Rethinking Deep Policy Gradients via State-Wise Policy Improvement

Kai-Chun Hu
Department of Applied Mathematics
National Chiao Tung University
Hsinchu, Taiwan
hukaichun.am05g@nctu.edu.tw

Ping-Chun Hsieh
Department of Computer Science
National Chiao Tung University
Hsinchu, Taiwan
pinghsieh@nctu.edu.tw

Ting-Han Wei
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada
tinghan@ualberta.ca

I-Chen Wu
Department of Computer Science
National Chiao Tung University
Hsinchu, Taiwan
icwu@csie.nctu.edu.tw

Abstract

Deep policy gradient is one of the major frameworks in reinforcement learning, and it has been shown to improve parameterized policies across various tasks and environments. However, recent studies show that the key components of the deep policy gradient methods, such as gradient estimation, value prediction, and optimization landscapes, fail to reflect the conceptual framework. This paper aims to investigate the mechanism behind the deep policy gradient methods through the lens of state-wise policy improvement. Based on the fundamental properties of policy improvement, we propose an alternative theoretical framework to reinterpret the deep policy gradient update as training a binary classifier, with labels provided by the advantage function. This framework obviates the statistical difficulties in the gradient estimates and predicted values of the deep policy gradient update. Experimental results are included to corroborate the proposed framework.

1 Introduction

Reinforcement learning (RL) is a branch of machine learning that achieves optimal sequential decision making by interacting with the environment and learning from the underlying random process. RL is often considered to be an optimization problem, and the major approaches that follow this principle are the *policy gradient methods* [8, 22, 15, 23]. The foundation of the policy gradient methods is built upon the theory of REINFORCE [30] or its equivalent representation called the policy gradient theorem [26]. The existing RL algorithms built upon the policy gradient theorem were often unable to evaluate the policy gradient by its original formula, and therefore an alternative statistical representation was introduced by leveraging the idea of discounted state visitation distribution. This statistical representation of the policy gradient is often considered as a variant of stochastic optimization, which is typically appraised by the update noise in the measurements regarding the objective. In the context of policy gradient, this noise corresponds to the joint effect of the error in the predicted value function and the variance of the gradient estimate.

Despite the great success of the policy gradient methods, their theoretical foundation is currently being challenged. For example, Thomas [27] pointed out that there is a missing term in the standard update rule of the value function approximator, and this term is required to reflect the discounted state visitation distribution in the statistical representation of the policy gradient. The fact that the default practice is missing this term leads to additional bias in the gradient estimates and could affect the convergence guarantees. Subsequently, Nota and Thomas [17] showed that many of the existing estimates of the policy gradient do not follow the gradient direction of any function due to this missing term, and therefore these methods are not guaranteed to converge to a reasonable stationary point. On the other hand, Ilyas et al. [7] empirically studied the value prediction error and the quality of the gradient estimates for the benchmark algorithms TRPO [22] and its simplified variant PPO with a clipped objective [23], and the results showed that the value prediction accuracy was poor and the gradient estimates suffered from rather high variance. Surprisingly, both algorithms TRPO and PPO can still improve the policy during the training process by operating under such restrictions. This mismatch suggests a deeper look into the theoretical justification of the deep policy gradients.

This paper is meant to take the first step toward understanding the reason behind this mismatch between theory and practice in the deep policy gradient methods. To address this phenomenon, we propose an alternative theoretical framework to reinterpret the deep policy gradient methods through the lens of *state-wise policy improvement*. Specifically, the main contributions of this paper are:

- We identify two fundamental results of state-wise policy improvement, which provide useful sufficient conditions for ensuring policy improvement. Based on these conditions, we reinterpret the deep policy gradient update as training a binary classifier via empirical risk minimization. Through the lens of state-wise policy improvement, the sample distributions in the policy gradient are thereby not required in improving a policy, and hence the inherent variance problem vanishes.
- We proceed to propose a generic classification-based policy update scheme, in which we use the sign of the advantage as the label and view the change in the action probability as the classifier. As an example, we present a practical training algorithm by leveraging a large margin classifier and the hinge loss. Under this design, the classifier attempts to meet the minimum requirement of the sufficient conditions for state-wise policy improvement by skipping those data samples that satisfy the sufficient conditions by a large margin. Through a proper rearrangement of the objective, the resulting scheme is equivalent to a variant of the PPO algorithm with a clipped objective.
- Finally, we empirically validate the proposed theoretical framework through an experiment of the classic optimal time control problem. Through a comparison with the baseline method, we demonstrate that the proposed classification-based scheme with data skipping is indeed effective in achieving steady policy improvement.

2 Preliminaries

In this section, we describe the problem setup and formally introduce the notations used in this paper. For ease of notation, we use Δ_Z to denote the set of all probability distributions over a finite support set Z. All vectors are column vectors by default, unless stated otherwise. We consider a discounted Markov Decision Process (MDP) defined by a tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$ [20], where $\mathcal{S}$ is the finite state space, $\mathcal{A}$ is the finite action space, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the Markov transition kernel, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in (0, 1)$ is the discount factor. Given an MDP described above, a policy $\pi \in \Delta_{\mathcal{A}}^{|\mathcal{S}|}$ specifies the action distributions at all states $s \in \mathcal{S}$. With a slight abuse of notation, we also use π to denote the $|S||\mathcal{A}|$-dimensional vectorized version of the policy π. The goal of an RL algorithm is to find a policy that maximizes the expected accumulated discounted rewards. For any policy π, the state value function V^π and the action value function Q^π, in terms of expected discounted accumulated rewards, are defined by

$$V^\pi(s) := \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r(s_t, a_t) \Big| s_0 = s, a_t \sim \pi, s_{t+1} \sim P(\cdot|s_t, a_t)\right], \tag{1}$$

$$Q^\pi(s, a) := \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r(s_t, a_t) \Big| s_0 = s, a_0 = a, a_t \sim \pi, s_{t+1} \sim P(\cdot|s_t, a_t)\right]. \tag{2}$$

Under this formulation, the standard definition of *state-wise policy improvement* is formalized by the following partial ordering relation.

Definition 2.1 (Partial ordering over policies). *Let π and μ be two policies. We say that π improves μ (denoted by $\pi \geq \mu$) if and only if $V^\pi(s) \geq V^\mu(s), \forall s \in \mathcal{S}$.*

A well-known result is that there exists a policy π^* that maximizes $V^\pi(s)$ over all policies, for all state $s \in \mathcal{S}$. In order to apply the numerical optimization techniques, such as gradient ascent, a natural alternative performance metric is introduced by taking a weighted average of the values at all states to construct a total ordering over policies, i.e.

$$\eta_\nu(\pi) := \sum_{s\in\mathcal{S}} \nu(s) V^\pi(s), \tag{3}$$

where $\nu \in \Delta_{\mathcal{S}}$ is some static distribution over $\mathcal{S}$. We also define the corresponding *discounted state visitation distribution* under a policy π as

$$\rho_\nu^\pi(s) := (1-\gamma)\sum_{t\geq 0}\gamma^t P(s_t = s|\pi, s_0 \sim \nu), \tag{4}$$

where $1-\gamma$ serves as a normalization factor. The theoretical foundation of the policy gradient methods, namely, the policy gradient theorem [26], is formulated as follows (in terms of statistical representation with advantage function):

$$\nabla_\theta \eta(\pi) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim\rho_\nu^{\pi_\theta}(s), a\sim\pi}\left[\left(Q^{\pi_\theta}(s,a) - b(s)\right)\frac{\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)}\right], \tag{5}$$

where $b(s)$ is called a baseline function and $\pi(a|s)$ in the denominator comes from the importance weight. One typical choice of the baseline function is $V^\pi(s)$. In this case, the baseline function is called the advantage function defined as $A^\pi(s,a) := Q^\pi(s,a) - V^\pi(s)$, for a policy π. To apply the policy gradient defined in (5), the existing deep policy gradient algorithms estimate $\nabla_\theta\eta(\pi)$ by following two major design principles as the default practice:

- The value functions, also referred to as the critic in an actor-critic architecture, can be estimated accurately by TD learning [24, 21] or its variants, such as n-step TD, TD(λ), GAE, and Retrace(λ), all of which is built on the weighted average approach [16, 29, 2].
- The policy, also referred to as the actor in an actor-critic architecture, can be trained by estimating the policy gradient in (5) via a sample average. Some variants of the sample average are: (i) [19, 1] incorporate probability distribution differences using importance sampling; (ii) [8, 22] leverage geometrical direction correction, such as Fisher information matrix and the second order approximation of Kullback-Leibler (KL) divergence; (iii) [23] introduces external penalties such as entropy or KL divergence.

The default practice presumes that the performance of an RL algorithm is mainly determined by the underlying accuracy of the estimated gradient and the estimated value functions. However, as mentioned in Section 1, these presumptions do not seem to hold. This motivates us to rethink the default practice and develop an alternative theoretical framework to better understand the underlying rationale of the deep policy gradient methods.

3 An Alternative Theoretical Framework for Deep Policy Gradients

In this section, we present a theoretical framework based on state-wise policy improvement to reinterpret the deep policy gradient methods. We start by summarizing the fundamental results of state-wise policy improvement. Based on these results, we describe the connection between state-wise policy improvement and the policy gradient methods and then present the proposed training algorithms. We close this section by presenting an experiment to justify our theory. The proofs of all the theoretical results are provided in Appendix A.

3.1 Fundamental Theorems of Policy Improvement

To begin with, we state two fundamental results that provide useful sufficient conditions for state-wise policy improvement.

Theorem 3.1. *Given any arbitrary policy μ, a policy π improves μ if π satisfies either one of the following conditions:*

$$\sum_{a\in\mathcal{A}} \pi(a|s)A^\mu(s,a) \geq 0,\ \forall s \in \mathcal{S} \tag{6}$$

$$\sum_{a\in\mathcal{A}} \mu(a|s)A^\pi(s,a) \leq 0,\ \forall s \in \mathcal{S} \tag{7}$$

It is easy to verify that both (6) and (7) automatically hold if π and μ are identical. Note that (6) can be viewed as the state-wise version of the classic performance difference lemma in [9, 22], which characterizes the difference in value function under some state distribution ν between any two policies π and μ as

$$\eta_\nu(\pi) - \eta_\nu(\mu) = \frac{1}{1-\gamma}\sum_{s\in\mathcal{S}} \rho_\nu^\pi(s) \sum_{a\in\mathcal{A}} \pi(a|s) A^\mu(s,a). \tag{8}$$

Specifically, for any state s, $V^\pi(s) - V^\mu(s)$ can be recovered from (8) by setting ν to be a Dirac measure with the atom located in state s. By interchanging the roles of π and μ in (8), the condition in (7) can be obtained by a similar argument (with its proof provided in Appendix A for completeness). Note that (8) provides a sufficient condition for policy improvement in terms of the weighted average value $\eta_\nu(\cdot)$. By contrast, (6)-(7) serve as sufficient conditions for state-wise policy improvement in the sense of Definition 2.1. In the sequel, we will show that this seemingly subtle change of perspective obviates the inherent statistical difficulty in the deep policy gradient update.

Remark 1. Various popular policy update schemes can be justified by Theorem 3.1. For example: (i) Given any policy μ, the classic one-step greedy policy improvement constructs an improved deterministic policy π by choosing $\bar{\mu}(s) = \arg\max_{a\in\mathcal{A}} Q^\mu(s,a)$, for each state s. It is easy to verify that this design satisfies (6); (ii) Moreover, by applying Theorem 3.1, one can show that the convex mixture of μ and $\bar{\mu}$ considered in [9, 28], i.e., $\alpha\bar{\mu} + (1-\alpha)\mu$, is also an improved policy over μ, for any $\alpha \in [0,1]$; (iii) By a similar argument, the one-step ϵ-greedy policy improvement also follows directly from Theorem 3.1 [25]. (iv) In the regularized policy optimization using a quadratic penalty function [5], given a policy μ, an updated policy π_{reg} is constructed by solving $\pi_{\text{reg}} = \arg\max_\pi \sum_a Q^\mu(s,a)\pi(a|s) - C\|\mu - \pi\|^2$, where C is a positive constant. Again, by verifying the conditions in Theorem 3.1, one can show that π_{reg} is an improved policy over μ. Based on the above discussion, we thereby view Theorem 3.1 as one fundamental theorem of policy improvement.

By combining Theorem 3.1 and the fact that $\sum_{a\in\mathcal{A}} \pi(a|s)A^\pi(s,a) = 0$ for any policy π and for any state s, we present alternative sufficient conditions that are checked on a per state-action-pair basis.

Theorem 3.2. *Given an arbitrary policy μ, a policy π improves μ if it satisfies either one of the following conditions: (i) $(\pi(a|s) - \mu(a|s))A^\pi(s,a) \geq 0$, $\forall s \in \mathcal{S}, \forall a \in \mathcal{A}$; (ii) $(\pi(a|s) - \mu(a|s))A^\mu(s,a) \geq 0$, $\forall s \in \mathcal{S}, \forall a \in \mathcal{A}$.*

Theorem 3.2 indicates that policy improvement can be achieved by determining the sign of the advantage of each state-action pair (regardless of its magnitude) and adjusting the action probabilities accordingly. In this way, no additional constraints, such as the KL divergence constraint used in TRPO [22], are needed to ensure policy improvement. This also naturally motivates the design of using the signs of the advantage function as labels in determining the direction of the policy update. More specifically, we can draw an analogy between the conditions in Theorem 3.2 and the training of a linear classifier: (i) The state-action pair serves as the feature vector of a training sample; (ii) The sign of $A^\pi(s,a)$ or $A^\mu(s,a)$ plays the role of a binary label; (iii) $\pi(a|s) - \mu(a|s)$ resembles the prediction of a linear classifier. In Section 3.3, we discuss the practical issues in achieving policy improvement via classification and present an exemplary training algorithm.

3.2 Connecting Deep Policy Gradients With State-Wise Policy Improvement

Before stating the proposed training algorithms, we first highlight the statistical issue inherent in the deep policy gradient methods and then connect Theorem 3.2 with the deep policy gradients.

Difficulty in estimating deep policy gradients. Here we consider parameterized policies π_θ, where θ denotes the parameters of the underlying parametric model (e.g. a neural network). Recall from (5) that the policy gradient of a parameterized policy π_θ can be explicitly characterized by

$$\nabla_\theta \eta_\nu(\pi_\theta) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim\rho_\nu^{\pi_\theta}, a\sim\pi_\theta}\left[\frac{\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)} A^{\pi_\theta}(s,a)\right] \tag{9}$$

$$\approx \frac{1}{(1-\gamma)|\mathcal{B}|}\sum_{(s,a)\in\mathcal{B}} \frac{\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)} A^{\pi_\theta}(s,a), \tag{10}$$

where $1/\pi_\theta(a|s)$ is the importance weight and $\mathcal{B}$ denotes a mini-batch of samples drawn under π_θ. While most of the deep policy gradient methods are built on the mini-batch estimate of the policy gradient, the gradient $\nabla_\theta \eta_\nu(\pi_\theta)$ is fundamentally difficult to estimate via sample average, as shown by the gradient estimate quality in [7]. Furthermore, [7] also found that the deep policy gradient methods can work under a small number of training samples (roughly 2×10^3), in which most of state-action pairs in the training set are sampled only at most once. Therefore, the importance weight of each state-action pair can hardly take effect in correcting the distribution mismatch, which is inherent in the default practice of the deep policy gradient methods.

Remark 2. The theory of TRPO considers a gradient update scheme that is slightly different from (10) by leveraging a surrogate objective adapted from (8). To be more specific, TRPO introduces a surrogate objective by approximating $\rho_\nu^\pi(s)$ by $\rho_\nu^\mu(s)$ in (8), under the condition that π is sufficiently close to μ. However, the gradient of the surrogate objective is still fundamentally difficult to estimate via sample average and suffers from the same variance issue as (10), as shown by the surrogate objective landscape in [7].

Interpreting deep policy gradients via state-wise policy improvement. Given the above gap between theory and the default practice, we propose to reinterpret the deep policy gradient methods through the lens of state-wise policy improvement. We start by leveraging the fact that $\sum_{a\in\mathcal{A}} \mu(a|s)A^\mu(s,a) = 0$ and rewriting (8) as

$$\eta_\nu(\pi) - \eta_\nu(\mu) = \frac{1}{1-\gamma} \sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} \rho_\nu^\pi(s) A^\mu(s,a) \left(\pi(a|s) - \mu(a|s)\right). \tag{11}$$

Note that (11) holds without any approximation. Moreover, a policy π that satisfies the sufficient conditions in Theorem 3.2 for state-wise policy improvement shall also improve μ in terms of the weighted-average value $\eta_\nu(\cdot)$, as shown in (11). More importantly, we can reinterpret the sample average in (10) with the help of Theorem 3.2 as follows: This sample average can be viewed as a weighted sum of $\nabla_\theta \pi_\theta(a|s) A^{\pi_{\theta_0}}(s,a)$ across the data samples in the mini-batch. For each sampled state-action pair (s,a), $\nabla_\theta \pi_\theta(a|s) A^{\pi_{\theta_0}}(s,a)$ serves as an improvement direction for this specific pair (s,a), and this improvement direction depends on the sign of $A^{\pi_{\theta_0}}(s,a)$ but not on the magnitude of the advantage. Therefore, the deep policy gradient approach using sample average can be viewed as the construct of state-wise policy improvement. We thereby draw an analogy between the deep policy gradient in (10) and the training of classic empirical risk minimization problems for classification: (i) The state-action pair (s,a) plays the role of a feature vector; (ii) Each $\nabla_\theta \pi_\theta(a|s) A^{\pi_{\theta_0}}(s,a)$ is analogous to the gradient of the loss incurred by a training sample; (iii) $A^{\pi_{\theta_0}}(s,a)$ plays the role of the label of a training sample. Hence, similar to the empirical risk minimization, what matters the most in the deep policy gradient update is the training data. This interpretation suggests that training a classifier based on Theorem 3.2 is a viable way to improve a policy in terms of $\eta_\nu(\cdot)$.

3.3 Deep Policy Gradient as a Binary Classification Problem

In this section, we expand on the insight provided by Theorem 3.2, which suggests that policy improvement can be achieved by training a classifier based on the advantage. We use $\hat{A}^\mu(s,a)$ to denote the estimated advantage of a policy μ for the state-action pair (s,a).

A prototypical linear classification method for policy update. One prototypical method is to use the sign of $\hat{A}^\mu(s,a)$ as the binary label, $\pi(a|s) - \mu(a|s)$ as the prediction of the linear classifier, and $-(\pi(a|s) - \mu(a|s))\hat{A}^\mu(s,a)$ as the loss incurred by each training sample. At each training step, given a batch of samples $\mathcal{D}$, the policy is updated by minimizing the loss $\sum_{(s,a)\in\mathcal{D}}(\mu(a|s) - \pi(a|s))\hat{A}^\mu(s,a)$. Compared to the typical sample-average-based deep policy gradient update in (10), one major benefit of this classification-based update scheme is that the error of this scheme only comes from the estimation error of the advantage function, and does not involve the sample distribution of the state-action pair. Hence, the classification-based approach completely obviates the statistical difficulty resulting from the sample average in the default practice of deep policy gradients.

Compared to the conventional classification problem, one salient feature of this classification-based policy update is that it needs to produce valid action distributions, i.e. an increase in the probability of a certain action implies the simultaneous decrease in the probabilities of the others. If there is a training sample with a particularly large $(\pi(a|s) - \mu(a|s))\hat{A}^\mu(s,a)$, then it is relatively easy to construct a counterexample with degraded performance. Moreover, this issue can be quite severe as the limited amount of collected data from the environment in each training step will most likely be unrepresentative of the problem space (e.g., only a single action per visited state). For the above reasons, a more careful design of the training algorithm is needed.

An exemplary training algorithm with data skipping. To address the above issue, we propose to consider a different loss function for the linear classifier. Inspired by [4], we formulate the policy improvement objective based on the large margin classification and the hinge loss, i.e.,

$$\min_{\theta} \sum_{(s_i,a_i)\in\mathcal{D}} \max\left\{0, \epsilon - \hat{A}^{\mu}(s_i,a_i)(\pi(a_i|s_i) - \mu(a_i|s_i))\right\}, \tag{12}$$

where $\epsilon > 0$ denotes the margin. By definition, the hinge loss does not take into consideration the samples that satisfy

$$\hat{A}^{\mu}(s,a)(\pi(a|s) - \mu(a|s)) \geq \epsilon. \tag{13}$$

In other words, the minimizer of the objective in (12) attempts to meet the minimum requirement of Theorem 3.2 for the training set $\mathcal{D}$ by skipping those data samples that satisfy the sufficient condition in Theorem 3.2 by a large margin.

Connecting PPO and the proposed training algorithm. It is worth noting that if the classifier is set to be $\frac{\pi}{\mu} - 1$ and the margin set to be $\epsilon \mapsto \epsilon|\hat{A}(s_i,a_i)|$, the resulting update scheme is equivalent to the PPO-clip objective and was proved by Pi et al. [18], which is independent of the sampling distribution and the choice of regularizer. This property also provides a theoretical basis that can explain the algorithms proposed by Han and Sung [6], Luo et al. [14], where they combined IMPALA [2] with PPO-clip. Theoretically, the sufficient conditions in Theorem 3.2 can be applied on top of an arbitrary smooth monotone increasing function $\mathcal{F}$ to construct an objective function as

$$\min_{\theta} \sum_{(s_i,a_i)\in\mathcal{D}} \max\left\{0, \epsilon - \hat{A}(s_i,a_i)(\mathcal{F}(\pi(a_i|s_i)) - \mathcal{F}(\mu(a_i|s_i))\right\}. \tag{14}$$

The empirical performance of the hinge-loss-based objective has been studied by Schulman et al. [23], Han and Sung [6], Luo et al. [14], Pi et al. [18]. The theoretical basis is now provided by Theorem 3.2.

Remark 3. The idea of casting RL as a classification problem has been investigated by [11, 12], which view the one-step greedy policy update (e.g. in Q-learning) as a binary classification problem. Despite the high-level resemblance, this paper is fundamentally different from the prior works [11, 12] as this paper is meant to study the theoretical foundation of the deep policy gradient methods, from the perspective of state-wise policy improvement.

3.4 Labeling Quality

In this section, we discuss the effect of the estimated advantage function on the labelling quality of the proposed classification-based scheme. One fundamental challenge is that the quantification of the accuracy of the estimated value functions with non-linear function approximation remains an open problem. Despite this, since (12) is a binary classifier, the component that matters the most is the sign of estimated advantage for training this classifier. This feature allows us to further study the interplay between the noisy estimation of the advantage function and its corresponding labels for the classifier.

To begin with, we present a necessary and sufficient condition for the correctness of the labels in the following theorem. Here we consider a more general baseline function denoted by $\phi(s,a)$. As the correctness of the label is a state-wise property, to make the notations uncluttered, we drop the inputs s and a from the value functions. Let $\hat{Q}$ be the estimated action value function. With a slight abuse of notation, we define $\hat{A} := \hat{Q} - \phi$ to be the advantage function under the more general baseline ϕ. Under a policy π, we also define $\Delta Q := \hat{Q} - Q^{\pi}$, $\Delta V = \phi - V^{\pi}$, and $\Delta A := \Delta Q - \Delta V = (\hat{Q} - Q^{\pi}) - (\phi - V^{\pi})$.

Theorem 3.3. *Given a policy π, for any baseline function ϕ and any state-action pair, the label is correct if and only if*

$$|\hat{A} - \frac{1}{2}\Delta A| > \frac{1}{2}|\Delta A|. \tag{15}$$

As the error term is caused by $\Delta Q - \Delta V$, the worst-case error is $\max|\Delta Q| + \max|\Delta V|$. Note that if $\hat{Q}$ and the baseline ϕ are correlated, the error bound may be further reduced. The following corollary is an example.

Corollary 3.4. *For a deterministic transition environment, assume that V^π and $\hat{V}$ are locally Lipchitz continuous, i.e.,*

$$|V^\pi(s_{t+1}) - V^\pi(s_t)| \leq L\|s_{t+1} - s_t\| \text{ and } |\hat{V}(s_{t+1}) - \hat{V}(s_t)| \leq \hat{L}\|s_{t+1} - s_t\|, \tag{16}$$

and that the function approximation error is bounded by ϵ. Then, if $\hat{A}$ is estimated by TD

$$\hat{A} = r_t + \gamma \hat{V}(s_{t+1}) - \hat{V}(s_t) \Rightarrow |\Delta A| \leq \gamma \left(L + \hat{L}\right) \|s_{t+1} - s_t\| + (1-\gamma)\epsilon. \tag{17}$$

Remark 4. Given the way TD-learning is formulated, an interesting question is why TD updates work despite the inaccurate value function approximation. Corollary 3.4 answers this question as follows. We first postulate that given a sufficiently small state change, $\max|\Delta Q| + \max|\Delta V| = (1+\gamma)\epsilon$ (derivation provided in Appendix A). Since the value function error is represented by the term ϵ, we can see from (17) that with γ close to 1, $|\Delta A|$ is dominated by L, $\hat{L}$, and $\|s_{t+1} - s_t\|$, where L is determined by the reward design, $\hat{L}$ is determined by the value function approximator, and $\|s_{t+1} - s_t\|$ is determined by the feature design. In other words, these three factors dictate how successfully the TD update performs, rather than the accuracy of the value function approximators.

3.5 Experimental Justification

In this section, we empirically investigate the rationale of the proposed classification-based scheme. In the experiment, we use IMPALA [2] as the baseline. For comparison, to illustrate the proposed scheme, we update the policy based on the minimum requirement of Theorem 3.2, under which we only need to determine the sign of the advantage. We ignore the magnitude of the advantage and incorporate the data-skipping mechanism into the finial objective. In the standard off-policy setting, the policy gradient is estimated by

$$\hat{A}(s,a)\frac{\nabla\pi(a|s)}{\mu(a|s)}, \tag{18}$$

where μ is the behavior policy used for collecting data. We use incorrect weights for importance sampling by replacing the behavior policy μ with the current policy π. The resulting objective is

$$\min_\theta \sum_{(s_i,a_i)\in\mathcal{D}} \max\left\{0, \epsilon - \frac{\hat{A}(s_i,a_i)}{|\hat{A}(s_i,a_i)|}\big(\log\pi(a_i|s_i) - \log\mu(a_i|s_i)\big)\right\}. \tag{19}$$

Keep in mind that according to Theorem 3.2, the only difference between (19) and IMPALA in this experiment is that we ignore specific cases in the training data while the baseline IMPALA algorithm does not. Furthermore, it is not possible for the objective (19) to approximate the policy gradient theorem due to the incorrect importance weights and the incorrect advantage estimation. The details of the used algorithms as well as the environment configurations are provided in Appendix B.

In order to avoid the problem of hyper-parameter dependence, the target environment needs to be kept as simple as passable. Therefore, we use a sample problem proposed by Evans [3] involving a railroad car powered by rocket engines on either side. Our goal is to move the car from an arbitrary initial position and ultimately have the car stop in a target position. If we attempt to complete this task in minimal time, intuitively, the strategy would be to choose maximum output in the target direction to get maximum acceleration, then at some critical point in time switch to maximum deceleration so that the car stops right at the target. This problem is also referred to as the optimal time control problem, whose model follows the following equation of motion:

$$m\ddot{x}(t) = a(t), \tag{20}$$

where $x(t)$ is the position of the car at time t, m is the mass of the car, and $a(t) \in [-1,1]$ is the total thrust generated by the rocket engines.

To apply the learning-based approach to this control problem, the dynamical system can be approximated by the forward Euler method [13]

$$s_{n+1} = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix} s_n + \Delta t \frac{a_n}{m}\begin{pmatrix} 0 \\ 1 \end{pmatrix}, \tag{21}$$

where $s = (x,v)^T$, $a_n \in [-1,1]$ is the action output at step n. We define the reward (or in this case, the cost) to be the negative distance to the target position, $r(s_n,a_n) = -|x_{n+1}|$, which allows us to apply RL to the problem.

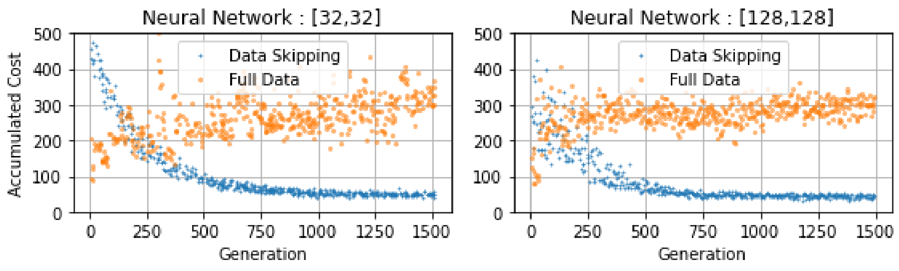

Figure 1: Results for the railroad car experiment. For the legend, "full data" refers to the standard policy gradient scheme (IMPALA), while "data skipping" refers to the scheme that updates the policy using the hinge loss and the sign of the advantage as the label. $[m, n]$ in the subfigure titles represents the width of each hidden layer (for a total of 2 layers). Each data point is the average of 64 trials, where each trial starts from a random initial position and the accumulated cost is summed over an episode consisting of 200 time steps. Each update consists of 64 most recent and 128 sampled gradients in the replay buffer, and each sample consists of 1 episode. All learning rates were initially set to 0.001 and adjusted by the Adam optimizer.

The support of the Gaussian policy is the entire real line, while the feasible action belongs to the interval $[-1, 1]$, i.e., the sampled action is truncated. To avoid over-representation of the sampled actions at the interval boundaries, the mean $\hat{a}$ and σ of the Gaussian policy must be controlled. Additional modifications are summarized as follows

$$\pi(a|s) = \mathcal{N}(\hat{a}(s), \sigma),\ \hat{a}(s) = \sin(NN(s;\theta)),\ \sigma \in [0.1, 0.5], \tag{22}$$

where $NN(s;\theta)$ is a neural network parameterized by θ. Since function approximation is used, the training set needs to be distributed over the entire range of the states of interest. Therefore, in our experiment the initial state is randomly initialized over this range.

The result in Figure 1 shows that the scheme of the standard policy gradient obtains a low total cost at early training stages, and thereafter the distribution of accumulated costs becomes widely varied. In contrast, the scheme with data skipping has a high accumulated cost initially, but its cost monotonically drops to a low value that shows stable performance improvement. Since the only difference between the two methods is data skipping, this experiment clearly shows that the performance of the baseline is poor due to training with undesired training data instances and that by skipping these instances, the performance can be steadily improved.

4 Conclusion

This paper re-investigates deep policy gradient methods via state-wise policy improvement, and attempts to bridge the gap between theory and practice. We present two fundamental policy improvement theorems, and based on these theorems we show the policy gradient methods are equivalent to training a binary classifier. Moreover, we propose a generic classification-based policy update scheme with data skipping and identify that the PPO-clip objective is equivalent to training a perceptron classifier. Finally, we conduct an experiment to validate the proposed theoretical framework.

Acknowledgement

This research is partially supported by the Ministry of Science and Technology (MOST) under Grant Number MOST 109-2634-F-009-019 through Pervasive Artificial Intelligence Research (PAIR) Labs, Taiwan. This material is also based upon work supported in part by the Ministry of Science and Technology of Taiwan under Contract No. MOST 108-2636-E-009-014.

References

[1] T. Degris, M. White, and R. S. Sutton. Linear off-policy actor-critic. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress, 2012.

[2] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, S. Legg, and K. Kavukcuoglu. IMPALA: scalable distributed deep-rl

with importance weighted actor-learner architectures. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80, pages 1406–1415. PMLR, 2018.

[3] L. C. Evans. An introduction to mathematical optimal control theory version 0.2, 1983. URL `http://math.berkeley.edu/~evans/control.course.pdf`.

[4] Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999.

[5] M. Geist, B. Scherrer, and O. Pietquin. A theory of regularized markov decision processes. In *International Conference on Machine Learning*, pages 2160–2169, 2019.

[6] S. Han and Y. Sung. Dimension-wise importance sampling weight clipping for sample-efficient reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97, pages 2586–2595. PMLR, 2019.

[7] A. Ilyas, L. Engstrom, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph, and A. Madry. A closer look at deep policy gradients. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

[8] S. M. Kakade. A natural policy gradient. In *Advances in Neural Information Processing Systems 14 , NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada*, pages 1531–1538. MIT Press, 2001.

[9] S. M. Kakade and J. Langford. Approximately optimal approximate reinforcement learning. In C. Sammut and A. G. Hoffmann, editors, *Machine Learning, Proceedings of the Nineteenth International Conference (ICML 2002), University of New South Wales, Sydney, Australia, July 8-12, 2002*, pages 267–274. Morgan Kaufmann, 2002.

[10] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[11] M. G. Lagoudakis and R. Parr. Reinforcement learning as classification: Leveraging modern classifiers. In T. Fawcett and N. Mishra, editors, *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 424–431. AAAI Press, 2003.

[12] A. Lazaric, M. Ghavamzadeh, and R. Munos. Analysis of a classification-based policy iteration algorithm. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 607–614, 2010.

[13] R. J. LeVeque. *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*, volume 98. SIAM, 2007.

[14] M. Luo, J. Yao, R. Liaw, E. Liang, and I. Stoica. IMPACT: importance weighted asynchronous architectures with clipped target networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[15] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1928–1937. JMLR.org, 2016.

[16] R. Munos, T. Stepleton, A. Harutyunyan, and M. G. Bellemare. Safe and efficient off-policy reinforcement learning. In D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29, NIPS 2016, December 5-10, 2016, Barcelona, Spain*, pages 1046–1054, 2016.

[17] C. Nota and P. S. Thomas. Is the policy gradient a gradient? In A. E. F. Seghrouchni, G. Sukthankar, B. An, and N. Yorke-Smith, editors, *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '20, Auckland, New Zealand, May 9-13, 2020*, pages 939–947. International Foundation for Autonomous Agents and Multiagent Systems, 2020. URL `https://dl.acm.org/doi/abs/10.5555/3398761.3398871`.

[18] C.-H. Pi, K.-C. Hu, S. Cheng, and I.-C. Wu. Low-level autonomous control and tracking of quadrotor using reinforcement learning. *Control Engineering Practice*, 95:104222, 2020. ISSN 0967-0661. doi: https://doi.org/10.1016/j.conengprac.2019.104222.

[19] D. Precup, R. S. Sutton, and S. P. Singh. Eligibility traces for off-policy policy evaluation. In *Proceedings of the 17th International Conference on Machine Learning, ICML 2000, Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*, pages 759–766. Morgan Kaufmann, 2000.

[20] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics, 1994.

[21] G. Rummery, M. Niranjan, and U. of Cambridge. Engineering Department. *On-line Q-learning Using Connectionist Systems*. University of Cambridge, Department of Engineering, 1994.

[22] J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1889–1897. JMLR.org, 2015.

[23] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. In *ArXiv*, volume abs/1707.06347, 2017.

[24] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3: 9–44, 1988.

[25] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[26] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12, NIPS 1999, Denver, Colorado, USA, November 29 - December 4, 1999*, pages 1057–1063. The MIT Press, 1999.

[27] P. Thomas. Bias in natural actor-critic algorithms. volume 32 of *Proceedings of Machine Learning Research*, pages 441–448, Bejing, China, 22–24 Jun 2014. PMLR. URL `http://proceedings.mlr.press/v32/thomas14.html`.

[28] N. Vieillard, O. Pietquin, and M. Geist. Deep conservative policy iteration. *CoRR*, abs/1906.09784, 2019.

[29] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas. Sample efficient actor-critic with experience replay. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[30] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

A Proof of Theorems

The proofs for the theorems in Section 4 are given in this Appendix.

Theorem A.1. *Given any arbitrary policy μ, a policy π improves μ if π satisfies either one of the following conditions:*

$$\sum_{a\in\mathcal{A}} \pi(a|s)A^{\mu}(s,a) \geq 0,\ \forall s \in \mathcal{S} \tag{23}$$

$$\sum_{a\in\mathcal{A}} \mu(a|s)A^{\pi}(s,a) \leq 0,\ \forall s \in \mathcal{S} \tag{24}$$

Proof. For any given policy π, the expected reward provided by a state s is

$$\sum_{a\in\mathcal{A}} \pi(a|s)r(s,a), \tag{25}$$

and the total discounted expected reward provided by s can be represented by

$$\left(\sum_{t\geq 0} \gamma^t P(s_t = s)\right) \sum_{a\in\mathcal{A}} \pi(a|s)r(s,a) \tag{26}$$

where $P(s_t = s)$ is the probability that visits the state s at time step t. From (26),

$$V^{\pi}(s) = \sum_{x\in\mathcal{S}} \rho_s^{\pi}(x) \sum_{a\in\mathcal{A}} \pi(a|x)r(x,a), \tag{27}$$

if following policy π for $P(s_t = s)$. Following this idea and consider a new reward defined by

$$r'(s,a,s') = r(s,a) + \gamma V^{\mu}(s') - V^{\mu}(s). \tag{28}$$

Obviously

$$V^{\pi}(s) - V^{\mu}(s) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r'(s_t,a_t,s_{t+1})\middle| s_0 = s, \forall a_t \sim \pi(a|s_t), s_{t+1} \sim P(s|s_t,a_t)\right]. \tag{29}$$

The expected reward r' provided by s is

$$\sum_{a\in\mathcal{A}} \pi(a|s) \sum_{s'\in\mathcal{S}} P(s'|s,a)r'(s,a,s'). \tag{30}$$

By definition

$$\sum_{s'\in\mathcal{S}} P(s'|s,a)r'(s,a,s') = A^{\mu}(s,a). \tag{31}$$

Hence,

$$V^{\pi}(s) - V^{\mu}(s) = \sum_{x\in\mathcal{S}} \rho_s^{\pi}(x) \sum_{a\in\mathcal{A}} \pi(a|x)A^{\mu}(x,a). \tag{32}$$

This formula holds for all states $s \in \mathcal{S}$. Since all are non-negative from (23), the above value in (32) is non-negative too. Thus, this implies that the values of (32) are all non-negative for all states. Thus, we obtain that the policy π is improved from μ. In the case that the formula (24) is satisfied, we can prove that the policy π is improved from μ in a similar way, which is omitted here. $\square$

Property A.2 shows the existence of some actions with non-negative values for all states, namely, there exist a action a do not worst then current policy. This is useful in the proof of Theorem 3.2.

Property A.2. *For arbitrary policy π, the advantage function*

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s) \tag{33}$$

has non-negative value support. Formally,

$$\forall s \in \mathcal{S}, \left\{a \middle| A(s,a) \geq 0\right\} \neq \emptyset \tag{34}$$

Proof. Suppose not,

$$\Big\{a\Big|A^{\pi}(s,a) \geq 0\Big\} = \emptyset \tag{35}$$

then

$$\sum_{a\in\mathcal{S}} \pi(a|s)A^{\pi}(s,a) < 0, \tag{36}$$

which contradiction to $\sum_{a\in\mathcal{A}} \pi(a|s)A^{\pi}(s,a) = 0$. □

Theorem A.3 (Second Fundamental Theorem of Policy Improvement)**.** *Given a policy μ, if a new policy π is selected according to*

$$(\pi(a|s) - \mu(a|s))A^{*}(s,a) \geq 0, \tag{37}$$

where $*$ can be either π or μ, then $\pi \geq \mu$.

Proof. Since

$$\sum_{a\in\mathcal{A}} \mu(a|s)A^{\mu}(s,a) = 0, \tag{38}$$

we obtain the following equation

$$\sum_{a\in\mathcal{A}} \pi(a|x)A^{\mu}(s,a) = \sum_{a\in\mathcal{A}} \left(\pi(a|x) - \mu(a|x)\right) A^{\mu}(s,a). \tag{39}$$

According to Property A.2 there exists non-negative advantages of actions for all states. Thus, there exists a new policy π that can be selected to increase the probability of selecting the actions with non-negative advantages of the origin policy μ and decrease those for others. Thus, the new policy π satisfies Theorem 3.1, and this theorem when $* = \mu$. When $* = \pi$, the proof is similar to above by exchanging π with μ. □

Theorem 3.2 above turns the policy gradient method into a classification approach, and the classifier is set to the formula

$$\pi(a|s) - \mu(a|s). \tag{40}$$

In the other words, if the new policy is selected according Theorem 3.2, then whether an action is expected to be good or not can be simply predicted by the sign of classifier (40).

Theorem A.4 (Labelling Quality of Stochastic Policy)**.** *Given a policy π, for arbitrary state-action value estimation $\hat{Q}(s,a)$ and arbitrary baseline function $\phi(s,a)$, denote*

$$\hat{A} = \hat{Q} - \phi,\ \Delta Q = \hat{Q} - Q^{\pi},\ \Delta V = \phi - V^{\pi} \tag{41}$$

and

$$\Delta A = \Delta Q - \Delta V = (\hat{Q} - Q^{\pi}) - (\phi - V^{\pi}). \tag{42}$$

Then, the label is correct if

$$|\hat{A} - \frac{1}{2}\Delta A| > \frac{1}{2}|\Delta A|, \tag{43}$$

and incorrect otherwise.

Proof. The label is correct if the true advantage has the same sign of the estimated advantage, namely

$$(Q^{\pi} - V^{\pi})(\hat{Q} - \phi) > 0. \tag{44}$$

Rewrite the left item as

$$\begin{aligned}(Q^{\pi} - V^{\pi})(\hat{Q} - \phi) = &\Big[(\hat{Q} - \phi) - (\Delta Q - \Delta V)\Big]\,\hat{A} \\ =&\hat{A}^2 - \hat{A}(\Delta Q - \Delta V) \\ =&\hat{A}^2 - \hat{A}(\Delta A) \\ =&(\hat{A} - \frac{\Delta A}{2})^2 - (\frac{\Delta A}{2})^2\end{aligned} \tag{45}$$

Thus, (43) is satisfied from (44) and (45). □

Corollary A.5. *For a deterministic transition environment, assume that V^π and $\hat{V}$ are locally Lipchitz continuous, i.e.,*

$$|V^\pi(s_{t+1}) - V^\pi(s_t)| \leq L\|s_{t+1} - s_t\| \text{ and } |\hat{V}(s_{t+1}) - \hat{V}(s_t)| \leq \hat{L}\|s_{t+1} - s_t\|, \tag{46}$$

and that the function approximation error is uniform bounded, $|V^\pi(s) - \hat{V}(s)| < \epsilon$. Then, if $\hat{A}$ is estimated by TD

$$\hat{A} = r_t + \gamma\hat{V}(s_{t+1}) - \hat{V}(s_t) \Rightarrow |\Delta A| \leq \gamma\left(L + \hat{L}\right)\|s_{t+1} - s_t\| + (1-\gamma)\epsilon. \tag{47}$$

Proof. Since the transition is deterministic, the Q value error is determined by

$$\begin{aligned}\Delta Q =& r_t + \gamma\hat{V}(s_{t+1}) - r_t - \gamma V^\pi(s_{t+1}) \\ =& \gamma\left(\hat{V}(s_{t+1}) - V^\pi(s_{t+1})\right).\end{aligned} \tag{48}$$

Therefore

$$\begin{aligned}\Delta Q - \Delta V =& \gamma\left(\hat{V}(s_{t+1}) - V^\pi(s_{t+1})\right) - \left(\hat{V}(s_t) - V^\pi(s_t)\right) \\ =& \gamma\left(\hat{V}(s_{t+1}) - \hat{V}(s_t)\right) - \gamma\left(V^\pi(s_{t+1}) - V^\pi(s_t)\right) \\ &+ (\gamma - 1)\left(\hat{V}(s_t) - V^\pi(s_t)\right).\end{aligned} \tag{49}$$

Its absolute value are

$$\begin{aligned}|\Delta Q - \Delta V| \leq& \gamma\left|\hat{V}(s_{t+1}) - \hat{V}(s_t)\right| + \gamma\left|V^\pi(s_{t+1}) - V^\pi(s_t)\right| \\ &+ (1-\gamma)\left|\hat{V}(s_t) - V^\pi(s_t)\right| \\ \leq& \gamma\left(L + \hat{L}\right)\|s_{t+1} - s_t\| + (1-\gamma)\epsilon.\end{aligned} \tag{50}$$

□

From above,

$$\sup|\Delta Q| = \gamma\sup|\hat{V}(s_{t+1}) - V^\pi(s_{t+1})| < \gamma\epsilon, \tag{51}$$

and

$$\sup|\Delta V| = \sup|\hat{V}(s_t) - V^\pi(s_t)| < \epsilon. \tag{52}$$

Thus, in the worse case,

$$|\Delta A| < \sup|\Delta Q| + \sup|\Delta V| < (1+\gamma)\epsilon. \tag{53}$$

B Pseudo Code and Experiment Configuration

Algorithm 1: Trial Algorithm

```
Result: Learned policy $\pi$
$\hat{V}(s;\theta_v)$: random initialized value network ;
$\hat{a}(s;\theta_\pi)$: random initialized policy network ;
$\sigma = 0.5$ ;
$\mathcal{D}$: the replay buffer fulled by policy $\pi$ defined in (22);
while True do
    Sample $N$ newest trajectory $\tau_{new} = \{(s_i, a_i, r_i, \mu_i, s_{i+1}) \sim \pi, i = 1, 2, \cdots, T\}$;
    Sample $M$ trajectory $\tau_{old} \in \mathcal{D}$;
    $\mathcal{B} = \{\tau_{new}\} \cup \{\tau_{old}\}$;
    for $\tau \in \mathcal{B}$ do
        $tmp = 0$;
        for $i = T$ to $1$ do
            $\hat{Q}_i = \min\{r_t + \gamma \hat{V}(s_{t+1}), \frac{1}{1-\gamma}\}$;
            $A_i = \hat{Q}_i - V(s_i)$;
            $A_i^{retrace} = A_i + \gamma \times tmp$;
            $tmp = \min(\frac{\pi_i}{\mu_i}, 1) A_i^{retrace}$;
            $V_i^{retrace} = V_i + tmp$;
        end
        $(\theta_\pi, \sigma) \leftarrow (\theta_\pi, \sigma) - \alpha \nabla_{\theta_\pi, \sigma} \frac{1}{T} \sum_{i=1}^{T} \max\left\{0, \epsilon - \frac{A_i^{trace}}{|A_i^{trace}|} \log \frac{\pi_i}{\mu_i}\right\}$;
        $\theta_v \leftarrow \theta_v - \alpha \nabla \frac{1}{T} \sum_{i=1}^{T} \left(V_i^{retrace} - \hat{V}(s_i)\right)^2$;
        $\sigma \leftarrow clip(\sigma, 0.1, 0.5)$
    end
    $\mathcal{D} = \mathcal{D} \cup \{\tau_{new}\}$;
    Remove oldest $N$ trajectory from $\mathcal{D}$
end
```

Algorithm 2: IMPALA

```
Result: Learned policy $\pi$
$\hat{V}(s;\theta_v)$: random initialized value network ;
$\hat{a}(s;\theta_\pi)$: random initialized policy network ;
$\sigma = 0.5$ ;
$\mathcal{D}$: the replay buffer fulled by policy $\pi$ defined in (22);
while True do
    Sample $N$ newest trajectory $\tau_{new} = \{(s_i, a_i, r_i, \mu_i, s_{i+1}) \sim \pi, i = 1, 2, \cdots, T\}$;
    Sample $M$ trajectory $\tau_{old} \in \mathcal{D}$;
    $\mathcal{B} = \{\tau_{new}\} \cup \{\tau_{old}\}$;
    for $\tau \in \mathcal{B}$ do
        $tmp = 0$;
        for $i = T$ to $1$ do
            $\hat{Q}_i = \min\{r_t + \gamma\hat{V}(s_{t+1}), \frac{1}{1-\gamma}\}$;
            $A_i = \hat{Q}_i - V(s_i)$;
            $A_i^{retrace} = A_i + \gamma \times tmp$;
            $tmp = \min(\frac{\pi_i}{\mu_i}, 1)A_i^{retrace}$;
            $V_i^{retrace} = V_i + tmp$;
        end
        $(\theta_\pi, \sigma) \leftarrow (\theta_\pi, \sigma) + \alpha\nabla_{\theta_\pi,\sigma}\frac{1}{T}\sum_{i=1}^{T} A_i^{trace}\frac{\pi_i}{\mu_i}$;
        $\theta_v \leftarrow \theta_v - \alpha\nabla\frac{1}{T}\sum_{i=1}^{T}\left(V_i^{retrace} - \hat{V}(s_i)\right)^2$;
        $\sigma \leftarrow clip(\sigma, 0.1, 0.5)$
    end
    $\mathcal{D} = \mathcal{D} \cup \{\tau_{new}\}$;
    Remove oldest $N$ trajectory from $\mathcal{D}$
end
```

Table 1: Training configuration for examine our theory.

Environment	
m	0.1
a	[-1,1]
ΔT	0.01
Position initial	[-2,2]
Velocity initial	[-2,2]
Horizon step	200
Reward	1$-$cost
Training Configuration	
Value Network	$[128, 128]$
Policy Network	$[128, 128]$, $[32, 32]$
Activation Function	ReLU
Learning Rate	0.001
Optimizer	Adam [10]
Batch Size	1-Horizon
Each Generation	64 (Newest) $+$128 (Old)
Replay Buffer Size	5000-Horizon