Adaptive Temporal Resolutions in Continuous-Time Reinforcement Learning

February 28, 2024

1 Introduction

Questions:

- We talked about finding per-state scales, however when doing control it seems like the most useful scale would change over time as well – even for a fixed data budget. Are we looking to build something that can tackle this too?

Motivation:

- Typically, an agent interacts with an environment a pre-determined rate. This rate may not be ideal for the learning and performing in the environment. Furthermore, the ideal rate is likely to be state and budget dependent. In this work we propose to dynamically adapt the interaction rate for improved performance.
- Concern: we must pay for finding a good interaction rate.
  - This is true, we could not compete with someone having given us the right interaction rate.
  - However, this is not typically the case, especially considering that the right rate is dependent on the state, and potentially the stage in learning. This makes our method superior in cases where finding a

Intuitions:

- Intuitively, we should work with an environment (both in terms of acting in it and reasoning about it) at the highest sensible scale for good progress on learning. (See discussion later). We can refine timescales as needed as we get higher data budgets.
- To find a good scale: from a state, consider "action-value functions" that are also a function of action repetition for the first immediate action, and afterwards uses whatever action scale makes sense (Assume that rewards scales are taken care of.) If sticking to the action longer has value comparable to no repetition, it tells us we can work with the environment at that scale.

  - This seems related to the technique for numeric integration Alex mentioned.

- The above works in fully tabular situations, but if there is an underlying space with distances (e.g. the state space in mountain car) this could be exploited for generalization.

Techniques:

- bla

2 Preliminaries

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

The actual objective of the learner is to maximize 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 for $\gamma \in [0, 1]$ and $T < \infty$, i.e., in an episodic task. The latter return is influenced by the actions taken by the agent, thus its behavior defined through a policy $\pi(a|s)$, which is a probability distribution over actions given a state. Then the agent's goal is to learn the optimal policy $\pi^\star$, which is the policy that maximizes the expected return.

Through the process of policy iteration [**?**], Monte-Carlo algorithms strive to maximize the expected return by approximating the value-function, which is indeed defined as its expectation. The state-value function quantifies the agent's expected return starting in state s and following policy π, i.e. $v^\pi(s) = \mathbb{E}_{[\pi][G_t|S_t=s]}$. Given that the task in RL is to face the control problem going beyond prediction, 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.

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]. \tag{1}$$

In many RL problems of interest, the state-space is prohibitively large and function approximation is needed in order to enable sample-efficient learning. Note that the algorithmic ideas presented in this paper will be given as linear solution methods, however their applicability is broader as they can be extended to arbitrary function approximation schemes. Given a set of linear features $\phi : \times \to \mathbb{R}^d$ and a trajectory $S_1, A_1, R_1, \ldots, R_T, S_{T+1}$ collected following policy π in M, the action values can be approximated by solving the least-squares problem

$$\hat{\theta} = \operatorname*{arg\,min}_{\theta \in \mathbb{R}^d} \sum_{t=1}^{T} \Big(\phi(S_t, A_t)^\top \theta - G_t \Big)^2 \tag{2}$$

and assuming that the feature yield a good linear representation of the action-value function, it holds that $q_\pi(s,a) \approx \phi(s,a)^\top \hat{\theta}$.

2.1 Monte-Carlo Policy Iteration

The main objective in *policy iteration* is to maximize the expected return by selecting the policy that maximizes the approximated action-value function, which is learnt from observations collected via interaction with the environment. Given that the number of point in a trajectory negatively affects computational time of the procedure and memory usage, ideally this should be done using as few samples as possible. Monte-Carlo policy iteration works by first performing policy evaluation to learn the action-values using the returns and then acting greedily with respected to the learned action-values. Monte-Carlo policy iteration can be summarized as

1. (Policy evaluation): Collect $N \in \mathbb{N}$ trajectories $S_1, A_1, R_1, \ldots, R_T, S_{T+1}$ following policy π in MDP M. Then solve the least squares problem

$$\hat{\theta} = \arg\min_{\theta \in \mathbb{R}^d} \sum_{i=1}^{N} \sum_{t=1}^{T} \left(\phi(S_{t,i}, A_{t,i})^\top \theta - G_{t,i} \right)^2$$

   where $S_{t,i}$ corresponds to the t-th state in the i-th trajectory for $i \in \{1, \ldots, N\}$ and $t \in \{1, \ldots, T\}$.

2. (Greedify): Take the new policy to be the policy that is greedy with respect to the approximate action-values, i.e. $\pi(s) = \arg\max_{a\in} \phi(s,a)^\top \hat{\theta} \approx \arg\max q^{\pi'}(s,a)$ where π' is the previous policy.

The main question this paper aims to answer is if the number of samples from a trajectory used to approximate the action-value function can be reduced by means of an adaptive algorithm. Usually the system is assumed to have a fixed discretization step, and current algorithms makes use of all the samples they are given. However it has been recently established by [**?**] that using only a subset of the tuples of the trajectory yields a better result when performing Monte-Carlo value estimation. In their work, they show, both theoretically and empirically, that a simple uniform discretization over the trajectory, i.e. updating using every M-th tuple in the trajectory, leads to significantly better value estimation when the number of updates is fixed. In this work, we also propose further discretizing, or subsampling, the trajectory in order to learn with as few updates as possible. By simply considering a coarse discretization we might lose too much information about the behaviour of the agent, while keeping the whole trajectory would lead to a computationally expensive algorithm. We will show that adapting the discretization step allows the algorithm to use less samples while still producing a good approximation of the action-value function, and consequently achieving a good policy. In the next section we will introduce both a uniform scheme for discretizing the Monte-Carlo updates and an adaptive scheme for discretizing the updates.

3 Adaptive Quadrature

Adaptive quadrature refers to a family of algorithms that use small step sizes in the parts of the domain of integration where it is hard to get good accuracy and large step sizes in the parts of the domain of integration where it is easier to get good accuracy.

We illustrate this idea by using Simpson's rule to integrate $\int_a^b f(x)dx$ with error tolerance ε.

Step 1 Start by applying Simpson's rule, combined with Richardson extrapolation to get an error estimate, with the largest possible step size h. Namely, $h = (b-a)/2$, compute

$$f(a), \quad f\left(a+\frac{h}{2}\right), \quad f(a+h) = f\left(\frac{a+b}{2}\right), \quad f\left(a+\frac{3h}{2}\right), \quad f(a+2h) = f(b)\,. \tag{3}$$

4 Algorithm

5 Analysis

6 Numerical Experiments

7 Related Work

Keywords and places to look into

1. Semi Markov Decision Process
2. Temporal Abstraction of Reinforcement Learning
3. Continuous Time Control in Reinforcement Learning
4. Time Discretization for Reinforcement Learning
5. There is a whole line of work on action xrepetition

7.1 Time Discretization-Invariant Safe Action Repetition for Policy Gradient Methods

[**?**]

Main points:

- Learn policy (with PPO, TRPO, or A2C) that outputs action and commitment level. The idea is that SGD will adjust the commitment level, per state, to maximize returns.
- The innovation: commitment level is based on distance in state space, as opposed to time. This makes it more robust, handling stochastic environments as well. (Previous work did committing to a number of repetitions.)
- Overall, this seems like good work, they have results on Mujoco that they are discretization invariant, an analysis of algo behaviour on InvertedPendulum.
- Main criticism in reviews: distance metric on the state-space is fixed.

- Opportunity? Learn more from the data: we can always learn about shorter commitments based on data from longer commitments.

The paper talks about a very similar idea of what I have been thinking, and it provides a good references to works which talk about action repeating as a way of temporal abstraction. They do connect it to options, but not explicitly the way they train it. They use δ to refer to the time step that the environment takes. First they define a continuous time MDP with deterministic environments governed by a dynamical system equation, i.e. MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, F, \gamma)$, where the state and action space are continous and at the same time the state transition is governed as

$$s(t) = s(0) + \int_0^t F(s(t'), a(t'))dt'$$
$$R(\tau) = \int_0^\infty \gamma^{t'} r(s(t'), a(t'))dt'$$

where $s(t)$ and $a(t)$ are states and action at time t and τ represents the whole trajectory, in some sense the return is R.

Using [**?**] they define a discretized version of this MDP $\mathcal{M}$ as $\mathcal{M}_\delta = (\mathcal{S}, \mathcal{A}, r_\delta, F_\delta, \gamma_\delta)$ with a discretization time scale of $\delta > 0$, where the agent observes a state and performs an action every δ. So the state s_i in $\mathcal{M}_\delta$ is actually state $s_(i\delta)$ for $\mathcal{M}$, and action a_i is maintained during the time interval $[i\delta, (i+1)\delta]$. Details for this I will skip to the paper. **The main Ideas** The paper mentions that people have previously solved this issue by doing action repetitions, but that action repetition has not been state dependent, what this means is that if there a sudden change in the state , the agent is unable to interrupt and the action keeps continuing. Previous methods often produced a action, and the number of times, the action should be repeated. What they propose is a safe region around the state, where the action can be executed, what this means is that the policy at every decision step produces an action and radius of safety zone for that action, i.e. $\pi(a_i, d_i|s_i)$, where the action is executed until the agent stays within the $\triangle(s, s_i) \leq d_i$ distance to the state for which the action was selected. This region is defined as the safe regions. For this paper (i.e. in the experiments), they use the $\triangle(s, s_i) = \|\tilde{s} - \tilde{s_i}\|_1 / dim(\mathcal{S})$. They show that their method's performance is invariant to the δ of the environment. **Pros and Contributions** (1) Provide a proof of the variance explosion for PG estimators for $\delta \to 0$, and show this can be solved with temporally abstracted actions. (2) A state dependent action repetition strategy , rather than a precomputed value for the whole environment. (3) Followed by experiments. **Cons** (1) The don't draw exactly good connections to the options literature exactly. (2) They use a radius a measure for safe distance for action , I think this boils down to using the right distance metric on the state space of the agent, and this might be difficult to predetermine. (3) They use moving average to normalize the state representation, which is good. Maybe option termination functions might be a better way to do this. (3) Need to look closely, but we want a policy iteration strategy where the actions start with a very large repetition window, but converges to the optimal repetition windows which is balance between small and big and maintains the optimal policy.

Reviews for the paper The meta reviewer points out the issue with the distance metric for states . One reviewer mentions about the possiblity of variance not exploding because stochasticity will reduce in the environment.

I think another issue lies in the fact that the agent needs some non markovian information of the current state and the starting state and also need a interrupt check controller of some form. I need to think a bit more about this.

7.2 Reinforcement learning in continuous time: Advantage updating

[?]

This paper talks about the value function collapse in the case of decrease in time window size discretization. The paper reasons that learning a Q function might not be appropriate for very small time window as the Q function might simply collapse into the Value function , as the effect of a different action at given state might reduce in the overall optimal trajectory and hence it might be difficult to extract the policy from the Q function.

Instead the paper proposes to learn an advantage function $A(x, u)$, which tells the advantage of a specific action in a given state over the other actions. $\max_u A(x, u) = 0$,i.e. at convergence and optimal policy the advatange of the optimal action is going to be zero, and advantage of the suboptimal action is going to be negative.

$$A^*(x, u) = \frac{1}{\triangle t}\left[Q^*(x, u) - \max_{u'} Q^*(x, u')\right]$$

They learn this function instead of the Q functions , by having bellman like updates for the same, and hence show that the learn on continuous time systems (with fine discretization) better then Q learning and Value iteration.

7.3 Dynamic Action Repetition for Deep Reinforcement Learning

[?]

This is a very simple paper and only empirical. The paper proposes to double the action space, where each action is repeated twice, with the first being a smaller rate and the second one having a larger rate of repetition. Then they learn a DQN / PG policy on top of that and show results for that. This is a simple approach but they don't have any theory around it and focus mostly on games.

They do talk about state dependent dynamic Action Repetition Rate (ARR) for future work though.

7.4 Learning to Repeat: Fine Grained Action Repetition for Deep Reinforcement Learning

[?] They just augment the policy to output actions and the number of repetitions for that action, and train that. I am not sure about the objective, the paper is again mostly empirical.

7.5 Making Deep Q-learning methods robust to time discretization

[?]

Theorem 2 : There is no Q function in continuous time. Contributions

- Formally show that Q function collapses to the Value Function (V) in near continuous time systems.
- Propose an algorithm which shows significant advantages.

I am not sure if its just a restatement, but they propose a way to learn the Advantage function, which is invariant of the discretization i.e. it converges to some function in the limit of $\delta t \to 0$. I.e. $\lim_{\delta t \to 0} A^{\pi}_{\delta t}(s,a) = A^{\pi}(s,a)$, what this says is that this function exists in the limit , and to me this paper just seems like a better and more refined way of writing [**?**] with more formal proofs.

$$Q^{\pi}_{\delta t}(s,a) = V^{\pi}_{\delta t}(s) + \delta t A^{\pi}_{\delta t}(s,a)$$

So we re scale $A^{\pi}_{\delta t}$ with the appropriate **Cons : The biggest CON is see is the learning algorithm needs to be aware of the δt of the environment.** Will just list theorems for ease

Theorem 1 : $V^{\pi}_{\delta t}(s)$ converges to $V^{\pi}(s)$ when $\delta t \to 0$

Theorem 2 : $Q^{\pi}_{\delta t}(s,a) = V^{\pi}_{\delta t}(s) + O(\delta t)$, when $\delta t \to 0$, for every (s,a).

Theorem 3 : $A^{\pi}_{\delta t}(s,a)$ has a limit for $\delta t \to 0$.

7.6 Lazy-MDPs: Towards Interpretable Reinforcement Learning by Learning When to Act

[**?**] The paper introduces a new framework in the form of an augmentation on top of an MDP, by adding a lazy action, in which case the agent falls back on executing a default policy. I.e., lets say we have an MDP $M = (\mathcal{S}, \mathcal{A}, P, r, \gamma, d_0)$ in the standard case, and we define another MDP, $M_+ = (M, \bar{a}, \bar{\pi}, \eta)$, where $\bar{a}$ is the lazy action, and $\bar{\pi}$ is the default policy, $\eta \in \mathbb{R}$ is the penalty that the agent receives for taking any other action expect the lazy action. Whenever the action takes the lazy action the agent uses the default policy $\bar{\pi}$ to execute an action, hence the new MDP becomes

$$M_+ = (\mathcal{S}, \mathcal{A}_+, \gamma, r_+, P_+, d_0)$$

where the new action space, transition function and reward function are defined as follows:

$$\mathcal{A}_+ = \mathcal{A} \cup \{\bar{a}\}$$

$$r_+(s,a) = \begin{cases} r(s,a) - \eta & \text{if } a \in \mathcal{A} \\ \sum_{a \in \mathcal{A}} \bar{\pi}(a|s) r(s,a) & \text{if } a = \bar{a} \end{cases}$$

$$P_+(s'|s,a) = \begin{cases} P(s'|s,a) & \text{if } a \in \mathcal{A} \\ \sum_{a \in \mathcal{A}} \bar{\pi}(a|s) P(s'|s,a) & \text{if } a = \bar{a} \end{cases}$$

l

7.7 Control Frequency Adaptation via Action Persistence in Batch Reinforcement Learning

[?]

7.8 Temporally-Extended ϵ-Greedy Exploration

[?]

Improve exploration by committing to the same action (action-repeat) for an extended period when taking an exploratory action.

In particular, using a heavy-tailed distribution (zeta) for the duration worked well and is ecologically motivated, too. A different parameter of the zeta distribution works best for different environments.

7.9 Nyquist Sampling

References

[Baird, 1994] Baird, L. (1994). Reinforcement learning in continuous time: Advantage updating. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, volume 4, pages 2448–2453 vol.4.

[Dabney et al., 2020] Dabney, W., Ostrovski, G., and Barreto, A. (2020). Temporally-Extended {\epsilon}-Greedy Exploration.

[Jacq et al., 2022] Jacq, A., Ferret, J., Pietquin, O., and Geist, M. (2022). Lazy-MDPs: Towards Interpretable Reinforcement Learning by Learning When to Act. arXiv:2203.08542 [cs].

[Lakshminarayanan et al.,] Lakshminarayanan, A. S., Sharma, S., and Ravindran, B. Dynamic Action Repetition for Deep Reinforcement Learning. page 7.

[Metelli et al., 2020] Metelli, A. M., Mazzolini, F., Bisi, L., Sabbioni, L., and Restelli, M. (2020). Control Frequency Adaptation via Action Persistence in Batch Reinforcement Learning. In *Proceedings of the 37th International Conference on Machine Learning*, pages 6862–6873. PMLR. ISSN: 2640-3498.

[Park et al., 2021] Park, S., Kim, J., and Kim, G. (2021). Time discretization-invariant safe action repetition for policy gradient methods. *CoRR*, abs/2111.03941.

[Sharma et al., 2020] Sharma, S., Srinivas, A., and Ravindran, B. (2020). Learning to Repeat: Fine Grained Action Repetition for Deep Reinforcement Learning. *arXiv:1702.06054 [cs].*

[Tallec et al., 2019] Tallec, C., Blier, L., and Ollivier, Y. (2019). Making Deep Q-learning methods robust to time discretization. *arXiv:1901.09732 [cs, stat].*