Re: A Family of Robust Stochastic Operators for Reinforcement Learning

Ajay Balaji, Rigel Galgana, Isaiah Liu, Andrew Park, Benjamin Vu, James Wang
Department of Computer Science
Brown University
Providence, RI 02703
{ajay_balaji, rigel_galgana, isaiah_liu,
andrew_park1, benjamin_vu, james_wang1}@brown.edu

Abstract

Various Q-learning operators have been proposed to improve performance in the presence of approximation or estimation errors. Recently, in *'A Family of Robust Stochastic Operators for Reinforcement Learning'*, the authors prove rigorously and show experimentally that stochastic operators can outperform more classical operators, such as the Bellman and Consistent Bellman operator. In this paper, we replicate their experimental results and find not only results mostly consistent with the paper, but also demonstrate that these stochastic operators can converge more rapidly to an optimal policy.

1 Introduction

Q-learning is a major technique in Reinforcement Learning, which functions by updating the agent's estimates of the value of state-action pairs according to some operator. While Q-learning using Bellman operator based updates can be shown to exactly solve for the optimal policy in the absence of approximation or estimation errors, this may not necessarily be the case for more complex problems. In addition, when the values of different actions at a particular state are close, these approximation errors further make learning the true optimal action difficult. Recent efforts to mitigate these effects involve constructing Q-learning operators that are *optimality preserving* and *action-gap increasing* [2]. Optimality preserving operators guarantee convergence to the optimal policy while gap-increasing operators improves Q-learning's rate of convergence. Most recently proposed operators with the above properties have been deterministic, which carry with them an inherent trade-off between optimality preservation and action-gap increasing. More specifically, operators which increase the action gap too much may violate optimality [3].

In *'A Family of Robust Stochastic Operators for Reinforcement Learning'*, the authors propose adding randomness with certain properties to the standard Bellman operator to circumvent this trade-off. The authors provide rigorous justification that these stochastic operators are both optimality-preserving and gap increasing in a stochastic sense. They then provide experimental evidence of these claims as well as directly compare the performance of their stochastic operator, the standard Bellman operator and Consistent Bellman Operator, a type of deterministic gap increasing and optimality preserving operator, on OpenAI gym games. In section 2, we will briefly explain relevant notation and the relevance of the stochastically action-gap increasing and optimality-preserving properties in our replication. In section 3, we compare the results found by [5] and our replicated results. In section 4, we analyze the performance of RSO under different exploration schemes, such as the ϵ-greedy and softmax methods. In section 5, we compare the performance of different distributions for the perturbation term in the stochastic operator. In section 6, we summarize our findings.

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

2 Properties of Robust Stochastic Operators (RSO)

In this section, we provide a brief overview of notation and then reiterate the proposed family of robust stochastic operators described in [5], as well as explain why any sequence of these family of operations are both stochastically optimality-preserving and gap-increasing. The MDP is given by $(\mathbb{X}, \mathbb{A}, \mathbb{P}, R, \gamma)$, where $\mathbb{X}$ and $\mathbb{A}$ denote the state and action spaces respectively with transition probabilities $\mathbb{P} : \mathbb{X} \times \mathbb{A} \times \mathbb{X} \to [0, 1]$, reward function $R : \mathbb{X} \times \mathbb{A} \to [0, \infty)$ and discount factor $\gamma \in [0, 1)$. Let $\mathbb{V}$ and $\mathbb{Q}$ denote the set of real valued functions over $\mathbb{X}$ and $\mathbb{X} \times \mathbb{A}$, where for $Q \in \mathbb{Q}$, $x \in \mathbb{X}$, and $a \in \mathbb{A}$, the value function $V \in \mathbb{V}$ is defined as $V(x) = \max_a \mathbb{E}_{x' \in \mathbb{P}(\cdot|x,a} Q(x', a)$. Let $\{\mathcal{T}_{\beta_k}\}_{k \in \mathbb{N}}$ denote a sequence of operators on $\mathbb{Q}$ where $\{\beta_k\}_{k \in \mathbb{N}}$ is a sequence of independent, non-negative, and bounded random variables with expectation between 0 and 1 inclusive. Finally, a robust stochastic operator is defined as the following:

$$\mathcal{T}_{\beta_k} Q_k(x, a) := R(x, a) + \gamma \mathbb{E}_{x' \in \mathbb{P}(\cdot|x,a)} V_k(x') - \beta_k \left[V_k(x) - Q_k(x, a) \right] \tag{1}$$

where $V_k \in \mathbb{V}$ and $Q_k \in \mathbb{Q}$ denote the agent's updated value and state-action value tables after k iterations. The authors further show this family of operators is also optimality-preserving in the stochastic sense. That is, these operators guarantee that Q-learning converges to a policy returning the true optimal actions at each state. This implies that that even training in the continuous sense is possible with loss of optimality. Furthermore, the guarantee of optimality in the stochastic sense shows that a convergence in an infinite dimensional space of random variables is possible, in contrast with convergence in a discretized, finite dimensional space explored by the standard Bellman operators.

Among these sequences that are optimality-preserving, the papers also states that for sequences with larger variance for β_k with the same mean, we also have a larger variance for $Q_k(x, a)$. Intuitively, an action-gap increasing operator widens the difference between the values of the actions of the true optimal action and all sub-optimal actions for each state. Since the sequences are optimality-preserving, we know that the value $Q_k(x, a)$ will be very close to the optimal value $Q^*(x, b^*)$, for some optimal action b^* in state x. Therefore, by using a sequence of β_k with larger variance, there is higher likelihood of the value $Q_k(x, a)$ being smaller, implying a larger action gap. Since this does not affect the ultimate convergence of the operators, this simple implementation decision regarding the variance β_k can be used to exploit the stochastic ordering result explained previously.

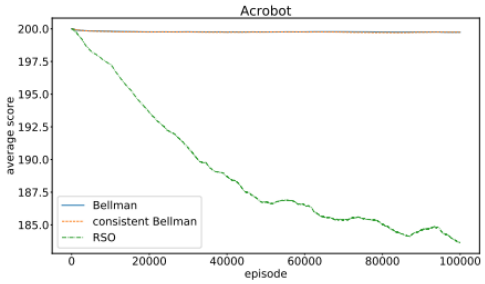
As we will be comparing the performance of RSO to the classic and consistent Bellman operators, we also note the differences among them. The classical Bellman operator makes no adjustment for potential approximation or estimation errors, and hence $\beta_k = 0$ for all k. In contrast, the consistent Bellman operator modifies the Q function such that action a is taken again if a yielded no state transition; $x' = x$. Bellemare [2] showed that this operator is action-gap increasing and optimality preserving and thus, given its simplicity, will serve as a representative benchmark amongst deterministic operators to compare RSO against. The experimental results of these operators on OpenAI Gym games are shown below.

3 Figures

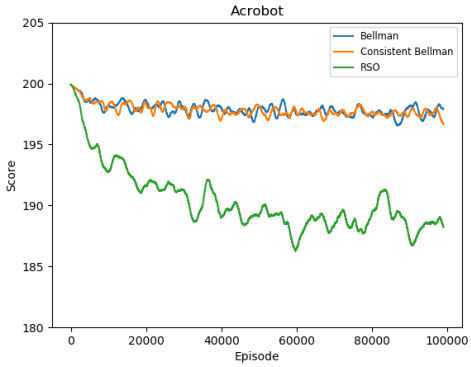
In this section, we show and explain our experimental replication results for some of the OpenAI Gym environments used in the paper: Acrobot, Mountain Car, and Lunar Lander. The authors observed that "simply replacing the Bellman operator or the consistent Bellman operator with the RSO generally results in significant performance improvements" in their experiments; our aim in this section is to re-do the analysis from scratch to see if that observation can be verified, as well as attempt to recreate the figures. This formed the bulk of our replication effort.

3.1 Acrobot

This environment is introduced in [4]. The authors [5] set up the state space as a 6-dimensional vector with three actions each, with the goal of minimizing the total score. The position and velocity values were discretized into 8 bins, while the other states were discretized into 10. Over 20 experiments, they trained the three agents for many episodes (an explicit number was not given). The graph in Figure 1a displays the average score of each of these agents, averaged over a 1000 episode moving average.



(a) Original Figure



(b) Our Figure

Figure 1: Acrobot Training Graphs

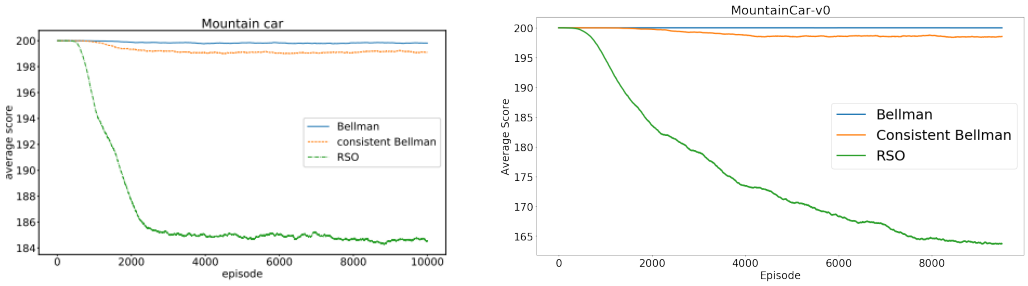
We attempted to replicate this environment by copying their specifications and using the hyperparameters $\alpha = 0.1$, $\gamma = 0.999$, $Q_{init} = 0$, and $T = 1$. Our results are displayed in Figure 1b. Ultimately, although the overall graph seems to be consistent with the fact that the average score of the RSO agent outperforms the other two, our graph seems to have much more variance compared to the original, as confirmed by the "jaggedness" of the visualization. This could be because we do not perform any sort of annealing method to our learning algorithm, while the original authors may have utilized an annealing method on top of the sliding window smoothing on the figure. In addition, we investigated their action gap values achieved in each of the agents by running each of the three algorithms for 100,000 episodes average over 72 trials. Although this is not consistent with the 40 trials they averaged in the paper, we assume that the mean action gap should still be similar between our experiment and the original. Indeed, our replicated action gap values are consistent with the original. Although our action gap values are larger than the original, this is still consistent with the fact that our replication performed slightly better overall. In addition, the paper did not provide any methodology for calculating the average action gap, so we computed this value by weighting the action gap value for each state by the empirical visited frequency.

Table 1: Original vs Replicated Action Gaps

Experiment	Original	Replication
Bellman	0.1436	1.1966
Consistent	0.1263	1.4587
RSO	0.9004	2.3851

3.2 Mountain Car

This OpenAI Gym environment is introduced in [1]. The authors [5] replicated the set up of Mountain Car by discretizing the 2-D state space into a 40x40 grid; there are three actions possible at each state and the goal of the problem is to minimize the player's score. Over 20 experiments, they trained the three agents for 10,000 episodes of up to 200 steps, and reported the training graph shown in Figure 2(a) (note that the graphs use a 500 episode moving average to smooth the curves). Their observation was that the RSO agent performed much better on average than the Bellman and Consistent Bellman agents.



(a) Original Figure

(b) Our Figure

Figure 2: Mountain Car Training Graphs

We attempted to replicate this environment by not only using the specifications above, but also choosing the hyperparameters $\alpha = 0.01$, $\gamma = 0.99$, $Q_{init} = -50$, and $T = 1$. Our results, shown in Figure 2(b), match their observation that the average score of the RSO agent is much better than that of the other two agents. From this standpoint, we can verify that replicating the training phase of this experiment was relatively straightforward. The only hurdle was realizing a softmax distribution over actions should be used to choose actions during the training phase; for more discussion on this please see section 4.

The authors also reported testing results: after the 10,000 episodes of training, they used the final policy to run the agent for 1,000 steps. They reported average testing scores and standard deviations of 129.93 and 32.68 for the Bellman agent, 127.58 and 30.90 for the Consistent Bellman agent, and 122.70 and 7.25 for the RSO agent. These results were harder to replicate: the policy yielded by the our training graph above in Figure 2(b) led to average testing scores and standard deviations of 123.69 and 9.48 for the Bellman agent, 119.45 and 3.66 for the Consistent Bellman agent, and 119.43 and 3.68 for the RSO agent.

In the paper, the RSO agent had by far the best testing performance; it outperformed the other two operators and also had a much lower standard deviation. Our results paint a less straightforward picture. The standard deviation of the RSO agent was just marginally better than the classic Bellman agent and marginally worse than the Consistent Bellman agent on average. To delve deeper, we repeated the same process with different training lengths and saw the following results (scores and standard deviations):

Table 2: Training Time vs Testing Results

Experiment	500 (Episodes)	1000	2500	5000	10000
Bellman	193.14 (17.19)	151.78 (25.94)	126.81 (4.61)	124.54 (3.72)	123.69 (9.48)
Consistent	149.21 (24.20)	128.14 (4.47)	120.68 (3.41)	119.96 (3.43)	119.45 (3.66)
RSO	129.67 (5.16)	120.87 (3.30)	119.42 (3.70)	119.43 (3.69)	119.43 (3.68)

As the training time decreases, we see a larger divergence between the RSO agent and the classical agents. With only 1000 episodes of training, the RSO agent attains testing results very close to those attained after 10,000 episodes of training (120.87 vs 119.43). In contrast, after 1000 episodes of training the Bellman and Consistent Bellman agents lag far behind. Furthermore, with short training the testing results of the RSO agent have lower variance than those of the classical agents. This is in line with our prior that the RSO agent learns faster and more stably than with the other two operators, but does no better in the long term. Hence the authors' claim of the RSO agent's superior performance seems valid in the context of this experiment, as it finds an equally good policy in a shorter amount of time than the other agents.

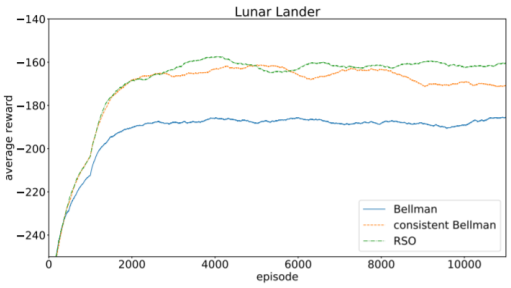
3.3 Lunar Lander

We used the discrete LunarLander-v2 OpenAI Gym environment to evaluate the three agents: Bellman, consistent Bellman, and RSO with epsilon-greedy and softmax policies. Additionally, learning rate

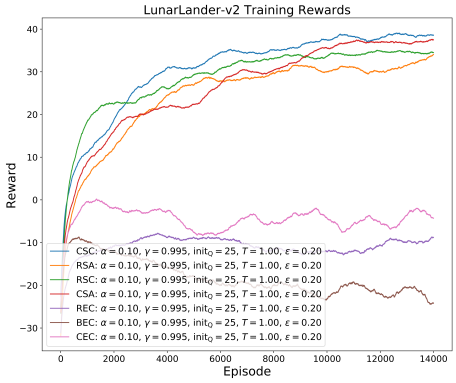
annealing was performed on the softmax agents for a total of 9 agents: BSC, BSA, BEC, CSC, CSA, CEC, RSC, RSA, and REC, where the first character B/C/R indicates the type of update (Bellman, consistent Bellman, RSO), the second character E/S indicates ϵ-greedy/softmax, and the third C/A indicates constant or annealed learning rate. Under this naming scheme, the original paper's agents would be BEC, CEC, and REC. The learning rate, α, was varied between the three values 0.01, 0.1, and 0.5, but only the agents with learning rate 0.1 did well.

The state vector (6 real numbers + 2 bits) was discretized into 4 bins (given by the four intervals in $[-\infty, -0.5, 0, 0.5, \infty]$) for each of the continuous entries. The total size of the state space was 16384.

The training rewards for the top 5 (out of 27) and BEC, CEC agent-hyperparameter pairs over 15000 episodes (100 moving average) are shown below:



(a) Original Figure

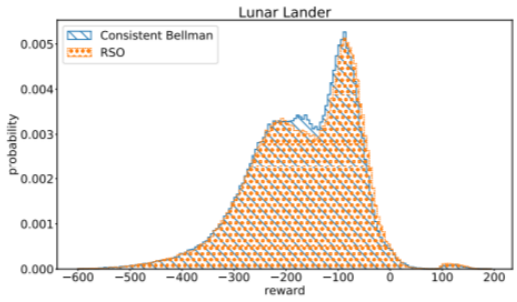


(b) Our Figure

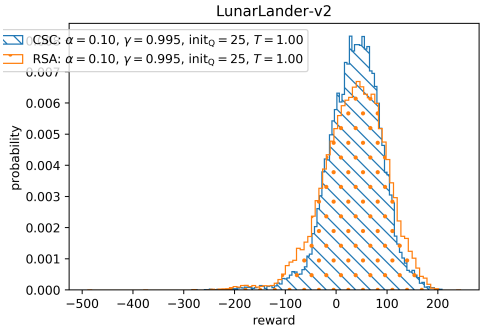
Figure 3: Lunar Lander Training Graphs

Notably, our rewards were much higher than those given in the original paper. There was little difference between the performance of the consistent Bellman agents and the RSO agents. The reward curves were relatively smooth (given the unknown smoothing factor on the original graph).

Over 3000 testing episodes and 20 experiments, a histogram was generated of the test rewards for the top two agents (CSC and RSA):



(a) Original Figure



(b) Our Figure

Figure 4: Lunar Lander Training Graphs

Notably, the distribution we got is centered much farther to the right (in the positive reward area) than the one given in the original paper. This was true for all agents except BEC with $\alpha = 0.5$. Additionally, the distributions are approximately normal with low variance compared to the original

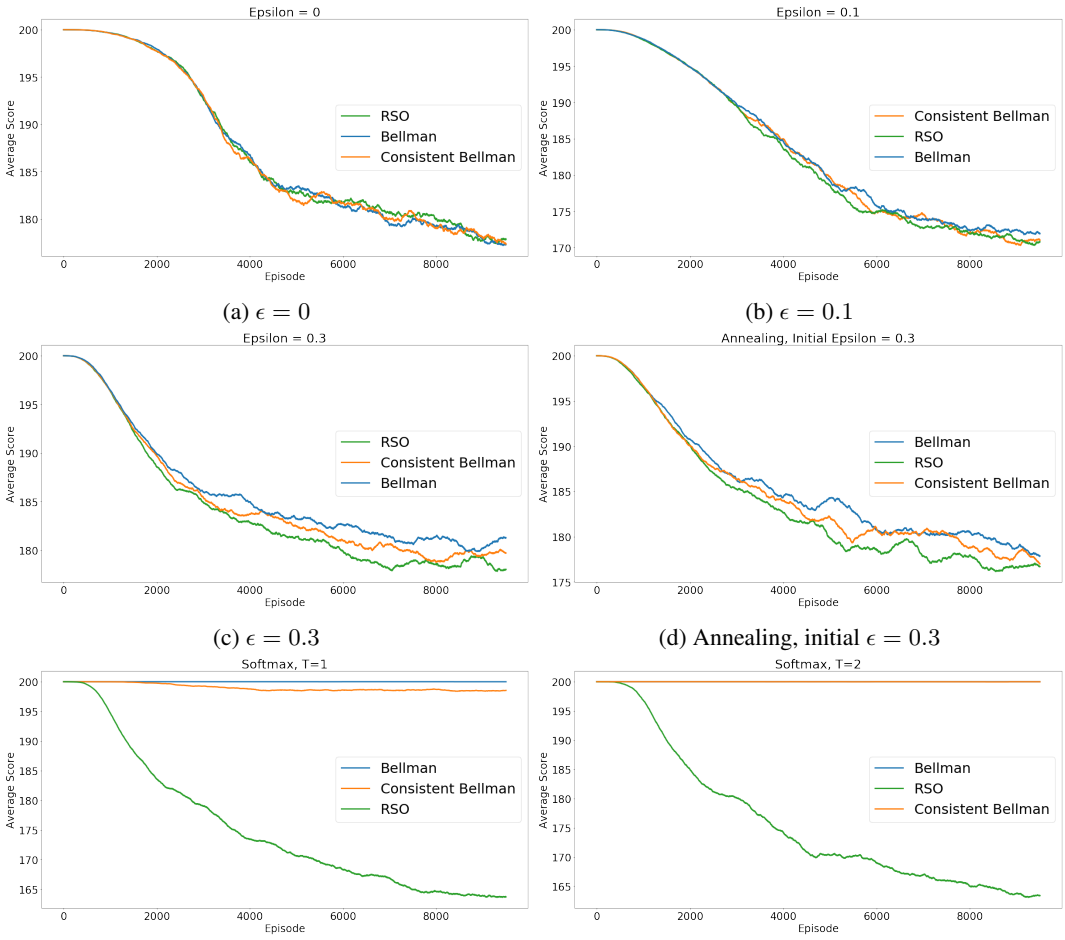
figure. The CSC agent had an marginally better center (36.38 vs 35.66) and lower standard deviation (52.07 vs. 61.47) than the RSA agent. This would seem to indicate that the performance CSC is at least equivalent to, if not superior to, that of RSA (and RSC, REC) in the LunarLander-v2 domain. Hence we cannot clearly verify that the RSO agent performs better than the classical agents here.

4 Exploration Schemes

Although the authors mentioned that they used the ϵ-greedy to balance exploitation and exploration, they did not specify which values of ϵ were used in each experiment. In particular, the authors mentioned that the relative performance of the three different agents did not depend much on the value of ϵ.

For the Mountain Car experiments (we focused on these as they had the shortest runtime), we found that ϵ did somewhat affect the relative performance of the agents. This could be hyperparameter dependent: perhaps our setup led to dependence on ϵ while the authors' setup did not. However, we document our observations regarding ϵ tuning below; note that the other hyperparameters were fixed at the values $\alpha = 0.01$, $\gamma = 0.99$, and $Q_{init} = -50$.

Below are figures showing the average scores over 20 experiments for 10,000 episodes of training for different exploration schemes. We have documented 3 values of ϵ as well as the cases where the softmax distribution with temperatures 1 and 2 are used to construct action probabilities. We tried annealing epsilon as well. Each curve has a 500 episode moving average. Testing results are provided in the table below the figures.



(a) $\epsilon = 0$

(b) $\epsilon = 0.1$

(c) $\epsilon = 0.3$

(d) Annealing, initial $\epsilon = 0.3$

(e) softmax, T=1

(f) softmax, T=2

Figure 5: Mountain Car Training Graphs For Different Exploration Schemes

Table 3: Testing Results for Different Exploration Schemes

	$\epsilon = 0$	$\epsilon = 0.1$	$\epsilon = 0.3$	Annealed, $\epsilon = 0.3$	softmax, $T = 1$	$T = 2$
Bellman	171.55	162.28	130.52	150.56	123.69	146.58
Consistent	188.80	165.60	129.90	135.81	119.45	121.82
RSO	184.79	161.30	127.58	138.74	119.43	119.42

There appears to be a difference between using an ϵ-greedy training policy and using a softmax approach. The softmax approach seems to lead to much greater differentiation *in the training phase* between the RSO and classical agents. In the ϵ-greedy graphs, the performance of the three agents is in the same ballpark, but in the softmax graphs the RSO agent performs drastically better than the other two.

As for testing results, it seems like both exploration schemes follow a similar trend: a higher propensity to explore seems to result in better RSO performance relative to the classical agents. This seems to be the case in both the training and testing phases, at least for the Mountain Car problem. For example, in the training graphs we can see that for the ϵ-greedy case, as ϵ increases the RSO curves starts to seperate a little from the other two curves. In the softmax training case, the classical operators really struggle in the higher temperature setting. We also can see that the gap between RSO and the other agents widens in the more exploratory columns of the testing results table. Furthermore, annealing the learning rate did not seem to have a large effect.

Although the authors explicitly mentioned they used the ϵ-greedy method in their experiments, the reader will note that the softmax graphs in Figure 5 look much more like the graphs in the paper than the ϵ-greedy graphs do. We took the step of emailing the authors to clarify this point; they confirmed that they used the softmax operation to construct action probabilities for their agents.

To reiterate, we found that more exploratory behavior tended to improve the performance of the RSO agent *relative to* the performance of the classical agents; in addition, the softmax method seemed to produce more differentiation than the ϵ-greedy method during the training phase.

5 Distribution of β

As a proof of concept that higher variance distributions for $\{\beta_k\}$ are more action-gap increasing, the authors of RSO also compared the performance of the RSO agents using distributions $\{\beta_k\} \sim U[0, 1)$ and $\{\beta_k\} = 1$ against the original $\{\beta_k\} \sim U[0, 2)$. While the authors claim that $\{\beta_k\} \sim U[0, 2)$ yields consistently superior performance, our results are mixed, suggesting that higher mean distributions tend to perform better. We tested the three distributions $U[0, 1)$, $U[0, 2)$, and fixed 1 that the authors used, as well as fixed 0, $U[0, \frac{1}{2})$, $U[0, \frac{3}{2})$, $U[\frac{1}{2}, \frac{3}{2})$, and $\{\beta_k\} \in \{0, 2\}$ with equal probability.

Table 4: Different Distributions of $\{\beta_k\}$ with mean 1, in order of increasing variance

Experiment	500 (Episodes)	1000	2500	5000	10000
$\beta_k = 1$	129.40 (5.62)	121.31 (3.67)	119.43 (3.71)	119.40 (3.68)	119.42 (3.67)
$\beta_k \sim U[\frac{1}{2}, \frac{3}{2})$	129.68 (8.83)	121.06 (3.40)	119.46 (3.66)	119.41 (3.70)	119.43 (3.67)
$\beta_k \sim U[0, 2)$	129.67 (5.16)	120.87 (3.30)	119.42 (3.70)	119.43 (3.69)	119.43 (3.68)
$\beta_k \in \{0, 2\}$	130.83 (5.83)	121.12 (3.46)	119.43 (3.67)	119.42 (3.68)	119.37 (3.72)

Interestingly, there is no noticeable difference amongst these distributions, which all have mean 1 but different variances. This is inconsistent with the authors' [5] claim of higher variance improving performance, theoretically and experimentally. However, performance seems to be positively correlated with the mean, as shown below.

Table 5: Different Distributions of $\{\beta_k\}$ in order of increasing mean

Experiment	500 (Episodes)	1000	2500	5000	10000
$\beta_k = 0$	193.14 (17.19)	151.78 (25.94)	126.81 (4.61)	124.54 (3.72)	123.69 (9.48)
$\beta_k \sim U[0, \frac{1}{2})$	184.69 (22.89)	141.90 (19.27)	124.55 (4.12)	121.73 (3.32)	120.42 (3.39)
$\beta_k \sim U[0, 1)$	172.53 (29.26)	127.42 (5.26)	121.57 (3.45)	120.10 (3.39)	119.55 (3.62)
$\beta_k \sim U[0, \frac{3}{2})$	143.02 (18.81)	124.25 (3.95)	119.87 (3.47)	119.42 (3.69)	119.43 (3.70)
$\beta_k \sim U[0, 2)$	129.67 (5.16)	120.87 (3.30)	119.42 (3.70)	119.43 (3.69)	119.43 (3.68)

As the mean of β deviates from 1, the performance seems to decrease as well. This suggests that distributions with higher mean may be optimal. As the authors showed that the expectation of $\{\beta_k\}$ being bounded between 0 and 1 is necessary condition for stochastic optimally-preserving.

6 Conclusion

We summarize our replication results below:

- In the replicated experiments (Acrobot, Mountain Car), we confirmed the authors' findings that the RSO agent achieved better training and testing performance than the classical agents.
- We were able to replicate the action gap results from the Acrobot experiment. We found that the RSO agent actually had a larger action gap relative to the classical agents than stated in the paper.
- We did not see the same negatively skewed, bimodal distribution of testing scores in the Lunar Lander experiment that the authors saw. We also achieved a significantly higher reward.
- We found that more exploratory hyperparameters (larger ϵ in the ϵ-greedy setting, larger temperature in the softmax setting) led to a larger gap in performance between the RSO agent and the classical agents. This contradicts the paper's finding that the value of ϵ did not affect the relative performance of the three operators (the paper did not explicitly mention the softmax case).
- We found that using distributions with higher mean for β_k led to better performance in the Mountain Car experiment. However, the alternative distributions with equivalent mean such as fixed 1, $U[\frac{1}{2}, \frac{3}{2})$, $\beta_k \in \{0, 2\}$ had comparable performances, contradicting the authors' claim that higher variance yields better performance.

Ultimately, our replication efforts seems to support the idea that using a Robust Stochastic Operator for Q-Learning can often lead to better training and testing performance than the classical Bellman and Consistent Bellman operators, although it does not seem to completely dominate the other two agents as suggested in the original paper.

References

[1] Alzantot M. Solution of mountaincar OpenAI Gym problem using Q-learning. https://gist.github.com/malzantot /9d1d3fa4fdc4a101bc48a135d8f9a289, 2017.

[2] Bellemare M. G.. & Ostrovski G., D. & Guez A. & Thomas P. S. & Munos R. (2016) Increasing the action gap: New operators for reinforcement learning. In *Proc. Thirtieth AAAI Conference on Artifical Intelligence*, AAAI' 16, pages 1476-1483. AAAI Press. TELOS/Springer–Verlag.

[3] Farahmand A. (2011) Action-gap phenomenon in reinforcement learning. *Advances in Neural Information Processing Systems 24*, Cambridge, MA: MIT Press.

[4] R. S. Sutton. Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding. Advances in Neural Information Processing Systems, 8:1038–1044, 1996.

[5] Yingdong L. & Squillante M. & Wu C. (2018) A Family of Robust Stochastic Operators for Reinforcement Learning. In G. Tesauro, D.S. Touretzky and T.K. Leen (eds.), *Advances in Neural Information Processing Systems 32*, Cambridge, MA: MIT Press.