Reward Centering

Abhishek Naik[1,2], **Yi Wan**[3], **Manan Tomar**[1,2], **Richard S. Sutton**[1,2]
{abhishek.naik,mtomar,rsutton}@ualberta.ca, yiwan@meta.com
[1]University of Alberta [2]Alberta Machine Intelligence Institute [3]Meta AI

Abstract

We show that discounted methods for solving continuing reinforcement learning problems can perform significantly better if they center their rewards by subtracting out the rewards' empirical average. The improvement is substantial at commonly used discount factors and increases further as the discount factor approaches one. In addition, we show that if a *problem's* rewards are shifted by a constant, then standard methods perform much worse, whereas methods with reward centering are unaffected. Estimating the average reward is straightforward in the on-policy setting; we propose a slightly more sophisticated method for the off-policy setting. Reward centering is a very general idea, so we expect almost every reinforcement-learning algorithm to benefit by the addition of reward centering.

Reinforcement learning is a computational approach to learning from interaction, where the goal of a learning agent is to obtain as much reward as possible (Sutton & Barto, 2018). In many problems of interest, the stream of interaction between the agent and the environment is *continuing* and cannot be naturally separated into disjoint subsequences or *episodes*. In continuing problems, agents experience infinitely many rewards, hence a viable way of evaluating performance is to measure the *average reward* obtained per step, or the rate of reward, with equal weight given to immediate and delayed rewards. The *discounted-reward* formulation offers another way to interpret a sum of infinite rewards by discounting delayed rewards in favor of immediate rewards. The two problem formulations are typically studied separately, each having a set of solution methods or algorithms.

In this paper, we show that the simple idea of estimating and subtracting the average reward from the observed rewards can lead to a significant improvement in performance when using standard *discounting* methods such as TD-learning (Sutton, 1988a) or Q-learning (Watkins & Dayan, 1992). The underlying theory dates back to 1962 with Blackwell's seminal work on dynamic programming in discrete Markov decision processes (MDPs). We are still realizing some of its deeper implications, and we shall discuss two in particular.

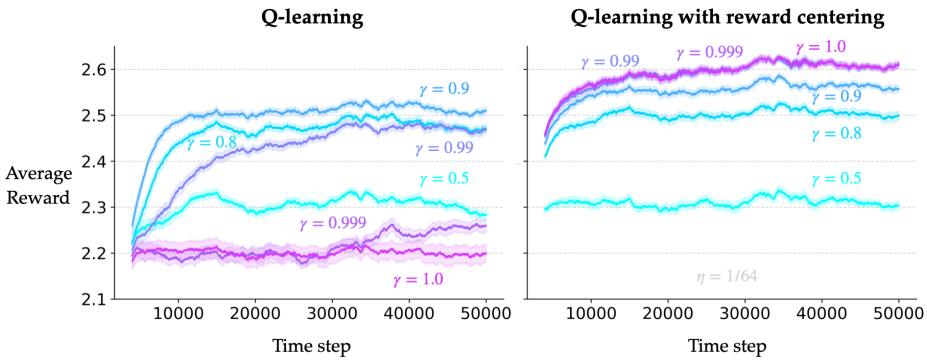


Figure 1: Learning curves showing the difference in performance of Q-learning with and without reward centering for different discount factors on the classic Access-Control Queuing problem. Plotted is the average per-step reward obtained by the agent across 50 runs w.r.t. the number of time steps of interaction. The shaded region denotes one standard error. Details in Section 4.

1. Mean-centering the rewards removes a state-independent constant (that scales inversely with $1-\gamma$, where γ denotes the discount factor) from the value estimates, enabling the value-function approximator to focus on the relative differences between the states and actions. As a result, values corresponding to discount factors arbitrarily close to one can be estimated relatively easily (e.g., without any degradation in performance; see Figure 1).
2. Furthermore, mean-centering the rewards (unsurprisingly) makes standard methods robust to any constant offset in the rewards. This can be useful when applying RL algorithms in problems where the properties of the reward signal are unknown or changing.

We begin with *what* reward centering is and *why* it can be beneficial (Section 1). We then show *how* reward centering can be done, starting with the simplest form (within the prediction problem), and show that it can be highly effective when used with discounted-reward algorithms such TD-learning (Section 2). However, the off-policy setting requires more sophistication, and we propose another way of reward centering based on recent advances in the *average-reward* formulation for RL (Section 3). Next, we perform a case study of using reward centering with Q-learning, in which we (a) present a convergence result based on recent work by Devraj and Meyn (2021), and (b) showcase consistent trends across a series of control problems that require tabular, linear, and non-linear function approximation (Section 4). Finally, we discuss the limitations of the proposed methods and propose directions of future work (Section 5).

1 Theory of Reward Centering

We formalize the interaction between the agent and the environment by a finite MDP $\mathcal{M} \doteq (\mathcal{S}, \mathcal{A}, \mathcal{R}, p)$, where $\mathcal{S}$ denotes the set of states, $\mathcal{A}$ denotes the set of actions, $\mathcal{R}$ denotes the set of rewards, and $p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ denotes the transition dynamics. At time step t, the agent is in state $S_t \in \mathcal{S}$, takes action $A_t \in \mathcal{A}$ using a behavior policy $b : \mathcal{A} \times \mathcal{S} \to [0, 1]$, observes the next state $S_{t+1} \in \mathcal{S}$ and reward $R_{t+1} \in \mathcal{R}$ according to the transition dynamics $p(s', r \mid s, a) = \Pr(S_{t+1} = s', R_t = r \mid S_t = s, A_t = a)$. We consider continuing problems, where the agent-environment interaction goes on *ad infinitum*. The agent's goal is to maximize the average reward obtained over a long time (formally defined in (2)). We consider methods that try to achieve this goal by estimating the expected discounted sum of rewards from each state for $\gamma \in [0, 1)$: $v^\gamma_\pi(s) \doteq \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \mid S_t = s, A_{t:\infty} \sim \pi], \forall s$. Here, the discount factor is not part of the problem but an algorithm parameter (see Naik et al. (2019) or Sutton & Barto's (2018) Section 10.4 for an extended discussion on objectives for continuing problems).

Reward centering is a simple idea: subtract the empirical average of the observed rewards from the rewards. Doing so makes the rewards appear *mean-centered*. The effect of mean-centered rewards is well known in the bandit setting. For instance, Sutton and Barto (2018: Section 2.8) demonstrated that estimating and subtracting the average reward from the observed rewards can significantly improve the rate of learning. The benefits not only extend to the full RL problem, but are magnified as the discount factor γ approaches one (bandit problems correspond to $\gamma = 0$).

The reason underlying the benefits of reward centering is revealed by the Laurent-series decomposition of the discounted value function. The discounted value function can be decomposed into two parts, one of which is a constant that does not depend on states or actions and hence is not involved in, say, action selection. Mathematically, for the tabular discounted value function $v^\gamma_\pi : \mathcal{S} \to \mathbb{R}$ of a policy π corresponding to a discount factor γ:

$$\boxed{v^\gamma_\pi(s) = \frac{r(\pi)}{1-\gamma} + \tilde{v}_\pi(s) + e^\gamma_\pi(s),} \quad \forall s, \tag{1}$$

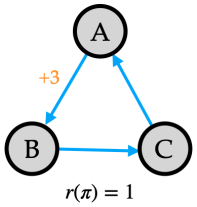
where $r(\pi)$ is the state-independent average reward obtained by policy π and $\tilde{v}_\pi(s)$ is the *differential* value of state s, each defined for ergodic MDPs (for ease of exposition) as (e.g., Wan et al., 2021):

$$r(\pi) \doteq \lim_{n\to\infty} \frac{1}{n} \sum_{t=1}^{n} \mathbb{E}\big[R_t \mid S_0, A_{0:t-1} \sim \pi\big], \quad \tilde{v}_\pi(s) \doteq \mathbb{E}\left[\sum_{k=1}^{\infty} \big(R_{t+k} - r(\pi)\big) \mid S_t = s, A_{t:\infty} \sim \pi\right], \tag{2}$$

and $e^\gamma_\pi(s)$ denotes an error term that go to zero as the discount factor goes to one (Blackwell, 1962: Theorem 4a; also see Puterman's (1994) Corollary 8.2.4). This decomposition of the state values (1) also implies a similar decomposition for state–action values.

The constant state–action-independent term explains the improvements in the bandit setting (see Sutton & Barto's (2018) Figure 2.5), where the action-value estimates are initialized to zero and the true values are centered around +4. The actions are selected based on their *relative* values, but each action-value estimate independently learns the constant offset. Approximation errors in estimating the offset can easily mask the relative differences in actions, especially if the offset is large.

In the full RL problem, the state(–action)-independent offset can indeed be quite large. For intuition, consider a three-state Markov reward process in Figure 2 induced by a policy π in some MDP. There is a reward of +3 on going from state s_A to s_B; 0 otherwise. The average reward $r(\pi)$ is 1. The discounted state values for three discount factors are shown in Figure 2. Note the magnitude of the values and especially the jump when the discount factor is increased. Now consider the values with the constant offset subtracted from each state, $v^\gamma_\pi(s) - r(\pi)/(1-\gamma)$, which we call the *centered* discounted values: $\tilde{v}^\gamma_\pi$. The magnitudes of the centered values are much smaller, and differ only slightly when the discount factor is increased. The differential values are also shown for reference. These trends hold in general: for any problem, the magnitude of the discounted values shoots up as the discount factor approaches one (and may cause computational or numerical instability); meanwhile, the centered discounted values do not change much and approach the differential values.



	$\frac{r(\pi)}{1-\gamma}$
$\gamma = 0.8$	5
$\gamma = 0.9$	10
$\gamma = 0.99$	100

		s_A	s_B	s_C
Standard discounted values	$\gamma = 0.8$	6.15	3.93	4.92
	$\gamma = 0.9$	11.07	8.97	9.96
	$\gamma = 0.99$	101.01	98.99	99.99
Centered discounted values	$\gamma = 0.8$	1.15	-1.07	-0.08
	$\gamma = 0.9$	1.07	-1.03	-0.04
	$\gamma = 0.99$	1.01	-1.01	-0.01
Differential values		1	-1	0

Figure 2: Comparison of the standard discounted values and the centered discounted values on a simple three-state problem.

Formally, the centered discounted values are the expected discounted sum of mean-centered rewards:

$$\tilde{v}^\gamma_\pi(s) \doteq \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \big(R_{t+1} - r(\pi)\big) \mid S_t = s, A_{t:\infty} \sim \pi\right], \quad v^\gamma_\pi(s) = \frac{r(\pi)}{1-\gamma} + \overbrace{\tilde{v}_\pi(s) + e^\gamma_\pi(s)}^{\tilde{v}^\gamma_\pi(s)}, \quad \forall s, \tag{3}$$

where $\gamma \in [0, 1]$. When $\gamma = 1$, the centered discounted values are the same as the differential values, that is, $\tilde{v}^\gamma_\pi(s) = \tilde{v}_\pi(s), \forall s$; when $r(\pi) = 0, \tilde{v}^\gamma_\pi(s) = v^\gamma_\pi(s), \forall s$. More generally, the centered discounted values are the differential values plus the error terms from the Laurent-series decomposition (1).

Reward centering thus enables capturing all the information within the discounted value function via two components: (1) the constant average reward and (2) the centered discounted value function. Such a decomposition can be immensely valuable: (a) As $\gamma \to 1$, the discounted values tend to explode but the centered discounted values remain small and tractable. (b) If the *problems*' rewards are shifted by a constant c, the magnitude of the discounted values increases by $c/(1-\gamma)$; the centered discounted values are unchanged because the average reward increases by c. Both of these effects are demonstrated in the following sections.

Reward centering also enables the design of algorithms in which the discount factor (an algorithm parameter) can be changed within the lifetime of a learning agent. This is usually inefficient or ineffective with standard algorithms since the uncentered discounted values can change massively (see, e.g., Figure 2). In contrast, the centered values may not change much, and the changes become minuscule as the discount factor approaches 1. We discuss this exciting direction in the final section.

To avail these potential benefits, we need to estimate the average reward from data. We shall now see that even the simplest method can take us quite far.

2 Simple Reward Centering

The simplest way to estimate the average reward is to maintain a running average of the rewards observed so far. That is, if $\bar{R} \in \mathbb{R}$ denotes the estimate of the average reward, after t time steps, $\bar{R}_t = \sum_{k=1}^{t} R_k$. More generally, the estimate can be updated with a step-size parameter β_t:

$$\bar{R}_{t+1} \doteq \bar{R}_t + \beta_t(R_{t+1} - \bar{R}_t). \tag{4}$$

This update leads to an unbiased estimate of the average reward $r(\pi)$ corresponding to the policy π with which the data is generated (that is, in the on-policy setting) if the step sizes follow the standard conditions (Robbins & Monro, 1951). To estimate the centered discounted values for a given policy π, we can use standard algorithms with the rewards centered using the current estimate of the average reward. For instance, after the transition $(S_t, A_t, R_{t+1}, S_{t+1})$, the TD-learning algorithm can update its centered-value estimates $\tilde{V}^\gamma : \mathcal{S} \to \mathbb{R}$ using a step-size parameter α_t as:

$$\tilde{V}_{t+1}(S_t) \doteq V_t(S_t) + \alpha_t\big[(R_{t+1} - \bar{R}_t) + \gamma \tilde{V}_t^\gamma(S_{t+1}) - \tilde{V}_t^\gamma(S_t)\big]. \tag{5}$$

We call this on-policy algorithm that updates the value and average-reward estimates as in (4) and (5) as *TD-learning with simple reward centering*. As we demonstrate shortly, this simple approach to reward centering works quite well.

Consider an MDP with seven states in a row with two actions in each state. The right action from the rightmost state leads to the middle state with reward +7 and the left action from the leftmost state leads to the middle state with reward +1; all other transitions have zero rewards. The target policy takes both actions in each state with equal probability, that is, $\pi(\text{left}|\cdot) = \pi(\text{right}|\cdot) = 0.5$. The average reward corresponding to this policy is 0.25. We tested three variants of the TD-learning algorithm: (a) standard TD-learning, (b) TD-learning with rewards that are centered by an oracle, (c) TD-learning with simple reward centering. We performed the same experiment with two discount factors, $\gamma = 0.9$ and 0.99, and evaluated the root mean-squared value error (RMSVE) of the estimates and the true discounted values at each time step w.r.t. the steady-state distribution of the states induced by the target policy π. The centering and oracle-centered methods estimate the centered discounted value function $\tilde{\mathbf{v}}_\pi^\gamma$, so for an apples-to-apples comparison, we added $\bar{R}_t/(1-\gamma)$ to the centered estimates to compute the uncentered value estimates at each time step.

The leftmost column of Figure 3 shows the learning curves for this on-policy experiment. The plots also contain the learning curves for another centering method that we shall discuss in the next section. In each plot, the curves correspond to the value of α that resulted in the fastest rate of learning for uncentered TD-learning baseline over the training period. For the centering methods, we chose a good value of η, where $\beta_t = \eta\alpha_t$ (without loss of generality), from a coarse search over a broad range. Each solid point represents the RMSVE averaged over the 50 independent runs; the shaded region denotes one standard error estimated from the data. Taken together, these plots help assess the average high-level trends that we are interested in. There is no separate training and testing period. All the experiments in this paper follow this general experimental setup; more details (including number of runs, range of parameter values tested) are in Appendix C.

First, note that the learning curves start much lower when the rewards are centered by an oracle; for the other approaches, the first error is of the order $r(\pi)/(1-\gamma)$. Standard TD-learning (blue) eventually converges to the same error rate as the oracle-centered version (orange), as expected. Learning the average reward and subtracting it (green) indeed helps reduce the RMSVE much faster compared to when there is no centering. However, the eventual error rate is slightly higher, which is expected because the average-reward estimate is changing over time, leading to more variance in the updates compared to the uncentered or oracle-centered version. Similar trends hold for the larger discount factor (lower left), with the uncentered approach appearing much slower in comparison (note the difference in axes' scales). In both cases, we verified that the average-reward estimate across the runs was around 0.25.

These experiments show that the simple reward-centering technique can be quite effective in the on-policy setting, and the effect is more pronounced for larger discount factors.

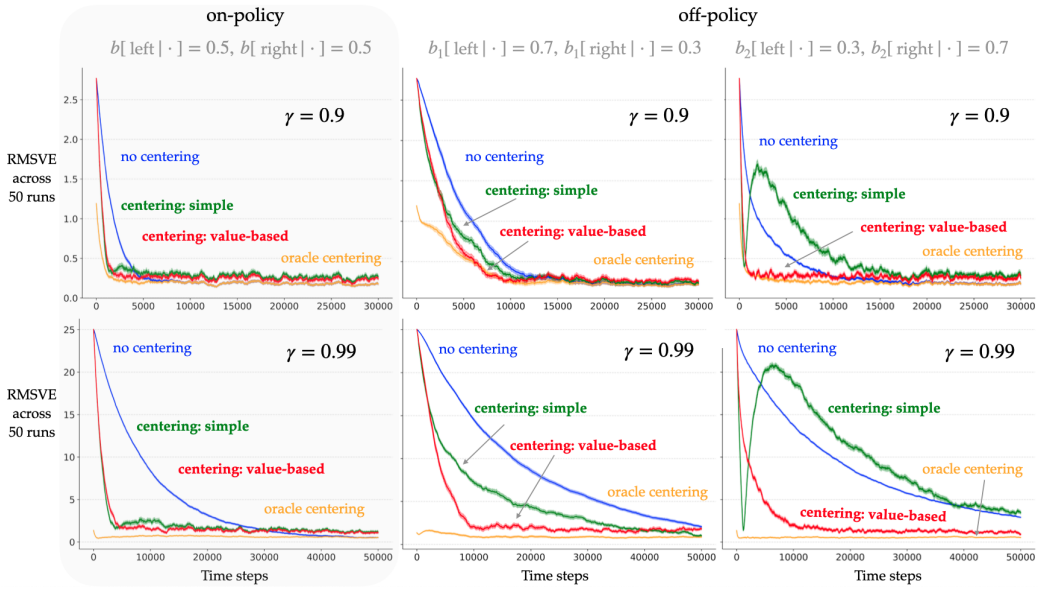


Figure 3: Learning curves demonstrating the performance of TD-learning with and without reward centering on one on-policy problem and two off-policy problems.

Limitations in the Off-policy Setting: (4) leads to an unbiased estimate of the behavior policy's average reward, which means that in the *off-policy* setting, the average-reward estimate $\bar{R}$ will converge to $r(b)$, not to $r(\pi)$. Adding an importance-sampling ratio to the update is not enough to guarantee convergence to $r(\pi)$ because importance sampling only corrects the mismatch in action distributions, not the mismatch in the resulting state distributions.

Let us consider the effect of an inaccurate estimate of the average reward. First, note that the centered discounted value function also satisfies a recursive Bellman equation:

$$\tilde{v}^\gamma(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r \mid s, a)\big[r - \bar{r} + \gamma \tilde{v}^\gamma(s')\big], \quad \text{or,} \quad \tilde{\mathbf{v}}^\gamma = \mathbf{r}_\pi - \bar{r}\mathbf{1} + \gamma \mathbf{P}_\pi \tilde{\mathbf{v}}^\gamma, \tag{6}$$

where, $\tilde{\mathbf{v}}^\gamma$ denotes a vector in $\mathbb{R}^{|\mathcal{S}|}$, $\mathbf{r}_\pi$ is the vector of the expected one-step reward from each state, $\bar{r}$ is a scalar variable, $\mathbf{1}$ is a vector of all ones, and $\mathbf{P}_\pi$ is the state-to-state transition matrix induced by the policy π. It is easy to verify that the solution tuples $(\tilde{\mathbf{v}}^\gamma, \bar{r})$ of (6) are of the form $\big(\tilde{\mathbf{v}}^\gamma_\pi + c\mathbf{1}, r(\pi) - c(1-\gamma)\big), \forall c \in \mathbb{R}$, where $\tilde{\mathbf{v}}^\gamma_\pi$ denotes the centered differential value function (3) corresponding to policy π and discount factor γ. Equivalently, we can write the family of solutions as $\big(\tilde{\mathbf{v}}^\gamma_\pi + \frac{k}{1-\gamma}\mathbf{1}, r(\pi) - k\big), \forall k \in \mathbb{R}$, which shows that if the average-reward estimate is off by k, then the centered discounted values each have a constant offset of $k/(1-\gamma)$. This is undesirable. The primary motivation of reward centering is to eliminate the potentially large offset from the estimates. So we desire a way to estimate the target policy's average reward while behaving according to a different behavior policy.

However, note that an inaccurate estimation of the average reward is not a deal-breaker: standard algorithms that do not center the rewards can be perceived as using a fixed inaccurate estimate of the average reward (zero), yet they are guaranteed to converge to the true values of the target policy in the tabular case. So the issue is less about convergence and more about the rate of learning. Estimating the average reward accurately may yield better sample-complexity bounds when using standard methods than simply estimating the uncentered values (e.g., the bounds for Q-learning involve powers of $1/(1-\gamma)$ (Qu & Wierman, 2020; Wainwright, 2019; Even-Dar et al., 2003)). We also saw in Figure 3 that when the rewards are centered by an oracle, the rate of learning is much higher compared to when there is no centering.

In summary, the effectiveness of reward centering increases with the accuracy of the average-reward estimate. Thus, even the simple method of reward centering (4) can be highly effective when the average reward of the behavior policy is close to that of the target policy. This may be true when the

two policies are similar, like a greedy target policy and an ϵ-greedy behavior policy with a relatively small value of ϵ. However, the benefits of reward centering in terms of rate of learning may reduce and even disappear as the difference in the two policies increases. We shall now consider a subtly advanced approach to estimate the average reward more accurately in the off-policy setting.

3 Value-based Reward Centering

We draw inspiration from the *average-reward* formulation, where estimating the average reward in the off-policy setting is a pertinent problem. In particular, Wan et al. (2021) recently showed that using the *temporal-difference* (TD) error in average-reward RL (instead of the *conventional* error in (4)) leads to an unbiased estimate of the reward rate in the tabular off-policy setting. It turns out that this idea from the average-reward formulation is quite effective even in the discounted-reward formulation, which is the focus of this paper. We show that if the behavior policy takes all the actions that the target policy does (the exact distribution over actions may differ arbitrarily), then we get a good approximation of the average reward of the target policy using the TD error:

$$\tilde{V}^{\gamma}_{t+1}(S_t) \doteq \tilde{V}^{\gamma}_{t}(S_t) + \alpha_t \, \rho_t \, \delta_t, \tag{7}$$

$$\bar{R}_{t+1} \doteq \bar{R}_t + \eta \, \alpha_t \, \rho_t \, \delta_t, \tag{8}$$

where, $\delta_t \doteq (R_{t+1} - \bar{R}_t) + \gamma \tilde{V}^{\gamma}_{t}(S_{t+1}) - \tilde{V}^{\gamma}_{t}(S_t)$ is the TD error and $\rho_t \doteq \pi(A_t|S_t)/b(A_t|S_t)$ is the importance-sampling ratio. Since this centering approach involves values in addition to the reward, we call it *value-based* centering. Unlike with simple centering, the convergence of the average-reward estimate and the value estimates is now interdependent. We present a convergence result in the next section for the control problem.

The first column of Figure 3 shows plots for value-based centering in the on-policy problem from the previous section, where the target policy picks both actions with equal probability. Value-based centering (red) appears as good as simple centering (green) in terms of the rate of learning and asymptotic error. The other two columns show plots for two off-policy experiments with behavior policies $[b_1(\text{left}|\cdot), b_1(\text{right}|\cdot)] = [0.7, 0.3]$, $[b_2(\text{left}|\cdot), b_2(\text{right}|\cdot)] = [0.3, 0.7]$. The two different behavior policies are symmetric but result in different trends. Corresponding to b_1, we saw that value-based centering resulted in a lower RMSVE faster than simple centering for both values of γ, and the final error rate was roughly the same. As expected, the simple approach estimated the average reward incorrectly and hence the learned values were relatively larger than with value-based centering (but not as large as when there was no centering). The results with b_2 were more interesting. The RMSVE reduced rapidly at first with simple centering, then rose sharply, and then reduced again. This is because the average-reward estimate was initialized to zero and it converged to around 0.5 (because b_2 skews the agent's state distribution towards the more-rewarding right side). When the estimate passed the true value of 0.25, the RMSVE was quite low, however, the estimate quickly climbed to 0.5, resulting in the peak in RMSVE. Eventually the value estimates settled to values corresponding to an average-reward estimate of around 0.5. In contrast, the average-reward estimate was much closer to the true value when using value-based centering, resulting in a smoother learning curve. The effects were amplified with the larger discount factor (bottom row).

Overall, we observed that reward centering can improve the rate of learning of discounted-reward prediction algorithms such as TD-learning, especially for large discount factors. While the simple way to center rewards is quite effective, value-based reward centering is better suited for general off-policy problems. We shall now consider reward centering within the control setting.

4 Case Study: Q-learning with Reward Centering

In this section, we examine the effects of reward centering when used alongside the Q-learning algorithm (Watkins & Dayan, 1992). In particular, we first present a convergence result based on recent work by Devraj and Meyn (2021). Next, using various control problems, we empirically study the effects of reward centering on tabular, linear, and non-linear variants of Q-learning.

Theory: The prevalence of Q-learning can be largely attributed to it being an off-policy algorithm: in the tabular case, it is guaranteed to converge to the value function of optimal policy while collecting data from an arbitrary behavior policy—even a random policy. Given its off-policy nature, we augment Q-learning with value-based reward centering. Since we use tabular, linear, and non-linear versions of this algorithm, we present a general form of its updates. At each time step, given an observation, the agent converts it into a feature vector $\mathbf{x}_t \in \mathbb{R}^d$, selects an action A_t, observes the reward signal R_{t+1} and the next observation, which it converts into $\mathbf{x}_{t+1}$, and so on. In the tabular case, $\mathbf{x}_t$ is a one-hot vector of the size of the state space; in the linear case, $\mathbf{x}_t$ may be a tile-coding representation; in the non-linear case, $\mathbf{x}_t$ is the output of the last non-linear layer of an artificial neural network. In each case, the agent linearly combines the feature vector with an action-specific weight vector $\mathbf{w}^a \in \mathbb{R}^d, \forall a$ to obtain the action-value estimate $\hat{q}$. At time step t, with the knowledge of transition $(\mathbf{x}_t, A_t, R_{t+1}, \mathbf{x}_{t+1})$, Q-learning with value-based reward centering updates the average-reward estimate and the per-action weights:

$$\mathbf{w}_{t+1}^{A_t} \doteq \mathbf{w}_t^{A_t} + \alpha_t\, \delta_t\, \nabla_{\mathbf{w}_t} \hat{q}(\mathbf{x}_t, A_t), \tag{9}$$

$$\bar{R}_{t+1} \doteq \bar{R}_t + \eta\, \alpha_t\, \delta_t, \tag{10}$$

$$\text{where,} \quad \delta_t \doteq R_{t+1} - \bar{R}_t + \gamma \max_a (\mathbf{w}_t^a)^\top \mathbf{x}_{t+1} - (\mathbf{w}_t^{A_t})^\top \mathbf{x}_t.$$

The full pseudocode for all algorithms is in Appendix A. We present the informal convergence-theorem statement here; the full theorem statement, proof, and analysis are in Appendix B.

Theorem 1. *If the Markov chain induced by the stationary behavior policy is irreducible and a per-state–action step size is reduced appropriately, tabular Q-learning with value-based reward centering (9–10) converges almost surely: Q_t and $\bar{R}_t$ converge to a particular solution $(\tilde{q}^\gamma, \bar{r})$ of the following Bellman equations:*

$$\tilde{q}^\gamma(s,a) = \sum_{s',r} p(s', r \mid s, a)\big(r - \bar{r} + \gamma \max_{a'} \tilde{q}^\gamma(s', a')\big). \tag{11}$$

The convergence proof is a consequence of important recent work by Devraj and Meyn (2021), who showed that subtracting a quantity from the rewards in Q-learning can result in a significantly better sample-complexity bound. Depending on the quantity subtracted, there is a whole family of Q-learning variants that converge almost surely in the tabular case to $\tilde{\mathbf{Q}}_\infty^\gamma = \mathbf{q}_*^\gamma - k/(1-\gamma)\mathbf{1}$, where $\tilde{\mathbf{Q}}_\infty^\gamma$ denotes the vector of asymptotic value estimates, $\mathbf{q}_*^\gamma$ denotes the discounted action-value function of the optimal policy π_γ^* corresponding to the discount factor γ, and k depends on $\mathbf{q}_*^\gamma$ and two algorithm parameters μ and κ. Recall that the standard (uncentered) discounted value function $\mathbf{q}_*^\gamma$ has a state–action-independent offset of $r(\pi_\gamma^*)/(1-\gamma)$. Relative Q-learning can remove $k/(1-\gamma)$ of it. This is very promising. Devraj and Meyn left the choice of μ and κ as open questions. We show that Q-learning with value-based centering can be seen as an instance of their algorithm family with particular choices of μ and κ. We further show (in Appendix B) that these choices can significantly reduce the state-independent offset. The equivalence enabled us to use their theoretical machinery to show almost-sure convergence and inherit strong variance-reduction properties.

Experiments: We now present results of Q-learning with and without centering on a set of control problems with tabular, linear, and non-linear function approximation. The problems are primarily from CSuite (Zhao et al., 2022). The repository specifies each problem in detail; we provide high-level descriptions here. We start the assessment in a tabular problem and then proceed to problems that require function approximation.

The Access-Control Queuing problem (Sutton & Barto, 2018) is a continuing problem in which the agent manages the access of incoming jobs to a set of servers. A job arrives at the front of the queue with one of four priorities with equal probability, and the agent has to decide at each time step whether to accept or reject the job based on the number of free servers left. If a job is accepted, the agent gets a positive reward proportional to the job's priority ($\{1, 2, 4, 8\}$); if rejected, the job is removed from the queue and the agent gets zero reward. At each time step, occupied servers get free with a certain probability, and the agent can observe the number of servers that are currently free as well as the priority of the job at the front of the queue.

Figure 1 shows the results of standard Q-learning (without centering) and Q-learning with value-based centering. For Q-learning, the curves correspond to the step-size parameters that resulted in the fastest learning over the training period (quantified by the area under the learning curve). For Q-learning with centering, they correspond to the best step-size parameters for a fixed value of η (shown in grey in the figure); this does not always mean the best (α, η) pair but that is okay since the results were robust to the choice of η. Throughout this section we followed this same practice of picking hyperparameters to plot learning curves.

The performance of Q-learning with centering did not degrade when the discount factor was close to one, unlike when there was no centering. For each discount factor, the performance with centering matched or exceeded that of the standard uncentered method. To verify if centering indeed helped remove the potentially large state-independent term, we checked the magnitude of the learned values. One way is to compute the magnitude across all state-action pairs. However, this approach typically leads to a poor approximation of the magnitude of learned values because many states (especially ones with low true values) may not occur frequently in the agent's ϵ-greedy interactions with the environment and hence their estimated values may stay close to their initialization. Instead, we checked the values of states that actually occur in the agent's stream of experience, in particular the maximum action value (used to choose the argmax action) of the last 10% states that occurred during training. Table 1 shows these values for the parameters corresponding to Figure 1's learning curves. As γ increased, the magnitude of learned values increased sharply with standard Q-learning but remained small with reward centering (as expected from the theory in Section 1).

Table 1: Magnitude of learned values

γ	**Without centering**	**With centering**
0.5	4.78	0.17
0.8	12.95	0.17
0.9	26.57	0.12
0.99	267.91	0.42
0.999	1434.47	0.51

These trends were quite general across the range of parameter values tested. Figure 4 shows the performance sensitivity to the methods' parameters. In particular, the x-axis denotes the step-size parameter α and the y-axis denotes the average reward obtained during the entire training period (which reflects the rate of learning). For both methods, the different curves correspond to different discount factors. The three plots on the right correspond to different values of the centering step-size parameter η. We saw the performance of Q-learning without centering deteriorated with large discount factors for a broad range of the step-size parameter α. In contrast, with centering, the performance did not degrade; in fact, it improved all the way till $\gamma = 1$ for a wide range of η values. In addition, its performance was not sensitive to the choice of η.

We also observed the rate of learning of the standard Q-learning algorithm is significantly affected by a constant shift in the *problems'* rewards. Note that adding a constant to all the rewards does not change the ordering of the policies according to the total-reward or the average-reward criterion in continuing problems. Figure 5 shows the behaviors of Q-learning with and without centering when applied to five problem variants with one of $\{-8, -4, 0, 4, 8\}$ added to all the rewards. To compare the resulting rate of rewards across the problems, the plots are shifted post-hoc (for instance, in the problem variant where the rewards were shifted by 8, after training, the same number was subtracted from all the rewards that the agent obtained). The behavior of Q-learning without centering was substantially different on all the problem variants. Q-learning with centering, unsurprisingly, results

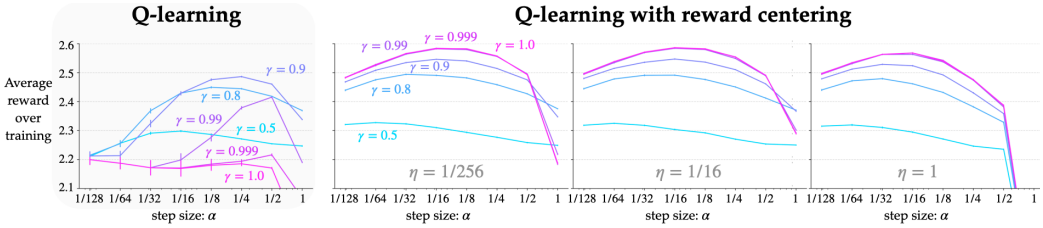


Figure 4: Parameter studies showing the sensitivity of the algorithms' performance to their parameters on the Access-Control problem. The error bars indicate one standard error, which at times is less than the width of the lines.

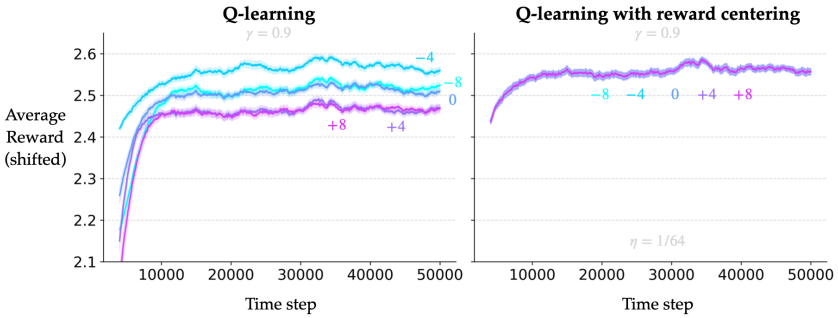
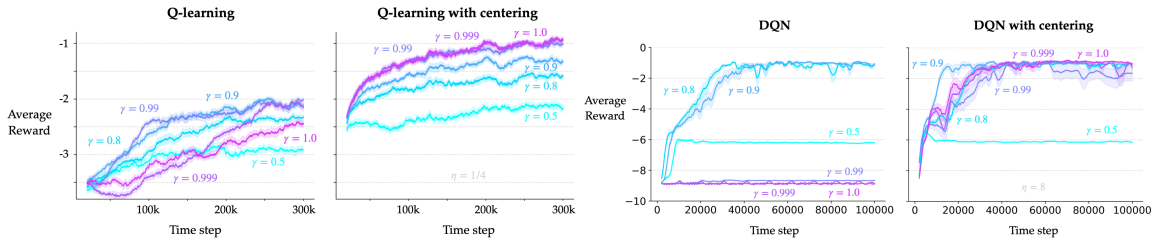


Figure 5: Learning curves on slight variants of the Access-Control Queuing problem with all the rewards shifted by a constant integer. The y-axis is shifted to compare learning curves for all the variants on the same scale. More details in-text.

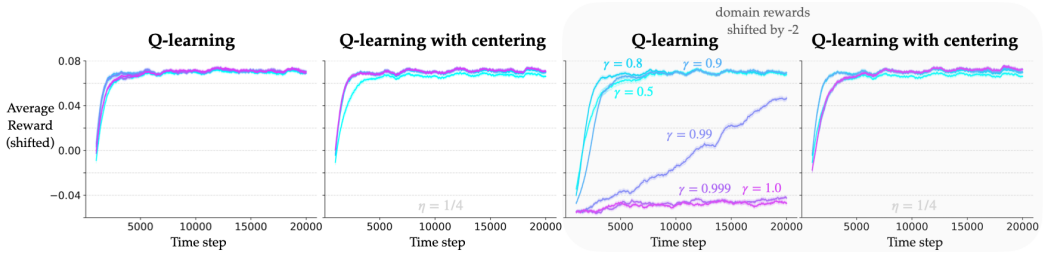
in similar behavior. We verified that the average-reward estimate indeed learns the average reward for every variant quickly. These trends were also consistent across values of the step-size parameters (the parameter studies are in Appendix C).

We observed similar trends on other continuing problems with linear and one with non-linear function approximation. In PuckWorld, the agent has to take a puck-like object to randomly changing goal positions in a square rink. At each time step, the agent observes six real numbers—the puck's position and velocity and the goal position in x and y directions—and gets a reward proportional to the negative distance to the goal. In Pendulum, the agent has to control the torque at the base of a one-link pendulum to take and maintain it in an upright position. At each time step, the agent observes three real numbers—the sine and cosine of the pendulum's angle w.r.t. the direction of gravity, and the pendulum's angular velocity—and gets a reward proportional to the negative angular distance of the pendulum from the upright position. In Catch, the agent moves a crate in the bottom row of a 2D pixel grid to catch falling fruits. For this problem, there are two kinds of observation vectors available to an agent: a 3D real vector containing the x coordinate of the crate and the (x, y) coordinates of the lowermost fruit; a 50D binary vector which is the flattened version of the entire pixel grid. The agent gets a $+1$ reward on successfully catching a fruit, -1 on dropping one, and 0 otherwise. All the problems are continuing; there are no resets.



(a) PuckWorld

(b) Pendulum



(c) Catch

Figure 6: Learning curves with and without centering corresponding to different values of γ on different problems. In the bottom row, the two plots on the right correspond to a variant of the Catch problem where all the rewards shifted by -2.

We used linear function approximation with tile-coded features for PuckWorld and the variant of Catch in which the agent observes the 3D real-valued features. For Pendulum and the variant of Catch with the 50D binary features, we used non-linear function approximation using artificial neural networks (Mnih et al.'s (2015) DQN). The complete experimental details are in Appendix C.

The trends were similar to those observed with Access-Control Queuing. In PuckWorld and Pendulum (top row of Figure 6), without centering, performance first improved as the discount factor γ increased and then degraded. However, with centering, the performance did not degrade for large values of γ. In Catch with linear function approximation (bottom row of Figure 6), the leftmost plot shows that the performance without centering was good even for large discount factors. However, it varied significantly when the problem rewards were shifted up or down by a constant; the third plot from the left demonstrates this for a shift of -2. On the other hand, with centering, the performance was good for all discount factors and unaffected by any shifts in the rewards.

These trends are further supplemented by the two plots on the left of Figure 7, which shows the sensitivity of the algorithms to variants of the Catch problem with rewards shifted by a constant. On the x-axis is the effective step size for the linear function approximators and on the y-axis is the reward rate averaged over the entire training period. As before, the y-axis is adjusted to compare the performance on all the problem variants at the same scale. We observed that the performance without reward centering was problem-dependent, whereas with centering, the rate of learning was roughly the same regardless of the problem variant. The two plots on the right of Figure 7 show that the trends were similar with non-linear function approximation.

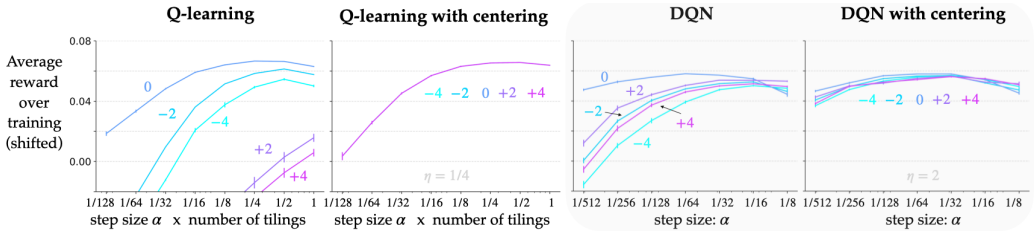


Figure 7: Parameter studies showing the sensitivity of the algorithms to their step-size parameter and to variants of the Catch problem, using both linear and non-linear function approximation.

From the results so far, we observed that reward centering can improve the performance of tabular, linear, and non-linear variants of the Q-learning algorithm on various problems. The improvement in the rate of learning is larger for discount factors close to 1. Furthermore, there is an improvement in the robustness of the algorithms to shifts in the problems' rewards. The parameter studies in this section indicate that the benefits of reward centering are quite robust to the choice of its parameter η. Appendix C contains additional learning curves and parameter studies that further reinforce the trends observed in this section.

5 Discussion, Limitations, and Future Work

The reward-centering idea is very general can be added to any discounted-reward RL algorithm for solving continuing problems. We focused on the reward-centered versions of TD-learning and Q-learning in this paper; we expect similar trends for all other algorithms that estimate values, like Sarsa (Rummery & Niranjan, 1994) or the family of actor-critic methods[1] (Konda & Tsitsiklis, 1999; Schulman et al., 2016). Our preliminary experiments of adding reward centering to the PPO algorithm (Schulman et al., 2017) showed improvements on several Mujoco problems (Todorov et al., 2012) that we made continuing (see Appendix C). A comprehensive empirical investigation of reward centering added to other RL algorithms is an exciting direction of future work. In parallel, it is also pertinent to extend theoretical results from the tabular setting to that of function approximation, beginning with linear function approximation.

[1]The advantage function or the baseline in policy-based methods have orthogonal benefits to reward centering. We discuss these in Appendix D.

Reward centering is related to but different from simply learning a bias parameter. In theory, if a bias weight (corresponding to a bias feature of one) is equal to the state–action-independent offset $r(\pi)/(1-\gamma)$, then the rest of the function-approximation capacity can estimate the relative values. However, on testing this we found the learned bias weight was rarely close to true value of the offset. Moreover, it is desirable to separate $r(\pi)$ from the scaling effect of $1/(1-\gamma)$. Along these lines, Tsitsiklis and Van Roy (2002) suggested setting a linear approximator's bias feature proportional to $1/(1-\gamma)$ and learning the corresponding bias weight. Reward centering takes this idea forward. Since we know what the corresponding bias weight is, we can have an explicit update rule for it (à la Sutton (1988b)), which can lead to faster learning. Besides, in the control problem, there is no need to estimate the offset within the standard discounted values, as Tsitsiklis and Van Roy propose doing. Instead, via reward centering, we can simply estimate the centered (relative) values.

Note that reward centering adds an additional element of non-stationarity to the learning problem because the average-reward estimate changes over time. As a result, the asymptotic variance of the updates is larger compared to when there is no centering (see, e.g., Figure 3). The increased variance of updates also ties in to how the average reward is estimated. Ideally, the average reward is accurately estimated as quickly as possible and does not change too much per step. The simple centering technique quickly estimates the average reward but is affected by the per-step stochasticity. Value-based centering additionally relies on changing values and hence can be relatively slower, but because it relies on the value estimates, per-step stochasticity in the problem dynamics may not affect it as much.[2] A step-size adaptation technique would be pertinent for estimating the average reward: larger step sizes when the errors are consistently large, smaller step sizes otherwise. Such an adaptation should be possible throughout the lifetime of the agent, since it may encounter new parts of the world having different kinds of reward signals.

Speaking of different kinds of reward signals, reward centering should be combined with approaches that deal with different *scales* of rewards. Reward centering makes learning algorithms robust to *shifts* in the rewards. However, the relative values themselves may have large magnitude (e.g., if all the rewards in a problem are multiplied by a constant, then the relative values will be scaled by the same constant). It is thus pertinent to combine reward centering with reward scaling. The general idea of scaling quantities to a tractable range is quite common (e.g., van Hasselt et al., 2016; Pohlen et al., 2018). Building on this body of work, Schaul et al. (2021) recently proposed a lightweight trick that re-scales TD errors into an optimization-friendly range using problem-agnostic information in the agent's data stream. With some care, their techniques can likely be extended from the episodic to the continuing case.

One of the most intriguing follow-up directions based on reward-centering is a method that adapts the discount factor over time. Learning both the average reward and the centered discounted estimates allows the agent to rapidly estimate the discounted value function corresponding to *any* discount factor. Concretely, consider the agent has estimated the average reward R and the centered discounted value function $\tilde{\mathbf{v}}^{\gamma_1}$ to some level of accuracy. With just this information, the agent can form an estimate of the standard discounted value function corresponding to another discount factor γ_2 via $\bar{R}/(1-\gamma_2) + \tilde{\mathbf{v}}^{\gamma_1}$. This is an estimate, of course, but it can be improved quickly with a few samples of experience—potentially with old experience from a buffer or a parameterized model. In contrast, with standard methods, it would take comparatively longer to raise the estimates to the new mean value and adapt the relative values. Hence, with reward centering, we can imagine efficient methods that adapt their discount factors over time: a low discount rate to learn quickly amidst a lot of uncertainty—like in the beginning of training—and when the world is more predictable, a higher discount rate to estimate the policy that maximizes the total reward obtained by the agent.

Reward centering is a simple idea with potentially large benefits. When combined with appropriate step-size adaptation and reward-scaling techniques, we think it will be a key enabler for agents to learn quickly and continually over their lifetimes.

[2]Consider a fully deterministic tabular problem. The rewards are different at each step in general and so the simple centering technique would always involve non-zero errors. On the other hand, the per-step TD error would be zero in the value-based approach.

Acknowledgments

We would like to thank Huizhen Yu, Arsalan Sharifnassab, Khurram Javed, and other members of the RLAI lab for discussions that helped improve the quality and clarity of the paper. We are also grateful for the computing resources provided by the Digital Research Alliance of Canada.

References

Blackwell, D. (1962). Discrete Dynamic Programming. *The Annals of Mathematical Statistics.*

Borkar, V., & Meyn, S. (2000). The ODE Method for Convergence of Stochastic Approximation and Reinforcement Learning. *SIAM Journal on Control and Optimization.*

Devraj, A., & Meyn, S. (2021). Q-learning with Uniformly Bounded Variance. *IEEE Transactions on Automatic Control.* Also *ArXiv:2002.10301.*

Engstrom, L., Ilyas, A., Santurkar, S., Tsipras, D., Janoos, F., Rudolph, L., & Madry, A. (2019). Implementation Matters in Deep RL: A Case Study on PPO and TRPO. *International Conference on Learning Representations.*

Even-Dar, E., Mansour, Y., & Bartlett, P. (2003). Learning Rates for Q-learning. *Journal of Machine Learning Research.*

Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. *ArXiv:1412.6980.*

Konda, V. R., & Tsitsiklis, J. N. (1999). Actor-Critic Algorithms. *Advances in Neural Information Processing Systems.*

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-Level Control Through Deep Reinforcement Learning. *Nature.*

Naik, A., Shariff, R., Yasui, N., Yao, H., & Sutton, R. S. (2019). Discounted Reinforcement Learning Is Not an Optimization Problem. *Optimization Foundations for Reinforcement Learning Workshop at the Conference on Neural Information Processing Systems.* Also *ArXiv:1910.02140.*

Ng, A., Harada, D., & Russell, S. (1999). Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping. *International Conference on Machine Learning.*

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E. Z., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., & Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems.*

Pohlen, T., Piot, B., Hester, T., Azar, M. G., Horgan, D., Budden, D., Barth-Maron, G., van Hasselt, H., Quan, J., Večerík, M., Hessel, M., Munos, R., & Pietquin, O. (2018). Observe and Look Further: Achieving Consistent Performance on Atari. *ArXiv:1805.11593.*

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* John Wiley & Sons.

Qu, G., & Wierman, A. (2020). Finite-Time Analysis of Asynchronous Stochastic Approximation and Q-learning. *Conference on Learning Theory.*

Robbins, H., & Monro, S. (1951). A Stochastic Approximation Method. *The Annals of Mathematical Statistics.*

Rummery, G. A., & Niranjan, M. (1994). *On-line Q-learning Using Connectionist Systems.* Technical Report, Engineering Department, Cambridge University.

Schaul, T., Ostrovski, G., Kemaev, I., & Borsa, D. (2021). Return-Based Scaling: Yet Another Normalisation Trick for Deep RL. *ArXiv:2105.05347*.

Schneckenreither, M. (2020). Average Reward Adjusted Discounted Reinforcement Learning: Near-Blackwell-Optimal Policies for Real-World Applications. *ArXiv:2004.00857*.

Schulman, J., Moritz, P., Levine, S., Jordan, M., & Abbeel, P. (2016). High-Dimensional Continuous Control Using Generalized Advantage Estimation. *International Conference on Learning Representations*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. *ArXiv:1707.06347*.

Singh, S., Jaakkola, T., & Jordan, M. (1994) Learning Without State-Estimation in Partially Observable Markovian Decision Processes. *Machine Learning Proceedings*.

Sutton, R. S. (1988a). Learning to Predict by the Methods of Temporal Differences. *Machine Learning*.

Sutton, R. S. (1988b). *NADALINE: A Normalized Adaptive Linear Element That Learns Efficiently*. GTE Laboratories Technical Report.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT Press.

Todorov, E., Erez, T., & Tassa, Y. (2012). MuJoCo: A Physics Engine for Model-Based Control. *International Conference on Robotics and Automation*.

Tsitsiklis, J. N., & Van Roy, B. (2002). On Average Versus Discounted Reward Temporal-Difference Learning. *Machine Learning*.

Van Hasselt, H., Guez, A., Hessel, M., Mnih, V., & Silver, D. (2016). Learning Values Across Many Orders of Magnitude. *Advances in Neural Information Processing Systems*.

Wainwright, M. J. (2019). Stochastic Approximation With Cone-Contractive Operators: Sharp ℓ_∞-Bounds for Q-learning. *ArXiv:1905.06265*.

Wan, Y., Naik, A., & Sutton, R. S. (2021). Learning and Planning in Average-Reward Markov Decision Processes. *International Conference on Machine Learning*.

Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine Learning*.

Zhao, R., Abbas, Z., Szepesvári, D., Naik, A., Holland, Z., Tanner, B., & White, A. (2022). CSuite: Continuing Environments for Reinforcement Learning, *Github:* google-deepmind/csuite

A Pseudocode

In this section we present the pseudocode for value-based reward-centering added to the tabular, linear, and non-linear variants of Q-learning.

Algorithm 1: Tabular Q-learning with value-based reward centering

Input: The behavior policy b (e.g., ϵ-greedy)
Algorithm parameters: discount factor γ, step-size parameters α, η
Initialize $Q(s,a)\ \forall s, a; \bar{R}$ arbitrarily (e.g., to zero)
Obtain initial S
for *all time steps* **do**
  Take action A according to b, observe R, S'
  $\delta = R - \bar{R} + \gamma \max_a Q(S', a) - Q(S, A)$
  $Q(S, A) = Q(S, A) + \alpha\, \delta$
  $\bar{R} = \bar{R} + \eta\, \alpha\, \delta$
  $S = S'$
end

Algorithm 2: Linear Q-learning with value-based reward centering

Input: The behavior policy b (e.g., ϵ-greedy)
Algorithm parameters: discount factor γ, step-size parameters α, η
Initialize $\mathbf{w}_a \in \mathbb{R}^d\ \forall a, \bar{R}$ arbitrarily (e.g., to zero)
Obtain initial observation $\mathbf{x}$
for *all time steps* **do**
  Take action A according to b, observe $R, \mathbf{x}'$
  $\delta = R - \bar{R} + \gamma \max_a \mathbf{w}_a^\top \mathbf{x}' - \mathbf{w}_A \mathbf{x}$
  $\mathbf{w}_A = \mathbf{w}_A + \alpha\, \delta\, \mathbf{x}$
  $\bar{R} = \bar{R} + \eta\, \alpha\, \delta$
  $\mathbf{x} = \mathbf{x}'$
end

Algorithm 3: (Non-linear) DQN with value-based reward centering

Input: The behavior policy b (e.g., ϵ-greedy)
Algorithm parameters: discount factor γ, step-size parameters α, η
Initialize value network, target network; initialize $\bar{R}$ arbitrarily (e.g., to zero)
Obtain initial observation $\mathbf{x}$
for *all time steps* **do**
  Take action A according to b, observe $R, \mathbf{x}'$
  Store tuple $(\mathbf{x}, A, R, \mathbf{x}')$ in the experience buffer
  if *time to update estimates* **then**
    Sample a minibatch of transitions $(\mathbf{x}, A, R, \mathbf{x}')^b$
    For every i-th transition: $\delta_i = R_i - \bar{R} + \gamma \max_a \hat{q}(\mathbf{x}'_i, a) - \hat{q}(\mathbf{x}_i, A_i)$
    Perform a semi-gradient update of the value-network parameters with the δ^2 loss
    $\bar{R} = \bar{R} + \eta\, \alpha\, \text{mean}(\delta)$
    Update the target network occasionally
  end
  $\mathbf{x} = \mathbf{x}'$
end

We recommend two small but useful optimizations to these general pseudocodes in Appendix C.

B Theoretical Details

This section presents (a) the complete convergence result for Q-learning with value-based centering using Devraj and Meyn's (2021) analysis, and (b) quantifies the reduction in constant state–action-independent offset in the value estimates.

Suppose the agent's interaction with the MDP follows a stationary behavior policy $b \in \Pi$. Let S_t, A_t denote the state-action pair occurring at time step t, followed by the reward R_{t+1} and next state S_{t+1}. Let $\nu_t(s,a)$ denote the number of times a state-action pair (s,a) has occurred up to and including time step t. The update rules of Q-learning with value-based centering are:

$$Q_{t+1}(S_t, A_t) \doteq Q_t(S_t, A_t) + \alpha_{\nu_t(S_t,A_t)}\, \delta_t, \tag{12}$$

$$\bar{R}_{t+1} \doteq \bar{R}_t + \eta\, \alpha_{\nu_t(S_t,A_t)}\, \delta_t, \tag{13}$$

$$\text{where,} \qquad \delta_t \doteq R_{t+1} - \bar{R}_t + \gamma \max_{a'} Q_t(S_{t+1}, a') - Q_t(S_t, A_t), \tag{14}$$

$\eta > 0$, and $\alpha_n = c/(n+d)$ where $c, d > 0$ for all $n \geq 1$.[3]

Theorem 1. (Formal) *If the joint process $\{S_t, A_t\}$ induced by the stationary behavior policy is an irreducible Markov chain, that is, starting from every state-action pair, there is a non-zero probability of transitioning to any other state-action pair in a finite number of steps, then $(Q_t, \bar{R}_t)$ in tabular Q-learning with value-based centering (12–14) converges to a solution of $(\bar{\mathbf{q}}^\gamma, \bar{r})$ in (11).*

Proof. We first show that Q-learning with value-based centering is a member of the large family of Devraj and Meyn's (2021) Relative Q-learning algorithms with particular choices of μ and κ. This allows us to utilize their convergence results.

The general Relative Q-learning algorithm updates its tabular estimates $\tilde{Q}^\gamma : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ at time step t using $(S_t, A_t, R_{t+1}, S_{t+1})$ as (in our notation):

$$\tilde{Q}^\gamma_{t+1}(S_t, A_t) \doteq \tilde{Q}^\gamma_t(S_t, A_t) + \alpha_t\big[R_{t+1} - f(\tilde{Q}^\gamma_t) + \gamma \max_{a'} \tilde{Q}^\gamma_t(S_{t+1}, a') - \tilde{Q}^\gamma_t(S_t, A_t)\big], \tag{15}$$

where $f(\tilde{Q}^\gamma_t) \doteq \kappa \sum_{s,a} \mu(s,a) \tilde{Q}^\gamma_t(s,a)$, $\kappa > 0$ is a scalar, and $\mu : \mathcal{S} \times \mathcal{A} \to [0,1]$ is a probability mass function.

Now note that updating both the average-reward and value estimates using the TD error (12 and 13) results in:

$$\bar{R}_t - \bar{R}_0 = \eta\Big(\sum_{s,a} Q_t(s,a) - \sum_{s,a} Q_0(s,a)\Big).$$

To simplify the analysis, we can assume $\bar{R}_0 = 0$ and $\mathbf{Q}_0 = \mathbf{0}$ without loss of generality. As a result, $\bar{R}_t = \eta \sum_{s,a} \tilde{Q}^\gamma_t(s,a)$. We can then combine the updates (9–10) in the tabular case to:

$$\tilde{Q}^\gamma_{t+1}(S_t, A_t) \doteq \tilde{Q}^\gamma_t(S_t, A_t) + \alpha_t\big(R_{t+1} - \eta \sum_{s,a} \tilde{Q}^\gamma_t(s,a) + \max_{a'} \tilde{Q}^\gamma_t(S_{t+1}, a') - \tilde{Q}^\gamma_t(S_t, A_t)\big). \tag{16}$$

Comparing (15) and (16), we can see that Q-learning with value-based reward centering is an instance of Relative Q-learning with:

$$\mu(s,a) = \frac{1}{|\mathcal{S}||\mathcal{A}|}\ \forall s, a, \quad \text{and} \quad \kappa = \eta|\mathcal{S}||\mathcal{A}|.$$

[3] Devraj and Meyn (2021) considered the step-size sequence $1/n$ in their algorithm but it can be easily verified that $\alpha_n = c/(n+d)$ also satisfies the step-size condition required by Borkar and Meyn's (2000) seminal result (that was used by Devraj & Meyn (2021) to show the convergence of their algorithm).

Devraj and Meyn's (2021) convergence result then applies. That is,

$$\begin{aligned}\tilde{\mathbf{Q}}_t^\gamma \to \tilde{\mathbf{Q}}_\infty^\gamma &\doteq \mathbf{q}_*^\gamma - \frac{\kappa}{1-\gamma+\kappa}\mu^\top \mathbf{q}_*^\gamma \mathbf{1} \\ &= \mathbf{q}_*^\gamma - \frac{\eta}{1-\gamma+\eta|\mathcal{S}||\mathcal{A}|}\sum_{s,a} q_*^\gamma(s,a)\mathbf{1}. \end{aligned} \tag{17}$$

Hence,

$$\begin{aligned}\bar{R}_t \to \bar{R}_\infty &\doteq \eta \sum_{s,a} q_*^\gamma(s,a) - \frac{\eta^2|\mathcal{S}||\mathcal{A}|}{1-\gamma+\eta|\mathcal{S}||\mathcal{A}|}\sum_{s,a} q_*^\gamma(s,a) \\ &= \frac{\eta(1-\gamma)}{1-\gamma+\eta|\mathcal{S}||\mathcal{A}|}\sum_{s,a} q_*^\gamma(s,a). \end{aligned} \tag{18}$$

We will now verify that $(\tilde{\mathbf{Q}}_\infty^\gamma, \bar{R}_\infty)$ satisfy the Bellman equations (11). Recall that the solutions of the Bellman equation are of the form $\big(\tilde{\mathbf{q}}_*^\gamma + \frac{k}{1-\gamma}\mathbf{1}, r(\pi_\gamma^*) - k\big)$. Since $\tilde{\mathbf{q}}_*^\gamma = \mathbf{q}_*^\gamma - \frac{r(\pi_\gamma^*)}{1-\gamma}$, we can re-write the solution class in terms of the discounted value function: $\big(\mathbf{q}_*^\gamma + \frac{(k-r(\pi_\gamma^*))}{1-\gamma}\mathbf{1}, r(\pi_\gamma^*) - k\big)$, or $\big(\mathbf{q}_*^\gamma - \frac{d}{1-\gamma}\mathbf{1}, d\big)$. For $d = \frac{\eta(1-\gamma)}{1-\gamma+\eta|\mathcal{S}||\mathcal{A}|}\sum_{s,a} q_*^\gamma(s,a)$, we can see that $(\tilde{\mathbf{Q}}_\infty^\gamma, \bar{R}_\infty)$ is a solution tuple of the Bellman equations. □

We can now characterize how close $\bar{R}_\infty$ is to $r(\pi_\gamma^*)$. In general the expression for $\bar{R}_\infty$ (18) is cryptic. However, a special case can shed some light. We know that the average of the discounted value function for a policy w.r.t. that policy's steady-state distribution is: $\sum_{s,a} d_\pi(s,a) q_\pi^\gamma(s,a) = \frac{r(\pi)}{1-\gamma}$. Now suppose the steady-state distribution over state–action pairs is constant—$1/(|\mathcal{S}||\mathcal{A}|), \forall s, a$. For that policy, $\frac{1}{|\mathcal{S}||\mathcal{A}|}\sum_{s,a} q_\pi^\gamma(s,a) = \frac{r(\pi)}{1-\gamma}$. Substituting this in (18), we get:

$$\bar{R}_\infty = \frac{\eta|\mathcal{S}||\mathcal{A}|}{1-\gamma+\eta|\mathcal{S}||\mathcal{A}|} r(\pi_\gamma^*). \tag{19}$$

We can see that $\bar{R}_\infty$ approaches the true reward rate from below when $\eta|\mathcal{S}||\mathcal{A}| >> 1-\gamma$, which can be true in many problems of interest that have large state (and action) spaces. That being said, note that this insight comes from a special case. More generally, the convergence point of $\bar{R}_\infty$ (and hence $\tilde{\mathbf{Q}}_\infty^\gamma$) is hard to interpret, which is a shortcoming we wish to resolve in future work. However, (19) can serve as a rule of thumb.

We end this section with a property of the centered discounted values.

Lemma 1. *The centered discounted values $\tilde{\mathbf{v}}_\pi^\gamma$ are on average zero when weighted by the on-policy distribution $\mathbf{d}_\pi$ induced by the policy π:*

$$\mathbf{d}_\pi^\top \tilde{\mathbf{v}}_\pi^\gamma = 0. \tag{20}$$

Proof. The proof is trivial after using the property that $\mathbf{d}_\pi^\top \mathbf{v}_\pi^\gamma = r(\pi)/(1-\gamma)$ (see Sutton & Barto's (2018) Section 10.4 or Singh et al.'s (1994) Section 5.3). Since $\tilde{\mathbf{v}}_\pi^\gamma = \mathbf{v}_\pi^\gamma - r(\pi)/(1-\gamma)\mathbf{1}$ (from (3)), $\mathbf{d}_\pi^\top \tilde{\mathbf{v}}_\pi^\gamma = 0$. □

C Experimental Details

Prediction 'TD-learning with rewards centered by an oracle' refers to a version of TD-learning with centering in which the average-reward estimate is fixed to the (somehow) known average reward of the target policy. In other words, the true average reward is known from the beginning and is subtracted from the observed rewards at each time step. This algorithm is a good baseline because its rate of learning is likely the theoretical best among all TD-based prediction algorithms (in stationary problems where the average reward of the fixed target policy does not change with time).

Each algorithm was run on the random-walk problem for 50,000 steps and repeated 50 times each. The step size α was decayed by 0.99999 at each step. The values estimates for all variants and the average-reward estimate for TD-with-centering were initialized to zero.

We tested $\alpha \in \{0.01, 0.02, 0.04, 0.08, 0.16, 0.32\}$ and picked the one which resulted in the lowest average RMSVE across the training period for standard uncentered approach ($\alpha = 0.04$ for $\gamma = 0.9$ and $\alpha = 0.08$ for $\gamma = 0.99$). Corresponding to these step sizes, we tested the centering approaches' parameter η within a coarse range of $\{1/640, 1/160, 1/40, 1/10\}$ and picked one based on the aforementioned criteria. As mentioned earlier, this does not result in the best choice of α, η for the centering approaches, which is okay; we made sure the baselines are tuned appropriately.

Control Table 2 contains a list of all the hyperparameters tested that are common across all the domains: γ, α, η. Note that setting $\eta = 0$ and initializing the average-reward estimate to zero, Q-learning with reward centering behaves exactly like standard Q-learning. For each set of parameters, the algorithms were run for N steps and repeated R times. The (N, R) tuples for each problem were: Access-Control Queuing: $(80k, 50)$; PuckWorld: $(300k, 20)$, Pendulum: $(100k, 15)$; Catch (linear): $(20k, 50)$; Catch (non-linear): $(80k, 15)$. For generating variants of the problems, we shifted the rewards by a range of numbers roughly proportional to the scale of rewards in the original problem: Access-Control Queuing and PuckWorld: {-8, -4, 0, 4, 8}; Pendulum: {-12, -6, 0, 6, 12}; Catch: {-4, -2, 0, 2, 4}.

The agent's behavior policy was always ϵ-greedy with fixed $\epsilon = 0.1$. For all the experiments, the average-reward estimate was initialized to zero. The value-estimation weights were initialized to zero in the tabular and linear experiments; the weights were initialized to small values around zero in the non-linear experiments (the default initialization in PyTorch (Paszke et al., 2019)). For the linear experiments we used 16 tiles of size $4 \times 4 \times 4$ for Catch and 32 tiles of size $4 \times 4 \times 4 \times 4 \times 4 \times 4$ for PuckWorld. These numbers and sizes were not specifically optimized for any problem or algorithm.

We set commonly used values for the various parameters of the deep RL (non-linear) experiments: the batch size was 64, the value-network and reward-rate parameters were updated every 32 steps,

Table 2: List of hyperparameters tested for each domain

	γ	α	η
Access-Control Queuing (tabular)	[0.5, 0.8, 0.9, 0.99, 0.999, 1]	[1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2, 1]	[0, 1/256, 1/64, 1/16, 1/4, 1]
PuckWorld (linear)	[0.5, 0.8, 0.9, 0.99, 0.999, 1]	[0.01, 0.1, 0.3, 0.5, 0.7, 0.9, 1.0, 1.1]	[0, 1/256, 1/64, 1/16, 1/4, 1]
Catch (linear)	[0.5, 0.8, 0.9, 0.99, 0.999, 1]	[1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2, 1]	[0, 1/256, 1/64, 1/16, 1/4, 1]
Catch (non-linear)	[0.5, 0.8, 0.9, 0.99, 0.999, 1]	[1/512, 1/256, 1/128, 1/64, 1/32, 1/16, 1/8]	[0, 1, 2, 4, 8, 16]
Pendulum (non-linear)	[0.5, 0.8, 0.9, 0.99, 0.999, 1]	[1/512, 1/256, 1/128, 1/64, 1/32, 1/16, 1/8]	[0, 1, 2, 4, 8, 16]

the target network was updated every 128 steps, the experience buffer size was 10,000. Apart for the main step-size parameter, the default parameters (set by PyTorch) were used for the Adam optimizer (Kingma & Ba, 2014).

Centering in the non-linear setting (that is, with DQN) in its current form requires a large value of η compared to the the tabular or linear versions. The reason is how a minibatch is used in the implementation of this deep RL algorithm. In line 10 of Algorithm 3, the mean of the TD errors of the minibatch of transitions is taken. The mean can make the overall gradient for the reward-rate update very small, so a large value of η can be used.

In our implementations we added two simple optimizations:

1. Make the average-reward estimate completely independent of its initialization: this can be done using the unbiased constant step-size trick (see Sutton & Barto's (2018) Exercise 2.7).
2. Propagate the changes to the average-reward estimate faster: this can be done by first computing the TD error, then updating the reward-rate estimate, then recomputing the TD error with the new reward-rate estimate, and finally updating the value estimate(s).

These optimizations did not affect the overall trends in the results but provided a small yet noticeable improvement for a tiny computational cost, hence we recommend using them.

For the experiments involving a shift in the problem rewards, the rewards obtained on each problem variant are not directly comparable. For intuition, imagine the first four rewards in the original problem be 2,0,3,1. In a variant of the problem with 5 added to all the rewards, the first four rewards may now appear to be 7,5,8,4. An agent solving the latter problem might trivially appear better than one solving the former problem even though its fourth reward was relatively lower. To compare them meaningfully, from the rewards obtained by an agent, we can subtract the constant that was added in the first place to all the problem's rewards. That is, we can shift the rewards back to make fair comparisons across problem variants. This is what we did when presenting the results of the shifting experiments; this is explicitly denoted by the word "shifted" in the y-axis label.

Figures 8–14 supplement the main trends shown in the main text: the effectiveness of centering increases as the discount factor approaches 1; with reward centering, the algorithms are more robust to any constant shifts in the rewards; the performance of reward centering is quite robust to the choice of the parameter η.

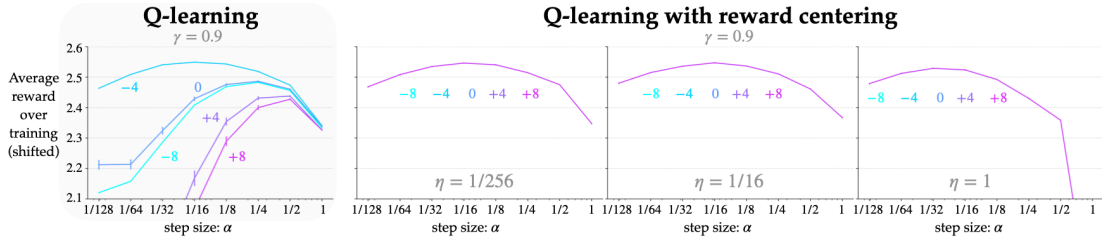


Figure 8: Parameter studies showing the sensitivity of the two algorithms' performance on variants of the Access-Control Queuing domain. The error bars indicate one standard error, which at times is less than the width of the lines. *Far left:* Without centering, the performance of Q-learning differed significantly on the different variants over a broad range of the step-size parameter α. *Center to right:* With centering, the performance was about the same across the problem variants, and was quite robust to the choice of its parameter η. All the curves correspond to $\gamma = 0.9$; the trends were consistent across other discount factors.

We also report preliminary results of PPO (Schulman et al., 2017) with and without centering. We chose to test these on the classic Mujoco problems (Todorov et al., 2012). Mujoco domains are typically implemented as episodic problems; we converted them to continuing problems by (a) setting the episode-truncation parameter to a very large number, and (b) if applicable, resetting the

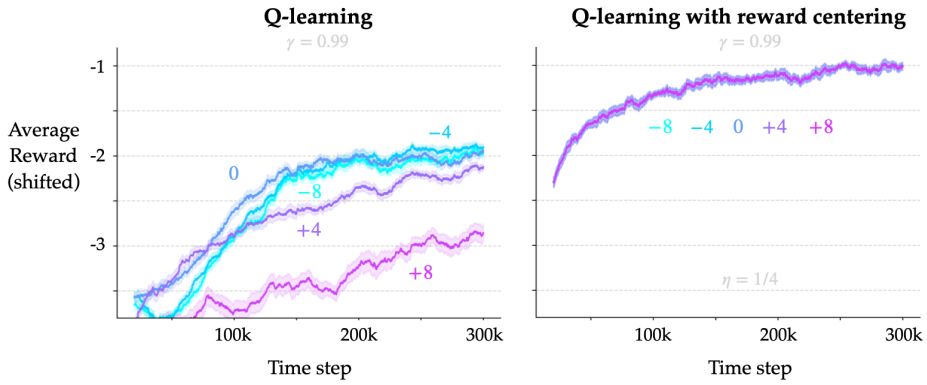


Figure 9: Learning curves for Q-learning with and without centering on variants of the PuckWorld problem when $\gamma = 0.99$. The performance without centering was different on each variant while that with centering was roughly the same. Reward centering also resulted in much faster learning. These trends were consistent across values of γ.

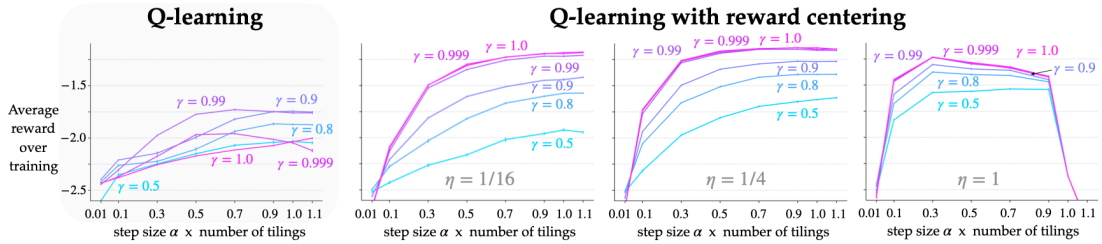


Figure 10: Parameter studies showing the sensitivity of the algorithms' performance to their parameters on the PuckWorld domain. *Far left:* Without centering, Q-learning's performance was relatively poor for a large range of α. *Center to right:* For each discount factor, the performance of Q-learning with centering was better across a broad range of α.

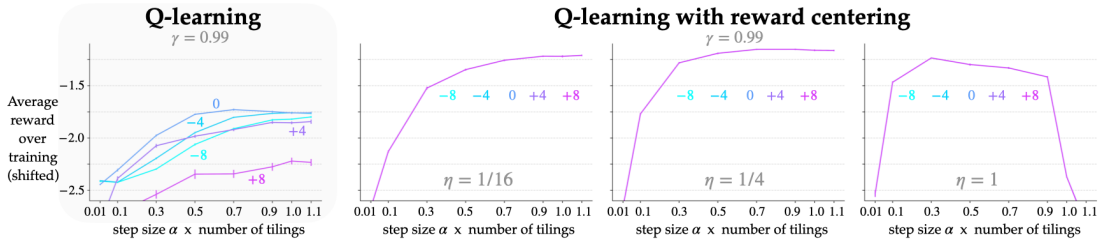


Figure 11: Parameter studies showing the sensitivity of the algorithms' performance to variants of the PuckWorld domain. The error bars indicate one standard error, which at times is less than the width of the lines. *Far left:* Without centering, the performance of Q-learning differed significantly on the different variants over a broad range of the step-size parameter α. *Center to right:* With centering, the performance was about the same across the problem variants, and was quite robust to the choice of its parameter η. All the curves correspond to $\gamma = 0.99$; the trends were consistent across other discount factors.

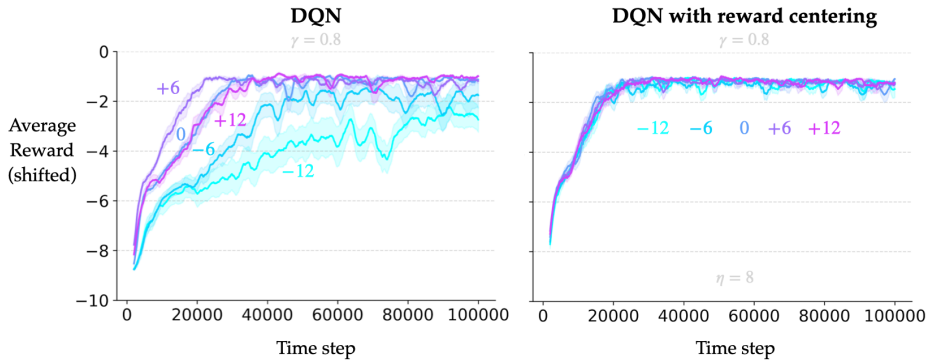


Figure 12: Learning curves for Q-learning with and without centering on variants of the Pendulum problem when $\gamma = 0.8$. The performance without centering was different on each variant while that with centering was roughly the same. Reward centering also resulted in much faster learning. These trends were consistent across values of γ.

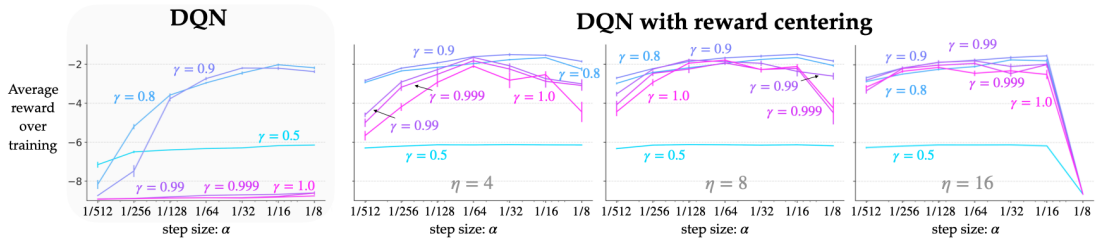


Figure 13: Parameter studies showing the sensitivity of the algorithms' performance to their parameters on the Pendulum domain. $\gamma = 0.5$ was too small to solve this problem. *Far left:* The performance of DQN suffered for discount factors larger than 0.9. *Center to right:* For each discount factor, the performance of DQN with centering was better across a broad range of α. Additionally, the performance was not too sensitive to the parameter η.

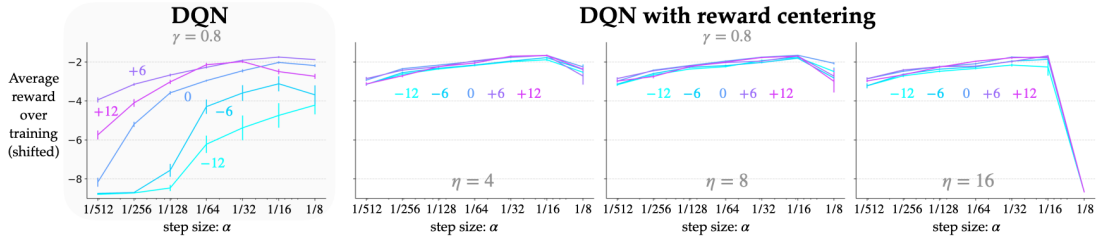


Figure 14: Parameter studies showing the sensitivity of the algorithms' performance with $\gamma = 0.8$ to variants of the Pendulum problem. *Far left:* Without centering, the performance of DQN differed significantly on the different variants. *Center to right:* With centering, the performance of DQN was about the same across the problem variants across a large range of the step size α, and was also quite robust to the choice of η.

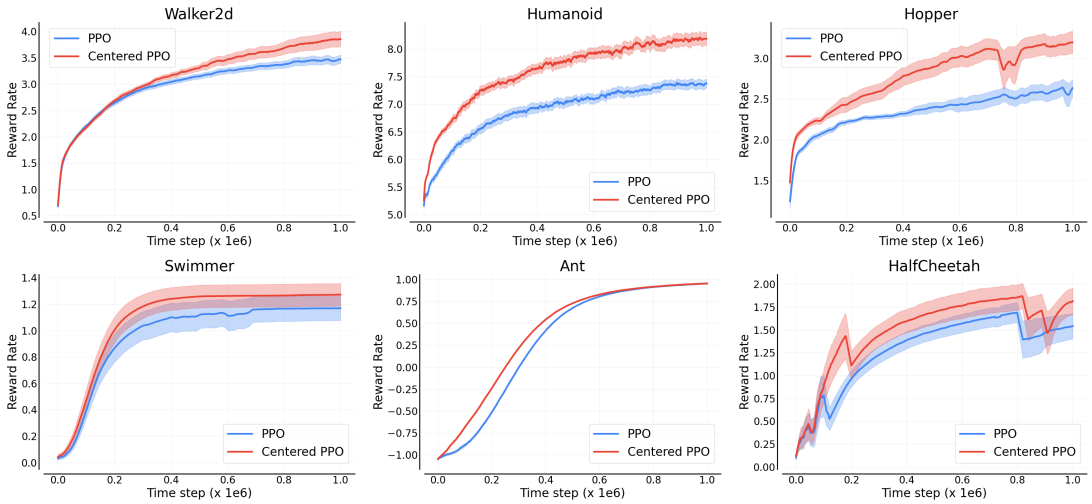


Figure 15: Learning curves for PPO with and without centering on continuing versions of six Mujoco domains. The solid lines and the shaded region denote the mean and one standard error over 10 independent runs.

domain to a starting state with a large negative reward if the agent enters an unrecoverable state. We used value-based centering (10), where δ_t corresponds to the advantage estimates computed by standard PPO.

Figure 15 shows the learning curves for PPO with and without centering. The y-axis shows the average reward obtained the agent over the last 1000 time steps. As with all the other experiments in this paper, the evaluation is online—there are no separate training or testing periods. A careful study will take more time due to the large number of hyperparameters; in our preliminary experiments with 10 runs each, we found that centering results in a slight improvement on all the problems, with the most pronounced improvements on the Humanoid problem. The step sizes corresponding to average-reward estimate for the different domains are: Hopper: 1E-4, HalfCheetah: 1E-3, Walker2D: 2E-5, Swimmer: 5E-5, Humanoid: 1E-2, Ant: 1E-4.

D Connections to Related Approaches

Concurrently with Devraj and Meyn (2021), Schneckenreither (2020) realized the Laurent series decomposition suggests that an explicit estimate of the average reward can completely remove the offset. So they proposed an algorithm which to estimate and subtract the average reward, with two important differences: (a) the average-reward estimate is updated only after non-exploratory actions, and (b) the algorithm has two discount factors to aim for the strongest optimality criterion—Blackwell optimality. Schneckenreither did not provide any convergence result for their algorithm. However, they analyzed that *if* the algorithm converged to the desired fixed point, then the resulting policy would be (Blackwell-)optimal. Wan et al. (2021) pointed out the average-reward estimate can be updated at every time step, including ones with exploratory actions, and showed almost-sure convergence of their algorithms. Combining those insights with Devraj and Meyn's, we show the convergence of Q-learning with value-based reward centering.

Reward centering and the advantage function have orthogonal benefits. The advantage function benefits the actor by reducing the variance of the updates in the policy space (Sutton & Barto, 2018; Schulman et al., 2016). On the other hand, reward centering benefits the critic's or baseline's estimation by eliminating the need to estimate the large state-independent constant offset. Both the quantities involved in the advantage function—$a^\gamma_\pi(s,a) = q^\gamma_\pi(s,a) - v^\gamma_\pi(s)\ \forall s, a$—have the large state-independent offset $r(\pi)/(1-\gamma)$. The net effect of the offset is zero when they are subtracted. But the key point is that both the state- and action-value estimates include the large offset. Reward

centering removes the need to estimate the large offset for both the state- and action-value function, which simplifies the critic-estimation problem. The actor update is left unchanged with reward centering because the advantage function itself remains unchanged: $\tilde{a}^\gamma_\pi(s,a) = \tilde{q}^\gamma_\pi(s,a) - \tilde{v}^\gamma_\pi(s)$, because $\tilde{q}^\gamma_\pi(s,a) = q^\gamma_\pi(s,a) - r(\pi)/(1-\gamma)$ and $\tilde{v}^\gamma_\pi(s) = v^\gamma_\pi(s) - r(\pi)/(1-\gamma)$. Hence, we expect reward centering to benefit all the algorithms that estimate values, which include all actor-critic methods that involve advantage estimation.

Dividing all the rewards with a (potentially changing) scalar number is typically referred to as reward scaling (see, e.g., Engstrom et al., 2020). Just like reward centering, reward scaling does not change the ordering of policies in a continuing problem. Scaling reduces the spread of the rewards, centering brings them close to zero, both of which can be favorable to complex function approximators such as artificial neural networks that are used for value estimation starting from a close-to-zero initialization. The popular stable_baselines3 repository scales (and clips[4]) the rewards by a running estimate of the variance of the discounted returns (github.com/DLR-RM/stable-baselines3/blob/master/stable_baselines3/common/vec_env/vec_normalize.py#L256). Mean-centering the rewards as well would be beneficial for continuing domains. Note that the mechanism of computing the mean and variance is more complicated in the off-policy setting than the on-policy setting. Our TD-error-based technique is likely part of the final solution for the off-policy setting. Simply maintaining a running estimate of the variance (as in the stable_baselines' approach) introduces a bias. As mentioned earlier, Schaul et al.'s (2021) technique is a good starting point.

Reward centering can be seen as reward shaping (Ng et al., 1999) with a constant state-independent potential function: $\Phi(s) = r(\pi)/(1-\gamma)\ \forall s$. Their Theorem 1 then reiterates that reward centering does not change the optimal policy of the problem. A possible drawback of reward shaping is that fully specifying the potential-based shaping function can be tricky, especially for problems with large state spaces. In the case of reward centering this is relatively easy: the potential function is constant across the entire state space, and we know how to learn the average reward reliably from data.

[4]Reward clipping in general changes the problem. Blinding the agent from large rewards can impose a performance ceiling or make some games impossible to solve (Schaul et al.'s (2021) Section 4.3 discusses this in the context of Atari problems).