Actor-Critic Deep Reinforcement Learning and Its Application in Wireless Communications

Data Science Seminar
April 8, 2022

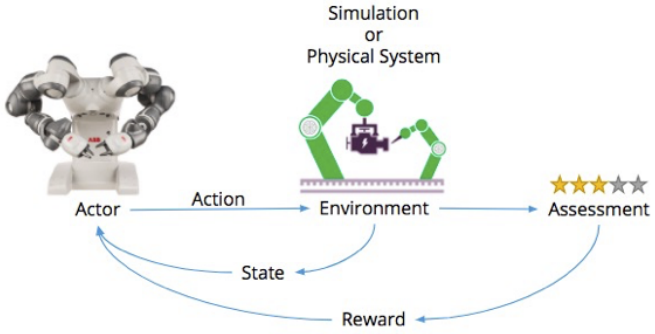
Prof. Liang Dong

Baylor University
SCHOOL OF ENGINEERING & COMPUTER SCIENCE
Department of Electrical & Computer Engineering

Outline

- Introduction to Reinforcement Learning
- Deep Reinforcement Learning Algorithm and Training
- Actor-Critic Deep Reinforcement Learning
- Deep Deterministic Policy Gradient Algorithm
- Application to Dynamic Spectrum Access and Sharing
- Conclusion and Future Work

Reinforcement Learning



- Reinforcement learning is a reward-driven interactive process of trial-and-error with the environment.
- Reward/punishment received from the environment are used to guide the learning process for future decisions.
- The agent learns to interact with the environment to achieve rewarding outcomes.

Suitability of Reinforcement Learning

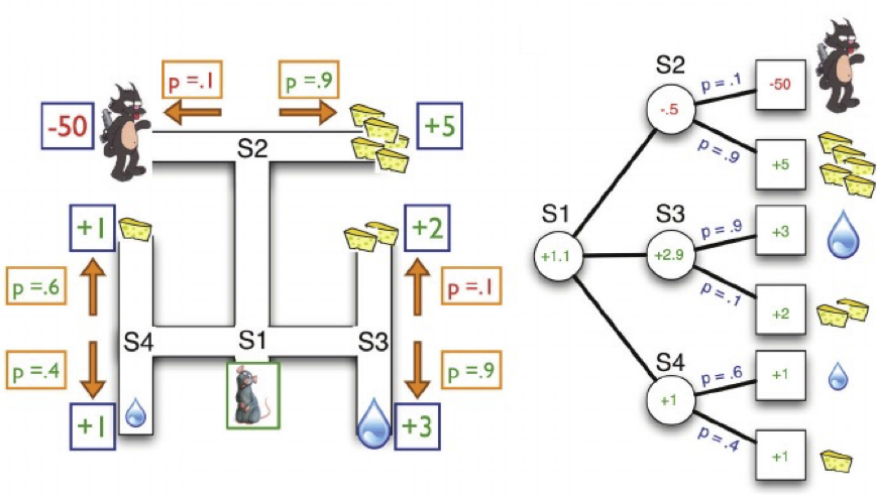
- When to use reinforcement learning methods?

- Reinforcement learning is suitable for tasks that are simple to evaluate but hard to specify.

- Reinforcement learning systems are end-to-end systems in which a complex task is usually not broken up into smaller components.

- Task execution can be evaluated with a simple reward or delayed rewards.

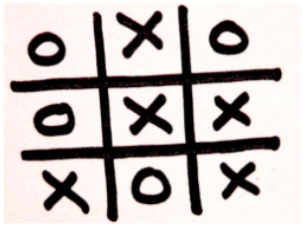
Reinforcement Learning and Markov Decision Process

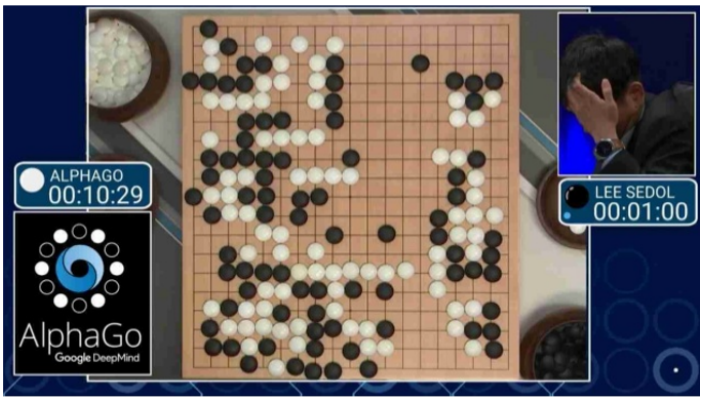


Markov decision process: $s_0 a_0 (r_0) s_1 a_1 (r_1) \ldots s_t a_t (r_t) \ldots s_N a_N r_N$.

(r_i) possibly absent or delayed rewards

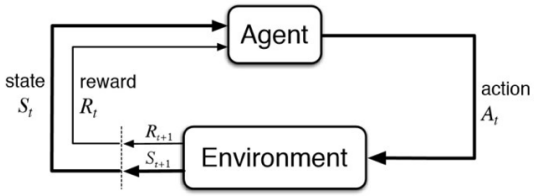
Reinforcement Learning for Board Games





- Board games: The actions are the moves made by the player.
- The reward is $+1$, 0, or -1 (win, draw, or loss) received at the end of the game.

Reinforcement Learning with States



- An agent uses the action to interact with the environment.
- The state of the environment is updated and observed.
- The environment gives the agent rewards.
- The goal is to determine the intrinsic values of actions in different states, regardless of the timing and stochasticity of the reward.

Value Function: Q-Function

- Let's look at an episode of the process represented by a Markov decision process

$$s_0\ a_0\ r_0\ s_1\ a_1\ r_1 \ldots s_t\ a_t\ r_t\ s_{t+1}\ a_{t+1}\ r_{t+1} \ldots$$

- Define the cumulative discounted reward at time t

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots = \sum_{\tau=t}^{\infty} \gamma^{\tau-t} r_\tau$$

where the discount factor $0 \leq \gamma \leq 1$.

Value Function: Q-Function

- Q-function of state-action pair (s_t, a_t) is a measure of the intrinsic value of performing action a_t in state s_t

$$Q(s_t, a_t) = \max \mathrm{E}[R_t \mid a_t]$$

Q-function represents the maximum reward over all combination of future actions.

- The chosen action from Set A of all possible actions at time t that maximize $Q(s_t, a_t)$ is

$$a_t^* = \arg\max_{a_t \in A} Q(s_t, a_t)$$

Value Function: Q-Function

- With the knowledge of the current reward r_t, we get an improved estimate of the Q-value using the Bellman equation[1].

$$Q(s_t, a_t) = r_t + \gamma \max_{a_{t+1}} \hat{Q}(s_{t+1}, a_{t+1})$$

- $\hat{Q}$ indicates that it is a predicted value.

"Looking ahead one step and predicting at state s_{t+1}."

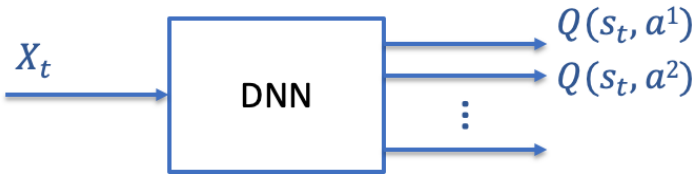
[1] Stochastic Dynamic Programming (Bellman 1957): Bellman Equation – It writes the value of a decision problem at a certain point in time in terms of the payoff and the value of the remaining decision problem. It breaks a dynamic optimization problem into a sequence of simpler subproblems.

Q-Learning Algorithm

- Where does deep learning come into play?
- Deep neural network (DNN) is a function approximator:

$$F(X_t, W, a) = \hat{Q}(s_t, a)$$

for each action $a \in A$.



where X_t is the feature input that is based on state s_t, W are the neural network weights.

Q-Learning Algorithm

- How to train the DNN?
- We need to find the loss from the target and the DNN output:

$$L_t = (Q(s_t, a) - \hat{Q}(s_t, a))^2$$

- However, there is a problem. The target $Q(s_t, a)$ is unknown.
- Solution: We use the one-step-ahead improved estimate of the Q-value as the ground-truth target.

$$Q(s_t, a) = r_t + \gamma \max_a \hat{Q}(s_{t+1}, a) = r_t + \gamma \max_a F(X_{t+1}, W, a)$$

Q-Learning Algorithm

- Therefore, the loss L_t of the DNN is

$$L_t = \left(\underbrace{r_t + \gamma \max_a F(X_{t+1}, W, a)}_{\text{treated as ground truth}} - F(X_t, W, a) \right)^2$$

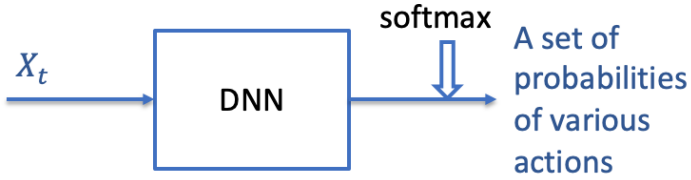
- Backpropagation

$$W \leftarrow W + \alpha \left(r_t + \gamma \max_a F(X_{t+1}, W, a) - F(X_t, W, a) \right) \frac{\partial F(X_t, W, a)}{\partial w}$$

Modeling States and Policy Gradient Methods

- Modeling States: DNN to learn the value $V(s_t)$ of a particular state instead of the state-action pair.

- Policy Gradient Methods: DNN computes $P(X_t, W, a)$, i.e., probability that action a should be performed.



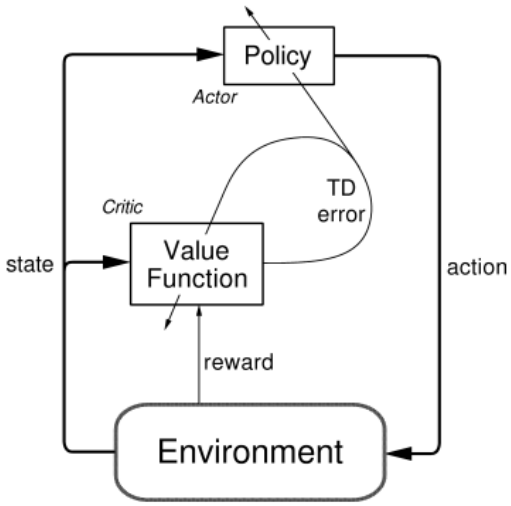
Actor-Critic Deep Reinforcement Learning

- Value-based algorithm: e.g., Deep Q-learning
- Policy-based algorithm: e.g., REINFORCE
- Actor-critic algorithm: A hybrid of the value-based and policy-based algorithms.

  e.g., Deep Deterministic Policy Gradient (DDPG) algorithm, Asynchronous Advantage Actor Critic (A3C) algorithm

Actor-Critic Deep Reinforcement Learning

- The actor-critic consists of two components: an actor to generate a policy and a critic to assess the policy.
- A better solution is learned through solving a multi-objective optimization problem and updating the parameters of the actor and the critic alternatively.



Actor-Critic DDPG Algorithm

- Practical Engineering Issues:

  (1) Large state space $\rightarrow$ Deep Q-networks
  (2) Large action space (continuous-valued actions) $\rightarrow$ Deep Deterministic Policy Gradient (DDPG) algorithm

- The DDPG algorithm is an actor-critic algorithm that integrates the deep Q-network with the deterministic policy gradient algorithm.

- It can concurrently learn a Q-function and a policy in a high-dimensional and continuous action space.

DDPG Algorithm: The Critic

- The critic estimates the action-value function, i.e., the Q-function.

$$\begin{aligned} Q^{\pi}(o_t, a_t) &= \mathbb{E}_{\pi}[R_t \mid o_t, a_t] \\ &= \mathbb{E}\left[r(o_t, a_t) + \gamma \mathbb{E}_{\pi}[Q^{\pi}(o_{t+1}, a_{t+1})]\right] \end{aligned}$$

  Taking an action a_t on observation o_t following policy π.

- The Q-function $Q^{\pi}(o_t, a_t)$ can be approximated by a Q-network parameterized with θ^Q.

$$Q(o_t, a_t \mid \theta^Q) = Q^{\pi}(o_t, a_t)$$

DDPG Algorithm: The Actor

- The actor updates the policy distribution in the direction suggested by the critic with policy gradients.

- When the policy is deterministic, the actor directly maps observation o_t to action a_t. This can be approximated by a policy network parameterized with θ^μ.

$$a_t = \mu(o_t \mid \theta^\mu)$$

- At training time, we add noise to the action to perform exploration.

$$a_t = \mu(o_t \mid \theta^\mu) + w$$

Deep Deterministic Policy Gradient (DDPG) Algorithm

- The DDPG algorithm maintains:

  (1) A Q-network (critic),
  (2) A policy network (actor),
  (3) A target Q-network (Q'),
  (4) A target policy network (μ').

  The target networks (3) and (4) are the time-delayed copies of the Q-network and the policy network, respectively. The target Q-network is parameterized with $\theta^{Q'}$, and the target policy network is parameterized with $\theta^{\mu'}$.

Deep Deterministic Policy Gradient (DDPG) Algorithm

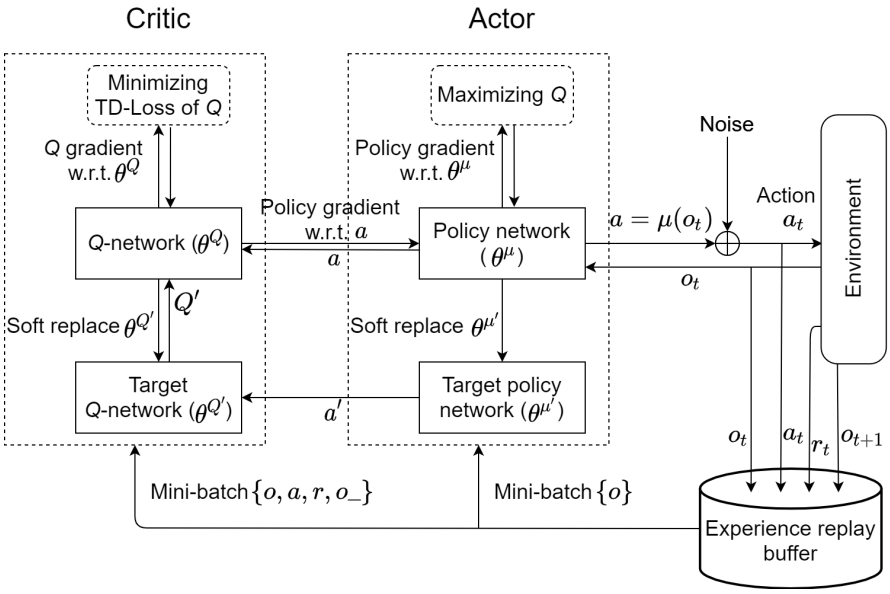


Figure: Actor-critic architecture of the algorithm that includes the four networks, a Q-network, a policy network, a target Q-network, and a target policy network.

Training the DDPG

- In the neural-network implementation, the updated Q-value ($\tilde{Q}$) according to the Bellman equation is given by

$$\tilde{Q}_t = r(o_t, a_t) + \gamma Q'(o_{t+1}, \mu'(o_{t+1} \mid \theta^{\mu'}) \mid \theta^{Q'}).$$

- The Q-network (Q) is updated by minimizing the temporal-difference (TD) loss as

$$\min \quad L = \sum \left[Q(o_t, a_t \mid \theta^Q) - \tilde{Q}_t \right]^2$$

where $\sum$ denotes the average sum over a mini-batch.

Training the DDPG

- With the actor's policy function μ, the objective is to maximize the expected return, i.e., the expected cumulative discounted reward, as

$$\max \quad J = \mathbb{E}\left[Q(o_t, a)|_{a=\mu(o_t)}\right].$$

- The policy network (μ) is updated by using the sampled policy gradient as

$$\nabla_{\theta^\mu} J \approx \sum \nabla_a Q(o_t, a \mid \theta^Q)|_{a=\mu(o_t)} \nabla_{\theta^\mu} \mu(o_t \mid \theta^\mu).$$

- The two target networks slowly track the learned parameters of the Q-network and the policy network.

$$\begin{aligned} \theta^{Q'} &\leftarrow \rho\theta^Q + (1-\rho)\theta^{Q'} \\ \theta^{\mu'} &\leftarrow \rho\theta^\mu + (1-\rho)\theta^{\mu'} \end{aligned}$$

where ρ $(0 < \rho \ll 1)$ controls the update rate to be slow to improve the stability of learning.

Twin Delayed Deep Deterministic Policy Gradient (TD3) Algorithm

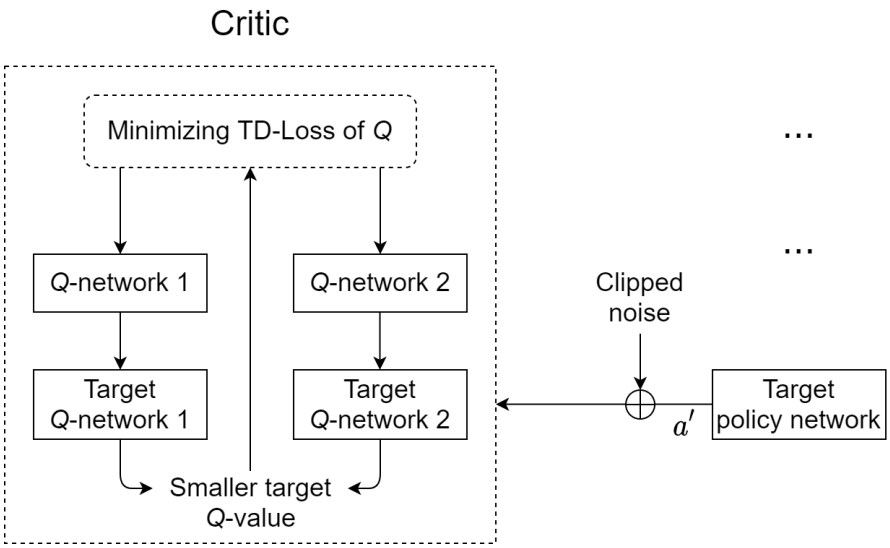


Figure: Part of structure of Twin Delayed Deep Deterministic Policy Gradient algorithm.

Wireless Communication Application: Dynamic Spectrum Access and Sharing

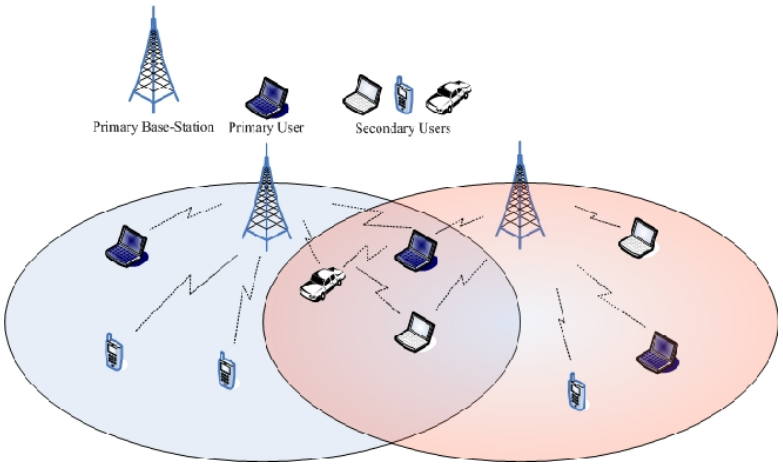


Figure: Dynamic spectrum access is an efficient solution to opportunistically allow secondary users to share licensed spectrum bands when it is not in use by their respective owners (primary users).

Use Cases

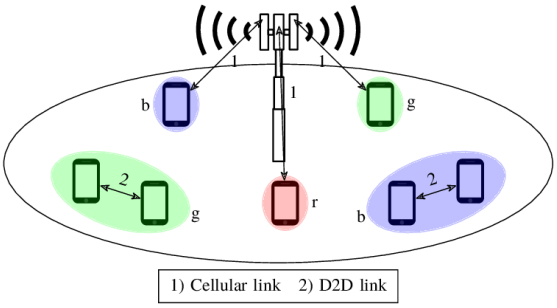


Figure: Use Case 1: Device-to-device (D2D) communication underlay cellular network.

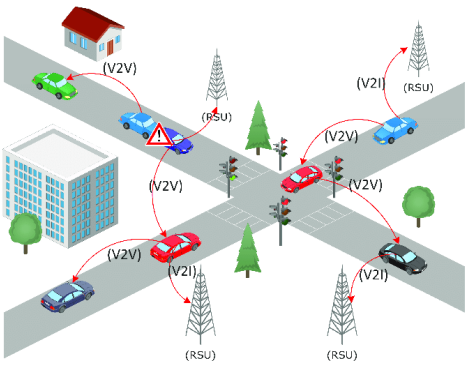


Figure: Use Case 2: Vehicular network – vehicle-to-vehicle (V2V) links and vehicle-to-infrastructure (V2I) links.

Transmission Strategy of Secondary Users

- The state of channel usage: $\mathbf{x} = [x_1, x_2, \ldots, x_N]^T$.
- There are K secondary users of the spectrum who want to transmit on the N frequency channels. The kth secondary user transmits on the nth channel. Its receiver can measure the received SINR on the nth channel as

$$\mathrm{SINR}_{k,n} = \frac{|h_{kk,n}|^2 p_{k,n}}{\sum_{l \in \mathcal{K} \setminus k} |h_{kl,n}|^2 p_{l,n} + W N_0}$$

- Information rate in bits per second per hertz (bit/s/Hz) is

$$\begin{aligned} q_k &= \sum_{n=1}^{N} \log_2(1 + \mathrm{SINR}_{k,n}) \\ &= \sum_{n=1}^{N} \log_2 \left(1 + \frac{|h_{kk,n}|^2 p_{k,n}}{\sum_{l \in \mathcal{K} \setminus k} |h_{kl,n}|^2 p_{l,n} + W N_0} \right). \end{aligned}$$

Transmission Strategy of Secondary Users

- If the kth secondary user transmits on the mth frequency channel that conflicts with any primary user, the primary user will issue a warning about the mth channel.
- After receiving the warning signal, the secondary user invalidates the communication effort on the mth channel by setting $\mathrm{SINR}_{k,m} = 0$ at the receiver.
- This effectively eliminates the contribution to the information rate q_k that is attributed to the transmission on the mth channel.

Optimization Problem Formulation

The transmission problem of the kth secondary user can be formulated as

$$\begin{array}{ll} \underset{\{p_{k,n}\}_{n\in\mathcal{N}}}{\text{maximize}} & q_k \\ \text{subject to} & \sum_{n\in\mathcal{N}} p_{k,n} \leq P \\ & \text{SINR}_{k,m} = 0, m \in \mathcal{M} \end{array}$$

where $\mathcal{M}$ ($\mathcal{M} \subseteq \mathcal{N}$) is the set of indexes of frequency channels that are occupied by the primary users.

Dynamic Spectrum Access and Sharing through DRL

- State and Observation

  $$o_t = \{\hat{\mathbf{x}}_{t-T_0}, \hat{\mathbf{x}}_{t-T_0+1}, \cdots, \hat{\mathbf{x}}_{t-1}\}$$

  where $\hat{\mathbf{x}}$ is the observation of the state vector, whose elements are ternary from $\{-1, 0, 1\}$.

- Action of the agent at time slot t is the transmission power as

  $$a_t = \{p_{n,t}\}_{n\in\mathcal{N}}$$

  where $p_{n,t}$ is the transmission power on the nth frequency channel.

- Reward

  $$r_t = q_t - \beta \sum_{m\in\mathcal{M}_t} p_{m,t}, \qquad R_t = \sum_{\tau=t}^{T} \gamma^{(\tau-t)} r_\tau, \;\; 0 \leq \gamma \leq 1$$

  where q_t is the information rate. The second term is the penalty due to conflict with the primary users.

Actor: Policy Network

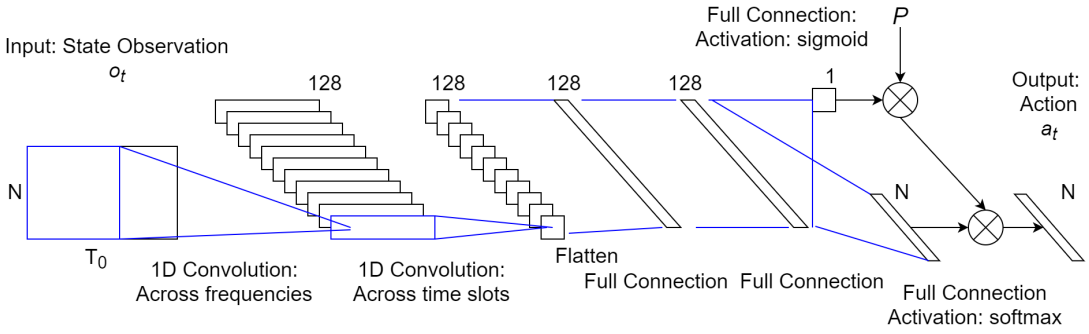


Figure: Neural-network architecture of the policy network.

Critic: Q-Network

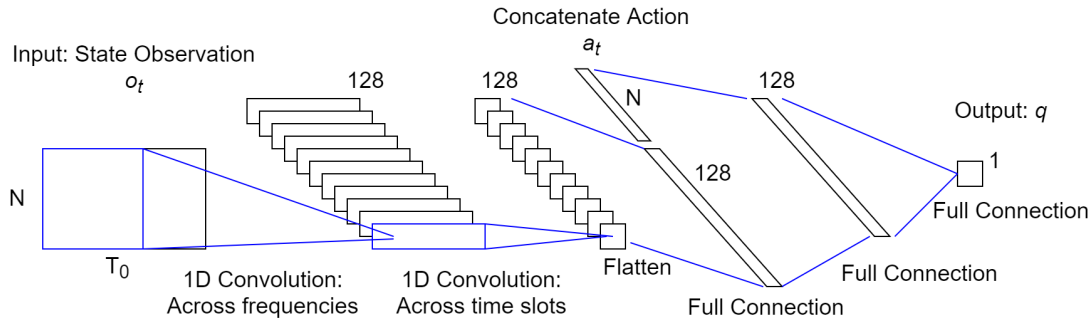


Figure: Neural-network architecture of the Q-network.

Experimental Methods

Figure: Experimental setup in the laboratory at Baylor BRIC.

Experimental Methods

Table: Parameters of the TD3 algorithm implemented by the secondary user.

Discount rate γ of cumulative reward	0.5
Learning rate of actor	0.0001
Learning rate of critic	0.0003
Update parameter of target networks ρ	0.001
TD3 delayed update of actor	1 actor update for 10 critic updates
Experience replay buffer size	$100,000$
Mini-batch size	128
State observation time span T_0	32 time slots
Reward coefficient β	0.05 (bit/s/Hz) / mW
Exploration noise w added to the action, decreasing during training	Start at $\sigma_w = 10$ mW
	$\sigma_{w,t+1} = 0.99995\sigma_{w,t}$

Results and Discussion

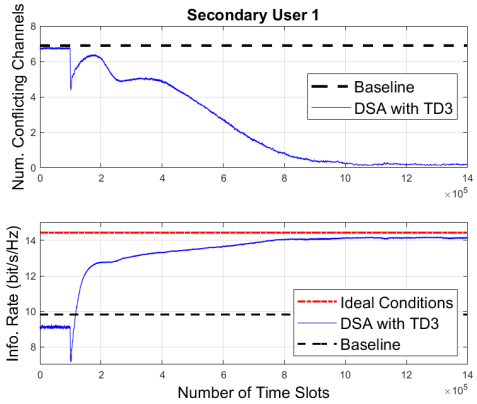


Figure: Secondary User 1 is active alone. Top: The number of frequency channels that conflict with the primary users over time slots of training iterations. Bottom: The information rate of the secondary user over time slots of training iterations.

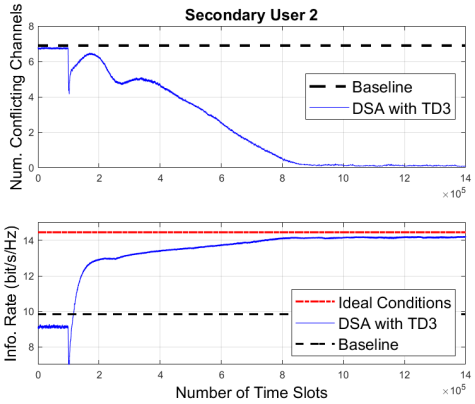


Figure: Secondary User 2 is active alone. Top: The number of frequency channels that conflict with the primary users over time slots of training iterations. Bottom: The information rate of the secondary user over time slots of training iterations.

Results and Discussion

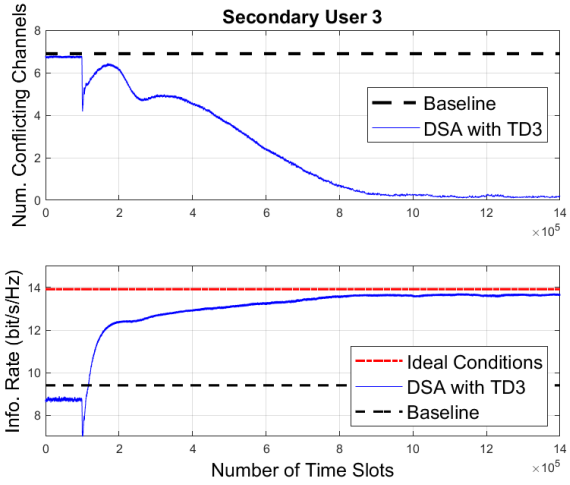


Figure: Secondary User 3 is active alone in the secondary user network. It performs dynamic spectrum access and sharing using the TD3 algorithm. Top: The number of frequency channels that conflict with the primary users over time slots of training iterations. Bottom: The information rate of the secondary user over time slots of training iterations.

Results and Discussion

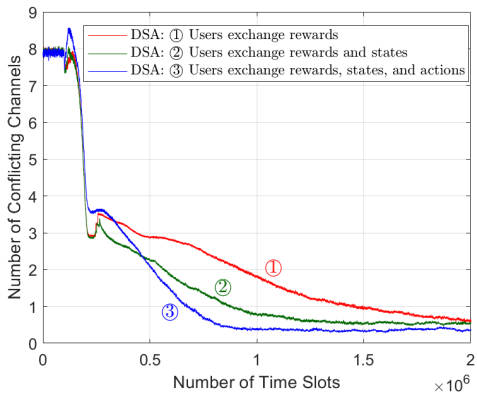


Figure: Three secondary users perform dynamic spectrum access and sharing. The plot shows the number of frequency channels that conflict with the primary users over time slots of training iterations.

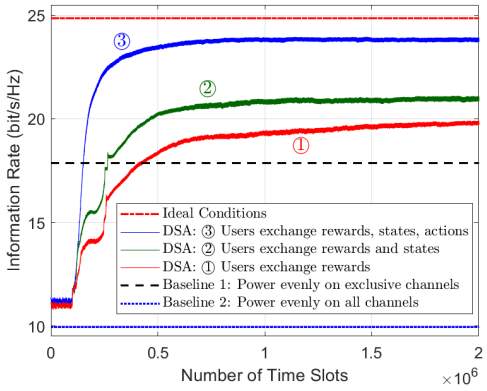


Figure: Three secondary users perform dynamic spectrum access and sharing. The plot shows the sum information rate of the secondary users over time slots of training iterations.

Conclusion and Future Work

- A framework of actor-critic deep deterministic policy gradient algorithm is tailored for dynamic spectrum access and sharing, and the deep neural networks are designed.
- Multiple secondary users implement multi-agent deep reinforcement learning with various degrees of coordination among the secondary users. The algorithms are effective, enabling the secondary users to quickly establish transmission policies that achieve good spectrum utilization.
- For future work, we plan to investigate the dynamic spectrum access and sharing problem with more random channel usage and switching patterns. Additionally, we plan to investigate the robustness and convergence speed of the TD3 algorithm.