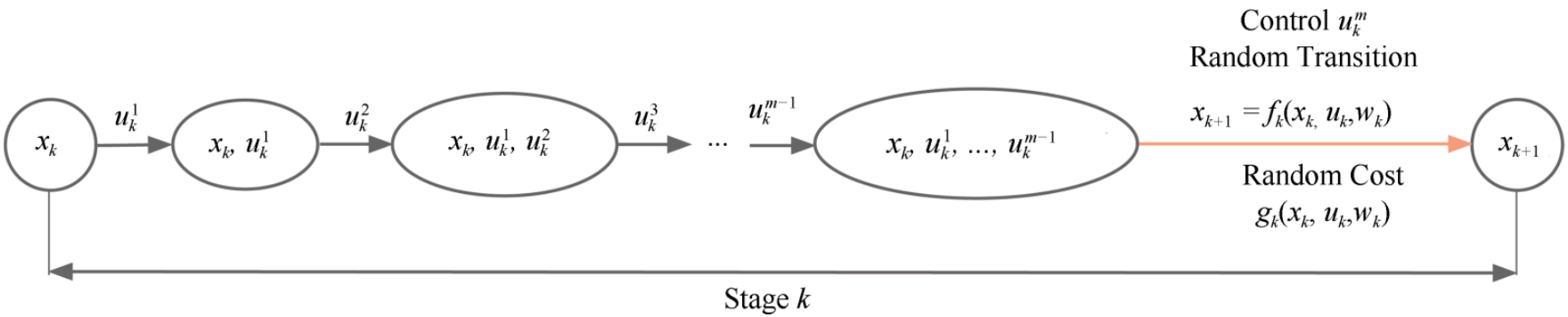

Figure: One-at-a-time action selection. Source: Bertsekas (2020)

1. Break down u_k into the sequence of m actions: $u_k^1, u_k^2, \ldots, u_k^m$
2. Introduce artificial states $\left(x_k, u_k^1\right), \left(x_k, u_k^1, u_k^2\right), \ldots, \left(x_k, u_k^1, \ldots, u_k^{m-1}\right)$
3. u_k^m marks the transition to the new state $x_{k+1} = f\left(x_k, u_k, w_k\right)$ incurring cost $g_k\left(x_k, u_k, w_k\right)$

Benefits of the Multiagent Rollout Algorithm

Past controls determined by the rollout policy, and the future controls determined by the base policy!

- Reducing the action space by increasing the state space. Reasonable since Q-factor minimization is performed for just one state at each stage.
- We reduce the computation complexity from $O(q^m)$ to $O(qm)$, $q = |U|$

Policy Improvement for Multiagent Rollout

Claim: multi-agent rollout performs about as well as standard rollout (all-at-once)
Proof: TODO

Ordering of Agents

- Instead of random order, at each step

Extensions

Reinforcement Learning Problem

- Learning how to map situations to actions
- Trial-and-error search
- Delayed feedback
- Trade-off between exploration and exploitation
- Sequential decision making
- Agent's actions affect the subsequent data it receives

Conclusion

Conclusion

- RL methods can be applicable to a wide variety of problems
- Out-of-the-box models work but require fine-tuning and take longer to converge
- Simple methods like state discretization are worth exploring when training speed and solution complexity are of the essence

References

 Dimitri Bertsekas – Multiagent Reinforcement Learning: Rollout and Policy Iteration (2020). Web: https://web.mit.edu/dimitrib/www/Multiagent_Sinica_2020.pdf

 Shimon Whiteson – Factored Value Functions for Cooperative Multi-Agent Reinforcement Learning (2020) [Seminar].

 Arup Kumar Sadhu, Amit Konar – Multi-Agent Coordination, A Reinforcement Learning Approach (2020).

Thanks for
your attention!