Formatting Instructions for RLC Submissions

Anonymous authors
Paper under double-blind review

Abstract

Hello

1 Introduction

In reinforcement learning, ...

2 Problem Setting

Reinforcement learning is often modeled as a Markov decision process (MDP) given by the tuple $M = (\mathcal{S}, \mathcal{A}, P, R)$. In this model, an agent and environment interact over a sequence of time steps t. At each time step, the agent receives a state $S_t \in \mathcal{S}$ from the environment, where $\mathcal{S}$ denotes the set of all possible states. The agent uses the information given by the state to select and action A_t from the set of possible actions $\mathcal{A}$. Based on the state of the environment and the agents behavior, i.e. action, the agent receives a scalar reward $R_t = R(S_t, A_t)$ and transitions to the next state $S_{t+1} \in \mathcal{S}$ according to the state-transition probability $P(s'|s, a) = P(S_{t+1} = s'|S_t = s, A_t = a)$ for each $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$.

The behavior of an agent is given by a policy $\pi(a|s)$, which is a probability distribution over actions given a state. The agent's goal is to learn the optimal policy, $\pi^\star$, which is the policy that maximizes the expected discounted return

$$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+1} + \cdots = \sum_{k=1}^{T-t} \gamma^{k-1} R_{t+k-1}$$

either for a discounted factor $\gamma \in [0, 1)$ when the task is continuing, $T = \infty$, or $\gamma \in [0, 1]$ and $T < \infty$ in episodic task.

Through the process of policy iteration, Monte-Carlo algorithms strive to maximize the expected return by computing value-functions that estimate the expected future returns. The state-value function is quantifies the agent's expected return starting in state s and following policy π, i.e. $v^\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$. When learning to control in RL, we often want to estimate the action-value function, which is the agent's expected return starting in state s, taking action a and then following policy π, i.e.

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t \,|\, S_t = s, A_t = a].$$

Often in RL, it is useful to use function approximation to learn the action-values.

3 Algorithm

Proposition 3.1. *Given tolerance $\tau \geq 0$ and a real-valued list xs of size N, Algorithm 1 returns an approximate sum Q that satisfies $|Q - sum(xs)| \leq \tau$.*

Algorithm 1 ADAPTIVE-QUADRATURE

Input: A list xs and tolerance $\tau \geq 0$
$N \leftarrow \text{length}(xs)$.
if N > 2 **then**
  $Q \leftarrow N \cdot (xs[0] + xs[-1])/2$.
else
  return $\text{sum}(xs)$.
end if
$\varepsilon \leftarrow |Q - \text{sum}(xs)|$
if $\varepsilon \geq \tau$ **then**
  $c \leftarrow \lfloor N/2 \rfloor$.
  $Q \leftarrow$ ADAPTIVE-QUADRATURE$(xs[:c], \tau/2)$ + ADAPTIVE-QUADRATURE$(xs[c:], \tau/2)$
end if
return Q.

Proof. This proof will follow by induction. Assume that $N < 3$. Then Algorithm 1 returns $Q = \text{sum}(xs)$. Therefore $|Q - \text{sum}(xs)| = |\text{sum}(xs) - \text{sum}(xs)| = 0 \leq \tau$ for all $\tau \geq 0$.

Now assume that for a fixed $N > 2$ and $\tau \geq 0$ it holds that $|Q - \text{sum}(xs)| \leq \tau$. □

A Experimental Details