Constrained Policy Improvement for Safe and Efficient Reinforcement Learning

Elad Sarafian
Bar-Ilan University, Israel
elad.sarafian@gmail.com

Aviv Tamar
Technion, Israel
aviv.tamar.mail@gmail.com

Sarit Kraus
Bar-Ilan University, Israel
sarit@cs.biu.ac.il

Abstract

We propose a policy improvement algorithm for Reinforcement Learning (RL) which is called Rerouted Behavior Improvement (RBI). RBI is designed to take into account the evaluation errors of the Q-function. Such errors are common in RL when learning the Q-value from finite past experience data. Greedy policies or even constrained policy optimization algorithms which ignore these errors may suffer from an improvement penalty (i.e. a negative policy improvement). To minimize the improvement penalty, the RBI idea is to attenuate rapid policy changes of low probability actions which were less frequently sampled. This approach is shown to avoid catastrophic performance degradation and reduce regret when learning from a batch of past experience. Through a two-armed bandit with Gaussian distributed rewards example, we show that it also increases data efficiency when the optimal action has a high variance. We evaluate RBI in two tasks in the Atari Learning Environment: (1) learning from observations of multiple behavior policies and (2) iterative RL. Our results demonstrate the advantage of RBI over greedy policies and other constrained policy optimization algorithms as a safe learning approach and as a general data efficient learning algorithm. A Github repository of our RBI implementation is found at `https://github.com/eladsar/rbi/tree/rbi`.

Keywords: Safe Reinforcement Learning, Constrained Policies Algorithms

Acknowledgements

This research has been partly supported by the Israel Innovation Authority and by the Ministry of Science and Technology, Israel.

1 Introduction

While Deep Reinforcement Learning (DRL) is the backbone of many of the recent Artificial Intelligence breakthroughs [14, 10], it suffers from several factors which inhibit deployment of RL systems to real-world tasks. Two of these elements are: (1) data efficiency and; (2) safety. DRL is notoriously data and time inefficient, requires up to billions of history states [6] or weeks of wall-clock time [5] to train to expert level. While it is partially due to the slow training process of deep neural networks, it is also due to inefficient, yet simple to implement, policy improvement routines. For example, a greedy policy improvement (with a fix exploration parameter) is known to have a higher regret than other methods such as Upper Confidence Bound (UCB) [2], but the latter is much more difficult to adjust to a deep learning framework [3]. This transformation from the countable state space of bandit and grid-world problems to the uncountable state-space in a DRL framework, calls for efficient improvement methods which fit into existing deep learning frameworks.

For some real-world problems, like autonomous cars [13], safety is a crucial factor. Random initialized policies and even a RL algorithm that may suffer from sudden catastrophic performance degradation are both unacceptable in such environments. While policy initialization may be solved with Learning from Demonstrations (LfD) algorithms [1], changing the policy in order to improve performance is still a risky task. Largely since the Q-value of a current policy can only be estimated from the past data. Therefore, for safe RL, it is desirable to design improvement algorithms that model the accuracy of the Q-value evaluation and can mitigate between fast improvement and a safety level [4, 17].

In this work, we propose a policy improvement method that addresses both the sample efficiency of the learning process and the problem of safe learning from incomplete past experience. We start by analyzing the improvement penalty of an arbitrary new policy $\pi(a|s)$ based on an estimated Q-function of a past behavior policy $\beta(a|s)$. We find that under a simplified model of learning the Q-values from i.i.d samples, the variance of a potential improvement penalty is proportional to $\frac{|\beta(a|s)-\pi(a|s)|^2}{\beta(a|s)}$. Therefore, we design a constraint, called reroute, that limits this term. We show that finding the optimal policy under the reroute constraint amounts to solving a simple linear program. Instead of optimizing this policy via a gradient descent optimization, we take a different approach and solve it in the non-parameterized space for every new state the actor encounters. In order, to learn the new improved policy with a parameterized Neural Network (NN), we store the calculated policy into a replay buffer and imitate the actor's policy with a KL regression.

While RBI is designed for safe learning from a batch of past experience, we show that it also increase data efficiency with respect to a greedy step and other constraints such as the Total Variation (TV) [7] and PPO [12]. In fact it is akin in practice to the forward KL constraint [18], however, unlike the KL constraint, it does not require different scaling for different reward signals and it is much more intuitive to design. We validate our findings both in simple environments such as a two-armed bandit problem with Gaussian distributed reward and also in a complex distributed actors framework when learning to play Atari.

2 Rerouted Behavior Improvement

Let us start by examining a single improvement step from a batch of past experience of a behavior policy. Define by β the behavior policy of a dataset $\mathcal{D}$ and by Q^β, and $\hat{Q}^\beta$ its true and approximated Q-functions. Theoretically, for an infinite dataset with infinite number of visitations in each state-action pair, one may calculate the optimal policy in an off-policy fashion [19]. However, practically, one should limit its policy improvement step over β when learning from a realistic finite dataset. To design a proper constraint, we analyze the statistics of the error of our evaluation of $\hat{Q}^\beta$. This leads to an important observation: the Q-value has a higher error for actions that were taken less frequently, thus, to avoid improvement penalty, we must restrict the ratio of the change in probability $\frac{\pi}{\beta}$. We will use this observation to craft the reroute constraint, and show that other well-known monotonic improvement methods (e.g. PPO and TRPO) overlooked this consideration, hence they do not guarantee improvement when learning from a finite experience.

2.1 Soft Policy Improvement

Before analyzing the error's statistics, we begin by considering a set of policies which improve β if our estimation of Q^β is exact. Out of this set we will pick our new policy π. Recall that the most naive and also common improvement method is taking a greedy step, i.e. deterministically acting with the highest Q-value action in each state. This is known by the policy improvement theorem [16], to improve the policy performance. The policy improvement theorem may be generalized to include a larger family of soft steps.

Lemma 2.1 (Soft Policy Improvement). *Given a policy β, with value and advantage V^β, A^β, a policy π improves β, i.e. $V^\pi \geq V^\beta$ $\forall s$, if it satisfies $\sum_a \pi(a|s)A^\beta(s,a) \geq 0$ $\forall s$ with at least one state with strict inequality. The term $\sum_a \pi(a|s)A^\beta(s,a)$ is called the improvement step.*[1]

[1]The proof adhere to the same steps of the greedy improvement proof in [16], thus it is omitted for brevity.

Essentially, every policy that increases the probability of taking positive advantage actions over the probability of taking negative advantage actions achieves improvement. Later, we will use the next Corollary to prove that RBI guarantees a positive improvement step.

Corollary 2.1.1 (Rank-Based Policy Improvement). *Let $(A_i)_{i=1}^{|\mathcal{A}|}$ be an ordered list of the β advantages in a state s, s.t. $A_{i+1} \geq A_i$, and let $c_i = \pi_i/\beta_i$. If for all states $(c_i)_{i=1}^{|\mathcal{A}|}$ is a monotonic non-decreasing sequence s.t. $c_{i+1} \geq c_i$, then π improves β.*

2.2 Standard Error of the Value Estimation

To provide a statistical argument for the expected error of the Q-function, consider learning $\hat{Q}^\beta$ with a tabular representation. The Q-function is the expected value of the random variable $z^\pi(s,a) = \sum_{k\geq 0} \gamma^k r_k | s, a, \pi$. Therefore, the Standard Error (SE) of an approximation $\hat{Q}^\beta(s,a)$ for the Q-value with N i.i.d. MC trajectories is

$$\sigma_{\varepsilon(s,a)} = \frac{\sigma_{z(s,a)}}{\sqrt{N_s \beta(a|s)}}, \tag{1}$$

where N_s is the number of visitations in state s in $\mathcal{D}$, s.t. $N = \beta(a|s) N_s$. Therefore, $\sigma_{\varepsilon(s,a)} \propto \frac{1}{\sqrt{\beta(a|s)}}$ and specifically for low frequency actions such estimation may suffer large SE.[2] Notice that we ignore recurrent visitations to the same state during the same episode and hence, the MC discounted sum of rewards are independent random variables.

2.3 Policy Improvement in the Presence of Value Estimation Errors

We now turn to the crucial question of what happens when one applies an improvement step with respect to an inaccurate estimation of the Q-function, i.e. $\hat{Q}^\beta$.

Lemma 2.2 (Improvement Penalty). *Let $\hat{Q}^\beta = \hat{V}^\beta + \hat{A}^\beta$ be an estimator of Q^β with an error $\varepsilon(s,a) = (Q^\beta - \hat{Q}^\beta)(s,a)$ and let π be a policy that satisfies lemma 2.1 with respect to $\hat{A}^\beta$. Then the following holds*

$$V^\pi(s) - V^\beta(s) \geq -\mathcal{E}(s) = -\sum_{s' \in \mathcal{S}} \rho^\pi(s'|s) \sum_{a \in \mathcal{A}} \varepsilon(s', a) \left(\beta(a|s') - \pi(a|s')\right), \tag{2}$$

where $\mathcal{E}(s)$ is called the improvement penalty and $\rho^\pi(s'|s) = \sum_{k\geq 0} \gamma^k P(s \xrightarrow{k} s'|\pi))$ is the unnormalized discounted state distribution induced by policy π.

Since $\varepsilon(s', a)$ is a random variable, it is worth to consider the variance of $\mathcal{E}(s)$. Define each element in the sum of Eq. (2) as $x(s', a; s) = \rho^\pi(s'|s)\varepsilon(s,a)(\beta(a|s') - \pi(a|s'))$. The variance of each element is therefore

$$\sigma^2_{x(s',a;s)} = (\rho^\pi(s'|s))^2 \sigma^2_{\varepsilon(s',a)} (\beta(a|s') - \pi(a|s'))^2 = \frac{(\rho^\pi(s'|s))^2 \sigma^2_{z(s',a)}}{N_{s'}} \frac{(\beta(a|s') - \pi(a|s'))^2}{\beta(a|s')}.$$

To see the the need for the reroute constraint, we can bound the total variance of the improvement penalty

$$\sum_{s',a} \sigma^2_{x(s',a;s)} \leq \sigma^2_{\mathcal{E}(s)} \leq \sum_{s',a,s'',a'} \sqrt{\sigma^2_{x(s',a;s)} \sigma^2_{x(s'',a';s)}},$$

where the upper bound is due to the Cauchy-Schwarz inequality, and the lower bound is since $\varepsilon(s,a)$ elements have a positive correlation (as reward trajectories overlap). Hence, it is evident that the improvement penalty can be extremely large when the term $\frac{|\beta - \pi|^2}{\beta}$ is unregulated and even a single mistake along the trajectory, caused by an unregulated element, might wreck the performance of the entire policy. However, by using the reroute constraint which tame each of these terms we can bound the variance of the improvement penalty.

While we analyzed the error for independent MC trajectories, a similar argument holds also for Temporal Difference (TD) learning [16]. [8] studied "bias-variance" terms in k-steps TD learning of the value function. Here we present their results for the Q-function error with TD updates. For any $0 < \delta < 1$, and a number t of iteration through the data for the TD calculation, the maximal error term abides

$$\varepsilon(s,a) \leq \max_{s,a} |\hat{Q}^\beta(s,a) - Q^\beta(s,a)| \leq \frac{1-\gamma^{kt}}{1-\gamma} \sqrt{\frac{3\log(k/\delta)}{N_s \beta(a|s)}} + \gamma^{kt}. \tag{3}$$

While the "bias", which is the second term in (3), depends on the number of iterations through the dataset, the "variance" which is the square root of the first term in (3) is proportional to $\frac{1}{\beta(a|s)N_s}$, therefore, bounding the ratio $\frac{|\beta-\pi|^2}{\beta}$ bounds the improvement penalty also for TD learning.

[2]Note that even for deterministic environments, a stochastic policy inevitably provides $\sigma_{z(s,a)} > 0$.

2.4 The Reroute Constraint

In order to confine the ratio $\frac{|\beta-\pi|^2}{\beta}$, we suggest limiting the improvement step to a set of policies based on the following constraint.

Definition 2.1 (Reroute Constraint)**.** Given a policy β, a policy π is a $reroute(c_{\min}, c_{\max})$ of β, if $\pi(a|s) = c(s,a)\beta(a|s)$ where $c(s,a) \in [c_{\min}, c_{\max}]$. Further, note that reroute is a subset of the TV constraint with $\delta = \min(1 - c_{\min}, \max(\frac{c_{\max}-1}{2}, \frac{1-c_{\min}}{2}))$.

With reroute, each element in the sum of (2.2) is proportional to $\sqrt{\beta(a|s)}|1 - c(s,a)|$ where $c(s,a) \in [c_{\min}, c_{\max}]$. Unlike reroute, other constraints such as the Total Variation (TV), forward and backward KL and PPO were not design to bound the improvement penalty.

2.5 Maximizing the Improvement Step under the Reroute Constraint

We now turn to the problem of maximizing the objective function $J(\pi)$ under the reroute constraint and whether such maximization yields a positive improvement step. Maximizing the objective function without generating new trajectories of π is a hard task since the distribution of states induced by the policy π is unknown. Therefore, usually we maximize a surrogate off-policy objective function $J^{OP}(\pi) = \mathbb{E}_{s\sim\beta}[\sum_a \pi(a|s)A^{\beta}(s,a)]$. It is common to solve the constrained maximization with a NN policy representation and a policy gradient approach [15, 11]. Here we suggest an alternative: instead of optimizing a parametrized policy that maximizes J^{OP}, the actor (i.e. the agent that interact with the MDP environment) may ad hoc calculate a non-parametrized policy that maximizes the improvement step $\sum_a \pi(a|s)A^{\beta}(s,a)$ (i.e. the argument of the J^{OP} objective) for each different state. This method maximizes also the J^{OP} objective since the improvement step is independent between states. Note that with an ad hoc maximization, the executed policy is guaranteed to maximize the objective function under the constraint whereas with policy gradient methods one must hope that the optimized policy avoided NN caveats such as overfitting or local minima and converged to the optimal policy.

For the reroute constraint, solving the non-parametrized problem amounts to solving the following simple linear program for each state

$$\begin{aligned} \text{Maximize: } & (\boldsymbol{A}^{\beta})^T \boldsymbol{\pi} \\ \text{Subject to: } & c_{\min}\boldsymbol{\beta} \le \boldsymbol{\pi} \le c_{\max}\boldsymbol{\beta} \\ \text{And: } & \sum \pi_i = 1. \end{aligned} \tag{4}$$

Where $\boldsymbol{\pi}$, $\boldsymbol{\beta}$ and $\boldsymbol{A}^{\beta}$ are vector representations of $(\pi(a_i|s))_{i=1}^{|\mathcal{A}|}$, $(\beta(a_i|s))_{i=1}^{|\mathcal{A}|}$ and $(A^{\beta}(s,a))_{i=1}^{|\mathcal{A}|}$ respectively. We term the algorithm that solves this maximization problem as Max-Reroute. Similarly, one may derive other algorithms that maximize other constraints.

Notice that Max-Reroute satisfies the conditions of Corollary 2.1.1, therefore it always provides a positive improvement step and hence, at least for a perfect approximation of Q^{β} it is guaranteed to improve the performance. In addition, notice that Max-Reroute uses only the action ranking information in order to calculate the optimized policy. We postulate that this trait makes it more resilient to value estimation errors. This is in contrast to policy gradient methods which optimize the policy according to the magnitude of the advantage function.

3 Two-armed bandit with Gaussian distributed rewards

To gain some insight into the nature of the RBI step, we examine it in a simplified model of a two-armed bandit with Gaussian distributed rewards [9]. To that end, define the reward of taking action a_i as $r_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$ and denote action a_2 as the optimal action s.t. $\mu_2 \ge \mu_1$. Consider the learning curve of off-policy learning where a behavior policy is mixed with a fix exploration parameter, i.e. $\beta(a) = \pi(a)(1-\varepsilon) + \frac{\varepsilon}{n_a}$ (where n_a is the number of actions and $\varepsilon = 0.1$). The Q-function is learned with $Q^{\pi}(a) = (1-\alpha)Q^{\pi}(a) + \alpha r$, where α is a learning rate, possibly decaying over time. We evaluate several constrained policies: (1) RBI with $(c_{\min}, c_{\max}) = (0.5, 1.5)$, (2) PPO with $\varepsilon = 0.5$, (3) TV with $\delta = 0.25$, (4) greedy step and; (5) forward KL with $\lambda = 1$. RBI, TV and PPO were all maximized with our maximization algorithms (without gradient ascent). To avoid absolute zero probability actions, we clipped the policy such that $\pi(a_i) \ge 10^{-3}$. In addition we added 10 random sample at the start of the learning process. The learning curves are plotted in Figure 1.

The learning curves exhibit two different patterns. For the scenario of $\sigma_1 > \sigma_2$, a fast convergence of all policies was obtained. Essentially, when the better action has low variance it is easy to discard the worse action by choosing it and rapidly improving its value estimation and then switching to the better action. On the other hand, for the case of $\sigma_1 < \sigma_2$ it is much harder for the policy to improve the estimation of the better action after committing to the worse action. We see that RBI defers early commitment and while it slightly reduces the rate of convergence in the first (and easy) scenario, it significantly increases the data efficiency in the harder scenario.

In the second scenario, RBI has the best and KL has the second-best learning curves in terms of initial performance. However, there is another distinction between the ideal learning rate (LR) of $\alpha = \frac{1}{n}$ and a constant rate of $\alpha = 0.01$. In the ideal LR case, the advantage of RBI and KL reduces over time. This is obvious since a LR of $\alpha = \frac{1}{n}$ takes into account the entire history and as such, for large history, after a large number of iterations, there is no need for a policy which learns well from a finite dataset. On the other hand, there is a stable advantage of RBI and KL for a fix LR as fix LR does not correctly weight the entire past experience. Notice that in a larger than 1-step MDP, it is unusual to use a LR of $\frac{1}{n}$ since the policy changes as the learning progress, therefore, usually the LR is fixed or decays over time (but not over state visitations). Hence, RBI has a positive advantage over greedy policies through the entire training process.

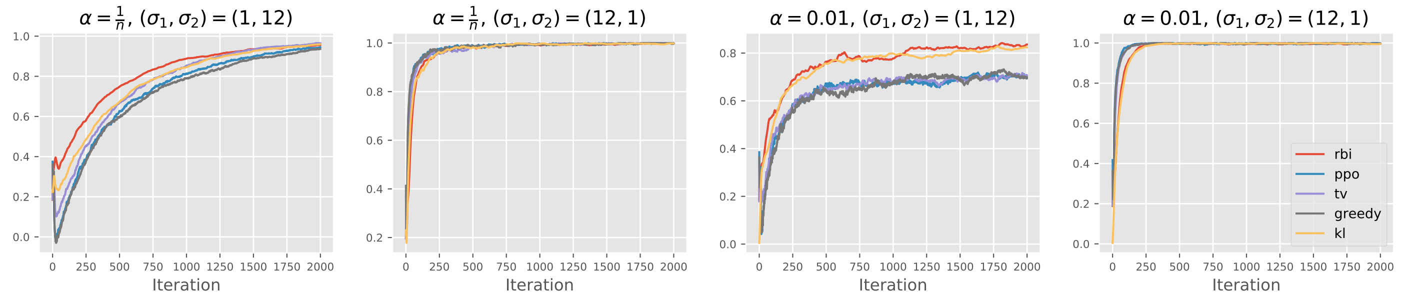

Figure 1: Different constrained policies performances in a two-armed bandit with Gaussian distributed reward setting

4 Experiments in the Atari environment

We implemented an RBI learning agent in the Atari Learning Environment. We adopted a distributed learning setting, similar to the setting of Ape-X [6]. In this experiment we set out to verify: (1) whether RBI is a good approach for Deep RL in terms of better final performance and (2) whether our approach of solving the optimal policy in the non-parametrized space as part of the actor's routine, can be generalized to iterative Deep RL.

To that end, we designed an actor that fetches a stored parametrized policy π_{θ_k} and Q-function $Q^{\pi}_{\phi_k}$. The actor solves the non-parametrized reroute constrained optimization problem (Eq. (4)) and generates an optimized constrained policy π. Our centralized learner imitates the actor's policy with a parametrized policy $\pi_{\theta_{k+1}}$ by minimizing a KL divergence loss $D_{KL}(\pi, \pi_{\theta_{k+1}})$. The learner keeps track of the history Q-function by minimizing the Huber loss $\mathcal{L}(Q^{\pi}_{\phi_{k+1}} - R)$, where R is an n-steps target value $R(s,a) = \sum_{k=0}^{n-1} \gamma^k r_k + \gamma^n \sum_{a'} \pi(s',a') Q^{\pi}_{\bar{\phi}}(s',a')$. Performance curves of 4 Atari games are presented in figure 3 and compared to our Ape-X algorithm implementation. The initial results demonstrate the benefit of using RBI as an efficient general RL learning algorithm.

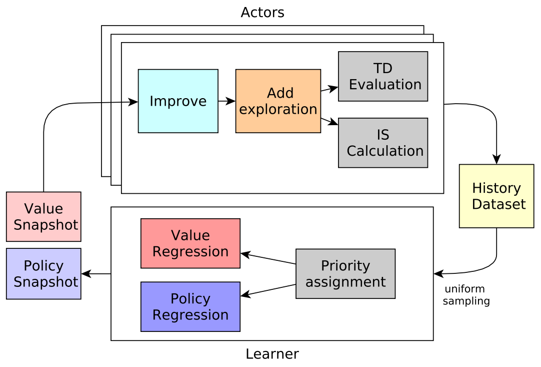

Figure 2: RBI in a distributed RL setting

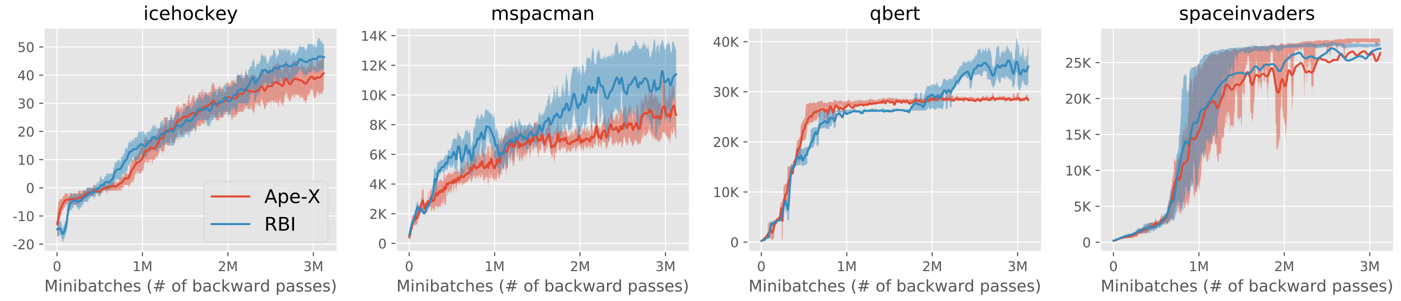

Figure 3: Performance curves of 4 Atari games. The second and third quartiles are shadowed.

References

[1] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[2] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[3] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, 2016.

[4] Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[5] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. *arXiv preprint arXiv:1710.02298*, 2017.

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

[7] Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *ICML*, volume 2, pages 267–274, 2002.

[8] Michael J Kearns and Satinder P Singh. Bias-variance error bounds for temporal difference updates. In *COLT*, pages 142–147. Citeseer, 2000.

[9] Andreas Krause and Cheng S Ong. Contextual gaussian process bandit optimization. In *Advances in Neural Information Processing Systems*, pages 2447–2455, 2011.

[10] OpenAI. Openai five, Jul 2018.

[11] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.

[12] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[13] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*, 2016.

[14] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[15] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[16] RS Sutton and AG Barto. Reinforcement learning: An introduction, (complete draft), 2017.

[17] Philip Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. High confidence policy improvement. In *International Conference on Machine Learning*, pages 2380–2388, 2015.

[18] Quan Vuong, Yiming Zhang, and Keith W Ross. Supervised policy update for deep reinforcement learning. 2018.

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