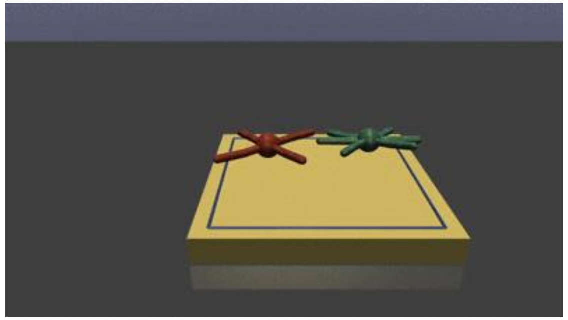
Sample Efficient Prioritized Experience Replay for Adversarial Environments

Tianyang Duan, Zongyuan Zhang, Zekai Sun, Heming Cui

Motivation

Robot soccer



RoboSumo

➢ **Adversarial environment**
- Non-stationary environment
- Zero-sum game
- Rapidly changing conditions and unpredictable opponents

➢ **Reinforcement learning(RL)** shows promising results in adversarial environments.
- Learn through interactions between agents and environments.
- Require a large number of trials and errors to converge

➢ **Sample efficiency** of RL needs to be improved in adversarial environments.
- Reduce the amount of data required for reaching target performance level.

Experience Replay

- **Experience replay** stores past experiences in a replay buffer and sampled randomly during training to improve the sample efficiency.

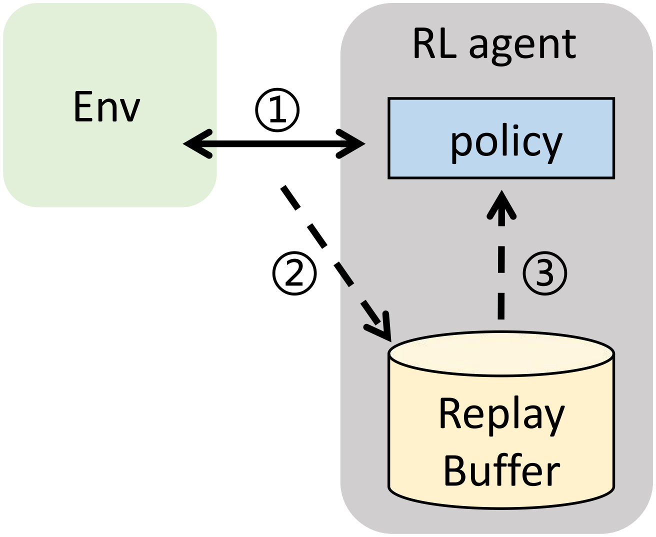
① Agent interacts with the environment.

- Agent receives a state S and out put action A
- Environment transforms to state S' and give a reward R

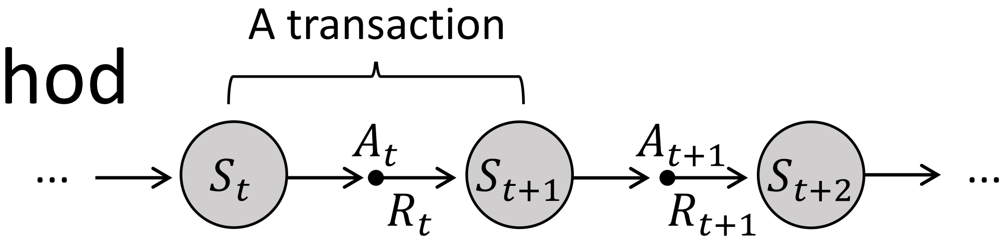
② Store the transaction (S, A, R, S') in replay buffer.

③ Sample a batch of transactions to train.

- **However, random sampling degrade sample efficiency, especially in non-stationary environments.**



Existing Prioritized Experience Replay Method



- **Prioritized Experience Replay(PER)** gives priority to the "important" transactions.
  - $Q(S_t, A_t)$: state-action value function, denotes the expected return with state S_t and action A_t

$$Q(S_t, A_t) \approx \sum_{k=t}^{T} R_k = R_t + \sum_{k=t+1}^{T} R_k \approx R_t + Q(S_{t+1}, A_{t+1})$$

  - Temporal Difference (TD)： The difference of expected and actual state-action value

$$TD\_error = R_t + Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)$$

  - Priority is calculated based on the TD error of the transaction.
  - Why TD error: TD error reflects the degree of deviation of a sample from the policy

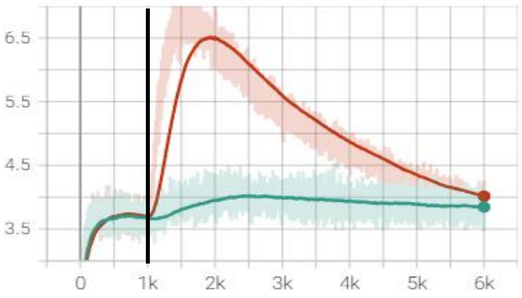
Problems of Existing PER Method in Adversarial Environment

High TD error

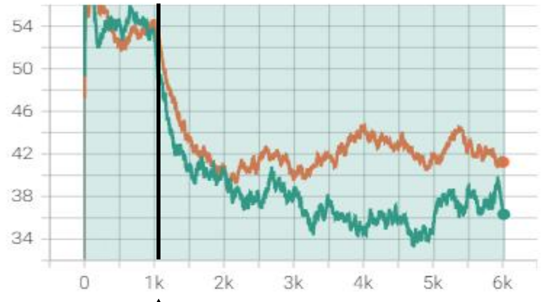
State-action space edge

Changes in environment

TD error



Reward



Environment changes at 1k step
Orange: motion friction
Green: opponent strategy change

- Prioritized transctions at the edge of the state-action space(with high TD error) hinder the fast convergence of RL[1].
- In the later training stages, outdated samples are prioritized which introduce noise.
- TD error cannot fully capture opponent strategy changes.
- **Thus, TD-error based PER approach is not ideal for adversarial environments**

[1] Cao, Xi, et al. "High-value prioritized experience replay for off-policy reinforcement learning." 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI). IEEE, 2019.

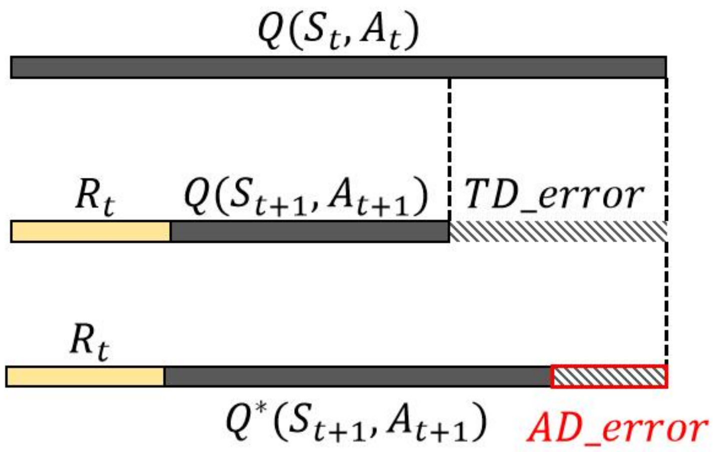
From Temporal Difference to Adversarial Difference

- We introduce the concept of Adversarial Difference (AD) as a measure to compare the expected returns from old and new opponents.
- Why AD:
  - TD indicates deviation of a transaction **versus the same opponent**
  - AD indicates deviation of a transaction **versus the different opponent**

- We propose **Adversarial Prioritized Experience Replay (APRE)** based on AD.
  - For improving sample efficiency in adversarial environments
  - APRE faces the following challenges:
    ① How to define old and new transactions
    ② How to trade off between old and new transaction
      - new experience: limited amount, help quickly converge
      - old experience: improving stability, avoiding catastrophic forgetting and overfitting

Key Idea: Adversarial Difference

- We propose Adversarial Difference(AD) which compares the expected returns between the old opponent and the new opponent.
  - A transaction $(S_t, A_t, R_t, S_{t+1}, A_{t+1})$
  - $AD\_error = R_t + Q^*(S_{t+1}, A_{t+1}) - Q(S_t, A_t)$
  - Q: the state-action value function of current opponent, Q^*: the state-action value function of old opponent



$$Q(S_t, A_t) \approx \sum_{k=t}^{T} R_k = R_t + \sum_{k=t+1}^{T} R_k$$ ← Expected return versus new opponent

$$TD\_error = R_t + Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)$$ ← The difference of expected return at time step t and $t+1$ versus new opponent

$$Q^*(S_{t+1}, A_{t+1}) = \sum_{k=t+1} R_k^*$$ ← Expected return versus old opponent

$$AD\_error = R_t + Q^*(S_{t+1}, A_{t+1}) - Q(S_t, A_t)$$ ← **The difference of expected return versus old opponent and new opponent**

$$\approx \sum_{k=t+1}^{T} R_k^* - \sum_{k=t+1}^{T} R_k$$

APER: Adversarial Prioritized Experience Replay

- **Ratio sampling** from "new" section and "old" section
  - Stage1: Start by detecting a rise in TD error, λ start from 1 and decays with training steps
  - Stage2: Starting from new/sum > λ
  - Stage3: $Starting\ from$ new/sum = 1

- **Prioritized sampling** from each individual section

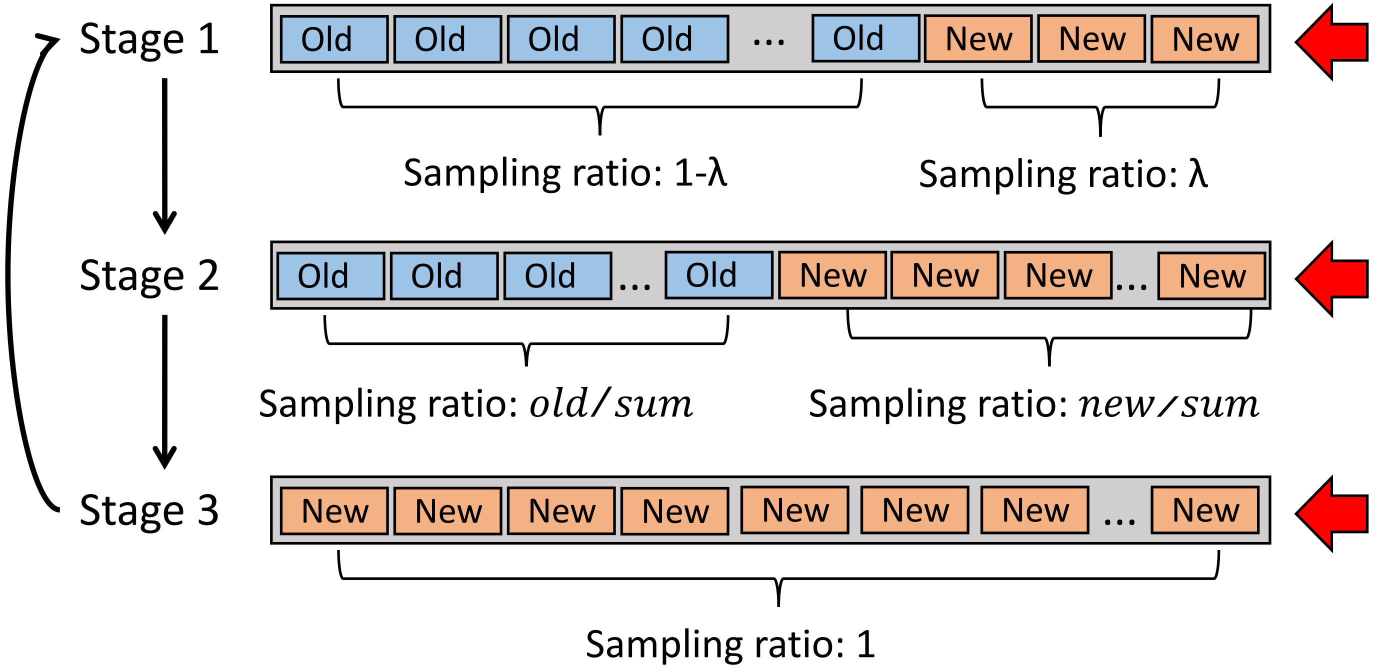
  In Stage 1,2:
  - old sample priority ∝ 1/|AD_error|
  - new sample priority ∝ |AD_error|

  Stage 3:
  - priority ∝ |TD_error|

The rise of TD error can be observed when the opponent change

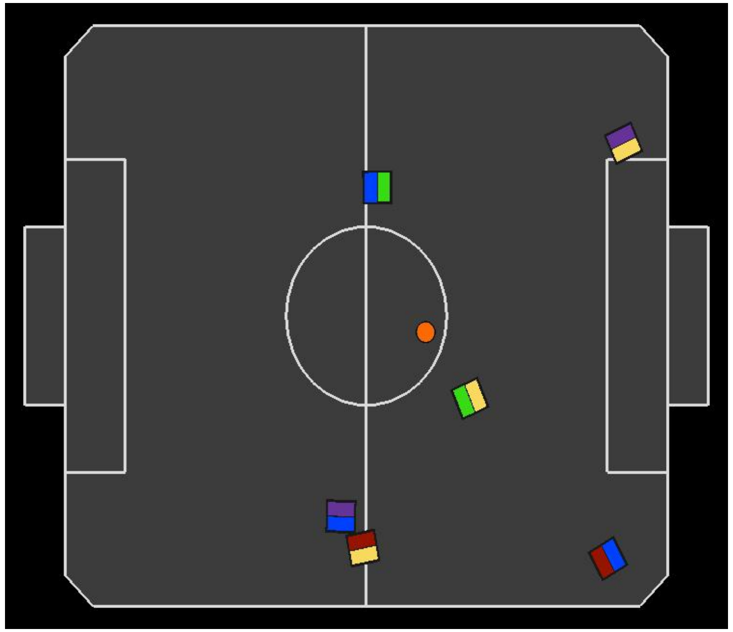
Replay Buffer

Store a transaction



Implementation and Evaluation

- **Evaluation environment:**
  - rSoccer - IEEE VSSS environment to simulate a robot soccer scenario.
- **RL algorithm:**
  - TD3 (Twin Delayed DDPG): a state-od-the-art RL algorithm for continuous control tasks
- **Baseline:**
  - Vanilla-ER
  - Prioritized Experience Replay (PER)
  - Combined Experience Replay (CER)
- **Evaluation settings:**
  - 3 vs 3 robot soccer match
  - Each agent controls one player
  - Follow the rewards shaping in rSoccer
  - Each match lasting 200 time steps



Evaluation Questions

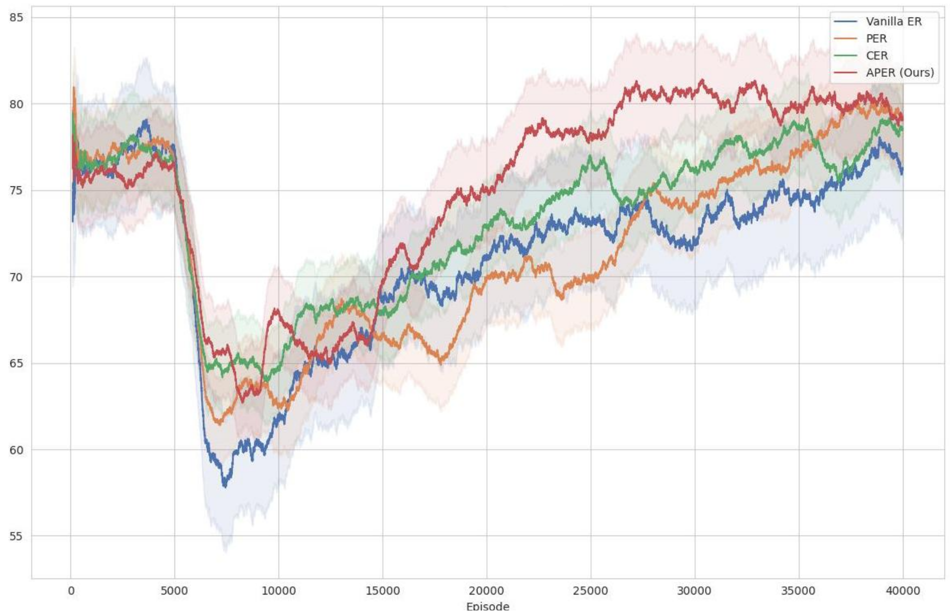
- Does the method improve sample efficiency in adversarial environments?
- How does AD-error-based priority help to reduce the efficiency damage caused by outdated transactions?
- What is the method's sensitivity to critical sampling-related hyperparameters?

End-to-end Performance

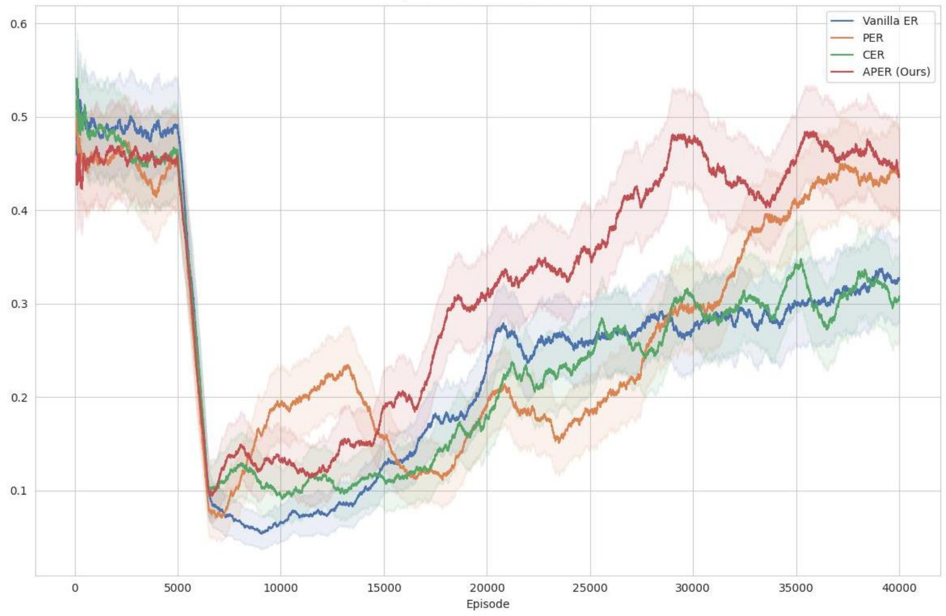
- **APER** (Red line) take a minimum of episodes to restore the average reward before changing the opponent policy, with **a 46.64% improvement to Vanilla ER**.
  - Play in the above experimental environment
  - At 5000 episodes, the opponent's strategy changed from offensive to defensive
  - All methods start from the same pre-trained model
  - All methods use the same hyperparameters, with learning rate = 1×10^{-4}, batch size = 1024, buffer size = 5×10^{5}.

Replay	Episodes spent to restore the original rewards	Improvement over vanilla ER
APER (Ours)	14695	46.64%
CER	19698	28.48%
PER	22064	19.89%
Vanilla ER	27543	-

Reward



Goal



Conclusion

- In this talk, we present APER, a sample efficient prioritized experience replay for adversarial environments.
  - We propose adversarial difference (AD), which compares the expected gains between different opponents.
  - APER detects the changes of the opponent by the rise of TD error.
  - APER uses ratio sampling to trade off between the new transactions and old transactions.
  - APER samples from new and old transactions according to AD and 1/AD priority, respectively.
- APER requires a minimum of episodes to restore the average reward before changing the opponent policy compared to baselines.
- Plan to submitted to 26th European Conference on Artificial Intelligence (ECAI 23) in April.

Future work

- Short-term work
  - Evaluate APER in other adversarial environments (e.g. RoboSumo, Pong)
  - Evaluate APER with other RL algorithms (e.g. A2C,DDPG)
- Long-term work
  - Apply APER in real robotic multi-agent reinforcement learning environments
    - Experience replay sharing in distributed training
  - Apply APER in Deep Reinforcement Learning