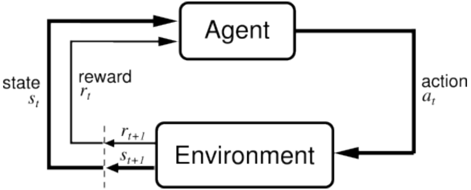
Hierarchical methods for Reinforcement Learning

Evgenii Nikishin

30.03.2018

Reinforcement Learning recap

Markov Decision Process (MDP):



- Agent actions $A_t \in \mathcal{A}$
- Environment states $S_t \in \mathcal{S}$
- Reward $R_t \in \mathbb{R}$
- Agent policy $A_t \sim \pi(a|s)$
- State transitions $S_{t+1}, R_{t+1} \sim p(s', r|s, a)$

Optimal policy maximizes the expected discounted return:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\pi}\left[R_1 + \gamma R_2 + \gamma^2 R_3 + \dots\right] = \arg\max_{\pi} \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k R_{k+1}\right]$$

Standard approaches

Value functions:

- $V_\pi(s) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s\right]$
- $Q_\pi(s, a) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a\right]$

Standard approaches

Value functions:

- $V_\pi(s) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s\right]$
- $Q_\pi(s,a) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a\right]$

Properties of value functions:

- $V_\pi(s) = \sum_a \pi(a|s) Q_\pi(s,a)$
- $Q_\pi(s,a) = \sum_{s',r} p(s',r|s,a)\left[r + \gamma V_\pi(s')\right]$

Standard approaches

Value functions:

- $V_\pi(s) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s\right]$
- $Q_\pi(s,a) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a\right]$

Properties of value functions:

- $V_\pi(s) = \sum_a \pi(a|s) Q_\pi(s,a)$
- $Q_\pi(s,a) = \sum_{s',r} p(s',r|s,a)\left[r + \gamma V_\pi(s')\right]$
- $V_*(s) = V_{\pi*}(s) = \max_a \sum_{s',r} p(s',r|s,a)\left[r + \gamma V_*(s')\right]$
- $Q_*(s,a) = Q_{\pi*}(s,a) = \sum_{s',r} p(s',r|s,a)\left[r + \gamma max_{a'} Q_*(s',a')\right]$

Standard approaches

Value functions:

- $V_\pi(s) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s\right]$
- $Q_\pi(s, a) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a\right]$

Properties of value functions:

- $V_\pi(s) = \sum_a \pi(a|s) Q_\pi(s, a)$
- $Q_\pi(s, a) = \sum_{s', r} p(s', r|s, a) \left[r + \gamma V_\pi(s')\right]$
- $V_*(s) = V_{\pi*}(s) = \max_a \sum_{s', r} p(s', r|s, a) \left[r + \gamma V_*(s')\right]$
- $Q_*(s, a) = Q_{\pi*}(s, a) = \sum_{s', r} p(s', r|s, a) \left[r + \gamma max_{a'} Q_*(s', a')\right]$

Learn optimal Q-function and act greedily (e.g. Q-learning:
$Q(s, a) \leftarrow Q(s, a) + \alpha\left(r(s, a) + \max_{a'} Q(s', a') - Q(s, a)\right))$

Standard approaches

Parametrize policy $\pi(a|s) = \pi_\theta(a|s)$ within a family of differentiable functions and update w.r.t. its gradient:

$$\nabla_\theta \mathbb{E}_{\pi_\theta}\left[\sum_{k=0}^{\infty} \gamma^k R_{k+1}\right] = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(A_t|S_t) Q_{\pi_\theta}(S_t, A_t)\right]$$

Standard approaches

Parametrize policy $\pi(a|s) = \pi_\theta(a|s)$ within a family of differentiable functions and update w.r.t. its gradient:

$$\nabla_\theta \mathbb{E}_{\pi_\theta}\left[\sum_{k=0}^{\infty} \gamma^k R_{k+1}\right] = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(A_t|S_t) Q_{\pi_\theta}(S_t, A_t)\right]$$

Q_{π_θ} is still unknown:

- use MC estimates
- parametrize and learn (Actor-Critic algorithm)

Options framework in RL

Option ω is a tuple $\langle \mathcal{I}_\omega, \pi_\omega, \beta_\omega \rangle$, where

- $\mathcal{I}_\omega \subseteq \mathcal{S}$ — option initiation set,
- $\pi_\omega(a|s)$ — option policy,
- $\beta_\omega(s)$ — probability of option termination at given state.

When option terminates, control is given to meta-policy $\pi_\Omega(\omega|s)$.
E. g., robot navigation: if there is no obstacle in front ($\mathcal{I}_\omega$), go forward (π_ω) until you get too close to another object (β_ω).
One can think of options as functions in program.

Discussion

Motivation of options:

- Sample efficiency
- Transfer of knowledge
- Consistent exploration
- Interpretability
- Way to address overfitting in RL

Discussion

Motivation of options:

- Sample efficiency
- Transfer of knowledge
- Consistent exploration
- Interpretability
- Way to address overfitting in RL (video)

Discussion

Motivation of options:

- Sample efficiency
- Transfer of knowledge
- Consistent exploration
- Interpretability
- Way to address overfitting in RL (video)

Arguments:

- There exist optimal deterministic policy in MDP
- Introducing options lead to larger number of parameters, which slows down learning

Option-Critic

Value functions extension:

$$Q_\Omega(s,\omega) = \sum_a \pi_\omega(a|s) Q_U(s,\omega,a)$$

Option-Critic

Value functions extension:

$$Q_\Omega(s,\omega) = \sum_a \pi_\omega(a|s) Q_U(s,\omega,a)$$

$$Q_U(s,\omega,a) = \sum_{s',r} p(s',r|s,a)\left[r(s,a) + \gamma U(\omega,s')\right]$$

Option-Critic

Value functions extension:

$$Q_\Omega(s,\omega) = \sum_a \pi_\omega(a|s) Q_U(s,\omega,a)$$

$$Q_U(s,\omega,a) = \sum_{s',r} p(s',r|s,a)\left[r(s,a) + \gamma U(\omega,s')\right]$$

$$U(\omega,s') = (1-\beta_\omega(s'))Q_\Omega(s',\omega) + \beta_\omega(s')V_\Omega(s')$$

Option-Critic

Value functions extension:

$$Q_\Omega(s,\omega) = \sum_a \pi_\omega(a|s) Q_U(s,\omega,a)$$

$$Q_U(s,\omega,a) = \sum_{s',r} p(s',r|s,a)\left[r(s,a) + \gamma U(\omega,s')\right]$$

$$U(\omega,s') = (1-\beta_\omega(s'))Q_\Omega(s',\omega) + \beta_\omega(s')V_\Omega(s')$$

$$V_\Omega(s') = \sum_{\omega'} \pi_\Omega(\omega'|s')Q_\Omega(s',\omega')$$

Option-Critic

Value functions extension:

$$Q_\Omega(s,\omega) = \sum_a \pi_\omega(a|s) Q_U(s,\omega,a)$$

$$Q_U(s,\omega,a) = \sum_{s',r} p(s',r|s,a)\left[r(s,a) + \gamma U(\omega,s')\right]$$

$$U(\omega,s') = (1-\beta_\omega(s'))Q_\Omega(s',\omega) + \beta_\omega(s')V_\Omega(s')$$

$$V_\Omega(s') = \sum_{\omega'} \pi_\Omega(\omega'|s')Q_\Omega(s',\omega')$$

$Q_U(s,\omega,a)$ is learnt via Q-learning algorithm.

Option-Critic

Main contributions:
Gradient w.r.t. option policy $\pi_\omega(a|s)$ params is given by

$$\mathbb{E}\left[\nabla \log \pi_\omega\left(a|s\right) Q_U(s, \omega, a)\right]$$

Natural result: take better primitive actions more often inside the option
Gradient w.r.t. option termination $\beta_\omega(s)$ params is given by

$$\mathbb{E}\left[-\nabla \beta_\omega\left(s\right)\left(Q_\Omega(s', \omega') - V_\Omega(s')\right)\right]$$

Also natural: lengthen options that have a large advantage.

Experiments

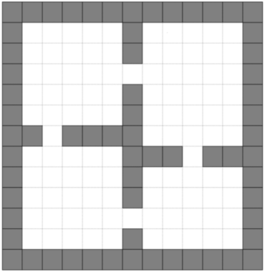


Figure: 4 rooms domain

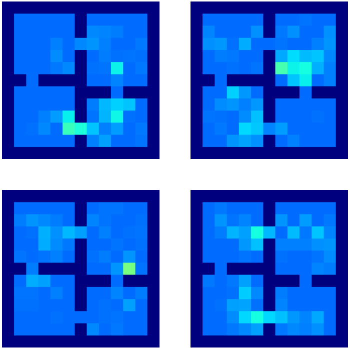


Figure: Termination probabilities for the option-critic agent learning with 4 options.

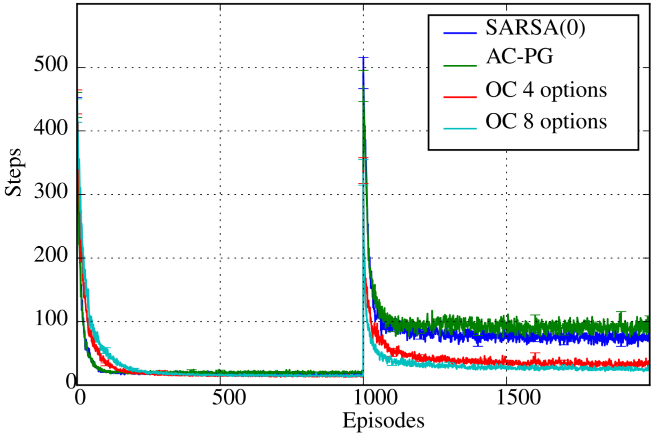


Figure: After a 1000 episodes, the goal location in the four-rooms domain is moved randomly.

Stochastic Neural Networks

Consider set of MDPs $\mathcal{M}$.
Assumption: for each MDP $M \in \mathcal{M}$, state space is decomposed into two components, $\mathcal{S}_{\mathrm{agent}}$, and $\mathcal{S}^{M}_{\mathrm{rest}}$. Also, MDPs share action space.
E.g. robot facing different tasks.

Stochastic Neural Networks

Consider set of MDPs $\mathcal{M}$.
Assumption: for each MDP $M \in \mathcal{M}$, state space is decomposed into two components, $\mathcal{S}_{\text{agent}}$, and $\mathcal{S}^{M}_{\text{rest}}$. Also, MDPs share action space.
E.g. robot facing different tasks.
Scenario:

1. One MDP is used to pretrain set of skills.
2. For other MDPs, policy on top of this skills is trained.

Skills training

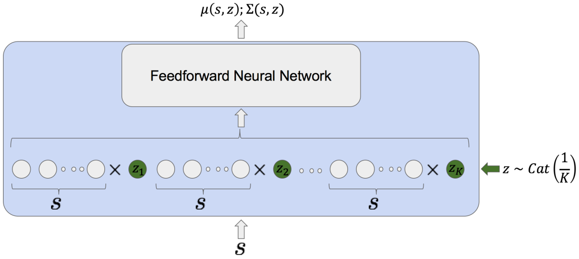


Figure: Skill network architecture

During for each episode separate z is sampled.

Skills training

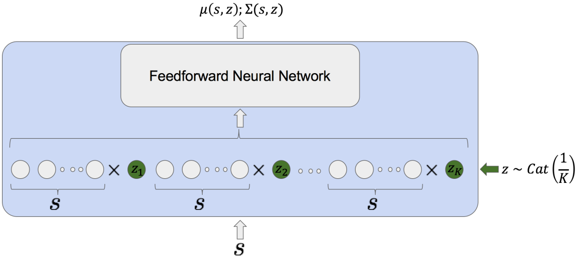


Figure: Skill network architecture

During for each episode separate z is sampled.
What makes agent to take into account z?

Mutual Information bonus

$$I(Z;C) = H(Z) - H(Z|C) = const + \mathbb{E}_{z,c}\left[\log p(Z=z|C=c)\right]$$

In order to forbid agent to ignore z, received reward modified:
Mutual Information between z and coordinates at timestep t is added:

$$R_t \leftarrow R_t + \alpha_H \log \hat{p}(Z=z|c_t)$$

In order to estimate $\log \hat{p}(Z=z^n|c_t^n)$, discretization is used: denote $m_c(z)$ visitation counts of how many times each cell c is visited when latent code z is sampled

$$\hat{p}(Z=z|c) = \frac{m_c(z)}{\sum_{z'} m_c(z')}$$

Skills usage

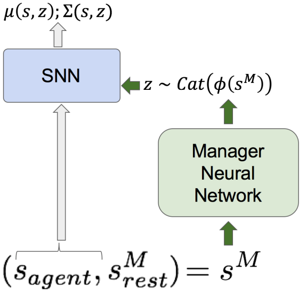


Figure: Hierarchical SNN architecture to solve downstream tasks

Manager operates in terms of skills: z now considered as actions.
Once manager selects a skill z, agent is committing to it for a fixed amount of steps $\mathcal{T}$.

Experimental setup

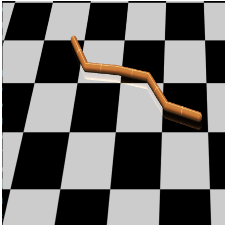


Figure: MDP 0: locomotion

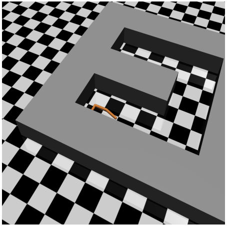


Figure: MDP 1: Maze 1

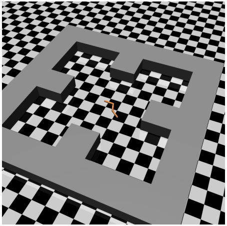


Figure: MDP 2: Maze 2

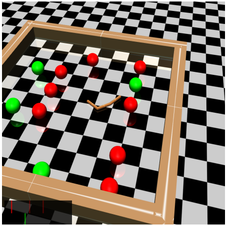
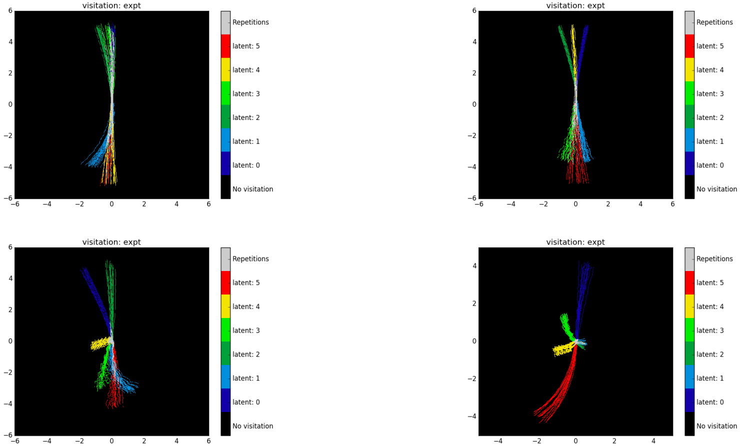


Figure: MDP 3: Food Gather

Experimental setup

- In MDP 0, we are interested in learning diverse set of skills
- Faster learning of the hierarchical architectures in the downstream MDPs:
    1. CoM reward: single policy with bonus speed reward
       (authors claim it accelerates learning)
    2. Multi-policy: independently trained policies + manager network upon
    3. SNN
    4. SNN + MI bonus

Experiments



Span of learnt skills in MDP 0, $\alpha_H = 0, 0.001, 0.01, 0.1$.

Experiments

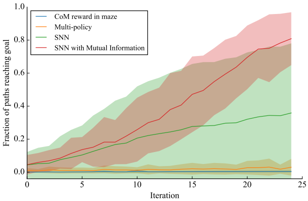


Figure: Maze 1

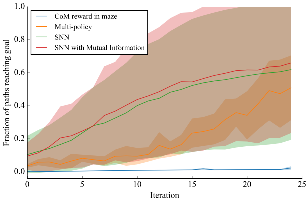


Figure: Maze 2

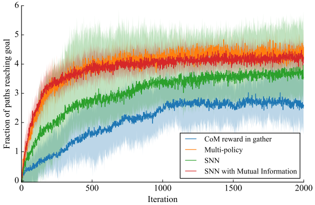


Figure: Food Gather

References

1. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning (Original options paper)
2. The Option-Critic Architecture
3. Stochastic Neural Networks for Hierarchical Reinforcement Learning