Multiagent (Deep) Reinforcement Learning

MARTIN PILÁT (MARTIN.PILAT@MFF.CUNI.CZ)

Reinforcement learning

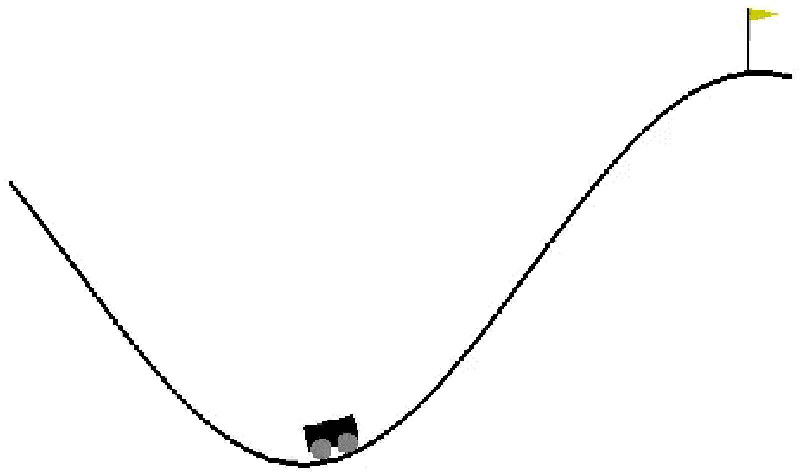
The agent needs to learn to perform tasks in environment

No prior knowledge about the effects of tasks

Maximized its utility

Mountain Car problem →

- Typical RL toy problem
- Agent (car) has three actions – left, right, none
- Goal – get up the mountain (yellow flag)
- Weak engine – cannot just go to the right, needs to gain speed by going downhill first



Reinforcement learning

Formally defined using a Markov Decision Process (MDP) (S, A, R, p)

- $s_t \in S$ – state space
- $a_t \in A$ – action space
- $r_t \in R$ – reward space
- $p(s', r|s, a)$ – probability that performing action a in state s leads to state s' and gives reward r

Agent's goal: maximize discounted returns $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots = R_{t+1} + \gamma G_{t+1}$

Agent learns its policy: $\pi(A_t = a|S_t = s)$

- Gives a probability to use action a in state s

State value function: $V^\pi(s) = E_\pi[G_t|S_t = s]$

Action value function: $Q^\pi(s, a) = E_\pi[G_t|S_t = s, A_t = a]$

Q-Learning

Learns the Q function directly using the Bellman's equations

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha(r_t + \gamma \max_a Q(s_{t+1}, a))$$

During learning – sampling policy is used (e.g. the ϵ-greedy policy – use a random action with probability ϵ, otherwise choose the best action)

Traditionally, Q is represented as a (sparse) matrix

Problems

- In many problems, state space (or action space) is continuous → must perform some kind of discretization
- Can be unstable

Watkins, Christopher JCH, and Peter Dayan. "Q-learning." Machine learning 8, no. 3-4 (1992): 279-292.

Deep Q-Learning

Q function represented as a deep neural network

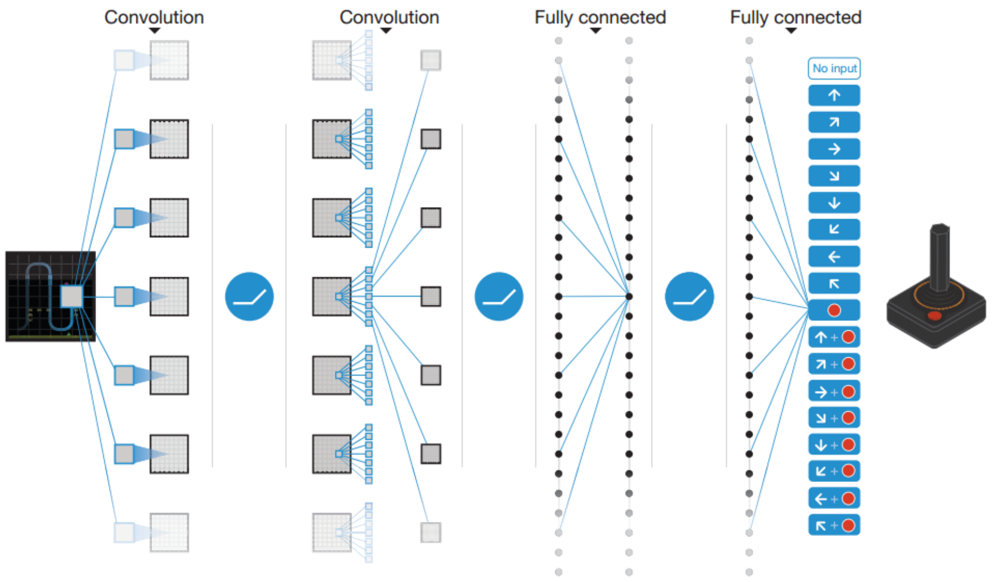
Experience replay
- stores previous experience (state, action, new state, reward) in a replay buffer – used for training

Target network
- Separate network that is rarely updated

Optimizes loss function

$$L(\theta) = E\left[\left(r + \gamma \max_{a'} Q(s, a; \theta_i^-) - Q(s, a; \theta)\right)^2\right]$$

- θ, θ^- - parameters of the network and target network



Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, et al. "Human-Level Control through Deep Reinforcement Learning." *Nature* 518, no. 7540 (February 2015): 529–33. https://doi.org/10.1038/nature14236.

Deep Q-Learning

Successfully used to play single player Atari games

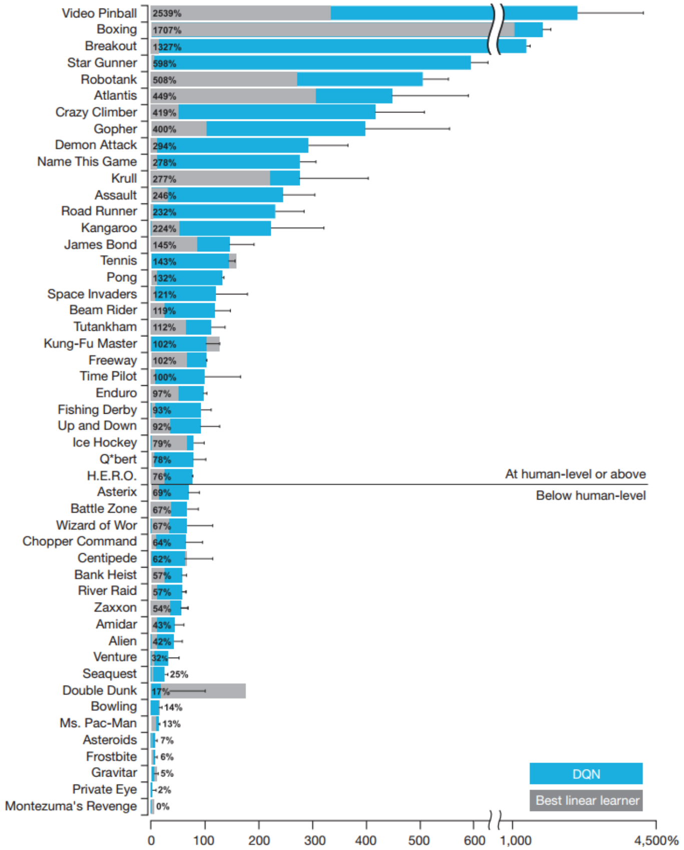
Complex input states – video of the game

Action space quite simple – discrete

Rewards – changes in game score

Better than human-level performance

- Human-level measured against “expert” who played the game for around 20 episodes of max. 5 minutes after 2 hours of practice for each game.



Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, et al. “Human-Level Control through Deep Reinforcement Learning.” *Nature* 518, no. 7540 (February 2015): 529–33. https://doi.org/10.1038/nature14236.

Actor-Critic Methods

The actor (policy) is trained using a gradient that depends on a critic (estimate of value function)

Critic is a value function

- After each action checks if things have gone better or worse than expected
- Evaluation is the error $\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$
- Is used to evaluate the action selected by actor
  - If δ is positive (outcome was better than expected) – probability of selecting a_t should be strengthened (otherwise lowered)

Both actor and critic can be approximated using NN

- Policy ($\pi(s,a)$) update - $\Delta\theta = \alpha \nabla_\theta (\log \pi_\theta(s,a)) q(s,a)$
- Value ($q(s,a)$) update - $\Delta w = \beta\big(R(s,a) + \gamma q(s_{t+1}, a_{t+1}) - q(s_t, a_t)\big) \nabla_\mathrm{w} \mathrm{q}(\mathrm{s_t}, \mathrm{a_t})$

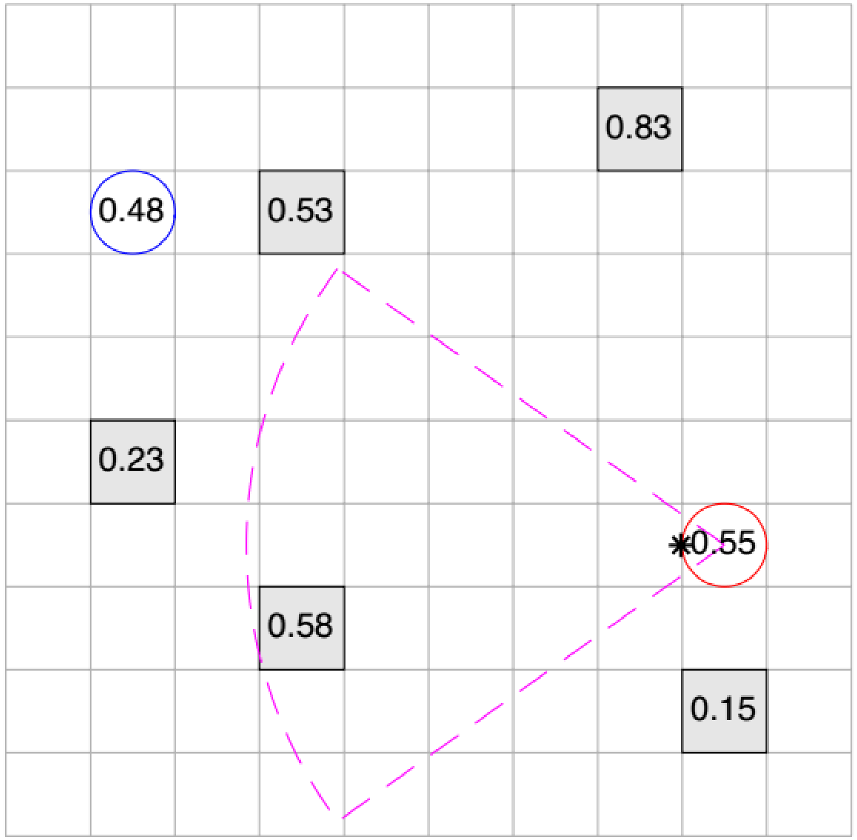
Works in continuous action spaces

Multiagent Learning

Learning in multi-agent environments more complex – need to coordinate with other agents

Example – level-based foraging (→)

- Goal is to collect all items as fast as possible
- Can collect item, if sum of agent levels is greater than item level



Goals of Learning

Minmax profile

- For zero-sum games – (π_i, π_j) is minimax profile if $U_i(\pi_i, \pi_j) = -U_j(\pi_i, \pi_j)$
  - Guaranteed utility against worst-case opponent

Nash equilibrium

- Profile $(\pi_1, \dots, \pi_n)$ is Nash equilibrium if $\forall i \forall \pi_i' : U_i(\pi_i', \pi_{-i}) \leq U_i(\pi)$
  - No agent can improve utility unilaterally deviating from profile (every agent plays best-response to other agents)

Correlated equilibrium

- Agents observe signal x_i with joint distribution $\xi(x_1, \dots, x_n)$ (e.g. recommended action)
- Profile $(\pi_1, \dots, \pi_n)$ is correlated equilibrium if no agent can improve its expected utility by deviating from recommended actions
- NE is special type of CE – no correlation

Goals of Learning

Pareto optimum

- Profile $(\pi_1, \dots, \pi_n)$ is Pareto-optimal if there is not other profile π' such that $\forall i: U_i(\pi') \geq U\_i(\pi)$ and $\exists i: U_i(\pi') > U_i(\pi)$
- Cannot improve one agent without making other agent worse

Social Welfare & Fairness

- Welfare of profile is sum of utilities of agents, fairness is product of utilities
- Profile is welfare or fairness optimal if it has the maximum possible welfare/fairness

No-Regret

- Given history $\mathrm{H}^{\mathrm{t}} = (a_0, \dots, a_{t-1})$ agent i's regret for not having taken action a_i is

$$R_i(a_i) = \sum_t u_{i(a_i, a_{-i}^t)} - u_i(\, a_i^t, a_{-i}^t)$$

- Policy π_i achieves no-regret if $\forall a_i: \lim_{t\to\infty} \frac{1}{t} R_i(a_i | H^t) \leq 0.$

Joint Action Learning

Learns Q-values for joint actions $a \in A$

- joint action of all agents $a = (a_1, \dots, a_n)$, where a_i is the action of agent i

$Q^{t+1}(a_t, s_t) = (1 - \alpha)Q^t(a_t, s_t) + \alpha u_i^t$

- u_i^t - utility received after joint action a_t

Uses opponent model to compute expected utilities of action

- $E(a_i) = \sum_{a_{-i}} P(a_{-i}) Q^{t+1}((a_i, a_{-i}), s_{t+1})$ – joint action learning
- $E(a_i) = \sum_{a_{-i}} P(a_{-i}|a_i) Q^{t+1}\left((a_i, a_{-i}), s_{t+1}\right)$ – conditional joint action learning

Opponent models predicted from history as relative frequencies of action played (conditional frequencies in CJAL)

ϵ – greedy sampling

Policy Hill Climbing

Learn policy π_i directly

Hill-climbing in policy space

- $\pi_i^{t+1} = \pi_i^t\left(s_i^t, a_i^t\right) + \delta$ if a_i^t is the best action according to $Q(s^t, a_i^t)$
- $\pi_i^{t+1} = \pi_i^t\left(s_i^t, a_i^t\right) - \frac{1}{|A_i|-1}$ otherwise

Parameter δ is adaptive – larger if winning and lower if losing

Counterfactual Multi-agent Policy Gradients

Centralized training and de-centralized execution (more information available in training)

Critic conditions on the current observed state and the actions of all agents

Actors condition on their observed state

Credit assignment – based on difference rewards

- Reward of agent $i \sim$ the difference between the reward received by the system if joint action a was used, and reward received if agent i would have used a default action
  - Requires assignment of default actions to agents
- COMA – marginalize over all possible actions of agent i

Used to train micro-management of units in StarCraft

Foerster, Jakob, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. "Counterfactual Multi-Agent Policy Gradients." *ArXiv:1705.08926 [Cs]*, May 24, 2017. http://arxiv.org/abs/1705.08926.

Counterfactual Multi-agent Policy Gradients

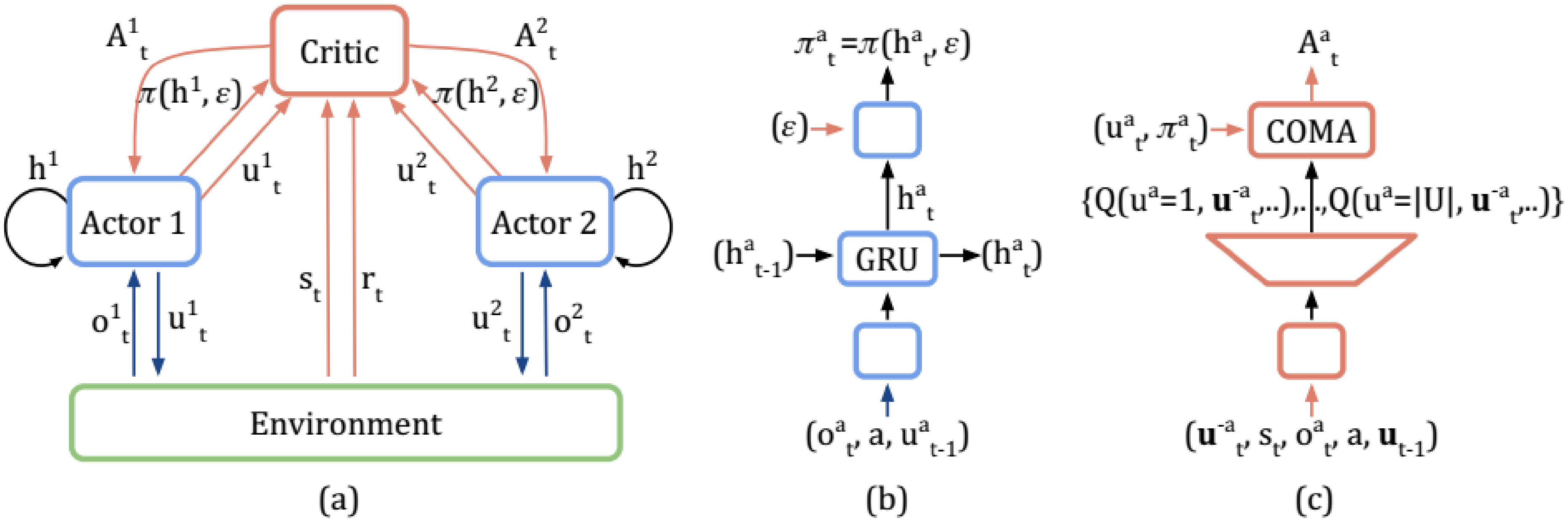


Figure 1: In (a), information flow between the decentralised actors, the environment and the centralised critic in COMA; red arrows and components are only required during centralised learning. In (b) and (c), architectures of the actor and critic.

Foerster, Jakob, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. "Counterfactual Multi-Agent Policy Gradients." *ArXiv:1705.08926 [Cs]*, May 24, 2017. http://arxiv.org/abs/1705.08926.

Ad hoc Teamwork

Typically whole team of agents provided by single organization/team.

- There is some pre-coordination (communication, coordination, …)

Ad hoc teamwork

- Team of agents provided by different organization need to cooperate
  - RoboCup Drop-In Competition – mixed players from different teams
- Many algorithms not suitable for ad hoc teamwork
  - Need many iterations of game – typically limited amount of time
  - Designed for self-play (all agents use the same strategy) – no control over other agents in ad hoc teamwork

Ad hoc Teamwork

Type-based methods

- Assume different types of agents
- Based on interaction history – compute belief over types of other agents
- Play own actions based on beliefs
- Can also add parameters to types

Other problems in MAL

Analysis of emergent behaviors

- Typically no new learning algorithms, but single-agent learning algorithms evaluated in multi-agent environment
- Emergent language
  - Learn agents to use some language
  - E.g. signaling game – two agents are show two images, one of them (sender) is told the target and can send a message (from fixed vocabulary) to the receiver; both agents receive a positive reward if the receiver identifies the correct image

Learning communication

- Agent can typically exchange vectors of numbers for communication
- Maximization of shared utility by means of communication in partially observable environment

Learning cooperation

Agent modelling agents

References and Further Reading

- Foerster, Jakob, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. “Counterfactual Multi-Agent Policy Gradients.” *ArXiv:1705.08926 [Cs]*, May 24, 2017. http://arxiv.org/abs/1705.08926.
- Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, et al. “Human-Level Control through Deep Reinforcement Learning.” *Nature* 518, no. 7540 (February 2015): 529–33. https://doi.org/10.1038/nature14236.
- Albrecht, Stefano, and Peter Stone. “Multiagent Learning - Foundations and Recent Trends.” http://www.cs.utexas.edu/~larg/ijcai17_tutorial/
  - Nice presentation about general multi-agent learning (slides available)
- Open AI Gym. https://gym.openai.com/
  - Environments for reinforcement learning
- Lillicrap, Timothy P., Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. “Continuous Control with Deep Reinforcement Learning.” *ArXiv:1509.02971 [Cs, Stat]*, September 9, 2015. http://arxiv.org/abs/1509.02971.
  - Actor-Critic method for reinforcement learning with continuous actions
- Hernandez-Leal, Pablo, Bilal Kartal, and Matthew E. Taylor. “Is Multiagent Deep Reinforcement Learning the Answer or the Question? A Brief Survey.” *ArXiv:1810.05587 [Cs]*, October 12, 2018. http://arxiv.org/abs/1810.05587.
  - A survey on multiagent deep reinforcement learning
- Lazaridou, Angeliki, Alexander Peysakhovich, and Marco Baroni. “Multi-Agent Cooperation and the Emergence of (Natural) Language.” ArXiv:1612.07182 [Cs], December 21, 2016. http://arxiv.org/abs/1612.07182.
  - Emergence of language in multiagent communication