[Re] A Family of Robust Stochastic Operators for Reinforcement Learning

Daniel Glickman
Brown University

Joseph Kijewski
Brown University

Haoze Zhang
Brown University

Lu Shao
Brown University

Ishaan Shah
Brown University

Nishant Kumar
Brown University

Abstract

We replicate a new family of robust stochastic operators (RSO) proposed in [1]. In reinforcement learning, the presence of approximation/estimation errors can increase the probability of sub-optimal actions being taken. One way to reduce the effects of these errors on performance is to increase the action gap, or the value difference between the best and next-best actions. RSO learning increases the action gap and as a result should be robust to approximation/estimation errors. The original paper tests the operator on problems with substantial approximation errors to demonstrate its effectiveness. We recreate the graph showing the performance of RSO relative to other operators on Mountain Car, and demonstrate that RSO performs better than the other operators regardless of ϵ. We additionally show that, even when optimal hyperparameters are selected for other operators, RSO still performs substantially better. Finally, we attempt to replicate the superiority of the authors' preferred RSO parameterization, but find that an alternative parameterization performs better.

1 Introduction

When a continuous-state, continuous-time Markov Decision Process (MDP) is modelled using a discrete-space, discrete-time system, there are substantial approximation errors. Such errors may lead to the selection of sub-optimal actions especially when the difference in value between actions is small, which results in slowed learning. Previous results from Farahmand et al. 2011 [2] have shown that methods to widen the action gap are useful in resisting approximation and estimation errors. Here, the action-gap refers to the difference between the Q-value of the optimal action and the next-best actions. The consistent Bellman operator introduced in Bellemare et al. 2016 [3] is a deterministic operator that maintains optimality while being gap-increasing unlike the classical Bellman operator. However, the paper also finds that deterministically increasing the action-gap above a certain region will lead to a violation of optimality. One of the open questions arising from Bellemare et al. 2016 [3] is the challenge of increasing the action-gap while preserving optimality. Lu et al. 2019 [1] aims to investigate this question and proposes a stochastic alternative (RSO) that preserves optimality and increases the action-gap.

One experiment in the paper compares the performance of RSO to the classical and consistent Bellman operators on a discretized version of the Mountain Car problem (`Mountain-Car-V0`) from the OpenAI Gym. Since Mountain Car has a continuous state space, the discrete-version is bound to have the approximation errors discussed above, meaning that the use of a gap-increasing operator would likely be beneficial. We attempt to reproduce the results of the three operators on Mountain Car as given in the paper. We also examine claims that the relative performance of the operators remain unchanged under variation of epsilon. Finally, we do a hyperparameter sweep to find the optimal

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

hyperparameter settings for the Bellman operator, the consistent Bellman operator, the authors' preferred parameterization of RSO, and a RSO with a different, deterministic parameterization. We then compare each operator using its optimal settings to assess the relative performance of the operators when run under the conditions best for each operator.

2 Background

The setting for our operators is a discrete, time-discounted MDP, defined by $(\mathbb{X}, \mathbb{A}, \mathbb{P}, R, \gamma)$, where $\mathbb{X}$ is the set of states, $\mathbb{A}$ is the set of actions, $\mathbb{P}$ is the transition probability kernel, R is the reward function, and γ is the discount factor. We use ϵ-greedy Q-learning with several different operators and a learning rate α. The initial values are taken to be the first reward encountered.

The first operator is the familiar Bellman operator. Updates are of the form

$$Q_{k+1}(s,a) = (1-\alpha)Q_k(s,a) + \alpha(R(s,a) + \gamma \max_{a\in\mathbb{A}} Q_k(s',a)) \tag{1}$$

where s is the current state and s' is the next state.

The second operator is the *consistent Bellman operator* introduced by Farahmand et al. It updates as follows:

$$Q_{k+1}(s,a) = (1-\alpha)Q_k(s,a) + \alpha\left(R(s,a) + \gamma\left(\mathbb{1}_{\{s!=s'\}} \max_{a\in\mathbb{A}} Q_k(s',a) + \mathbb{1}_{\{s=s'\}} Q_k(s,a)\right)\right) \tag{2}$$

The final operator is the family of robust stochastic operators introduced in the paper. It takes as a parameter a sequence β_k of independent nonnegative random variables with finite support and expectation $\mathbb{E}_{\mathbb{P}}[\beta_k] \in [0,1]$. Then the update is

$$Q_{k+1}(s,a) = (1-\alpha)Q_k(s,a) + \alpha\left(R(s,a) + \gamma\left(\max_{a\in\mathbb{A}} Q_k(s',a) - \beta_k\left(\max_{a\in\mathbb{A}} Q_k(s,a) - Q_k(s,a)\right)\right)\right) \tag{3}$$

To run our replication we adapted the same reinforcement learning code base that the authors of the original paper did [4]. Changes we made included adding the new operators, altering the structure of the code to enable easier testing, and adding in graphing functionality. In order to adapt the discrete-time, discrete-state operators to Mountain Car, we discretized the state space into a 40×40 grid as specified in Lu et al. [1] for every experiment in this paper (10×10 in the original codebase).

3 Reproduction And Verification

Our first goal was to replicate the graph showing the relative performance of the RSO where $\beta_k \sim U[0,2)$, Bellman, and consistent Bellman operators on Mountain Car. Due to the ambiguity in the parameter setting, we explore and find that ϵ decay is not used in the original graph. Afterward, we examine the claim made by the authors that the relative performance of the operators remains consistent across different values of ϵ.

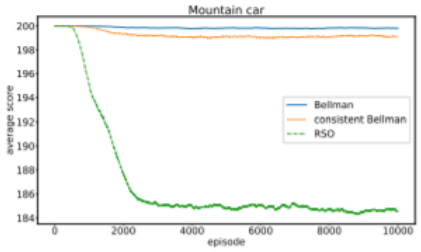
3.1 Graph Replication

The result of our graph replication is given in `Figure 1`. The authors of the original paper claim that they use the default hyper-parameters of the library [4], which are given in `Table 1`. It was unclear whether they used ϵ decay, which is optional in the program. But we discover that our graph resembles theirs when ϵ decay is not used. Thus in our replication of the graph we do not use ϵ decay but use the other default hyperparameters.

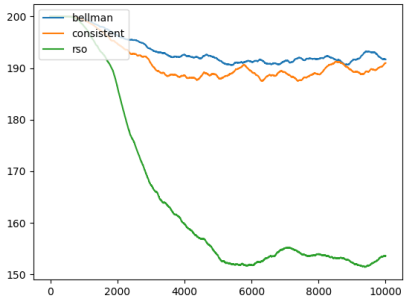
3.2 Effects of ϵ Decay

We now resolve the ambiguity of the choice on ϵ decay in the original paper by analyzing the effect of such decay on the relative performance of RSO, consistent Bellman and classical Bellman operators.

The result is demonstrated in `Figure 2`. We run all three operators on Mountain Car with two configurations. First with default parameters ($\alpha = 0.5, \epsilon = 0.1, \gamma = 0.9$) and second where we use



(b) Mountain Car problem.



(a) The original graph in [1].

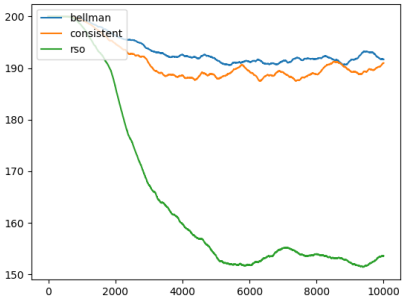
(b) Our replication.

Figure 1: The replication successfully demonstrates that RSO shows better convergence to a policy than the other two operators with the parameters given in table 1. RSO converges at a comparable rate in the replication but achieves a significantly better final score. The performance of Bellman and consistent Bellman are also marginally better in the replication.

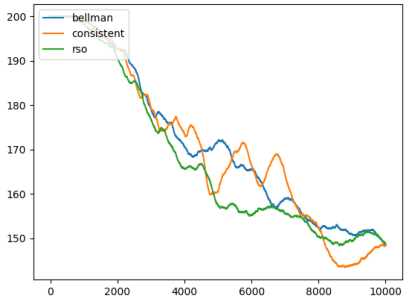
Table 1: Hyperparameters (α: learning rate, γ: discount factor, ϵ: exploration factor, β_k: RSO random variable) used in the generation of the graphs. The original paper uses the default parameters in a framework but leaves the use of parameter decaying undisclosed. We disable decaying since we later prove that decaying makes the performance difference between RSO and Bellman smaller.

	α	γ	ϵ	ϵ decay	β_k
Original	0.5	0.9	0.1	n/a	U(0,2)
Replication	0.5	0.9	0.1	Off	U(0,2)

default parameters and discount ϵ by 0.999 after each episode. The introduction of ϵ decay allows all operators to perform better and vastly improves the performance of the classical and consistent Bellman operators thus reducing the advantage of RSO. Consequently, such decay is almost certainly not used in the original paper and accordingly is not used in our replication.



(a) Without ϵ decay.



(b) With 0.999 per episode ϵ discount.

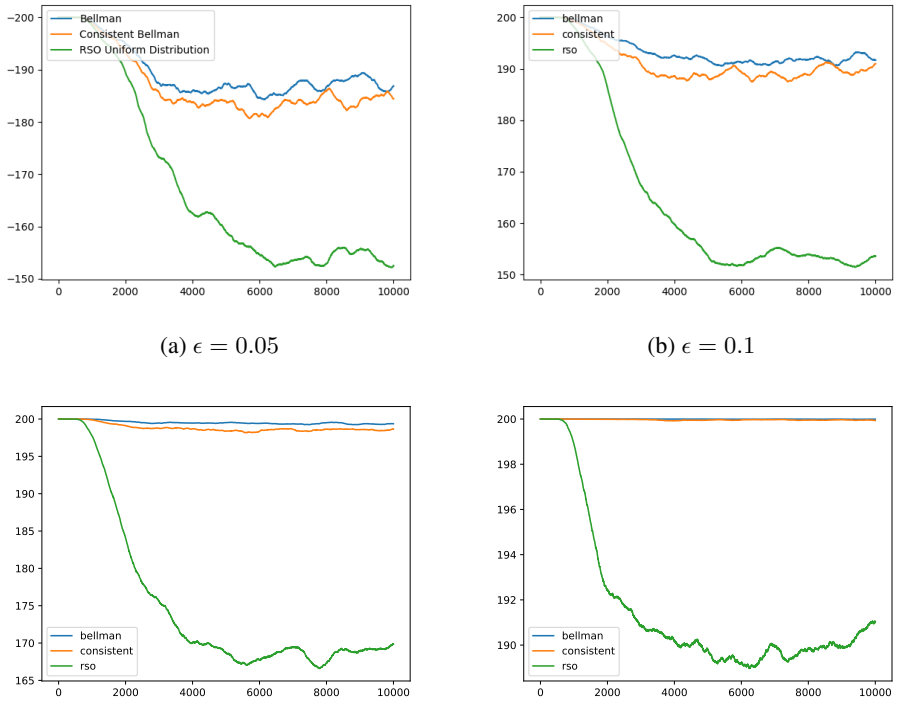
Figure 2: ϵ decay allows all the operators to converge to a better but similar policy. However, it considerably reduces the performance benefit of RSO.

3.3 Invariance of Relative Performance Under Variation of ϵ

In the original paper, the authors noted that the relative performances of Bellman, consistent Bellman and RSO generally don't change drastically with respect to the value of ϵ — the amount of exploration. To verify that, we fixed the value of alpha at 0.5 and that of gamma at 0.9, which are the default values

in the code that the original paper altered for their experiments, and experimented with different epsilons.

As demonstrated by `Figure 3`, the general trends of the graphs are indeed approximately invariant with $\epsilon = 0.05, 0.1, 0.3, 0.5$. RSO consistently converges faster than the other two and achieves the best result at the end. We also note that the graphs with $\epsilon = 0.3$ and $\epsilon = 0.5$ are very similar to the graph we were attempting to replicate, even though the authors stated that they replaced only the operator code in the original code, which has $\epsilon = 0.1$.



(a) $\epsilon = 0.05$ (b) $\epsilon = 0.1$

(c) $\epsilon = 0.3$ (d) $\epsilon = 0.5$

Figure 3: The plots show that RSO indeed outperforms both Bellman and consistent Bellman regardless of ϵ, in that RSO solves the Mountain Car problem much faster and obtains a significantly better score.

4 Optimizing Operator Performance Over Hyper-parameters

When a particular model is being used in research or industry, it is common to perform gradient descent or some other form of local search to find values for the model's hyper-parameters that maximize its performance. With that in mind, we performed a hyper-parameter sweep, trying different combinations of ϵ and ϵ decay with each operator. We additionally experimented with α decay, lower values of α, and different values of γ, but found that these changes either made little difference in performance or were detrimental. By comparing each operator using its optimal parameters, we were able to make a more fair comparison between the RSO with $\beta_k \sim U[0, 2)$ proposed by the authors and the other operators.

4.1 Methods

Our hyperparameter sweep was carried out over every combination of $\alpha = \{0.5\}, \epsilon = \{0.05, 0.1, 0.3\}, \gamma = \{0.9\}, \epsilon_{\text{decay}} = \{0.999, 1\}$ on RSO with $\beta_k \sim U[0, 2)$, the Bellman Operator, and the Consistent Bellman Operator. We additionally did the sweep over the RSO with $\beta_k = 1$ to verify the authors claim that $\beta_k \sim U[0, 2)$ was optimal. For each setting of hyperparameters in

Table 2: Average score for each operator given optimal hyperparameters.

Operator	Score
Bellman	148.21
Consistent Bellman	150.02
RSO Stochastic	140.17
RSO Deterministic:	122.35

the sweep, we took the score of policies learned by each operators for 1,000 episodes following 10,000 episodes of training and then averaged it over 20 trials. We then chose the hyperparameters that resulted in the best average score for each operator to constitute that operator's optimal hyperparameter setting. Finally, we compared the performance of all the operators with their respective optimal hyperparameter settings. The performance results can be seen in `Table 2` and the optimal hyperparameters for each operator can be seen in `Table 3`.

Table 3: Optimal hyperparameters (α: learning rate, γ: discount factor, ϵ: exploration factor, β_k: RSO random variable) for each operator.

Operators	α	γ	ϵ	ϵ decay	β_k
Bellman	0.5	0.9	0.1	On	
Consistent Bellman	0.5	0.9	0.05	On	
RSO Stochastic	0.5	0.9	0.3	Off	U(0,2)
RSO Deterministic	0.5	0.9	0.3	Off	1

4.2 Results

Our analysis reveals that when we use the classical Bellman, Consistent Bellman, and RSO with $\beta_k \sim U[0, 2)$ with their optimal hyperparameters, the RSO outperforms the others by a significant margin. Additionally, the deterministic operator (RSO with $\beta_k = 1$) substantially outperforms the RSO with the authors' chosen parameterization ($\beta_k \sim U[0, 2)$), which contradicts the authors' claims that $\beta_k \sim U[0, 2)$ performed better. For all of the operators, we found optimal hyperparameters different than those used by the authors. Compared to the default values, for the Bellman operator, the addition of epsilon decay resulted in the best performance; for the Consistent Bellman Operator, epsilon decay along with a smaller epsilon resulted in the best performance; and for the RSO with $\beta_k \sim U[0, 2)$ a greater epsilon resulted in the best performance.

5 Reflection

5.1 Graph Replication

We strictly followed the authors' methodology to generate the replication graph using the same existing codebase with relatively minor changes. However, although the trends are approximately the same, the values we eventually reach are considerably different from those in the original graph. As mentioned before, it seems that they may have used some ϵ of value between 0.3 and 0.5. (`Figure 3`) Due to limited compute, we did not search for that ϵ so as to generate the exact same graph. Additionally, we tried replicating the graph with and without epsilon decay to see what the authors originally did, since it's not specified in the paper. However, our graphs with epsilon decay were substantially different than theirs with the Bellman-like operators showing markedly better performance.

Overall, while the graph was not misleading and did show a real advantage of RSO over the Bellman operators in training, it would have been helpful if the authors had explicitly stated the hyperparameters they used. Additionally, while we did manage to replicate the authors' claim that changes in the value of ϵ did not have a large impact on the relative performance of the operators, it would have been helpful if the authors had included the results of these experiments or at least the ϵ values searched over in the appendix.

5.2 Hyperparameter Optimizations

After comparing each operator with optimal parameters, we clearly verified the superiority of RSO with $\beta_k \sim U[0, 2)$ to the Bellman Operators. Therefore, the authors' claim of RSO's superiority was replicated successfully. However, our results contradicted the authors' preferred parameterization of $\beta_k \sim U[0, 2)$, since our deterministic RSO ($\beta_k = 1$) performed substantially better.

Interestingly, our validation score was substantially worse than the authors across all the operators with the default hyperparameters, as can be seen in `Table 4`. However, the scorer of our RSO with $\beta_k = 1$ still managed to, at 122.35, be slightly better than the authors' reported error using the RSO with $\beta_k \sim U[0, 2)$.

Table 4: Average Score With Default Parameters

Operator	Authors' Score	Replication Score
Bellman	129.93	185.10
Consistent Bellman	127.58	182.18
RSO Stochastic	122.70	142.70

We were not able to resolve the reason for our worse validation error relative to the authors' results. One potential reason is that the hyperparameters they used for these results were different than the ones stated.

6 Conclusion

Overall, we found that the results of the work were largely replicable. While we could not replicate the graph exactly, the graph we generated showed similar relative performance between the operators. Furthermore, we found the authors' claims about the invariance of the operators' relative performance under various ϵ and the superiority of RSO for Mountain Car to be well-founded. However, we could not replicate the exact validation scores on Mountain Car — ours were all about 20 higher. Additionally, we found that the RSO with $\beta_k = 1$ was superior to the RSO with $\beta_k \sim U[0, 2)$ proposed by the authors, despite the authors' claims to the contrary.

References

[1] Yingdong Lu, Mark Squillante, and Chai Wu. *A Family of Robust Stochastic Operators for Reinforcement Learning*. Advances in Neural Information Processing Systems pre-proceedings, 2019.

[2] A. Farahmand. *Action-gap phenomenon in reinforcement learning*. Advances in Neural Information Processing Systems, 24, 2011.

[3] M. G. Bellemare, G. Ostrovski, A. Guez, P. S. Thomas, and R. Munos. *Increasing the action gap: New operators for reinforcement learning*. In Proc. Thirtieth AAAI Conference on Artificial Intelligence, AAAI'16, pages 1476–1483. AAAI Press, 2016.

[4] Víctor Mayoral Vilches. *Basic Reinforcement Learning*.
Online, available: `https://github.com/vmayoral/basic_reinforcement_learning`