Prioritized Experience Replay

Reminders

- TD error: $\delta_t = r_t + \gamma Q_{tgt}(s_t, \mu_{tgt}(s_t)) - Q(s_{t-1}, a_{t-1})$, and we have the transition $(s_{t-1}, a_{t-1}, r_t, s_t)$
- Experience replay was first mentioned by Lin in 1992
- Value/policy iteration algorithms belong to the family of planning algorithms

Overview

- Problems with regular RL algorithms:
  - Have strongly correlated updates that break i.i.d. assumption of many stochastic gradient-based algorithms,
  - Rapidly forgetting of useful but rare experiences
- Presented a deterministic and stochastic prioritization algorithms

Key ingredients

Prioritization with TD-error (deterministic)

- New transitions that arrive are assigned the highest priority (TD-error is not known)
- After fitting, they are returned back with the weight equal to the abs(TD-error)
- This method is expected to perform poorly when affected by noise - can create a TD peek, and with slow learning, a useless transition can be repeated multiple times
- Some transitions might not be replayed again
- Still shown to work a lot better than uniform sampling

Stochastic prioritization

- Interpolates between pure greedy and uniform sampling
- Sample from:

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$$

  for $\alpha = 0$, we get a uniform case, and $\alpha \to \infty$ the pure greedy case
- Importance sampling to reduce bias cased by non-uniform probabilities

Proportional case

- Priority: $p_i = \delta_i + \epsilon$

Rank based case

- Priority: $p_i = \frac{1}{\text{rank}(i)}$
- Likely to be more robust, since it is insensitive to outliers
- Awesome implementation trick: approximate as a binary heap

Comments

- Built on top of Prioritized Sweeping:
  - "To choose effectively where to spend a costly planning step, classic prioritized sweeping uses a simple heuristic to focus computation on the states that are likely to have the largest errors."
  - Here they basically recognize the importance of choosing which states to update next (in terms of value propagation) and create a general algorithm to do so
- Efficient implementations of each of the algorithms described in the paper

  - Only concern is how to sample from
- Prioritized replay speeds up learning by a factor of 2 **and** the overall performance on most of the tested scenarios
- On multiple runs across multiple games, proportional prioritization achieves a better maximum score, but rank based is better on average
- From the games tested, I think that rank based variant should work better in real world scenarios
- Also mentioned other prioritization schemes that might improve performance and robustness depending on the application
- **BUT**, what I do not understand is why we use the TD-error previously defined and why the TD-error is initially set to the max value, given that we could calculate it from beginning
  - Why don't we simply store the $Q_{tgt}(s, a)$ and assign to it a highest priority. Then, when we re-evaluate $Q_{tgt}(s, a)$, we set the priority to $Q_{tgt} - Q_{tgt}^{(}-)$, as this gives the actual TD-error