An Introduction to Deep Reinforcement Learning and its Application to Wireless Network Optimization

Dr. Nancy Nayak

Postdoctoral Research Associate

EEE Department, Imperial College, London

Reinforcement Learning (RL)

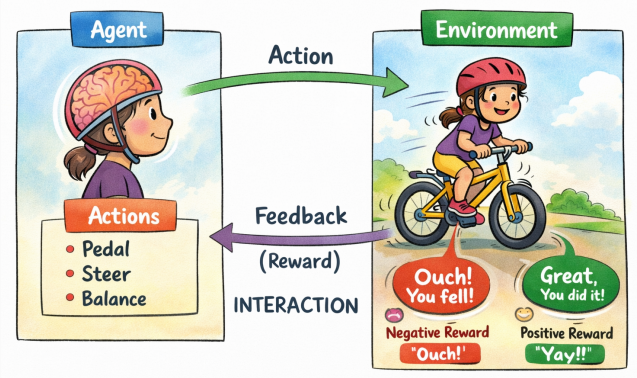


Figure 1: A kid is learning how to ride a bicycle[1]

What is RL? Learning through interactions with the environment to establish and remember "good" action policies

[1] Content and figures assisted by ChatGPT (OpenAI, 2026).

Why RL in Wireless Communications?

- Non-convex, complex, large-scale joint optimization objectives that are extremely difficult to solve in real time
- Relaxed or decomposed into smaller sub-problems - deviation from real-world conditions
- DRL learns a policy directly from interaction with the wireless environment

RL Components	**In Wireless Context**
Agent	Base station
Environment	Consists of wireless channel between users, BS, and RIS
State	CSI, SINR, interference, past actions
Action	Beam alignment, power allocation, RIS phases
Reward	Sum-rate, energy efficiency, QoS

Markov Decision Process (MDP)

MDP is the mathematical foundation of RL

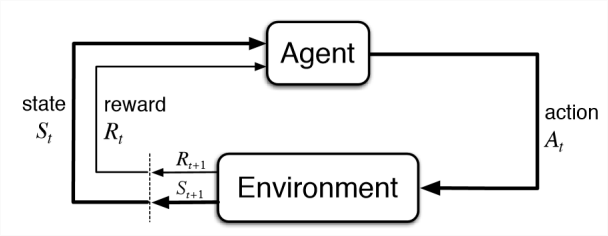


Figure 2: An RL agent learns by interacting with the environment and making sequential decisions.

- Observe the environment state S_t at time t
- Decisions taken by the agent are action A_t
- System state changes from S_t to S_{t+1} with probability $P(S_{t+1}|S_t, A_t)$
- Immediate feedback from the environment R_{t+1}

Markov Process

A Markov Process is defined by the tuple $(\mathcal{S}, \mathbf{P})$, where

- $\mathcal{S}$ is a finite set of states,
- $\mathbf{P} = (P_{ss'})$ is the state-transition probability matrix.

The transition probability from state s to state s' is $P_{ss'} = \Pr\left(S_{t+1} = s' \mid S_t = s\right)$.

Remarks:

- The above formulation assumes discrete time and discrete states - can also have continuous time and/or continuous states
- A Markov process may contain absorbing states — states that, once entered, cannot be left (game over in chess, terminal goal in navigation, task completion)

Markov Reward Process

A Markov Reward Process is a tuple $(\mathcal{S}, \mathbf{P}, R, \gamma)$, where

- S is finite set of states
- R is a reward function for every state s; $R_s = \mathbf{E}(R_{t+1}|S_t = s)$
- γ is a discount factor between 0 and 1

Remarks:

- Each state is associated with some reward
- Reward reflects how preferable to enter a particular state

Return of MRP G_t is the total discounted reward for the MRP starting from time step t:

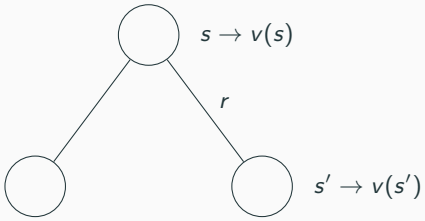
$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

Bellman Equation for Markov Reward Process

The state-value function $v(s)$ for any state s of an MRP includes:

- Immediate reward R_{t+1}
- Discounted value of the next state S_{t+1}: $\gamma v(S_{t+1})$

$$
\begin{aligned}
v(s) &= \mathbb{E}[G_t \mid S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \mid S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \cdots) \mid S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s] \\
&= R_s + \gamma \sum_{s' \in S} P_{ss'} v(s').
\end{aligned}
$$



Markov Decision Process

A Markov Decision Process (MDP) is a tuple $(\mathcal{S}, A, \mathbf{P}, R, \gamma)$, where

- A is a finite set of actions
- $P = P^a_{ss'}$ is the action-dependent, state-transition probability matrix

  $P^a_{ss'} = P[S_{t+1} = s' | S_t = s, A_t = a]$
- $R = R^a_s$ is a reward function for every possible action a and state s

  $R^a_s = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$
- An **Action Policy** π is a probability distribution over actions for every state

  $\pi(a|s) = P[A_t = a | S_t = s]$ - fully defines the agent's decision making - only depends on the current state (not the history) of MDP
- If policies are stationary (i.e., time-independent), $A_t \sim \pi(.|S_t)$

Relating MDP to MRP

For an MDP $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ and a policy π:

- The state sequence $S_1, S_2, S_3, \ldots$ forms a Markov process $(\mathcal{S}, \mathbf{P}^\pi)$.
- The state–reward sequence $S_1, R_2, S_2, R_3, S_3, R_4, \ldots$ forms a Markov reward process (MRP) $(\mathcal{S}, \mathbf{P}^\pi, \mathbf{R}^\pi, \gamma)$,

where

$$P^\pi_{s,s'} = \sum_{a \in \mathcal{A}} \pi(a \mid s)\, P^a_{s,s'}, \text{ and}$$

$$R^\pi_s = \sum_{a \in \mathcal{A}} \pi(a \mid s)\, R^a_s.$$

State-Value and Action-Value Functions for MDP

The state-value function $v_\pi(s)$ of an MDP is the expected return starting from state s and following policy π:

$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] .$$

The action-value function $q_\pi(s, a)$ is the expected return starting from state s, taking action a, and then following policy π:

$$q_\pi(s, a) = \mathbb{E}_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s,\ A_t = a] .$$

Optimal Value Function

The optimal state-value function $v^*(s)$ is the maximum state value for any state s among all policies:

$$v^*(s) = \max_{\pi} v_\pi(s).$$

The optimal action-value function $q^*(s, a)$ is the maximum action value for any action a in state s among all policies:

$$q^*(s, a) = \max_{\pi} q_\pi(s, a).$$

The optimal policy can be found by maximizing over $q^*(s, a)$:

$$\pi^*(a \mid s) = \begin{cases} 1, & \text{if } a = \arg\max_{a \in \mathcal{A}} q^*(s, a), \\ 0, & \text{otherwise.} \end{cases}$$

Systems of Bellman Optimality Equations for MDP

Bellman Optimality Equations for State Values

For all states $s \in \mathcal{S}$, $v^*(s) = \max_a \left[R_s^a + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a \, v^*(s') \right]$.

Bellman Optimality Equations for State-Action Values

For each state–action pair (s, a), $q^*(s, a) = R_s^a + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a \max_{a'} q^*(s', a')$.

Direct relationship between state and action values

$$q^*(s, a) = R_s^a + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a \, v^*(s').$$

State transition probabilities and the reward function are assumed to be known.

From Value Iteration to Q-Learning

Value Iteration using dynamic programming (model-based)

$$V_{k+1}(s) = \max_a \left[R(s,a) + \gamma \sum_{s'} P(s'|s,a) V_k(s') \right]$$

Move from state-value to action-value

$$Q_{k+1}(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q_k(s',a')$$

Requires: known dynamics $P(s'|s,a)$, known reward model. What if we do not know the environment dynamics? - **Model free**.

From Value Iteration to Q-Learning

Replace expectation by a real sample

Observed transition: (s, a, r, s')

$$Q(s,a) \approx r + \gamma \max_{a'} Q(s', a')$$

Stochastic and Stable update rule

$$Q_{k+1}(s,a) = (1-\alpha) \underbrace{Q_k(s,a)}_{\text{current estimate}} + \alpha \underbrace{\left[r + \gamma \max_{a'} Q_k(s', a') \right]}_{\text{TD target}}$$

$$= Q_k(s,a) + \alpha \underbrace{\left[r + \gamma \max_{a'} Q_k(s', a') - Q_k(s,a) \right]}_{\text{temporal-difference (TD) error}}$$

Deep Q-Networks (DQN)

Why Q-learning fails: Q-table size grows exponentially with state/action dimensions

DQN: Approximate $Q(s, a)$ with a neural network: $Q_\theta(s, a) \approx Q(s, a)$. The network $Q_\theta(s, a)$ takes the state s as input and outputs the Q-values for all possible actions.

TD target in DQN:

$$\underbrace{y}_{\text{target Q-value}} = \underbrace{r}_{\text{immediate reward}} + \gamma \underbrace{\max_{a'} Q_{\theta^-}(s', a')}_{\text{best future Q-value}}$$

Stability is provided with experience replay buffer and target network.

Intelligent Actions with Actor-Critic

Motivation:

- DQN handles discrete actions only
- High-dimensional or continuous actions (e.g., beamforming, power) require a different approach

Actor-Critic architecture:

- **Actor:** policy network $A(s|\theta_a)$ – outputs actions given state s
- **Critic:** action-value network $Q(s, A(s|\theta_a)|\theta_c)$ – evaluates the action suggested by the actor

Actor-Critic with Parameterized Q

For the i^{th} sample $\{s_i, a_i, r_i, s_{i+1}\}$ in the episode, critic update (Bellman loss):

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\Big(r_i + \gamma Q(s_{i+1}, A(s_{i+1}|\theta_a)|\theta_c) - Q(s_i, A(s_i|\theta_a)|\theta_c)\Big)^2$$

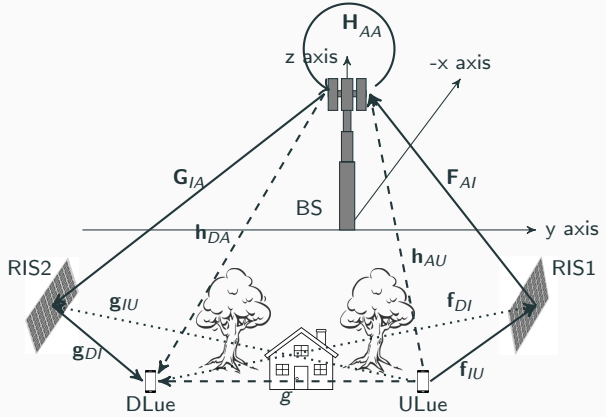
Actor update (Policy Gradient):

$$\nabla_{\theta_a} J(\theta_a) = \nabla_{\theta_a} Q(s, A(s|\theta_a)|\theta_c)$$

Critic evaluates the action; actor updates policy to maximize critic's Q-value.

Deep Reinforcement Learning to solve wireless problems

A DRL approach for RIS assisted Full Duplex Communication[2]



[2] Nayak, Nancy, Sheetal Kalyani, and Himal A. Suraweera. "A DRL approach for RIS-assisted full-duplex UL and DL transmission: Beamforming, phase shift and power optimization." IEEE Transactions on Wireless Communications (2024).

System model

- Full-duplex (FD) setup, one FD BS, one Half duplex (HD) ULue, one HD DLue
- The direct paths are blocked and, therefore have high path loss
- Two Reconfigurable Intelligent Surfaces (RIS) to facilitate communication when the users are not in LoS with the BS
- RIS1 and RIS2 are deployed as Uniform Planar Antenna (UPA) and have N_1 and N_2 reflecting elements, respectively
- The diagonal phase-shift matrices $\mathbf{\Theta}_U$ and $\mathbf{\Theta}_D$ of the two RISs are

$$\mathbf{\Theta}_U = \text{diag}\{e^{j\theta_1^U} \ \dots \ e^{j\theta_n^U} \ \dots \ e^{j\theta_{N_1}^U}\}, \text{ and } \mathbf{\Theta}_D = \text{diag}\{e^{j\theta_1^D} \ \dots \ e^{j\theta_n^D} \ \dots \ e^{j\theta_{N_2}^D}\} \quad (1)$$

System model

- BS has a uniform linear antenna (ULA) array with M_t transmit antenna elements and M_r receive antenna elements
- ULue and DLue are single-antenna HD user
- We also consider user to user and inter-RIS interferences
- s_U denotes the transmit signal from the ULue
- $p_U > 0$ denotes the transmit power of the ULue
- s_D denotes the transmit signal of the BS
- $p_A > 0$ denotes the transmit power of the BS

System model

- The received signal at the BS is given by

$$y_A = \mathbf{w}_R \mathbf{h}_{AU} \sqrt{p_U} s_U + \mathbf{w}_R \mathbf{F}_{AI} \mathbf{\Theta}_U \mathbf{f}_{IU} \sqrt{p_U} s_U + \mathbf{w}_R \mathbf{G}_{IA}^T \mathbf{\Theta}_D \mathbf{g}_{IU} \sqrt{p_U} s_U + \underbrace{\mathbf{w}_R \mathbf{H}_{AA} \mathbf{w}_T \sqrt{p_A} s_D}_{\text{residual SI}} + \mathbf{w}_R n_A, \tag{2}$$

- The signal received by the DLue is

$$y_D = \mathbf{h}_{DA} \mathbf{w}_T \sqrt{p_A} s_D + \mathbf{g}_{DI} \mathbf{\Theta}_D \mathbf{G}_{IA} \mathbf{w}_T \sqrt{p_A} s_D + \mathbf{f}_{DI} \mathbf{\Theta}_U \mathbf{F}_{AI}^T \mathbf{w}_T \sqrt{p_A} s_D + \underbrace{\mathbf{g}_{DI} \mathbf{\Theta}_D \mathbf{g}_{IU} \sqrt{p_U} s_U + \mathbf{f}_{DI} \mathbf{\Theta}_U \mathbf{f}_{IU} \sqrt{p_U} s_U}_{\text{inter-RIS}} + \underbrace{g \sqrt{p_U} s_U}_{\text{inter-user}} + n_D. \tag{3}$$

- Here transmit and receive beamformers are denoted by $\mathbf{w}_T$ and $\mathbf{w}_R$ respectively
- Highlighted term leads to a very high level of interference if not canceled

SINR and Data rate

- The Signal to Interference and Noise Ratio (SINR) at the BS and the DL user are given by

$$\begin{aligned}\gamma_{BS} &= \frac{p_U||\mathbf{w}_R(\mathbf{h}_{AU} + \mathbf{F}_{AI}\mathbf{\Theta}_U\mathbf{f}_{IU} + \mathbf{G}_{IA}^T\mathbf{\Theta}_D\mathbf{g}_{IU})||^2}{p_A||\mathbf{w}_R\mathbf{H}_{AA}\mathbf{w}_T||^2 + \mathbf{w}_R^2\sigma_A^2}, \text{ and} \\ \gamma_{DL} &= \frac{p_A||(\mathbf{h}_{DA} + \mathbf{g}_{DI}\mathbf{\Theta}_D\mathbf{G}_{IA}\mathbf{w}_T + \mathbf{f}_{DI}\mathbf{\Theta}_U\mathbf{F}_{AI}^T\mathbf{w}_T)||^2}{p_U||(g + \mathbf{g}_{DI}\mathbf{\Theta}_D\mathbf{g}_{IU} + \mathbf{f}_{DI}\mathbf{\Theta}_U\mathbf{f}_{IU})||^2 + \sigma_D^2},\end{aligned} \tag{4}$$

  respectively.

- Accordingly, the data rate at the DLue and the BS are given by

$$\begin{aligned}r_{DL} &= \log_2(1 + \gamma_{DL}), \text{ and} \\ r_{BS} &= \log_2(1 + \gamma_{BS}).\end{aligned} \tag{5}$$

Objective

The optimization problem is formulated as follows:

$$\mathcal{P}_1 : \max_{\Theta_D, \Theta_U, \mathbf{w}_T, \mathbf{w}_R, p_A, p_U} r_{BS} + r_{DL} \quad \text{s.t. } 0 \leq p_A \leq p_A^{max}, \quad 0 \leq p_U \leq p_U^{max}. \tag{6}$$

- Typically relaxed and broken into smaller subproblems
- Non convex high dimensional problem
- Conventional optimization and DL methods require CSI

Challenges in existing methods and our motivation

- Challenge - the **self-interference (SI)**
- Existing works assume negligible residual self-interference
- Or cancels the residual SI by using a beamformer that needs CSI
- What happens if the CSI is noisy and the residual SI is high/unknown?
- Proposed solution: **Two-stage Deep Reinforcement Learning (DRL) algorithm** that does not need CSI
- First stage: **cancel a major part of the SI interference by sending a pilot symbol**
- Second stage: **feed the data to an Intelligent agent** which learns to predict the RIS phaseshifts, beamformers and the transmit powers (agent: DRL algorithm)

Second stage: DRL based method

- The DRL agent located at the BS initializes the phaseshift, beamformers and transmit powers randomly, together called as an **action**
- Using these, the BS and ULue transmit and DLue and BS receives signals
- After receiving the signal at the BS, a significant amount of residual SI is cancelled[3] using a pilot signal (LSSIC); then the SINR at BS is obtained; used as **observation or state**
- The SINR at DLue (a scalar) is also calculated and the SINR is fed back to the agent to be used in second stage as **observation or state**
- Note, if an estimate of $\mathbf{H}_{AA}$ is already available at the BS, $\hat{h}$ can be estimated using this $\tilde{\mathbf{H}}_{AA}$; named as $\tilde{\mathbf{H}}_{AA}$ based SI cancellation (HSIC)

[3] DLue is HD, so no SI for DLue

Second stage: DRL based method

- Weighted sum of the corresponding data rates of these two SINRs are used as **reward**
- Based on the reward, the DRL agent predicts a different set of actions, according to which again the signals are transmitted from BS and ULue, achieving a new pair of UL and DL SINR (after LSSIC/HSIC at the first stage for UL) which takes the system to the **next state**
- The quality of the beamformers learned by the DRL agent in the second stage depends on the SI-cancelled signal from the first stage

MDP for our system model

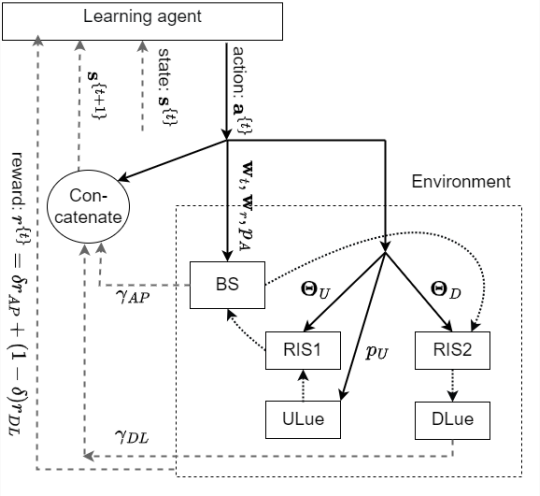


Figure 3: MDP formulation for RIS-based FD communication.

State, Action and Reward

- The predicted action for time step t is given by

$$\begin{aligned}\mathbf{a}^{\{t\}} = [&\theta_{11}^{\{t\}}, \theta_{12}^{\{t\}}, \ldots, \theta_{1N_1}^{\{t\}}, \theta_{21}^{\{t\}}, \theta_{22}^{\{t\}}, \ldots, \theta_{2N_2}^{\{t\}}, \text{real}(w_{T1}^{\{t\}}, w_{T2}^{\{t\}}, \ldots, w_{TM_t}^{\{t\}}),\\ &\text{imag}(w_{T1}^{\{t\}}, w_{T2}^{\{t\}}, \ldots, w_{TM_t}^{\{t\}}), \text{real}(w_{R1}^{\{t\}}, w_{R2}^{\{t\}}, \ldots, w_{RM_r}^{\{t\}}),\\ &\text{imag}(w_{R1}^{\{t\}}, w_{R2}^{\{t\}}, \ldots, w_{RM_r}^{\{t\}}), p_A^{\{t\}}, p_U^{\{t\}}].\end{aligned} \tag{7}$$

- New state which serves as next input to agent

$$\mathbf{s}^{\{t+1\}} = [\gamma_{BS}^{\{t\}}, \gamma_{DL}^{\{t\}}, a^{\{t\}}] \tag{8}$$

- The job of the RL agent is to learn the policy $\boldsymbol{\pi} : \mathcal{S} \rightarrow \mathcal{A}$ from the observations corresponding to each of the actions while maximizing the return $\mathbb{r}^{\{t\}}(\gamma) = \sum_{t'=t}^{\infty} \gamma^{t'-t} r^{\{t\}}(\mathbf{a}^{\{t\}}, \mathbf{s}^{\{t\}})$ where $\gamma \in [0, 1]$ is the discounting factor.

Deep Deterministic Policy Gradient (DDPG)

DDPG is an off-policy, model-free, actor–critic RL algorithm.

- An **actor-network** $\mathbb{A}$ parameterized by ω^a which predicts the action $\mathbf{a}^{\{t\}}$ based on the current state $\mathbf{s}^{\{t\}}$
- Gaussian noise is added to the action for exploration
- A **critic network** $\mathbb{C}$ parameterized by ω^c which estimates $Q(\mathbf{s}^{\{t\}}, \mathbf{a}^{\{t\}})$ that is essentially the quality of the action taken by actor-network $\mathbb{A}$
- A target actor and a target critic networks for stable updates
- Agent encourages actor-network to take **better actions through its feedback**
- The critic network $\mathbb{C}$ trains itself for **better prediction by observing the rewards after each action** [4]

[4] For more details regarding how these networks are trained, please refer to the paper.

Proposed actor network

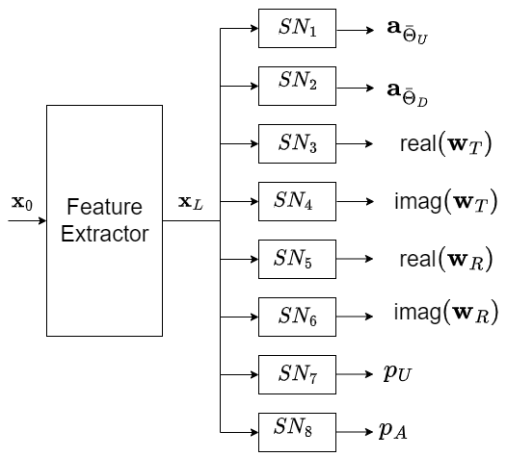


Figure 4: The proposed action predictor network. SN represents sub-network

DDPG

- Maintains a replay buffer $\mathcal{B}$ of finite size τ to get **stable, uncorrelated gradients** for policy improvement, and samples the observations from the buffer in mini-batches to update the parameters.
- At each timestep, the state $\mathbf{s}^{\{i\}}$ and the action taken $\mathbf{a}^{\{i\}}$ along with the reward obtained $r^{\{i\}}$ and the next state $\mathbf{s}^{\{i+1\}}$ is stored as an experience $(\mathbf{s}^{\{i\}}, \mathbf{a}^{\{i\}}, r^{\{i\}}, \mathbf{s}^{\{i+1\}})$ in $\mathcal{B}$.
- Uses target networks with parameters $\bar{\omega}^a$ and $\bar{\omega}^c$ to avoid divergence in value/return estimation
- Single-Step TD Target (Return/Reward Approximation) using target networks

$$y^{\{i\}} = r^{\{i\}} + \gamma \mathbb{C}(\mathbf{s}^{\{i+1\}}, \mathbb{A}(\mathbf{s}^{\{i+1\}}|\bar{\omega}^a)|\bar{\omega}^c). \tag{9}$$

DDPG critic update

- We observe a reward $r^{\{i\}}$ after taking an action $\mathbf{a}^{\{i\}}$,
- The loss function used for critic is,

$$\mathcal{L} = \frac{1}{N} \sum_i \left(y^{\{i\}} - \mathbb{C}(\mathbf{s}^{\{i\}}, \mathbf{a}^{\{i\}} | \boldsymbol{\omega}^c) \right)^2, \tag{10}$$

- $\mathbb{C}(\cdot)$ is the Q value predicted by critic network with parameter $\boldsymbol{\omega}^c$ corresponding to $\mathbf{s}^{\{i\}}$ and $\mathbf{a}^{\{i\}}$ before seeing the reward.
- The critic network parameters are updated as

$$\boldsymbol{\omega}^c \leftarrow \boldsymbol{\omega}^c - \eta_c \nabla_{\boldsymbol{\omega}^c} \mathcal{L}, \tag{11}$$

where $\eta_c \ll 1$ is the stepsize for the stochastic update.

DDPG actor update

- For the actor-network, we want to go towards an action that increases the Q value.
- Hence, we update parameters ω^a of actor network as

$$\omega^a \leftarrow \omega^a + \eta_a \frac{1}{N} \sum_i \left(\nabla_{\omega^a} \mathbb{A}(s) \nabla_a \mathbb{C}(s,a)|_{a=\mathbb{A}(s)} \right), \tag{12}$$

  where $\eta_a \ll 1$ is the update stepsize.

- Finally, the target network parameters are updated in every U timestep to provide stable value estimates using an exponentially weighted update as $\bar{\omega}^c \leftarrow \lambda \omega^c + (1-\lambda)\bar{\omega}^c$, and $\bar{\omega}^a \leftarrow \lambda \omega^a + (1-\lambda)\bar{\omega}^a$, with $\lambda \ll 1$.

Our method and baseline competitor methods

- **RandPSBF**: Agent does not receive any SINR feedback from the environment; predict the RIS phases, the beamformers, and transmit powers randomly
- **OUPSBF**: Makes use of the same DRL framework as ours, except for the action-noise where it adds the Ornstein Uhlenbeck noise to the RIS phases and beamformer
- Proposed Minimum Signalling Feedback (MSF) DRL method with LSSIC and HSIC (**MSF-DRL-LSSIC** and **MSF-DRL-HSIC**):
  - The critic network and the feature extractor actor-network are feed-forward networks with two layers, each with 100 neurons.
  - At the beginning of every episode[5], the MSF-DRL agent chooses actions randomly.
  - MSF-DRL uses a Gaussian action noise with zero mean and linearly decaying standard deviation (SD) with an initial SD of 0.3 decaying over 50 episodes.

[5] Episode is an independent game or sequence of states where the agent and environment interacts. It starts at an initial state and ends at a terminal state.

Contd.

- **PerfCSI-DRL** and **NoisCSI-DRL**:
  - A hypothetical experiment called "PerfCSI-DRL" to predict only the RIS phases where the perfect CSI knowledge including residual SI is available to the agent
  - The agent calculates the beamformers based on the ZF and MMSE principle that needs perfect CSI
  - Periodically receive the CSI to calculate the beamformers and therefore incurs overhead
  - CSI estimation methods may not be exact - NoisCSI-DRL

Comparison with static UE

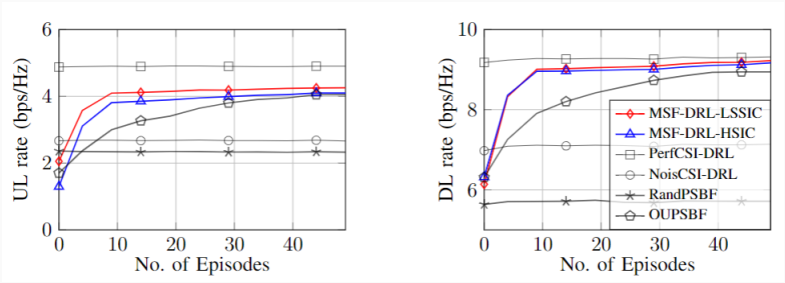


Figure 5: Rate evolution during learning in the static UE scenario. MSF-DRL-LSSIC and HSIC learns to predict and tries to reach the benchmark PerfCSI-DRL, performs much better than the case of NoisCSI-DRL.

Results with quantized phase MSF-DRL-LSSIC

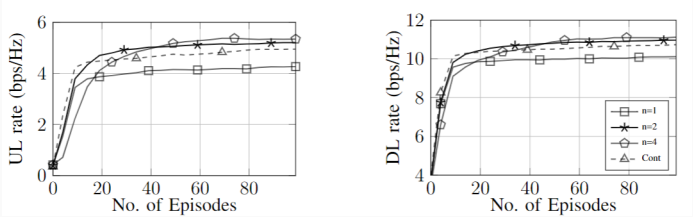


Figure 6: Effect of quantization on phaseshifts of RISs on MSF(Q)-DRL methods. When $n = 1$, the phase-shifts can take $Q = 2^n = 2$ values $\{0, \pi\}$; for $n = 0$, the phase-shifts are fixed to 0 and only the beamformers are learned.

Reduced signaling with two-stage DRL algorithm

- Uses only one **scalar feedback**, i.e., the weighted sum of UL and DL SINR from the environment instead of CSI which is in the form of Matrix for MIMO channels

Table 1: Proposed methods reduces signaling

Methods	FLOPs	Needs CSI, SI (#values)	Signaling from BS to RIS (in bits)
PerfCSI-DRL	2.9×10^4	Yes (813)	$64N_1 + 64N_2 = 4608$
MSF-DRL	3.3×10^4	**No** (2)	$64N_1 + 64N_2 = 4608$
MSF(Q)-DRL	5.46×10^4	**No** (2)	$2N_1 + 2N_2 = \mathbf{144}$

Future Directions

- Task aware quantization/pruning to make edge AI perform on par with SOTA
- Exploit geometry for faster convergence and training
- Hybrid algorithms which combine DL with traditional methods
- Continual learning to handle non stationary environments
- Robustness to naturally occurring adversarial samples in the context of wireless

Thank you!

Alternative solution PerfCSI-DRL using MRC-ZF - needs CSI

- Maximize sum of UL and DL SINR by maximizing the SNR towards reception

$$\begin{aligned} \mathcal{P}_2: \quad & \max_{\mathbf{w}_T} \; r_{DL} + r_{BS} \\ & \text{s.t.} \; ||\mathbf{w}_R^{MRC} \mathbf{H}_{AA} \mathbf{w}_T||^2 = 0, \quad ||\mathbf{w}_T||^2 = 1, \end{aligned} \tag{13}$$

where

$$\mathbf{w}_r^{MRC} = \frac{(\mathbf{h}_{AU} + \mathbf{F}_{AI}\mathbf{\Theta}_U\mathbf{f}_{IU} + \mathbf{G}_{IA}^T\mathbf{\Theta}_D\mathbf{g}_{IU})^H}{||\mathbf{h}_{AU} + \mathbf{F}_{AI}\mathbf{\Theta}_U\mathbf{f}_{IU} + \mathbf{G}_{IA}^T\mathbf{\Theta}_D\mathbf{g}_{IU}||}. \tag{14}$$

- Note that the knowledge of every interferer is not available so $\mathbf{g}_{IU}$ is not available

Contd.

- We want to minimize the self-interference using Zero-Forcing. The precoder $\mathbf{w}_T$ is in the orthogonal complement space of $\mathbf{w}_R^{MRC}\mathbf{H}_{AA}$. The orthogonal projection onto the orthogonal complement of the column space of $\mathbf{w}_R^{MRC}\mathbf{H}_{AA}$ is given by[6]

$$\Pi^{\perp}_{\mathbf{H}^{\dagger}_{AA}\mathbf{w}_R^{MRC\dagger}} = \mathbf{I}_{M_t} - \mathbf{H}^{\dagger}_{AA}\mathbf{w}_R^{MRC\dagger}(\mathbf{w}_R^{MRC}\mathbf{H}_{AA}\mathbf{H}^{\dagger}_{AA}\mathbf{w}_R^{MRC\dagger})^{-1}\mathbf{w}_R^{MRC}\mathbf{H}_{AA}.$$

- The optimal solution for transmit beamforming is

$$\mathbf{w}_T^{ZF} = \frac{\Pi^{\perp}_{\mathbf{H}^{\dagger}_{AA}\mathbf{w}_R^{MRC\dagger}}(\mathbf{h}_{DA} + \mathbf{g}_{DI}\mathbf{\Theta}_D\mathbf{G}_{IA} + \mathbf{f}_{DI}\mathbf{\Theta}_U\mathbf{F}_{AI}^T)^{\dagger}}{||\Pi^{\perp}_{\mathbf{H}^{\dagger}_{AA}\mathbf{w}_R^{MRC\dagger}}(\mathbf{h}_{DA} + \mathbf{g}_{DI}\mathbf{\Theta}_D\mathbf{G}_{IA} + \mathbf{f}_{DI}\mathbf{\Theta}_U\mathbf{F}_{AI}^T)^{\dagger}||}. \tag{15}$$

- This work can be extended to multiple users, too, and precoding will help us to beamform towards every user using the same antenna array.

[6] $\dagger$ represents conjugate transpose