Experience Replay in Reinforcement Learning – how to recycle data efficiently

Jamie Kang
(jamiekang@stanford.edu)
Management Science & Engineering, Stanford University

Overview

Motivation:

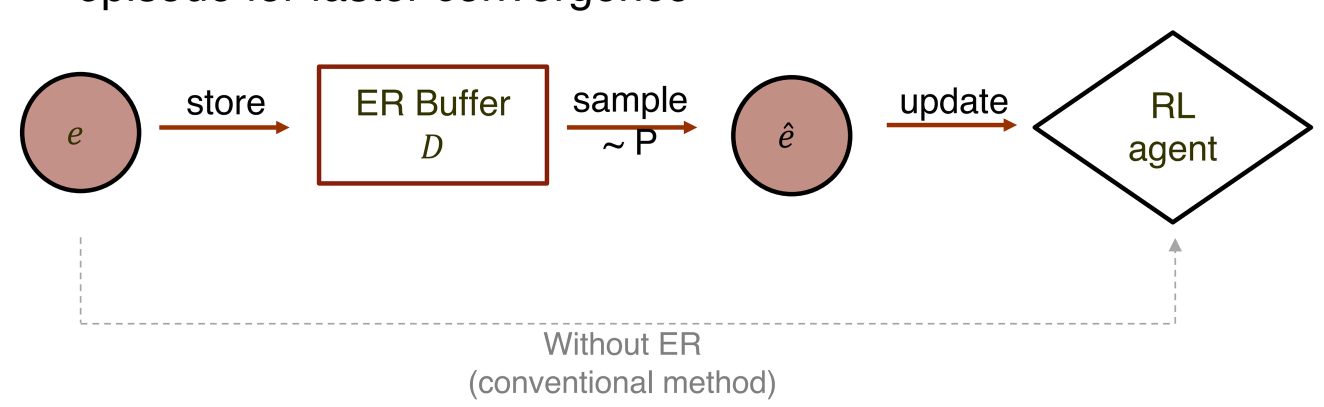
- An Online RL agent observes a stream of transitions and learns from each experience e = (s, a, r, s') incrementally
- Once used for update, each experience is discarded → waste
- Experience Replay has recently received much attention in the literature as a way to
  1) Efficiently recycle experience data
  2) Reduce correlation in training when using value function approximation (eg. DQN)

Goal:

- We focus on **1)**
- Investigate how efficiently Experience Replay and its variants use data
- Compare their effects through simulation
- Develop simple new algorithm that can better take advantage of unusual and/or rare data

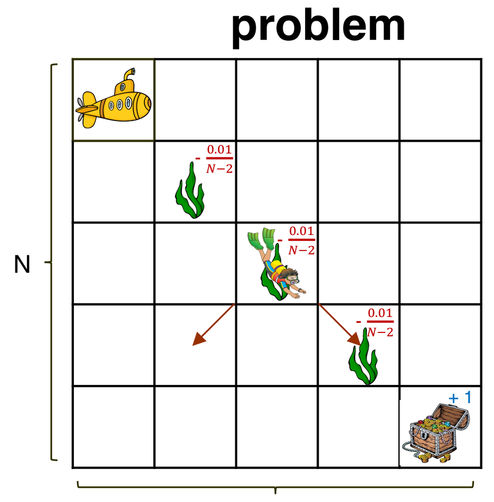
Method:

- To better focus on the "data recycle" aspect rather than correlation reduction, we use a tabular RL algorithm
- In particular, we use Q-Learning with ε-greedy exploration
- Step size in the Q-update starts big and is decreased over episode for faster convergence



Environment

- **(Simplified) Deep Sea problem**



|S| = N x N
|A| = 2 (A = {1,2})

Optimal regret per episode: 0.99
→ Obtained by iteratively choosing 'R' action (unknown) at each state

Each diagonal cell incurs - $\frac{0.01}{N-2}$ cost
→ Need rigorous exploration & efficient use of data

- Simple yet insightful environment that suits our purpose

Algorithms

Algorithm 1. Experience Replay

Initialize buffer D
Initialize value function matrix Q
for episode = 1, 2, … **do:**
  Sample initial state s
  while *not terminate* **do:**
    Pick action a according to ε-greedy using Q
    Observe reward r and next state s'
    Store the experience $e = (s, a, r, s')$ into D
    Sample an experience $\hat{e} = (\hat{s}, \hat{a}, \hat{r}, \hat{s'})$ from D
    Update Q using $\hat{e}$
    Move to next state s'
  end while
end for

1. Experience Replay

- Sample uniformly
- Issue: All experiences treated equally, including some experiences that may be more useful.

2. Prioritized Experience Replay (PER)

- Instead of sampling uniformly, prioritize experiences that agent will learn more from.
- Use TD-error (δ) as proxy
  -> given when using Q-learning

$$\delta = r + \max_{a' \in A} Q(s', a') - Q(s, a)$$

$$p \cong |\delta|, \qquad P(i) = \frac{p_i^\alpha}{\sum_{i=1}^{|D|} p_i^\alpha}$$

and sample experience using distribution P.

- Experience with higher TD-error (i.e. "unexpected" transition) will be recycled more frequently.

3. Asymmetric Prioritized Experience Replay (APER)

- PER with positive TD-error incentive

$$p \cong |\delta|(1 + \beta \cdot I\{\delta > 0\})$$

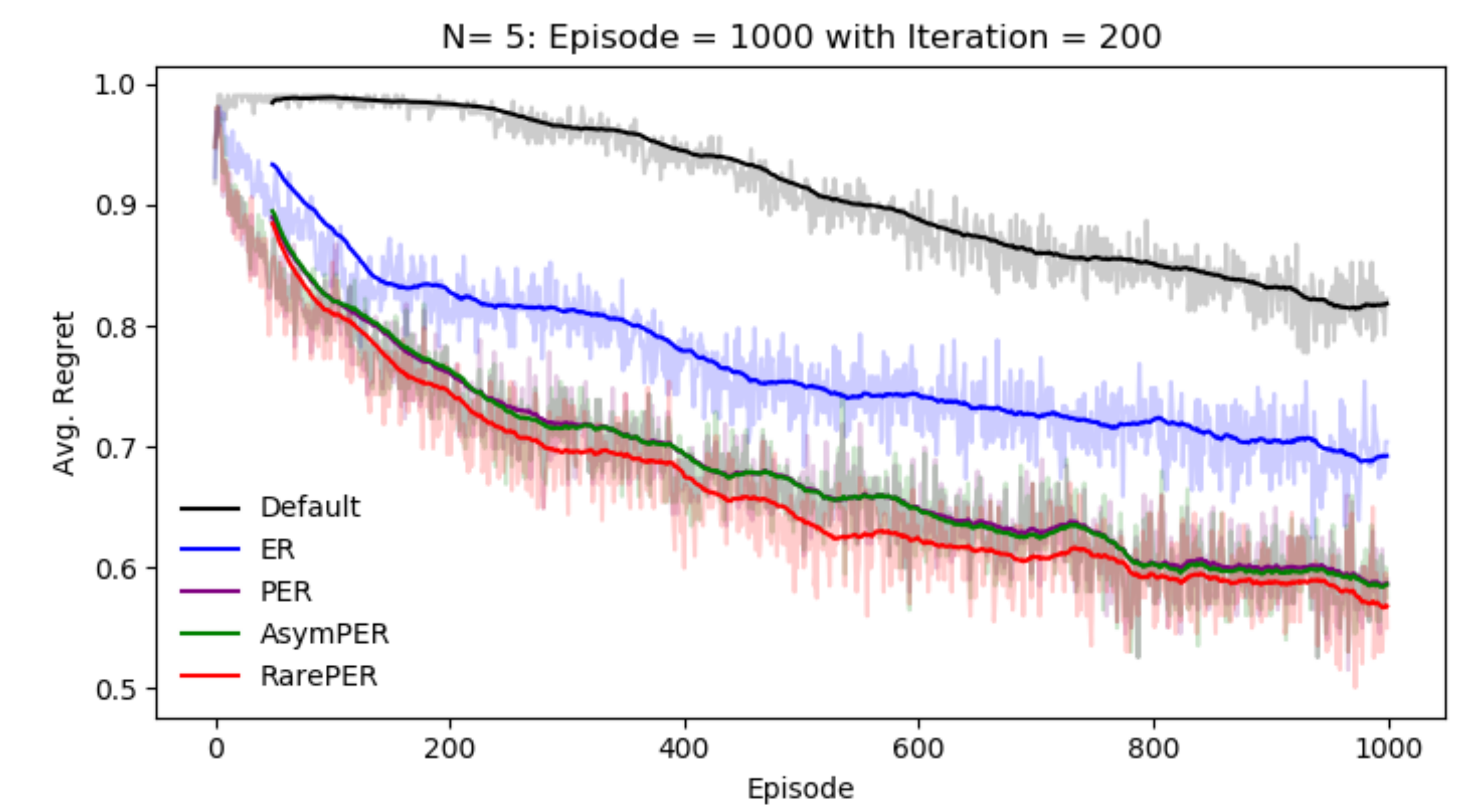
4. Rare Prioritized Experience Replay (RPER)

- PER with rare transition incentive

$$\tilde{p} \cong \frac{|\delta|}{F(s, a, r, s')}$$

where F matrix records the # of observation of each transition tuple (s, a, r, s')
→ rare experience has small F, and is prioritized over other experiences with the same TD-error

Results



- For simplicity we only present the results from tests on a N=5 Deep Sea world, but similar trends have been observed for larger N values.
- Experience Replay (regardless of its priority type) significantly reduces average regret per period.
  → Without Experience Replay, average regret remains far above 80% after 1,000 episodes.
  → Most efficient is RPER, reducing average regret to below 57% after 1,000 episodes.
- APER does not seem to be more useful than PER.
- Experience Replay proves to be very efficient in the tabular context as it is in generalized function approximation methods

Future work

In the order of personal curiosity

- Understand the mathematics (proof, bound, complexity) behind RPER
- Test the algorithms on problems with sparse MDP structure. We expect that RPER will be even more useful in this setting.
- Instead of ε-greedy, combine with deep exploration methods (eg. PSRL)
- In preliminary testing on Cart Pole problem, the experience algorithms did not perform as well.
  → Need better parameters?
  → Or are the algorithms neglecting some crucial component of this problem?
- Test RPER for value function approximation algorithms (eg. DQN)
- Implement buffer with more efficient data structure (eg. Sum Tree) to reduce computation cost

Conclusion

- We develop and examine two variants of Prioritized Experience Replay: APER and RPER
- Both variants can be implemented as add-ons at a small cost
- Simulation results show that RPER uses experience data most efficiently

References

[1] Ruishan Liu and James Zou. The effects of memory replay in reinforcement learning. arXiv preprint arXiv:1710.06574, 2017.
[2] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. Nature, 518(7540):529, 2015.
[3] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. arXiv preprint arXiv:1511.05952, 2015.
[4] Shangtong Zhang and Richard S Sutton. A deeper look at experience replay. arXiv preprint arXiv:1712.01275, 2017.