GREEDY MULTI-STEP OFF-POLICY REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper presents a novel multi-step reinforcement learning algorithms, named Greedy Multi-Step Value Iteration (GM-VI), under the off-policy setting. GM-VI iteratively approximates the optimal value function using a newly proposed multi-step bootstrapping technique, in which the step size is adaptively adjusted along each trajectory according to a greedy principle. With the improved multi-step information propagation mechanism, we show that the resulted VI process is capable of safely learning from arbitrary behavior policy without additional off-policy correction. We further analyze the theoretical properties of the corresponding operator, showing that it is able to converge to globally optimal value function, with a rate faster than the traditional Bellman Optimality Operator. Experiments reveal that the proposed method is reliable, easy to implement, and achieves state-of-the-art performance on a series of standard benchmark datasets.

1 INTRODUCTION

Multi-step reinforcement learning (RL) is a set of methods that are capable of flexibly adjusting the trade-off between the observed multi-step return and the estimated future return so as to meet the demands of a particular task. Recent advances on multi-step RL have achieved remarkable empirical success (Horgan et al., 2018; Barth-Maron et al., 2018). However, one major challenge of multi-step RL comes from how to achieve the right balance between the two terms. Such balance can be regarded as a kind of data-knowledge trade-off in some sense, as the future return is estimated through a value function representing the knowledge learnt so far. Particularly, a large step size tends to quickly propagate the information in the data, while a small one relies more on the estimation based on the learnt knowledge. Classical solution to address this issue is to impose a fixed prior distribution over every possible step size (Sutton & Barto, 2018; Schulman et al., 2016), ignoring the quality of data and on-going knowledge which dynamically improves over the learning process. Unfortunately, such a prior distribution has to be tuned case by case.

Another issue related to multi-step RL is off-policy learning, i.e., its capability to learn from data from other behavior policies. Previous research on this is mainly conducted under the umbrella of Policy Iteration (PI) (Sutton & Barto, 2018), with the goal to evaluate the value of a given target policy (Precup, 2000; Harutyunyan et al., 2016; Munos et al., 2016; Sutton & Barto, 2018; Schulman et al., 2016). Despite their success, those methods usually suffer from certain undesired side effects of off-policy learning, e.g., high variance due to the product of importance sampling (IS) ratios, and the restrictive premise of being able to access both behavior policy and the target policy (to compute the IS ratios). Most importantly, those methods also require the aforementioned prior distribution on step size, which usually need to be tuned case by case, e.g., TD(λ) (Sutton & Barto, 2018), GAE(λ) (Schulman et al., 2016).

In contrast with PI, Value Iteration (VI) methods propagate the value of the most promising action to approximate the optimal value function, without the need of policy evaluation (Sutton & Barto, 2018; Szepesvári, 2010). These characteristics of VI make it somewhat unnatural for multi-step learning — theoretically, this means that it has to search over the whole trajectory space to find one which achieves the highest (multi-step) return. For a method like this, the good side is that it can safely use data from any behavior policy without any off-policy correction (as no policy evaluation is needed), but at the cost of unrealistic computational burden. As mentioned in the next section, very few research (Horgan et al., 2018; Barth-Maron et al., 2018) in literatures address these issues.

In this paper, we propose a novel method named GM-VI that adopts a multi-step style scheme to iteratively approximates the optimal value function. The core idea of our method is to greedily chooses the largest value among various-step bootstrapping estimation, so as to approximate optimal value as quickly as possible. This greedy principle essentially allows us to adjust the step size adaptively during multi-step learning according to the quality of the trajectory data — a higher-quality trajectory data leads to a larger chosen step size compared to a lower-quality one. Furthermore, GM-VI naturally allows safely using data from any behavior policy without additional correction, while freely using multi-step data.

The key to the success of the proposed GM-VI method is a novel multi-step Bellman Optimality Operator, which is an extension to its classical one-step counterpart (Sutton & Barto, 2018; Szepesvári, 2010). We analyze the theoretical properties of this operator, showing that it is able to converge to globally optimal value function, with a rate faster than the traditional Optimal Bellman Operator. As a concrete implementation of this operator, we propose a novel algorithm named Greedy Multi-Step Q Learning, showing that it achieves state-of-the-art performance on standard benchmark tasks.

2 PRELIMINARIES

We begin with discussion on Value Iteration (VI) approach and Policy Iteration (PI) approach, which are two fundamental approaches of RL.

2.1 MARKOV DECISION PROCESSES

A Markov Decision Processes (MDP) is described by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma)$. $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $\gamma \in (0, 1)$ is the discount factor; $\mathcal{T}$ is the transition function mapping $s, a \in \mathcal{S} \times \mathcal{A}$ to distributions over $\mathcal{S}$; r is the reward function mapping s, a to distribution over $\mathbb{R}$, which we will use $r(\cdot|s, a)$ for stochastic case and $r(s, a)$ for deterministic case. The trajectory by executing policy π for N steps after executing action a_t at state s_t is defined as

$$\tau_{N,\pi}^{s_t,a_t} \triangleq r_t, s_{t+1}, a_{t+1}, r_{t+1}, s_{t+2}, a_{t+2}, \ldots, s_{t+N}, a_{t+N},$$

where $r_t \sim r(\cdot|s_t, a_t), s_{t+1} \sim \mathcal{T}(\cdot|s_t, a_t), a_{t+1} \sim \pi(\cdot|s_{t+1})$. The return is defined as the accumulated reward $R_t^\gamma = \sum_{n=0}^{\infty} \gamma^n r_{t+n}$. The value function of a policy π is defined as the expected return $Q^\pi(s, a) \triangleq \mathbb{E}_{\tau_{\infty,\pi}^{s_t,a_t}} [R_t^\gamma | s_t = s, a_t = a, \pi]$. Value-based RL methods aims to approximate the *optimal value function* $Q^*(s, a) = \max_\pi Q^\pi(s, a)$. The policy outputted by the algorithm is behaved according to the estimated value function Q, i.e., $\pi^Q(s) = \arg\max_a Q(s, a)$.

2.2 VALUE ITERATION

VI approach aims to approximate the optimal value function by back-propagating the optimal value.

One-Step VI. One-step VI back-propagates the optimal value on each state to its father state-action step by step, represented by Q learning (Watkins & Dayan, 1992). Formally, the value function is updated by the *One-Step Bellman Optimality Operator (we will briefly use **one-step optimality operator**)*

$$(\mathcal{B}^1 q)(s, a) \triangleq \mathbb{E}_{r_t, s_{t+1}} \left[r_t + \gamma \max_{a_{t+1}} q(s_{t+1}, a_{t+1}) \middle| s_t = s, a_t = a \right] \tag{1}$$

Such step-by-step back-propagation is often not efficient especially when the trajectory has a long horizon. As we will show in Section 3, such step-by-step back-propagation is not necessarily the best choice and it can be accelerated. Besides, when using function approximator for value function, the approximation error may be accumulated across the long-horizon back-propagation (Hasselt, 2010; Van Hasselt et al., 2015).

Multi-Step VI. One naive way to improve the propagation efficiency is to use multi-step bootstrapping (Horgan et al., 2018; Barth-Maron et al., 2018). The value function is updated by the *Multi-Step Bellman Optimality Operator (**multi-step optimality operator**)*

$$(\mathcal{B}_{\mathcal{P}}^N q)(s, a) \triangleq \mathbb{E}_{\pi \sim \mathcal{P}, \tau_{N,\pi}^{s_t,a_t}} \left[\sum_{n=0}^{N-1} \gamma^n r_{t+n} + \gamma^N \max_{a_{t+N}} q(s_{t+N}, a_{t+N}) \middle| s_t = s, a_t = a, \mathcal{P} \right] \tag{2}$$

where $\mathcal{P}(\pi)$ is the probability of choosing a behavior policy π, together with the trajectory $\tau_{N,\pi}^{s_t,a_t}$, formalizing sampling experience of different policies from the replay buffer. Recent advances have shown promising results in practice. This implies that multi-step bootstrapping can often benefit learning. However, it requires much effort and prior knowledge to tune the step size N. Furthermore, the final Q value function is highly dependent on the behavior policies and generally can not converge to global optimal value function (as we will show in Theorem 2).

2.3 POLICY ITERATION

Policy iteration approach iteratively performs policy evaluation and policy improvement, which naturally allow multi-step learning. At the policy evaluation phase, the algorithm evaluates the value function Q^π of the current policy π.

Multi-Step On-Policy Evaluation. On-policy methods abandon the off-policy data which may incur instability (Tesauro, 1995; Boyan, 1999). The value function is updated by the *Multi-Step On-Policy Bellman Operator*

$$(\bar{\mathcal{B}}_\pi^N q)(s,a) \triangleq \mathbb{E}_{\tau_{N,\pi}^{s_t,a_t}}\left[\sum_{n=0}^{N-1}\gamma^n r_{t+n} + \gamma^N q(s_{t+N},a_{t+N})\middle| s_t=s, a_t=a, \pi\right] \tag{3}$$

Classical implementation includes: (1) one-step methods: SARSA, Expected SARSA (Sutton & Barto, 2018), etc.; (2) infinity-step methods: Monte Carlo (Sutton & Barto, 2018); (3) trade-off: TD(λ) (Sutton & Barto, 2018), GAE(λ) (Schulman et al., 2016), which balance the trade-off by a discount factor λ. The corresponding operator is defined as $\bar{\mathcal{B}}_\pi^\lambda \triangleq (1-\lambda)\sum_{N\geq 1}\lambda^N \bar{\mathcal{B}}_\pi^N$, which assign exponentially-decay weight as N becomes larger. Roughly, the parameter λ represents the prior knowledge or our bias on the step size of bootstrap. Unfortunately, this parameter has to be tuned in a case-by-case manner.

Multi-Step Off-Policy Evaluation. A general idea to employ off-policy data is to set correction operation. One classical method is to use importance sampling (IS) with *Multi-Step Off-Policy Bellman Operator*,

$$(\widetilde{\mathcal{B}}_{\mathcal{P},\pi}^N q)(s,a) \triangleq \mathbb{E}_{\pi'\sim\mathcal{P},\tau_{N,\pi'}^{s_t,a_t}}\left[\sum_{n=0}^{N-1}\gamma^n \zeta_{t+1}^{t+n} r_{t+n} + \gamma^N \zeta_{t+1}^{t+N} q(s_{t+N},a_{t+N})\middle| \begin{matrix} s_t=s \\ a_t=a \\ \mathcal{P},\pi \end{matrix}\right] \tag{4}$$

where $\zeta_{t+1}^{t+n} \triangleq \prod_{t'=t+1}^{t+n}\frac{\pi(a_{t'}|s_{t'})}{\pi'(a_{t'}|s_{t'})}$ is the *importance sampling (IS) ratio*. The multiple product terms of IS make the value suffer from high variance, and has motivated further variance reduction techniques, e.g., TB(λ) (Precup, 2000), Q(λ) (Harutyunyan et al., 2016), Retrace(λ) (Munos et al., 2016). For these methods, we need to know not only the trajectory but also the behavior policy, i.e., the likelihoods of choosing the behavior action of the behavior policy, $\pi'(a_t|s_t)$.

3 GREEDY MULTI-STEP VALUE ITERATION

The general idea of our approach aims to approximate the optimal value function, while adaptively adjusting the step size by the quality of the trajectory data. The novel *Greedy Multi-Step Bellman Optimality Operator (**GM-optimality operator**)* is defined as

$$(\mathcal{G}_{\mathcal{P}}^N q)(s,a) \triangleq \mathbb{E}_{\pi\sim\mathcal{P},\tau_{N,\pi}^{s_t,a_t}}\left[\max_{1\leq M\leq N}\left[\sum_{n=0}^{M-1}\gamma^n r_{t+n} + \gamma^M \max_{a_{t+M}} q(s_{t+M},a_{t+M})\right]\middle| \begin{matrix} s_t=s, \\ a_t=a, \\ \mathcal{P} \end{matrix}\right] \tag{5}$$

Note that one-step GM-optimality operator $\mathcal{G}_{\mathcal{P}}^1$ is equivalent to one-step optimality operator $\mathcal{B}^1$ (eq. 1). One way to understand GM-optimality operator is from a forward view of value update. As shown in Figure 1, one-step optimality operator looks forward for one step, while multi-step optimality operator looks forward for several steps and take the corresponding bootstrap value. Our GM-optimality operator also looks forward for multiple steps but greedily chooses the optimal one among all these bootstrap values.

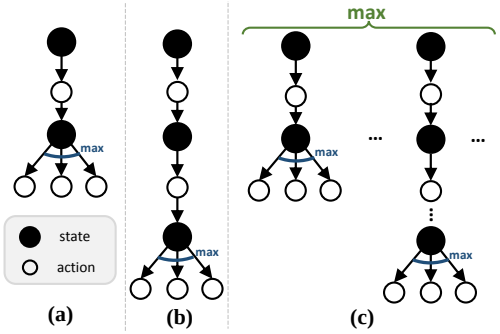


Figure 1: Backup diagram of (a) One-Step Optimality Operator; (b) Multi-Step Optimality Operator; (c) Greedy Multi-Step Optimality Operator.

Algorithm 1 Greedy Multi-Step Q Learning

Input: Initialized $Q^{(0)}$
1: Initialize buffer $B = \{\}$, iteration $k = 1$.
2: **repeat**
3: Execute policy with $Q^{(k)}$ (e.g., ϵ-greedy) and obtain a trajectory τ
4: **for** (s_t, a_t) in τ **do**
5: Put $(s_t, a_t, \tau_N^{s_t,a_t})$ into buffer B
6: **end for**
7: **for** i=1, M **do**
8: Sample $(s_t, a_t, \tau_N^{s_t,a_t})$ from B.
9: Update $Q^{(k+1)}$ with $(s_t, a_t, \tau_N^{s_t,a_t})$ and $Q^{(k)}$
10: **end for**
11: $k = k + 1$
12: **until**

One question readers may concern is that *why it can perform off-policy learning from arbitrary behavior policy without any correction?* The essential reason is that our goal is to estimate the optimal value function but not the value function of a specific policy. This implies that no policy evaluation is needed and thus naturally no correction is involved. From a technical view, given a trajectory $\tau_{N,\pi}^{s_t,a_t} \triangleq r_t, s_{t+1}, a_{t+1}, r_{t+1}, s_{t+2}, a_{t+2}, \ldots, s_{t+N}, a_{t+N}$, if it brings a high future return, then a long horizon along the trajectory is utilized. Otherwise if the trajectory is worse than it has explored before, we can still use an extremely short-step bootstrap, i.e., one-step bootstrap $r_t + \max_a Q(s_{t+1}, a)$, which only involve the (s_t, a_t, r_t, s_{t+1}) information from the environment but not concern about a specific policy.

The corresponding implementation of this operator is named *Greedy Multi-Step Q Learning* (GM-Q learning). Given a trajectory τ and a value function $Q^{(k)}$, the target value is computed by

$$Q_{\text{target}}^{(\tau,k)}(s,a) = \max_{1 \le M \le N} \left[\sum_{n=0}^{M-1} \gamma^n r_{t+n}^{(\tau)} + \gamma^M \max_{a_{t+M}} Q^{(k)}(s_{t+M}^{(\tau)}, a_{t+M}) \right], \tag{6}$$

One can use the average over all samples, $Q^{(k+1)}(s,a) \leftarrow \frac{1}{|\boldsymbol{\tau}^{s,a}|} \sum_{\tau \in \boldsymbol{\tau}^{s,a}} Q_{\text{target}}^{(\tau,k)}(s,a)$ with all experience state-action pair (s,a), where $\boldsymbol{\tau}^{s,a} \triangleq \{\tau^{s,a}\}$ is the set of truncated trajectories starting from s, a; or use temporal difference $Q^{(k+1)}(s,a) \leftarrow Q^{(k)}(s,a) + \alpha \left(Q_{\text{target}}^{(\tau,k)}(s,a) - Q^{(k)}(s,a) \right)$; or use mean square error $\min_q \sum_\tau \|q(s,a) - q_{\text{target}}^{(\tau,k)}(s,a)\|^2$ for parametrized function q. By replacing Line 9 in Algorithm 1 with one of these equations, one can obtain the GM-Q learning algorithm.

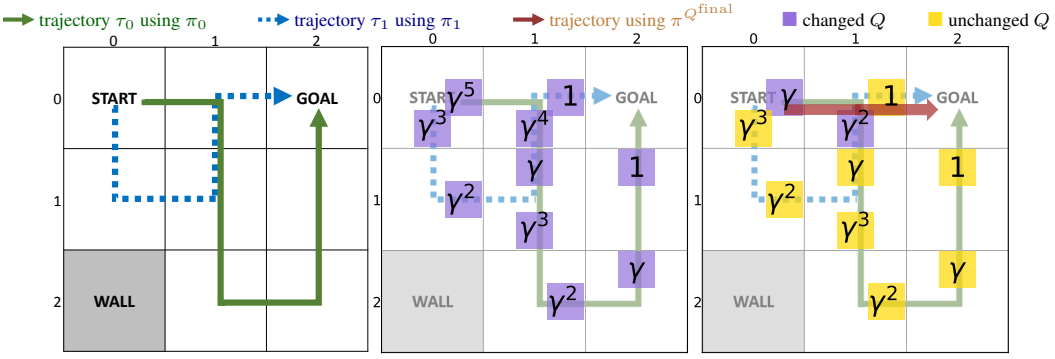


Figure 2: Update process of GM-Q learning with $N = \infty$. The Q value with purple background means that the value is changed at corresponding iteration, while one with the yellow background means that the value is unchanged. γ is the discount factor.

To gain more insight on how GM-Q learning works, let us simulate the update process of GM-Q learning on a small grid world example. Figure 2a depicts a grid world problem. The agent needs to find a shorted path from START state to GOAL state. The reward is 1 if the agent reach the GOAL state otherwise 0; the state is defined as $s = [\text{row}, \text{col}]$, e.g., for GOAL state $s = [0, 2]$; the action space $\mathcal{A} = \{\leftarrow, \rightarrow, \uparrow, \downarrow\}$. There are two existing trajectories. The Q values are initialized with 0 for all state-actions. At each iteration, the Q function is update for each state-action in the buffer. At iteration 1, GM-Q learning chooses the longest horizon until the end on all the state-actions. For example, as shown in Figure 2b, the $Q([0, 0], \downarrow)$ is updated by trajectory τ_1 until the end, i.e., $r([0, 0], \downarrow) + \gamma r([1, 0], \rightarrow) + \gamma^2 r([1, 1], \uparrow) + \gamma^3 r([0, 1], \rightarrow) + \gamma^4 \max_a Q([0, 2], a) = \gamma^3$. At iteration 2, For example, the one-step bootstrap value for $([0, 0], \rightarrow)$ is larger than any other bootstrap values. Therefore, we have $Q([0, 0], \rightarrow) = r([0, 0], \rightarrow) + \gamma \max_a Q([0, 1], a) = \gamma$. We also provide a simulation on how other algorithms behave in Appendix B. GM-Q learning only require 2 iterations to finish updating (while one-step Q learning requires 4) and it finally obtains the optimal policy (while multi-step Q learning gets a sub-optimal one, shown in Appendix Figure 10a).

4 THEORETICAL ANALYSIS

In this section, we analyze the convergence and the corresponding speed of GM-optimality operator $\mathcal{G}_{\mathcal{P}}^N$, comparing with the existing operators like multi-step optimality operator $\mathcal{B}_{\mathcal{P}}^N$. We consider finite state $\mathcal{S}$ and action spaces $\mathcal{A}$.

4.1 CONVERGENCE

First, we show the convergence property of GM-optimality operator $\mathcal{G}_{\mathcal{P}}^N$ (eq. 5), beginning with the contraction property.

Lemma 1 *(contraction of GM-optimality operator) For any $\mathcal{P}$, given any value function $q, q' \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$, $N \geq 1$, $\|\mathcal{G}_{\mathcal{P}}^N q - \mathcal{G}_{\mathcal{P}}^N q'\| \leq \gamma \|q - q'\|$.*

We provide all the proofs in Appendix A. Throughout, we write $\|\cdot\|$ for supremum norm. Here γ is the *contraction rate* of $\mathcal{G}_{\mathcal{P}}^N$. Note that this is a worst-case contraction rate of $\mathcal{G}_{\mathcal{P}}^N$, a more detailed analysis will be given in the next section.

Lemma 2 *For any $\mathcal{P}$, $N \geq 1$, given any value function $q \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$, $Q^\pi \leq \mathcal{G}_{\mathcal{P}}^N Q^\pi$.*

By the lemmas above, it's easy to obtain that GM-optimality operator will converge to Q^*.

Theorem 1 *(the fix point of GM-optimality operator; Greedy Multi-Step Bellman Optimality Equation) For any $\mathcal{P}$ and $N \geq 1$, $\mathcal{G}_{\mathcal{P}}^N Q^* = Q^*$.*

Proof: Let $\widehat{Q}^*$ denote the fix point of $\mathcal{G}_{\mathcal{P}}^N$ (it has a fix point by Lemma 1). For any policy π, by applying $\mathcal{G}_{\mathcal{P}}^N$, we have $Q^\pi \leq \mathcal{G}_{\mathcal{P}}^N Q^\pi \leq (\mathcal{G}_{\mathcal{P}}^N)^2 Q^\pi \leq \cdots = \widehat{Q}^*$, which implies $\widehat{Q}^*$ satisfies the definition of Q^*. □

We have finished the convergence property of GM-optimality operator $\mathcal{G}_{\mathcal{P}}^N$. Another problem worth concerning is the convergence property of multi-step optimality operator $\mathcal{B}_{\mathcal{P}}^N$ (eq. 2).

Lemma 3 *(contraction of multi-step optimality operator $\mathcal{B}_{\mathcal{P}}^N$) For any two vectors $q, q' \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$, $N \geq 1$, $\|\mathcal{B}_{\mathcal{P}}^N q - \mathcal{B}_{\mathcal{P}}^N q'\| \leq \gamma^N \|q - q'\|$.*

Theorem 2 *(the fix point of multi-step optimality operator $\mathcal{B}_{\mathcal{P}}^N$) Let $Q^*_{\mathcal{B}_{\mathcal{P}}^N}$ denote the fix point of $\mathcal{B}_{\mathcal{P}}^N$, i.e., $\mathcal{B}_{\mathcal{P}}^N Q^*_{\mathcal{B}_{\mathcal{P}}^N} = Q^*_{\mathcal{B}_{\mathcal{P}}^N}$.*

1) When $N = 1$, $Q^*_{\mathcal{B}_{\mathcal{P}}^1} = Q^*$, which means $\mathcal{B}^1 Q^* = Q^*$ (known as Bellman Optimality Equation).

2) For any $N \geq 2$, $Q^*_{\mathcal{B}_{\mathcal{P}}^N} \leq Q^*$. If and only if $\mathcal{P}$ satisfies for any $\pi \in \{\pi | \mathcal{P}(\pi) > 0\}$ we have $\pi(s) = \arg\max_a Q^*(s, a)$, then $Q^*_{\mathcal{B}_{\mathcal{P}}^N} = Q^*$.

The theorems above imply that multi-step optimality operator $\mathcal{B}_{\mathcal{P}}^{N}$ ($N \geq 2$) converges faster than one-step optimality operator $\mathcal{B}^{1}$ does. However, multi-step optimality operator $\mathcal{B}_{\mathcal{P}}^{N}$ generally converges to a suboptimal value function — converging to optima only happens when *all* the behavior policies are optimal, which is almost never guaranteed in practice. To best of our knowledge, this the first time giving this formal statement on the sub-optimal convergence of multi-step optimality operator.

4.2 Convergence Speed

In this section, we compare the convergence speed of GM-optimality operator $\mathcal{G}_{\mathcal{P}}^{N}$ and one-step optimality operator $\mathcal{B}^{1}$, as both these two operators converge to the optimal value function (while multi-step optimality operator $\mathcal{B}_{\mathcal{P}}^{N}$ cannot).

First, we give a basis on understanding why GM-optimality operator $\mathcal{G}_{\mathcal{P}}^{N}$ can generally converge faster. GM-optimality operator $\mathcal{G}_{\mathcal{P}}^{N}$ chooses the maximum one among various-step bootstrap values, while one-step optimality operator $\mathcal{B}^{1}$ only uses the one-step value. It's obvious that $\mathcal{G}_{\mathcal{P}}^{N} q \geq \mathcal{B}^{1} q$. If we additionally set a proper initial condition on the value function q, i.e., $q \leq Q^{*}$, then $\mathcal{G}_{\mathcal{P}}^{N} q$ tends to get closer to Q^{*} than $\mathcal{B}^{1} q$.

Lemma 4 *$\mathcal{G}_{\mathcal{P}}^{N} q \geq \mathcal{B}^{1} q$. If $q \leq Q^{*}$, then $\|\mathcal{G}_{\mathcal{P}}^{N} q - Q^{*}\| \leq \|\mathcal{B}^{1} q - Q^{*}\|$.*

Then, we give a general result that: with same given iteration, GM-optimality operator $\mathcal{G}_{\mathcal{P}}^{N}$ tends to get closer to optimal value function than one-step optimality operator $\mathcal{B}^{1}$ does. Let $\{Q_{\mathcal{G}_{\mathcal{P}}^{N}}^{(k)}\}$ and denote the sequence by applying GM-optimality operator $\mathcal{G}_{\mathcal{P}}^{N}$ starting from $Q^{(0)}$, i.e., $Q_{\mathcal{G}_{\mathcal{P}}^{N}}^{(k+1)} = \mathcal{G}_{\mathcal{P}}^{N} Q_{\mathcal{G}_{\mathcal{P}}^{N}}^{(k)}$ ($Q_{\mathcal{G}_{\mathcal{P}}^{N}}^{(0)} = Q^{(0)}$); and similarly $\{Q_{\mathcal{B}_{\mathcal{P}}^{N}}^{(k)}\}$ is the one by applying $\mathcal{B}_{\mathcal{P}}^{N}$.

Theorem 3 *If $Q^{(0)} \leq Q^{*}$, then we have $\|Q_{\mathcal{G}_{\mathcal{P}}^{N}}^{(k)} - Q^{*}\| \leq \|Q_{\mathcal{B}^{1}}^{(k)} - Q^{*}\|$ for any $N \geq 1, k \geq 1$.*

Next, we give a result under an ideal case: if all the behavior policies are the optimal ones, then the contraction rate of $\mathcal{B}_{\mathcal{P}}^{N}$ is γ^{N} and we can obtain the optimal value function by applying GM-optimality operator $\mathcal{G}_{\mathcal{P}}^{\infty}$ only *once* for any $q \leq Q^{*}$.

Theorem 4 *If for any $\pi \in \{\pi | \mathcal{P}(\pi) > 0\}$ $\pi(s) = \arg\max_{a} Q^{*}(s, a)$ and $q \leq Q^{*}$, then $\|\mathcal{G}_{\mathcal{P}}^{N} q - Q^{*}\| \leq \gamma^{N} \|q - Q^{*}\|$. Specially, for $N = \infty$, $\mathcal{G}_{\mathcal{P}}^{\infty} q = Q^{*}$.*

Finally, we give a more practical result that: with some relaxed conditions, the GM-optimality operator $\mathcal{G}_{\mathcal{P}}^{N}$ has a contraction rate of γ^{2}. We introduce a condition that the behavior policy output the optimal action on specific states.

Condition 1 *(condition on the distribution of behavior policy $\mathcal{P}$ and the value function q) Given $\mathcal{P}$ and $q \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$, for any $\pi \in \{\pi | \mathcal{P}(\pi) > 0\}$ satisfy the following condition: for any s which satisfies $\left|\min_{a} \mathcal{B}^{1} q(s, a) - \max_{a'} Q^{*}(s, a')\right| > \gamma \left\|q - Q^{*}\right\|_{\infty}$, $\pi(s) = \arg\max_{a} Q^{*}(s, a)$.*

Theorem 5 *(faster contraction rate with special condition for GM-optimality operator $\mathcal{G}_{\mathcal{P}}^{N}$) If $q \leq Q^{*}$ and distribution of behavior policy $\mathcal{P}$ satisfies Condition 1, then for any value function $q \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$, $N \geq 2$, $\|\mathcal{G}_{\mathcal{P}}^{N} q - \mathcal{G}_{\mathcal{P}}^{N} Q^{*}\| = \|\mathcal{G}_{\mathcal{P}}^{N} q - Q^{*}\| \leq \gamma^{2} \|q - Q^{*}\|$.*

Above all, our GM-optimality operator $\mathcal{G}_{\mathcal{P}}^{N}$ can converge to the optimal value function with a rate of γ (or γ, γ^{N} with some condition). It does *not* require off-policy correction, and only requires the access to the trajectory data while not necessary the behavior policy π (i.e., the policy distribution $\pi(\cdot|s)$). Furthermore, it can adaptively adjust the step size by the quality of the trajectory data. To best of our knowledge, none of the existing operators own all of these properties together. We summarize the properties of all the referred operators in Table 1.

Operator	Converge to	Contraction Rate	NOT requiring off-policy correction	NOT requiring knowing policy	Support adaptively adjusting step size
One-Step Optimal Operator $\mathcal{B}^1$ (eq. 1)	Q^*	γ	✓	✓	×
Multi-Step Optimal Operator $\mathcal{B}_{\mathcal{P}}^N$ (eq. 2)	$Q^*_{\mathcal{B}_{\mathcal{P}}^N} (\leq Q^*)$	γ^N	✓	✓	×
Multi-Step On/Off-Policy Operator (eq. 3, 4)	Q^π	γ^N	×	×	×
Greedy Multi-Step Optimal Operator $\mathcal{G}_{\mathcal{P}}^N$ (eq. 5)	Q^*	γ (γ^2, γ^N with condition)	✓	✓	✓

Table 1: Properties of the operators.

5 Experiment

We designed our experiments to investigate the following questions. 1) What are the performance characteristic of GM-Q learning in sample efficiency and reward? 2) Is multi-step learning necessary? Besides, as GM-Q learning adaptively choose the step size, how large step size will GM-Q learning choose? 3) How will the initialization of value function affect the convergence speed, as the analytical results we have given in Section 4.2?

5.1 Experimental Setup

For our GM-Q learning algorithm, we use $N = \infty$ (eq. 6) for all tasks, which means that it will check all the bootstrapping values with different step sizes until the end of the trajectory.

The following algorithms are compared. (a) *Q learning*: a classical algorithm Sutton & Barto (2018). (e) *Multi-Step Q learning*: a vanilla multi-step version of Q learning without any off-policy correction, which has shown promising result in practice (Horgan et al., 2018; Barth-Maron et al., 2018). (f) *SARSA*: a classical one-step on-policy algorithm (Sutton & Barto, 2018). (g) *Retrace(λ)*: a state-of-the-art off-policy algorithm by clipping the importance sampling ratio. All the proposed methods adopt the same implementations to ensure that the differences are due to the algorithm changes instead of the implementations. The implementation detail is provided in Appendix C.

The algorithms are evaluated on benchmark tasks implemented in OpenAI Gym (Brockman et al., 2016). Each algorithm was run with 10 random seeds. The trained policies are evaluated after sampling every 600 timesteps data.

5.2 Performance

Figure 3 shows the performance of the algorithms. GM-Q learning outperforms the classical and the state-of-the-art algorithms in both reward and sample efficiency on all the tasks. For example, on the mountain car task, GM-Q learning reaches a reward of -160 within 2×10^4 timesteps. While multi-step Q learning, which performs best among remaining tasks, achieves a reward -160 with twice the timesteps of GM-Q learning. The common characteristic of these two tasks is that the reward is sparse. For example, the mountain car task will provide a reward only when the car drives up the mountain on the right (while the car's engine is not strong enough to scale the mountain in a single pass). GM-Q learning outperforms other algorithms on these sparse reward tasks.

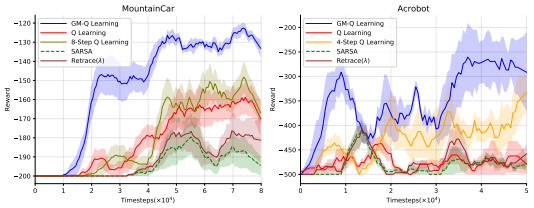


Figure 3: Episode rewards of the algorithms averaged over 10 random seeds. The shaded area corresponds to 40% confidence intervals.

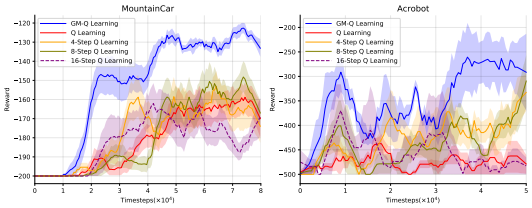


Figure 4: Episode rewards of multi-step Q-learning with different step sizes.

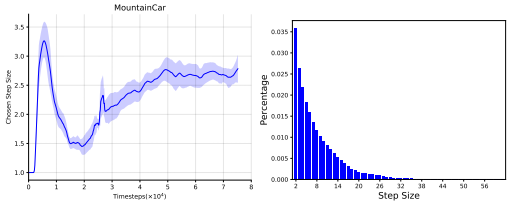


Figure 5: (Left) The chosen step size of GM-Q learning during training process. (Right) The distribution of the chosen step size over all data, in which the percentage of step size 1 is 77%.

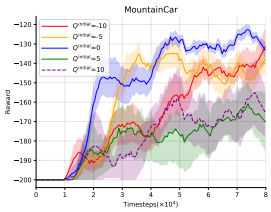


Figure 6: Episode rewards of GM-Q learning with different initialized Q values.

5.3 Discussion on the Components of GM-Q learning

Multi-Step Learning. As shown in Figure 4, with proper step size setting, multi-step Q learning performs better than one-step Q learning. For 16-step Q learning (purple dashed line), although it failed to achieve a higher score at the end, it reach a higher reward than other-step Q learning at the beginning (1×10^4 timestep). These results implies the efficiency of long-horizon information propagation of multi-step learning. However, vanilla multi-step highly depends on the behavior policy, restricting it from achieving further success.

Adaptive Adjustment of Step Size. As shown in Figure 4, multi-step Q learning performs better with step size 8 on the mountain car task, while step size 4 on the acrobot task. It requires parameter tuning for each task. Furthermore, the step sizes for multi-step Q learning is fixed for each state-action pair during the learning process, while those of our GM-Q learning is adaptively adjusted. As shown in Figure 5 (Left), at the later stage (after 2×10^4 timesteps), the chosen step sizes increase as the quality of data improves (shown by the increased score of GM-Q learning in Figure 3). Figure 5 (Right) plots the distribution of the chosen step size. Larger step sizes are less chosen than the smaller ones.

Initialization of the Value Function. As we have stated in Section 4.2, the initialization of Q value can affect the performance of GM-Q learning. We experiment with different initialized Q values by setting the parameter of DNN. As shown in Figure 6, with smaller initialized value (-10 or -5), GM-Q learning can achieve a high score but require relatively more iterations. While larger initialized value leads to a poor final score. Future work on proper initialization of value function needs studying.

6 Conclusion

In this work, we introduce a new multi-step Bellman Optimality Operator, which can be used to approximate the optimal value function Q^* with a sequence of monotonically improved Q-functions. When applying it for value function updating, the proposed operator has several advantages:1) it is able to adjust the step size adaptively during learning according to the quality of the trajectory data, with no task-specific hyperparameters; 2) like its one-step version, it supports learning from off-policy samples while no need for off-policy correction, hence will not suffer from the issues related to that, such as high variance; 3) it has guaranteed convergence with a faster rate. The feasibility and effectiveness of the proposed method has been demonstrated on a series of standard benchmark datasets with promising results.

References

Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva Tb, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributed distributional deterministic policy gradients. *arXiv preprint arXiv:1804.08617*, 2018.

Justin A Boyan. Least-squares temporal difference learning. In *ICML*, pp. 49–56. Citeseer, 1999.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016. cite arxiv:1606.01540.

Anna Harutyunyan, Marc G Bellemare, Tom Stepleton, and Rémi Munos. Q (λ) with off-policy corrections. In *International Conference on Algorithmic Learning Theory*, pp. 305–320. Springer, 2016.

Hado V Hasselt. Double q-learning. In *Advances in neural information processing systems*, pp. 2613–2621, 2010.

Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado Van Hasselt, and David Silver. Distributed prioritized experience replay. *arXiv preprint arXiv:1803.00933*, 2018.

Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc Bellemare. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 1054–1062, 2016.

Doina Precup. Eligibility traces for off-policy policy evaluation. *Computer Science Department Faculty Publication Series*, pp. 80, 2000.

John Schulman, Philipp Moritz, Sergey Levine, Michael I Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *international conference on learning representations*, 2016.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Csaba Szepesvári. Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning*, 4(1):1–103, 2010.

Gerald Tesauro. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3): 58–68, 1995.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *arXiv preprint arXiv:1509.06461*, 2015.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

A THEOREM PROOFS

Proof of Lemma 1:

$$
\begin{aligned}
&\left\|\mathcal{G}_{\mathcal{P}}^{N} q-\mathcal{G}_{\mathcal{P}}^{N} q'\right\| \\
&=\max_{s_t,a_t}\left|\mathbb{E}_{\pi\sim\mathcal{P},\tau_{N,\pi}^{s_t,a_t}}\left[\max_{1\leq M\leq N}\sum_{n=0}^{M-1}\gamma^n r_{t+n}+\gamma^M\max_{a_{t+M}} q\left(s_{t+M},a_{t+M}\right)\right]\right. \\
&\qquad\left.-\left(\mathbb{E}_{\pi\sim\mathcal{P},\tau_{N,\pi}^{s_t,a_t}}\left[\max_{1\leq M\leq N}\sum_{n=0}^{M-1}\gamma^n r_{t+n}+\gamma^M\max_{a_{t+M}} q'\left(s_{t+M},a_{t+M}\right)\right]\right)\right| \\
&\leq\max_{s_t,a_t}\mathbb{E}_{\pi\sim\mathcal{P},\tau_{N,\pi}^{s_t,a_t}}\left|\left(\max_{1\leq M\leq N}\sum_{n=0}^{M-1}\gamma^n r_{t+n}+\gamma^M\max_{a_{t+M}} q\left(s_{t+M},a_{t+M}\right)\right)\right. \\
&\qquad\left.-\left(\max_{1\leq M\leq N}\sum_{n=0}^{M-1}\gamma^n r_{t+n}+\gamma^M\max_{a_{t+M}} q'\left(s_{t+M},a_{t+M}\right)\right)\right| \\
&\leq\max_{s_t,a_t}\mathbb{E}_{\pi\sim\mathcal{P},\tau_{N,\pi}^{s_t,a_t}}\max_{1\leq M\leq N}\left|\left(\sum_{n=0}^{M-1}\gamma^n r_{t+n}+\gamma^M\max_{a_{t+M}} q\left(s_{t+M},a_{t+M}\right)\right)\right. \\
&\qquad\left.-\left(\sum_{n=0}^{M-1}\gamma^n r_{t+n}+\gamma^M\max_{a_{t+M}} q'\left(s_{t+M},a_{t+M}\right)\right)\right| \\
&\leq\max_{s_t,a_t}\mathbb{E}_{\pi\sim\mathcal{P},\tau_{N,\pi}^{s_t,a_t}}\max_{1\leq M\leq N}\gamma^M\left|\max_{a_{t+M}} q\left(s_{t+M},a_{t+M}\right)-\max_{a_{t+M}} q'\left(s_{t+M},a_{t+M}\right)\right| \\
&\leq\max_{1\leq M\leq N}\gamma^M\max_{s_{t+N},a_{t+M}}\left|\max_{a_{t+M}} q\left(s_{t+M},a_{t+M}\right)-\max_{a_{t+M}} q'\left(s_{t+M},a_{t+M}\right)\right| \\
&\leq\max_{1\leq M\leq N}\gamma^M\max_{s_{t+N},a_{t+M}}\left|q\left(s_{t+M},a_{t+M}\right)-q'\left(s_{t+M},a_{t+M}\right)\right| \\
&=\max_{1\leq M\leq N}\gamma^M\left\|q-q'\right\| \\
&=\gamma\left\|q-q'\right\|
\end{aligned}
$$

□

Proof of Lemma 2:

For any s_t, a_t, we have

$$\begin{aligned}
&\mathcal{G}_{\mathcal{P}}^{N} Q^{\pi}\left(s_t, a_t\right) \\
&=\mathbb{E}_{\pi \sim \mathcal{P}, \tau_{N, \pi}^{s_t, a_t}}\left[\max_{1 \leq M \leq N} \sum_{n=0}^{M-1} \gamma^{n} r_{t+n}+\gamma^{M} \max_{a_{t+M}} Q^{\pi}\left(s_{t+M}, a_{t+M}\right)\right] \\
&\geq \mathbb{E}_{s_{t+1}}\left[r_t+\gamma \max_{a_{t+1}} Q^{\pi}\left(s_{t+1}, a_{t+1}\right)\right] \\
&\geq \mathbb{E}_{s_{t+1}, a_{t+1}}\left[r_t+\gamma Q^{\pi}\left(s_{t+1}, a_{t+1}\right)\right] \\
&=Q^{\pi}\left(s_t, a_t\right)
\end{aligned}$$

□

Proof of Lemma 3: The proof similar to the one of Lemma 1.

□

Proof of Theorem 2: (sketch) As 1) is already known, we proof 2).

It's obvious that $Q^*_{\mathcal{B}^N_{\mathcal{P}}} \leq \mathcal{B}^1 Q^*_{\mathcal{B}^N_{\mathcal{P}}} \leq (\mathcal{B}^1)^2 Q^*_{\mathcal{B}^N_{\mathcal{P}}} \leq \cdots = Q^*$.

If there exists one behavior policy that does not output optimal action, then there exist at least one s, a such that $Q^*_{\mathcal{B}^N_{\mathcal{P}}}(s,a) < \mathcal{B}^1 Q^*_{\mathcal{B}^N_{\mathcal{P}}}(s,a) \leq Q^*(s,a)$.

□

Proof of Lemma 4: First, we prove that for any s_t, a_t

$$\begin{aligned}
&\mathcal{G}_{\mathcal{P}}^{N} q\left(s_t, a_t\right) \\
&=\mathbb{E}_{\pi \sim \mathcal{P}, \tau_{N, \pi}^{s_t, a_t}}\left[\max_{1 \leq M \leq N} \sum_{n=0}^{M-1} \gamma^{n} r_{t+n}+\gamma^{M} \max_{a_{t+M}} q\left(s_{t+M}, a_{t+M}\right)\right] \\
&\geq \mathbb{E}_{s_{t+1}}\left[r_t+\gamma \max_{a_{t+1}} q\left(s_{t+1}, a_{t+1}\right)\right] \\
&=\mathcal{B}^{1} q\left(s_t, a_t\right)
\end{aligned}$$

If $q \leq Q^*$, then $\mathcal{G}^N_{\mathcal{P}} q \leq \mathcal{G}^N_{\mathcal{P}} Q^* = Q^*$ and $\mathcal{B}^1 q \leq \mathcal{B}^1 Q^* = Q^*$.

Let $s', a' = \arg\max_{s',a'} \left|\mathcal{G}^N_{\mathcal{P}} q(s',a') - Q^*(s',a')\right|$, we have $\|\mathcal{G}^N_{\mathcal{P}} q - Q^*\| = \left|\mathcal{G}^N_{\mathcal{P}} q(s',a') - Q^*(s',a')\right| \leq \left|\mathcal{B}^1 q(s',a') - Q^*(s',a')\right| \leq \max_{s,a}\left|\mathcal{B}^1 q(s,a) - Q^*(s,a)\right| = \|\mathcal{B}^1 q - Q^*\|$

□

Proof of Theorem 3: Similar to the proof in Lemma 4, if $q \geq q'$, then $\mathcal{G}^N_{\mathcal{P}} q \geq \mathcal{B}^1 q$. Therefore, we have $Q^{(k)}_{\mathcal{G}^N_{\mathcal{P}}} \geq Q^{(k)}_{\mathcal{B}^1}$ with the same initial $Q^{(0)}$. Then similar to the proof in Lemma 4, we complete the proof. □

Proof of Theorem 4: Let $\mathcal{P}^*$ denote the distribution of behavior policy which satisfies for any $\pi \in \{\pi | \mathcal{P}^*(\pi) > 0\}$ $\pi(s) = \arg\max_a Q^*(s,a)$.

$$
\begin{aligned}
&\left\|\mathcal{G}_{\mathcal{P}}^N q - Q^*\right\| \\
&= \max_{s_t,a_t} \Bigg| \mathbb{E}_{\pi\sim\mathcal{P}^*,\tau_{N,\pi}^{s_t,a_t}} \left[\max_{1\le M\le N} \sum_{n=0}^{M-1} \gamma^n r_{t+n} + \gamma^M \max_{a_{t+M}} q\left(s_{t+M}, a_{t+M}\right) \right] \\
&\qquad\qquad -\mathbb{E}_{\pi\sim\mathcal{P}^*,\tau_{N,\pi}^{s_t,a_t}} \left[\sum_{n=0}^{N-1} \gamma^n r_{t+n} + \gamma^N \max_{a_{t+N}} Q^*\left(s_{t+N}, a_{t+N}\right) \right] \Bigg| \\
&\le \max_{s_t,a_t} \Bigg| \mathbb{E}_{\pi\sim\mathcal{P}^*,\tau_{N,\pi}^{s_t,a_t}} \left[\sum_{n=0}^{N-1} \gamma^n r_{t+n} + \gamma^N \max_{a_{t+N}} q\left(s_{t+N}, a_{t+N}\right) \right] \\
&\qquad\qquad - \left(\mathbb{E}_{\pi\sim\mathcal{P}^*,\tau_{N,\pi}^{s_t,a_t}} \left[\sum_{n=0}^{N-1} \gamma^n r_{t+n} + \gamma^N \max_{a_{t+N}} Q^*\left(s_{t+N}, a_{t+N}\right) \right] \right) \Bigg| \\
&\le \max_{s_t,a_t} \mathbb{E}_{\pi\sim\mathcal{P}^*,\tau_{N,\pi}^{s_t,a_t}} \Bigg| \left(\sum_{n=0}^{N-1} \gamma^n r_{t+n} + \gamma^N \max_{a_{t+N}} q\left(s_{t+N}, a_{t+N}\right) \right) \\
&\qquad\qquad - \left(\sum_{n=0}^{N-1} \gamma^n r_{t+n} + \gamma^N \max_{a_{t+N}} Q^*\left(s_{t+N}, a_{t+N}\right) \right) \Bigg| \\
&= \max_{s_t,a_t} \mathbb{E}_{\pi\sim\mathcal{P}^*,\tau_{N,\pi}^{s_t,a_t}} \gamma^N \left| \max_{a_{t+N}} q\left(s_{t+N}, a_{t+N}\right) - \max_{a_{t+N}} Q^*\left(s_{t+N}, a_{t+N}\right) \right| \\
&\le \gamma^N \left\| q - Q^* \right\|
\end{aligned}
$$

□

Proof of Theorem 5:

$$
\begin{aligned}
&\left\|\mathcal{G}_{\mathcal{P}}^N q - Q^*\right\| \\
&= \max_{s_t,a_t} \left| \mathbb{E}_{\pi\sim\mathcal{P},\tau_{N,\pi}^{s_t,a_t}} \left[\max_{1\le M\le N} \sum_{n=0}^{M-1} \gamma^n r_{t+n} + \gamma^M \max_{a_{t+M}} q\left(s_{t+M}, a_{t+M}\right) \right] - Q^*\left(s_t, a_t\right) \right| \\
&\le \max_{s_t,a_t} \left| \mathbb{E}_{\pi\sim\mathcal{P},\tau_{N,\pi}^{s_t,a_t}} \left[\sum_{n=0}^{1} \gamma^n r_{t+n} + \gamma^2 \max_{a_{t+2}} q\left(s_{t+2}, a_{t+2}\right) \right] - Q^*\left(s_t, a_t\right) \right| \\
&= \max_{s_t,a_t} \left| \mathbb{E}_{\pi\sim\mathcal{P},\tau_{N,\pi}^{s_t,a_t}} \left[\sum_{n=0}^{1} \gamma^n r_{t+n} + \gamma^2 \max_{a_{t+2}} q\left(s_{t+2}, a_{t+2}\right) \right] - \mathbb{E}_{\pi\sim\mathcal{P},\tau_{N,\pi}^{s_t,a_t}} \left[r_t + \gamma \max_{a_{t+1}} Q^*\left(s_{t+1}, a_{t+1}\right) \right] \right| \\
&\le \gamma \max_{s_{t+1},a_{t+1}} \left| \mathbb{E}_{\pi\sim\mathcal{P},\tau_{N,\pi}^{s_{t+1},a_{t+1}}} \left[r_{t+1} + \gamma \max_{a_{t+2}} q\left(s_{t+2}, a_{t+2}\right) - \max_{a_{t+1}} Q^*\left(s_{t+1}, a_{t+1}\right) \right] \right| \\
&= \gamma \max_{s_t,a_t} \left| \mathbb{E}_{\pi\sim\mathcal{P},\tau_{N,\pi}^{s_t,a_t}} \left[r_t + \gamma \max_{a_{t+1}} q\left(s_{t+1}, a_{t+1}\right) - \max_{a_t} Q^*\left(s_t, a_t\right) \right] \right|
\end{aligned}
$$

For any s_t which satisfies $\left|\min_a \mathcal{B}^1 q(s_t,a) - \max_{a'} Q^*(s_t,a')\right| > \gamma \left\|q - Q^*\right\|_\infty$, we have already assumed $\pi(s) = \arg\max_a Q^*(s,a)$ (Condition 1). The for those s_t, we have

$$
\begin{aligned}
&\left| \mathbb{E}_{\pi\sim\mathcal{P},\tau_{N,\pi}^{s_t,a_t}} \left[r_t + \gamma \max_{a_{t+1}} q\left(s_{t+1}, a_{t+1}\right) - \max_{t} Q^*\left(s_t, a_t\right) \right] \right| \\
&= \left| \mathbb{E}_{\pi\sim\mathcal{P},\tau_{N,\pi}^{s_t,a_t}} \left[r_t + \gamma \max_{a_{t+1}} q\left(s_{t+1}, a_{t+1}\right) - r_t - \gamma \max_{a_{t+1}} Q^*\left(s_{t+1}, a_{t+1}\right) \right] \right| \\
&= \gamma \left| \mathbb{E}_{\pi\sim\mathcal{P},\tau_{N,\pi}^{s_t,a_t}} \left[\max_{a_{t+1}} q\left(s_{t+1}, a_{t+1}\right) - \max_{a_{t+1}} Q^*\left(s_{t+1}, a_{t+1}\right) \right] \right| \\
&\le \gamma \left\| q - Q^* \right\|
\end{aligned}
$$

Finally, we obtain

$$
\begin{aligned}
&\left\|\mathcal{G}_{\mathcal{P}}^N q - Q^*\right\| \\
&\le \gamma \max_{s_t,a_t} \left| \mathbb{E}_{\pi\sim\mathcal{P},\tau_{N,\pi}^{s_t,a_t}} \left[r_t + \gamma \max_{a_{t+1}} q\left(s_{t+1}, a_{t+1}\right) - \max_{s_t} Q^*\left(s_t, a_t\right) \right] \right| \\
&\le \gamma^2 \left\| q - Q^* \right\|
\end{aligned}
$$

□

B THE BEHAVIOR OF ALGORITHMS ON THE GRID WORLD EXAMPLE

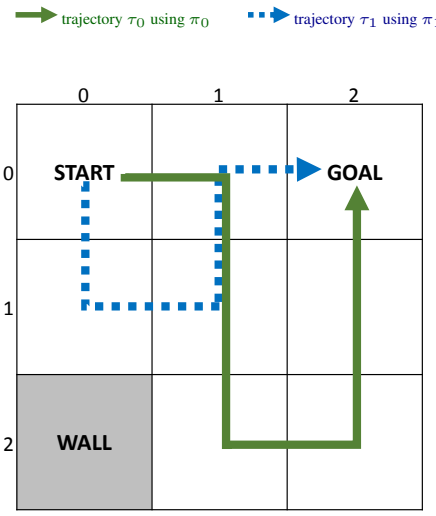


Figure 7: Grid World Problem.

Figure 8: The optimal value function Q^*. The value within the cell shows the Q value of that state and corresponding action, e.g., the value on the top of cell $[2, 1]$ shows $Q([2, 1], \uparrow)$.

B.1 Q LEARNING

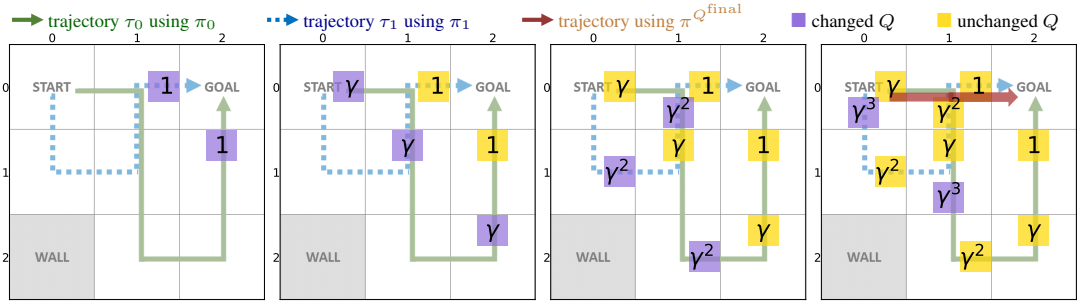
Q learning requires 4 iterations to complete back-propagation in the grid world example (as shown in Figure 9).

B.2 MULTI-STEP VALUE ITERATION

As shown in Figure 10a, for 2-step Q learning, the final value function $Q^{\text{final}}([0, 0], \rightarrow) = \gamma^3 \leq \gamma = Q^*([0, 0], \rightarrow)$. And the final policy also selects the down action $\downarrow$ as an optimal action at state $[0, 0]$, which in fact is a bad action. This is because $Q([0, 0], \rightarrow)$ is computed along the trajectory τ_0 for 2 steps, $Q([0, 0], \rightarrow) = r([0, 0], \rightarrow) + \gamma r([0, 1], \downarrow) + \gamma^2 \max_a Q([1, 1], a) = \gamma^3$. A formal statement is as follows. As a result, the final policy also selects down action $\downarrow$ as an optimal action at state $[0, 0]$, which in fact is a bad action.

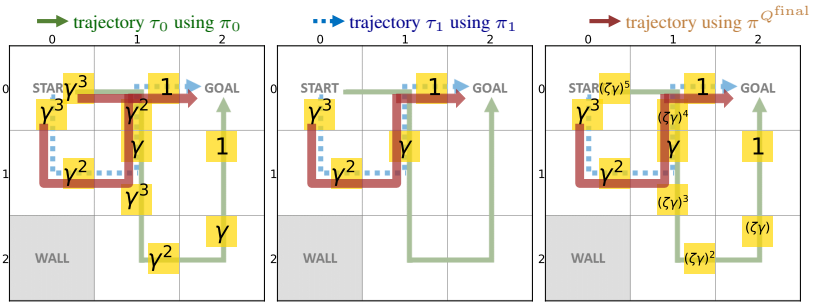
B.3 MULTI-STEP POLICY ITERATION

We choose π_1 as the target (current) policy (as it performs better with trajectory τ_1). Thus trajectory τ_0 is off-policy data to the target policy π_1. For convenience, assume that for any s_t, a_t in trajectory



(a) Iteration 1 (b) Iteration 2 (c) Iteration 3 (d) Iteration 4

Figure 9: Value function update process of Q learning. The Q value with purple background means that the value is changed at corresponding iteration, while one with the yellow background means that the value is unchanged. The initial Q values are 0, which are not plotted to make it look clear.



(a) 2-Step Q Learning (b) On-Policy Monte Carlo (c) Off-Policy Monte Carlo

Figure 10: Final value function Q^{final} of the algorithms and the trajectory executed by policy $\pi^{Q^{\text{final}}}$.

τ_0 , $\frac{\pi_1(a_t|s_t)}{\pi_0(a_t|s_t)} = \zeta$ and $\zeta < 1$ (as a_t are chosen by the behavior policy π_0 and usually $\pi_1(a_t|s_t) < \pi_0(a_t|s_t)$). The estimated value function of off-policy Monte Carlo is shown in Figure 10c.

Both on/off-policy PI-based algorithms can *not* output the optimal policy (as shown in Figure 10). This is because neither of the two policies are the global optimal policy, while PI approach attempts to approximate the value function of a specific policy but not the optimal value function.

C Implementation Details

Hyperparameter	**Value**
learning rate	3×10^{-3}
timesteps per epoch	1024
hidden units of Q network	(100) (Mountain Car) (128,128,128) (Acrobot)
buffer size	100 (Mountain Car) 50 (Acrobot)
start exploration rate ϵ	0.3

Table 2: Hyperparameters of the implemented algorithms.