entropy

MDPI

Article

Learn Quasi-Stationary Distributions of Finite State Markov Chain

Zhiqiang Cai [1,*], Ling Lin [2] and Xiang Zhou [1,3]

[1] School of Data Science, City University of Hong Kong, Tat Chee Ave, Kowloon, Hong Kong, China; xiang.zhou@cityu.edu.hk
[2] School of Mathematics, Sun Yat-sen University, Guangzhou 510275, China; linling27@mail.sysu.edu.cn
[3] Department of Mathematics, City University of Hong Kong, Tat Chee Ave, Kowloon, Hong Kong, China
* Correspondence: zqcai3-c@my.cityu.edu.hk

Abstract: We propose a reinforcement learning (RL) approach to compute the expression of quasi-stationary distribution. Based on the fixed-point formulation of quasi-stationary distribution, we minimize the KL-divergence of two Markovian path distributions induced by candidate distribution and true target distribution. To solve this challenging minimization problem by gradient descent, we apply a reinforcement learning technique by introducing the reward and value functions. We derive the corresponding policy gradient theorem and design an actor-critic algorithm to learn the optimal solution and the value function. The numerical examples of finite state Markov chain are tested to demonstrate the new method.

Keywords: quasi-stationary distribution; reinforcement learning; KL-divergence; actor-critic algorithm

Citation: Cai, Z.; Lin, L.; Zho, X. Learn Quasi-Stationary Distributions of Finite State Markov Chain. *Entropy* **2022**, *24*, 133. https://doi.org/10.3390/e24010133

Academic Editors: Michael Dellnitz, Carsten Hartmann and Feliks Nüske

Received: 7 November 2021
Accepted: 11 January 2022
Published: 17 January 2022

1. Introduction

Quasi-stationary distribution (QSD) is the long time statistical behavior of a stochastic process that will be surely killed when this process is conditioned to survive [1]. This concept has been widely used in applications, such as in biology and ecology [2,3], chemical kinetics [4,5], epidemics [6–8], medicine [9] and neuroscience [10,11]. Many works for rare events in meta-stable systems also focus on this quasi-stationary distribution [12,13]. In addition, some new Monte Carlo sampling methods, for instance, the Quasi-stationary Monte Carlo method [14,15], also arise by using QSD instead of true stationary distribution, for instance, the Quasi-stationary Monte Carlo method [14,15]

We are interested in the numerical computation of QSD and focus on the finite state Markov chain in this paper. Mathematically, the quasi-stationary distribution can be solved as the principal left eigenvector of a sub-Markovian transition matrix. Thus, traditional numerical algebra methods can be applied to solve the quasi-stationary distribution in finite state space, for example, the power method [16], the multi-grid method [17] and Arnoldi's algorithm [18]. These eigenvector methods can produce a stochastic vector for QSD instead of generating samples of QSD.

In search of efficient algorithms for large state space, stochastic approaches are in favor of either sampling the QSD or computing the expression of QSD, and these methods can be applied or extended easily to continuous state space. A popular approach for sampling quasi-stationary distribution is the Fleming–Viot stochastic method [19]. The Flemming–Viot method first simulates N particles independently. When any one of the particles falls into the absorbing state and becomes killed, a new particle is uniformly selected from the remaining $N-1$ surviving particles to replace the dead one, and the simulation continues. When time and N tend to infinity, the particles' empirical distribution can converge to the quasi-stationary distribution.

In [20–22], the authors proposed to recursively update the expression of QSD at each iteration based on the empirical distribution of a single-particle simulation. It is shown

in [21] that the convergence rate can be $O(n^{-1/2})$, where n is the iteration number. This method is later improved in [23,24] by applying the stochastic approximation method [25] and the Polyak–Ruppert averaging technique [26]. These improved algorithms have a choice of flexible step size but require a projection operator onto probability simplex, which carries some extra computational overhead increasing with the number of states. Ref. [15] extended the algorithm to the diffusion process.

In this paper, we focus on how to compute the expression of the quasi-stationary distribution, which is denoted by $\alpha(x)$ on a metric space $\mathcal{E}$. If $\mathcal{E}$ is finite, α is a probability vector, and if $\mathcal{E}$ is a domain in $\mathbb{R}^d$, then α is a probability density function on $\mathcal{E}$. We assume α can be numerically represented in parametric form α_θ and $\theta \in \Theta$. This family $\{\alpha_\theta\}$ can be in tabular form or any neural network. Then, the problem of finding the QSD α becomes answering the question of how to compute the optimal parameter θ in Θ. We call this problem the learning problem for QSD. In addition, we want to directly learn QSD and not use the distribution family $\{\alpha_\theta\}$ to fit the simulated samples generated by other traditional simulation methods.

Our minimization problem for QSD is similar to the variational inference (VI) [27], which minimizes an objective functional measuring the distance between the target and candidate distributions. However, unlike the mainstream VI methods such as evidence lower bound (ELBO) technique [28] or particle-based [29], flow-based methods [30], our approach is based on recent important progresses from reinforcement learning (RL) method [31], particularly the policy gradient method and actor-critic algorithm. We first regard the learning process of the quasi-stationary distribution as the interaction with the environment, which is constructed by the property of QSD. Reinforcement learning has recently shown tremendous advancements and remarkable successes in applications (e.g., [32–34]). The RL framework provides an innovative and powerful modeling and computation approach for many scientific computing problems.

The essential question is how to formulate the QSD problem as an RL problem. Firstly, for the sub-Markovian kernel K of a Markov process, we can define a Markovian kernel K_α on $\mathcal{E}$ (see Definition 1) and then QSD is defined by the equation $\alpha = \alpha K_\alpha$, which equals α as the initial distribution and the distribution after one step. Secondly, we consider an optimal α (in our parametric family of distribution) to minimize the Kullback–Leibler divergence (i.e., relative entropy) of two path distributions, denoted by $\mathbb{P}$ and $\mathbb{Q}$, associated with two Markovian kernels K_α and K_β where $\beta := \alpha K_\alpha$. Thirdly, inspired by the recent work [35] of using RL for rare events sampling problems, we transform the minimization of KL divergence between $\mathbb{P}$ and $\mathbb{Q}$ into the maximization of a time-averaged reward function and defined the corresponding value function $V(x)$ at each state x. This completes our modeling of RL for the quasi-stationary distribution problem. Lastly, we derive the policy gradient theorem (Theorem 1) to compute the gradient with respect to θ of the averaged reward for the learning dynamic for the averaged reward. This is known as the "actor" part. The "critic" part is to learn the value function V in its parametric form V_ψ. The actor-critic algorithm uses the stochastic gradient descent to train the parameter θ for the action α_θ and the parameter ψ for the value function V_ψ (see Algorithm 1).

Our contribution is that we first devise a method to transform the QSD problem into the RL problem. Similar to [35], our paper also uses the KL-divergence to define the RL problem. However, our paper fully adapts the unique property of QSD that is a fixed point problem $\alpha = \alpha K_\alpha$ to define the RL problem.

Our learning method allows the flexible parametrization of the distributions and uses the stochastic gradient method to train the optimal distribution. It is easy to implement optimization with scale up to large state spaces. The numerical examples we tested have shown our that methods converge faster than other existing methods [22,23].

Finally, we remark that our method works very well for QSD of the strict sub-Markovian kernel K but is not applicable to compute the invariant distribution when K is Markovian. This is because we transform the problem into the variational problem between two Markovian kernels K_α and K_β (where $\beta = \alpha K_\alpha$). Note that $K_\alpha(x, y) =$

$K(x,y) + (1 - K(x,\mathcal{E}))\alpha(y)$ (Definition 1), and our method is based on the fact that $\alpha = \beta$ if and only if $K_\alpha = K_\beta$. If K is Markovian kernel, then $K_\alpha \equiv K$ for any α, and our method cannot work. Thus, $K(x,\mathcal{E})$ has to be strictly less than 1 for some $x \in \mathcal{E}$.

This paper is organized as follows. Section 2 is a short review of the quasi-stationary distribution and some basic simulation methods of QSD. In Section 3, we first formulate the reinforcement learning problem by KL-divergence and derive the policy gradient theorem (Theorem 1). Using the above formulation, we then develop the actor-critic algorithm to estimate the quasi-stationary distribution. In Section 4, the efficiency of our algorithms is illustrated by four examples compared with the simulation methods in [24].

Algorithm 1: (**ac-α method**) Actor-critic algorithm for quasi-stationary distribution α_θ

Initialization

$t = 0; \theta = \theta_0; \psi = \psi_0; r_t = r_0;$

Sample $X_0 \sim \mu_{\theta_0}$, the stationary distribution of $K_{\alpha_{\theta_0}}$

for $t = 0, 1, 2, \ldots$ **do**

Sample X_{t+1} from the transition kernel $K_{\alpha_{\theta_t}}(X_t, X_{t+1})$

$\delta_t = V_{\psi_t}(X_{t+1}) - V_{\psi_t}(X_t) + R_{\theta_t}(X_t, X_{t+1}) - r_t$

$\theta_{t+1} = \theta_t + \eta_t^\theta \left(\delta_t \nabla_\theta \ln K_{\alpha_{\theta_t}}(X_t, X_{t+1}) + \nabla_\theta \ln K_{\beta_{\theta_t}}(X_t, X_{t+1})\right)$

$\psi_{t+1} = \psi_t + \eta_t^\psi \, \delta_t \nabla_\psi V_{\psi_t}(X_t)$

$r_{t+1} = r_t + \eta_t^r \, \delta_t$

$X_t \sim \mu_{\theta_{t+1}}$ the stationary distribution of $K_{\alpha_{\theta_{t+1}}}$

$t = t + 1$

2. Problem Setup and Review

2.1. Quasi-Stationary Distribution

We start with an abstract setting. Let $\mathcal{E}$ be a finite state equipped with the Borel σ-field $\mathcal{B}(\mathcal{E})$, and let $\mathcal{P}(\mathcal{E})$ be the space of probabilities over $\mathcal{E}$. A sub-Markovian kernel on $\mathcal{E}$ is defined as a map $K : \mathcal{E} \times \mathcal{B}(\mathcal{E}) \mapsto [0,1]$ such that for all $x \in \mathcal{E}$, $A \mapsto K(x,A)$ is a nonzero measure with $K(x,\mathcal{E}) \leq 1$ and for all $A \in \mathcal{B}(\mathcal{E})$, $x \mapsto K(x,A)$ is measurable. In particular, if $K(x,\mathcal{E}) = 1$ for all $x \in \mathcal{E}$, then K is called a Markovian kernel. Throughout the paper, we assume that K is strictly sub-Markovian, i.e., $K(x,\mathcal{E}) < 1$ for some x.

Let X_t be a Markov chain with values in $\mathcal{E} \cup \{\partial\}$ where $\partial \notin \mathcal{E}$ denotes an absorbing state. We define the extinction time

$$\tau := \inf\{t > 0 : \quad X_t = \partial\}.$$

We define the **quasi-stationary distribution (QSD)** α as the long time limit of the conditional distribution, if there exists a probability distribution ν on $\mathcal{E}$ such that the following is the case:

$$\alpha(A) := \lim_{t\to\infty} P_\nu(X_t \in A \mid \tau > t), \qquad A \in \mathcal{B}(\mathcal{E}). \tag{1}$$

where P_ν refers to the probability distribution of X_t associated with the initial distribution ν on $\mathcal{E}$. Such a conditional distribution well describes the behavior of the process before extinction, and it is easy to see that α satisfies the following fixed point problem:

$$P_\alpha(X_t \in A \mid \tau > t) = \alpha(A) \tag{2}$$

where P_α refers to the probability distribution of X_t associated with the initial distribution α on $\mathcal{E}$. Equation (2) is equivalent to the following stationary condition such that the following is the case:

$$\alpha = \frac{\alpha K}{\alpha K \mathbf{1}}, \quad \text{or } \alpha(y) = \frac{\sum_x \alpha(x)K(x,y)}{\sum_x \alpha(x)K(x,\mathcal{E})} \tag{3}$$

where α is a row vector and $\mathbf{1}$ denotes the column vector with all entries being one and

$$K(x,\mathcal{E}) = \sum_{x' \in \mathcal{E}} K(x,x').$$

For any sub-Markovian kernel K, we can associate K with a Markovian kernel $\tilde{K}$ on $\mathcal{E} \cup \{\partial\}$ defined by the following:

$$\begin{cases} \tilde{K}(x,A) = K(x,A) \\ \tilde{K}(x,\{\partial\}) = 1 - K(x,\mathcal{E}) \\ \tilde{K}(\partial,\{\partial\}) = 1. \end{cases}$$

for all $x \in \mathcal{E}, A \in \mathcal{B}(\mathcal{E})$. The kernel $\tilde{K}$ can be understood as the Markovian transition kernel of the Markov chain (X_t) on $\mathcal{E} \cup \{\partial\}$ for which its transitions in $\mathcal{E}$ is specified by K, but it is "killed" forever once it leaves $\mathcal{E}$.

In this paper, we assume $\mathcal{E}$ is a finite state space and the process in consideration has a unique QSD. Assume that K is irreducible, then existence and uniqueness of the quasi-stationary distribution can be obtained by the Perron–Frobenius theorem [36].

An important Markovian kernel is the following K_α, which is defined on $\mathcal{E}$ only and has a "regenerative probability" α.

Definition 1. *For any given $\alpha \in \mathcal{P}(\mathcal{E})$ and a sub-Markovian kernel K on $\mathcal{E}$, we define K_α, a Markovian kernel on $\mathcal{E}$, as follows:*

$$K_\alpha(x,A) := K(x,A) + (1 - K(x,\mathcal{E}))\alpha(A) \tag{4}$$

for all $x \in \mathcal{E}$ and $A \in \mathcal{B}(\mathcal{E})$.

K_α is a Markovian kernel because $K_\alpha(x,\mathcal{E}) = 1$. It is easy to sample $X_{t+1} \sim K_\alpha(X_t,\cdot)$ from any state $X_t \in \mathcal{E}$: run the transition as normal by using $\tilde{K}$ to have a next state denoted by Y, then $X_{t+1} = Y$ if $Y \in \mathcal{E}$; otherwise, sample X_{t+1} from α.

We know that α is the quasi-stationary distribution of K if and only if it is the stationary distribution of K_α.

$$\alpha = \alpha K_\alpha. \tag{5}$$

It is easy to observe that $\alpha = \beta$ if and only if $K_\alpha = K_\beta$ for any two distributions α and β. Moreover, for every α', $K_{\alpha'}$ has a unique invariant probability denoted by $\Gamma(\alpha')$. Then, $\alpha' \mapsto \Gamma(\alpha')$ is continuous in $\mathcal{P}(\mathcal{E})$ (i.e., for the topology of weak convergence), and there exists $\alpha \in \mathcal{P}(\mathcal{E})$ such that $\alpha = \Gamma(\alpha)$ or, equivalently, α is a QSD for K.

2.2. Review of Simulation Methods for Quasi-Stationary Distribution

According to the above subsection, the QSD α satisfies the fixed point problem as follows:

$$\alpha = \Gamma(\alpha), \tag{6}$$

where $\Gamma(\alpha)$ is the stationary distribution of K_α on $\mathcal{E}$. In general, (6) can be solved recursively by $\alpha_{n+1} \leftarrow \Gamma(\alpha_n)$.

The Fleming–Viot (FV) method [19] evolves N particles independently of each other as a Markov process associated with the transition kernel K_α until one succeeds in jumping to the absorbing state ∂. At that time, this killed particle is immediately reset to $\mathcal{E}$ as an initial state uniformly chosen from one of the remaining $N-1$ particles. The QSD α is approximated by the empirical distribution of the N particles in total, and these

particles can be regarded as samples from the quasi-stationary distribution α such as the MCMC method.

Ref. [37] proposed a simulation method by only using one particle at each iteration to update α. At iteration n, given an $\alpha_n \in \mathcal{P}(\mathcal{E})$, one can run a discrete-time Markov chain $X^{(n+1)}$ as normal on $\partial \cup \mathcal{E}$ with initial $X_0^{(n+1)} \sim \alpha_n$; then, α_{n+1} is computed as the following weighted average of empirical distributions:

$$\alpha_{n+1}(x) := \alpha_n(x) + \frac{1}{n+1} \sum_{k=0}^{\tau^{(n+1)}-1} \frac{I\left(X_k^{(n+1)} = x \mid X_0^{(n+1)} \sim \alpha_n\right) - \alpha_n(x)}{\frac{1}{n+1}\sum_{j=1}^{n+1} \tau^{(j)}} \tag{7}$$

where $n \geq 0$ and I are the indicator functions, and $\tau^{(j)} = \min\left\{k \geq 0 \mid X_k^{(j)} \in \partial\right\}$ is the first extinction time for the process $X^{(j)}$. This iterative scheme has a convergence rate of $O(\frac{1}{\sqrt{n}})$.

In [23,24], the above method is extended to the stochastic approximations framework:

$$\alpha_{n+1}(x) = \Theta_H\left[\alpha_n + \epsilon_n \sum_{k=0}^{\tau^{(n+1)}-1} \left(I\left(X_k^{(n+1)} = x | X_0^{(n+1)} \sim \alpha_n\right) - \alpha_n(x)\right)\right] \tag{8}$$

where Θ_H denotes the L_2 projection into the probability simplex, and ϵ_n is the step size satisfying $\sum \epsilon_n = \infty$ and $\sum \epsilon_n^2 < \infty$. Specifically, if $\epsilon_n = O(\frac{1}{n^r})$ for $0.5 < r < 1$, under a sufficient condition, they have $\sqrt{n^r}(\alpha_n - \alpha) \xrightarrow{\mathrm{d}} \mathcal{N}(0, V)$ for some matrix V [23,24]. If the Polyak–Ruppert averaging technique is applied to generate the following:

$$\nu_n := \frac{1}{n}\sum_{k=1}^{n} \alpha_k, \tag{9}$$

then the convergence rate of $\nu_n \to \alpha$ becomes $\frac{1}{\sqrt{n}}$ [23,24].

The simulation schemes (7) and (8) need to sample the initial states according to α_n and to add the empirical distribution and α_n at each x point wisely. Thus, they are suitable for finite state space where α is a probability vector saved in the tabular form. In (8), there is no need to record all exit times $\tau^{(j)}, j = 1, \ldots, n$, but the additional projection operation in (8) is computationally expensive since the cost is $O(m \log m)$ where $m = |\mathcal{E}|$ [38,39].

3. Learn Quasi-Stationary Distribution

We focus on the computation of the expression of the quasi-stationary distribution. In particular, when this distribution is parametrized in a certain manner by θ, we can extend the tabular form for finite-state Markov chain to any flexible form, even in the neural networks for probability density function in $\mathbb{R}^d$. However, we do not pursue this representation and expressivity issue here and restrict our discussion to finite state space only to illustrate our main idea first. In finite state space, $\alpha(x)$ for $x \in \mathcal{E} = \{1, \ldots, m\}$ can be simply described as a softmax function with $m-1$ parameter $\theta_i : \alpha(i) \propto e^{\theta_i}, 1 \leq i \leq m-1$ ($\theta_m = 0$). This introduces no representation error. For the generalization to continuous space $\mathcal{E}$ in jump and diffusion processes or even for a huge finite state space, a good representation of $\alpha_\theta(x)$ is important in practice.

In this section, we shall formulate our QSD problem in terms of reinforcement learning (RL) so that the problem of seeking optimal parameters becomes a policy optimization problem. We derive the policy gradient theorem to construct a gradient descent method for the optimal parameter. We then show a method for designing actor-critic algorithms based on stochastic optimization.

3.1. Formulation of RL and Policy Gradient Theorem

Before introducing the RL method of our QSD problem, we develop a general formulation by introducing the KL-divergence between two path distributions.

Let P_θ and Q_θ be two families of Markovian kernels on $\mathcal{E}$ in parametric forms with the same set of parameters $\theta \in \Theta$. Assume both P_θ and Q_θ are ergodic for any θ. Let $T > 0$ and denote a path up to time T by $\omega_0^T = (X_0, X_1, \ldots, X_T) \in \mathcal{E}^{T+1}$. Define the path distributions under the Markov chain kernel P_θ and Q_θ, respectively.

$$\mathbb{P}_\theta(\omega_0^T) := \prod_{t=1}^{T} P_\theta(X_t \mid X_{t-1}), \quad \mathbb{Q}_\theta(\omega_0^T) := \prod_{t=1}^{T} Q_\theta(X_t \mid X_{t-1}). \tag{10}$$

Define the KL divergence from $\mathbb{P}_\theta$ to $\mathbb{Q}_\theta$ on $\mathcal{E}^{T+1}$:

$$\mathsf{D}_{KL}(\mathbb{P}_\theta \mid \mathbb{Q}_\theta) := \sum_{\omega_0^T} \mathbb{P}_\theta(\omega_0^T) \ln \frac{\mathbb{P}_\theta(\omega_0^T)}{\mathbb{Q}_\theta(\omega_0^T)} = -\mathbf{E}_{P_\theta} \sum_{t=1}^{T} R_\theta(X_{t-1}, X_t), \tag{11}$$

where the expectation $\mathbf{E}_{P_\theta}$ is for the path $(X_0, X_1, \ldots, X_T)$ generated by the transition kernel P_θ, and the following is called the (one-step) **reward**.

$$R_\theta(X_{t-1}, X_t) := -\ln \frac{P_\theta(X_t \mid X_{t-1})}{Q_\theta(X_t \mid X_{t-1})}. \tag{12}$$

Define the **average reward** $r(\theta)$ as the time averaged negative KL divergence in the limit of $T \to \infty$.

$$r(\theta) := -\lim_{T\to\infty} \frac{1}{T} \mathsf{D}_{KL}(\mathbb{P}_\theta \mid \mathbb{Q}_\theta) = -\lim_{T\to\infty} \frac{1}{T} \mathbf{E}_{P_\theta} \sum_{t=1}^{T} R_\theta(X_{t-1}, X_t). \tag{13}$$

Due to ergodicity of P_θ, $r(\theta) = \sum_{x_0,x_1} R_\theta(x_0, x_1) P_\theta(x_1|x_0) \mu_\theta(x_0)$ where μ_θ is the invariant measure of P_θ, $r(\theta)$ is independent of initial state X_0. Obviously, $r(\theta) \leq 0$ for any θ.

Property 1. *The following are equivalent:*

1. $r(\theta)$ *reaches its maximal value* 0 *at* θ^*;
2. $\mathbb{P}_{\theta^*} = \mathbb{Q}_{\theta^*}$ *in* $\mathcal{P}(\mathcal{E}^{T+1})$ *for any* $T > 0$;
3. $P_{\theta^*} = Q_{\theta^*}$;
4. $R_{\theta^*} \equiv 0$.

Proof. We only need to show $(1) \Longrightarrow (3)$. It is easy to see that

$$r(\theta) = -\sum_{x_0} \mathsf{D}_{KL}(P_\theta(\cdot|x_0) \mid Q_\theta(\cdot|x_0))\, \mu_\theta(x_0).$$

If $r(\theta) = 0$, since $\mu_\theta > 0$, then

$$\mathsf{D}_{KL}(P_\theta(\cdot|x_0) \mid Q_\theta(\cdot|x_0)) = 0 \qquad \forall x_0.$$

Thus, we have $P_\theta = Q_\theta$. □

The above property establishes the relationship between the RL problem and QSD problem.

We show our theoretic main result below as the foundation of our algorithm to be developed later. This theorem can be regarded as one type of the policy gradient theorem for the policy gradient method in reinforcement learning [31].

Define the following **value function** ([31] Chapter 13).

$$V(x) := \lim_{T\to\infty} \sum_{t=1}^{T} \mathbf{E}_{P_\theta}[R_\theta(X_{t-1}, X_t) - r(\theta) \mid X_0 = x]. \tag{14}$$

Certainly, V also depends on θ, although we do not write θ explicitly.

Theorem 1 (**policy gradient theorem**). *We have the following two properties:*

1. *At any θ, for any $x \in \mathcal{E}$, the following Bellman-type equation holds for the value function V and the average reward $r(\theta)$:*

$$V(x) = \mathbf{E}_{Y\sim P_\theta(\cdot|x)}[V(Y) + R_\theta(x, Y) - r(\theta)]. \tag{15}$$

2. *The gradient of the average reward $r(\theta)$ is the following:*

$$\begin{aligned}\nabla_\theta r(\theta) = {} & \mathbf{E}[\nabla_\theta \ln Q_\theta(Y \mid X)] + \\ & \mathbf{E}\Big[\Big(V(Y) - V(X) + R_\theta(X, Y) - r(\theta)\Big)\nabla_\theta \ln P_\theta(Y \mid X)\Big],\end{aligned} \tag{16}$$

where expectations are for the joint distribution $(X, Y) \sim \mu_\theta(x)P_\theta(y \mid x)$ where μ_θ is the stationary measure of P_θ.

Proof. We shall prove the Bellman equation first and then we use the Bellman equation to derive the gradient of the average reward $r(\theta)$. For any $x_0 \in \mathcal{E}$, by writing $\omega_0^T = (x_0, \ldots, x_T)$ and defining

$$\Delta R_\theta(\omega_0^T) = \sum_{t=1}^{T} (R(x_{t-1}, x_t) - r(\theta)),$$

we have the following:

$$\begin{aligned} V(x_0) &= \lim_{T\to\infty} \mathbf{E}_{P_\theta}\Big[\Delta R_\theta(\omega_0^T) \mid X_0 = x\Big] \\ &= \lim_{T\to\infty} \sum_{x_2,\ldots,x_T} \sum_{x_1} \left(\left(\prod_{t=2}^{T} P_\theta(x_t \mid x_{t-1})\right) P_\theta(x_1 \mid x_0)\Delta R(\omega_0^T)\right) \\ &= \lim_{T\to\infty} \sum_{x_1} \left(P_\theta(x_1 \mid x_0) \sum_{x_2,\ldots,x_T} \left(\prod_{t=2}^{T} P_\theta(x_t \mid x_{t-1})\Big[\Delta R(\omega_1^T) + \Delta R(\omega_0^1)\Big]\right)\right) \\ &= \sum_{x_1} \left(P_\theta(x_1 \mid x_0)\left(\lim_{T\to\infty}\left[\sum_{x_2,\ldots,x_T} \prod_{t=2}^{T} P_\theta(x_t \mid x_{t-1})\Delta R(\omega_1^T)\right] + \Delta R(\omega_0^1)\right)\right) \\ &= \sum_{x_1} P_\theta(x_1 \mid x_0)[V(x_1) + R_\theta(x_0, x_1)] - r(\theta), \end{aligned} \tag{17}$$

which proves (15); in other words, we have the following.

$$r(\theta) = \mathbf{E}_{Y\sim P_\theta(\cdot|x)}[V(Y) + R_\theta(x, Y) - V(x)], \qquad \forall x \in \mathcal{E}.$$

Next, we compute the gradient of $r(\theta)$. By trivial equality of the following:

$$\sum_{x_1} P_\theta(x_1 \mid x_0)\nabla_\theta \ln P_\theta(x_1 \mid x_0) = \nabla_\theta \sum_{x_1} P_\theta(x_1 \mid x_0) = 0, \tag{18}$$

and the definition (12), we can write the gradient of $r(\theta)$ as follows.

$$\begin{aligned}\nabla_\theta r(\theta) =& \sum_y \nabla_\theta P_\theta(y \mid x)[V(y) + R_\theta(x,y) - V(x)] \\ &+ \sum_y P_\theta(y \mid x)[\nabla_\theta V(y) - \nabla_\theta V(x) + \nabla_\theta \ln Q_\theta(y \mid x)].\end{aligned}$$

We here keep the term $V(x)$ in the first line, even though it has no contribution here (in fact, to add any constant to $V(x)$ is also fine). Since this equation holds for all states x on the right-hand side, we take the expectation with respect to μ_θ, the stationary distribution of P_θ. Thus, we have the following.

$$\begin{aligned}\nabla_\theta r(\theta) =& \sum_{x,y} \mu_\theta(x) \nabla_\theta P_\theta(y \mid x)[V(y) + R_\theta(x,y) - V(x)] \\ &+ \sum_{x,y} \mu_\theta(x) P_\theta(y \mid x)[\nabla_\theta V(y) - \nabla_\theta V(x) + \nabla_\theta \ln Q_\theta(y \mid x)] \\ =& \sum_{x,y} \mu_\theta(x) \nabla_\theta P_\theta(y \mid x)[V(y) + R_\theta(x,y) - V(x)] \\ &+ \sum_y \mu_\theta(y) \nabla_\theta V(y) - \sum_x \mu_\theta(x) \nabla_\theta V(x) + \sum_{x,y} \mu_\theta(x) P_\theta(y \mid x) \nabla_\theta \ln Q_\theta(y \mid x) \\ =& \sum_{x,y} \mu_\theta(x) P_\theta(y \mid x) \Big[V(y) + R_\theta(x,y) - V(x) \Big] \nabla_\theta \ln P_\theta(y \mid x) \\ &+ \sum_{x,y} \mu_\theta(x) P_\theta(y \mid x) \nabla_\theta \ln Q_\theta(y \mid x).\end{aligned}$$

In fact, we can add any constant number b (independent of x and y) inside the squared bracket of the last line without changing the equality due to the following fact similar to (18): $\sum_{x,y} \mu_\theta(x) \nabla_\theta P_\theta(y \mid x) = \sum_y \mu_\theta(y) \nabla_\theta \sum_x P_\theta(x \mid y) = 0$. (16) is a special case of $b = r(\theta)$. □

Remark 1. *As shown in the proof,* (16) *holds if* $r(\theta)$ *at the right-hand side is replaced by any constant number* b. $b = r(\theta)$ *is a good choice to reduce the variance since* $r(\theta)$ *can be regarded as the expectation of* R_θ.

Remark 2. *If* $P_\theta = Q_\theta$, *then the first term of* (16) *vanishes due to* (18) *and the second term of* (16) *vanishes due to* (15).

Remark 3. *The name of "policy" here refers to the role of* θ *as the policy for decision makers to improve reward* $r(\theta)$.

3.2. Learn QSD

Now, we discuss how to connect QSD with the results in the previous subsection. In view of Equation (5), we introduce $\beta := \alpha K_\alpha$ as the one-step distribution if starting from the initial α; in other words, we have the following.

$$\beta(y) := \sum_{x \in \mathcal{E}} \alpha(x) K_\alpha(x,y), \qquad \forall y \tag{19}$$

By (5), α is a QSD if and only if $\beta = \alpha$. However, we do not directly compare these two distributions α and β. Instead, we consider their Markovian kernels induced by (4): K_α and K_β. Our approach is to consider KL divergence similar to (11) between two kernels K_α and K_β since $\alpha = \beta$ if and only if $K_\alpha = K_\beta$. In this manner, one can view K_α and K_β (note $\beta = \alpha K_\alpha$) as two transition matrices P_θ and Q_θ in the previous section, in which the parameter θ here is in fact the distribution α.

To have a further representation of the distribution α, which is a (probability mass) function on $\mathcal{E}$, we propose a parametrized family for α in the form α_θ where θ is a generic parameter. In the simplest case, α_θ takes the so-called *soft-max* form $\alpha_\theta(i) = \frac{e^{\theta_i}}{\sum_{j\geq 1} e^{\theta_j}}$ if $\mathcal{E} = \{1, \ldots, N\}$ for $\theta = (\theta_1, \ldots, \theta_{N-1}, \theta_N \equiv 0)$. This parametrization represents α without any approximation error for finite state space and the effective space of θ is just $\mathbb{R}^{N-1}$. For certain problems, particularly with large state space, if one has some prior knowledge about the structure of the function α on $\mathcal{E}$, one might propose other parametric forms of α_θ with the dimension of θ less than the cardinality $|\mathcal{E}|$ to improve the efficiency, although the extra representation error in this manner has to be introduced.

For any given $\alpha_\theta \in \mathcal{P}(\mathcal{E})$, the corresponding Markovian kernel K_{α_θ} is then defined in (4) and $\beta_\theta = \alpha_\theta K_{\alpha_\theta} i$ is defined by (19). K_{β_θ} is like-wise defined by (4) again. To use the formulation in Section 3.1, we chose $P_\theta = K_{\alpha_\theta}$ and $Q_\theta = K_{\beta_\theta}$. Define the objective function as before:

$$r(\theta) := -\lim_{T\to\infty} \frac{1}{T} \mathsf{D}_{KL}(\mathbb{P}_\theta \mid \mathbb{Q}_\theta) = -\lim_{T\to\infty} \frac{1}{T} \mathbf{E}_{P_\theta} \sum_{t=1}^{T} R_\theta(X_{t-1}, X_t).$$

where the following is the case.

$$R_\theta(x, y) = -\ln \frac{K_{\alpha_\theta}(x, y)}{K_{\beta_\theta}(x, y)}.$$

The value function $V(x)$ is defined similarly. Theorem 1 now provides the expression of the following gradient:

$$\begin{aligned} \nabla_\theta r(\theta) = \mathbf{E}[&\big(R_\theta(X, Y) - r(\theta) + V(Y) - V(X)\big) \nabla_\theta \ln K_{\alpha_\theta}(X, Y) \\ &+ \nabla_\theta \ln K_{\beta_\theta}(X, Y)] \end{aligned} \tag{20}$$

where $(X, Y) \sim \mu_\theta(x) K_{\alpha_\theta}(x, y)$ and where μ_θ is the stationary measure of K_{α_θ}.

The optimal θ^* for the QSD α_θ is to maximize $r(\theta)$, and this can be solved by the gradient descent algorithm:

$$\theta_{t+1} = \theta_t + \eta_t^\theta \nabla_\theta r(\theta_t). \tag{21}$$

where $\eta_t^\theta > 0$ is the step size. In practice, the stochastic gradient is applied:

$$\nabla_\theta r(\theta_t) \approx \nabla_\theta \ln K_{\alpha_\theta}(X_t, X_{t+1}) \times \delta(X_t, X_{t+1}) + \nabla_\theta \ln K_{\beta_\theta}(X_t, X_{t+1})$$

where X_t, X_{t+1} are sampled based on the Markovian kernel K_{α_θ} (see Algorithm 1) and the differential temporal (TD) error δ_t is as follows.

$$\delta_t = \delta(X_t, X_{t+1}) = R_\theta(X_t, X_{t+1}) - r(\theta_t) + V(X_{t+1}) - V(X_t). \tag{22}$$

Next, we need to address a remaining issue, which is the question of how to compute value functions V and $r(\theta_t)$ in the TD error (22). In addition, we also need to show the details of computing $\nabla_\theta K_{\alpha_\theta}$ and $\nabla_\theta K_{\beta_\theta}$.

3.3. Actor-Critic Algorithm

With the stochastic gradient method (21), we can obtain optimal policy θ^*. We refer to (21) as the learning dynamics for the policy, and it is generally known as *actor*. To calculate the value function V appearing in $\nabla r(\theta)$, we need to have a new learning dynamic, which is called *critic*. Then, the overall policy-gradient method is termed as the actor-critic method.

We start with the Bellman Equation (15) for the value function and considered the mean-square-error loss as follows:

$$\mathrm{MSE}[V] = \frac{1}{2}\sum_x \nu(x)\left(\sum_y K_{\alpha_\theta}(x,y)[V(y) + R_\theta(x,y) - r(\theta)] - V(x)\right)^2$$

where ν is any distribution supported on $\mathcal{E}$. $\mathrm{MSE}[V] = 0$ if and only if V satisfies the Bellman Equation (15), i.e., V is the value function. To learn V, we introduce function approximation for the value function, V_ψ, with the parameter ψ and considered to minimize the following:

$$\mathrm{MSE}(\psi) = \frac{1}{2}\sum_x \nu(x)\left(\sum_y K_{\alpha_\theta}(x,y)[V(y) + R_\theta(x,y) - r(\theta)] - V_\psi(x)\right)^2$$

by the semi-gradient method ([31], Chapter 9).

$$\begin{aligned}\nabla_\psi \mathrm{MSE}(\psi) &= -\sum_{x,y} \nu(x)K_{\alpha_\theta}(x,y)\left[V(y) + R_\theta(x,y) - r(\theta) - V_\psi(x)\right]\nabla_\psi V_\psi(x)\\ &\approx -\sum_{x,y} \nu(x)K_{\alpha_\theta}(x,y)\left[V_\psi(y) + R_\theta(x,y) - r(\theta) - V_\psi(x)\right]\nabla_\psi V_\psi(x)\end{aligned}$$

Here, the term $V(y)$ is frozen first and then approximated by V_ψ since it could be treated as a prior guess of the value function for the future state.

Then, for the gradient descent iteration $\psi_{t+1} = \psi_t - \eta_t^\psi \nabla_\psi \mathrm{MSE}_V(\psi_t)$ where η_t^ψ is the step size, we can have the following stochastic gradient iteration:

$$\psi_{t+1} = \psi_t + \eta_t^\psi\, \delta(X_t, X_{t+1})\, \nabla_\psi V_{\psi_t}(X_t) \tag{23}$$

where the differential temporal (TD) error δ is defined above in (22).

$$\delta_t = \delta(X_t, X_{t+1}) = R_{\theta_t}(X_t, X_{t+1}) - r(\theta_t) + V_{\psi_t}(X_{t+1}) - V_{\psi_t}(X_t).$$

Here, for the sake of simplicity, (X_t, X_{t+1}) are the same samples as in the actor method for θ_t. This means that distribution ν above is chosen as μ used for the gradient $\nabla_\theta r(\theta)$.

Next, we consider the calculation of the reward $r(\theta)$ by the following Bellman Equation (15).

$$\sum_x \mu(x)\sum_y K_{\alpha_\theta}(x,y)(R_\theta(x,y) - r(\theta) + V(y) - V(x)) = 0$$

Let r_t be the estimate of the reward $r(\theta_t)$ at time t. We can update our estimate of the reward every time a transition occurs as follows:

$$r_{t+1} = r_t + \eta_t^r \times \delta_t \tag{24}$$

where δ_t is the TD error before

$$\delta_t = \delta(X_t, X_{t+1}) = R_{\theta_t}(X_t, X_{t+1}) - r_t + V_{\psi_t}(X_{t+1}) - V_{\psi_t}(X_t).$$

In conclusion, (21), (23) and (24) together consist of the actor-critic algorithm, which is summarized in Algorithm 1. We remark that Algorithm 1 can be easily adapted to use the mini-batch gradient method where several copies of (X_t, X_{t+1}) are sampled, and the average is used to update the parameters. The stationary distribution μ_θ of K_{α_θ} is sampled by running the corresponding Markov chain for several steps with "warm start": the initial for θ_{t+1} is set as the final state generated from the previous iteration at θ_t. The length of this "burn-in" period can be set as just one step in practice for efficiency.

Remark 4. *Finally, we remark on the computation of $\nabla_\theta \ln K_{\alpha_\theta}$ and $\nabla_\theta \ln K_{\beta_\theta}$ in Algorithm 1. The details are shown in Appendix A. We comment that the main computational cost is the function $K(x, \mathcal{E})$, which has to be pre-computed and stored. If the problem has some special structure, the*

function could be approximated in parametric form. Another special case is our second example where $K(x, \mathcal{E}) = 0 \quad \forall x \in \{2, 3, \ldots, N\}$.

4. Numerical Experiment

In this section, we present two examples to demonstrate Algorithm 1. We call the algorithm (7), (8) and (9) in Section 2.2 used in [23,24], as **Vanilla Algorithm**, **Projection Algorithm** and **Polyak Averaging Algorithm**, respectively. Let 0 be the absorbing state and $\mathcal{E} = \{1, \ldots, N\}$ are non-absorbing states; the Markov transition matrix on $\{0, \ldots, N\}$ is denoted by the following:

$$\tilde{K} = \begin{bmatrix} 1 & 0 \\ * & K \end{bmatrix},$$

where K is an N-by-N sub-Markovian matrix. For Algorithm 1, distribution α_θ on $\mathcal{E}$ is always parameterized as follows:

$$\alpha_\theta = \frac{1}{e^{\theta_1} + \ldots + e^{\theta_{N-1}} + 1} \left[e^{\theta_1}, \ldots, e^{\theta_{N-1}}, 1 \right],$$

and the value function $V_\psi(x)$ is represented in tabular form for simplicity:

$$V_\psi = [\psi_1, \ldots, \psi_N]$$

where $\psi \in \mathbb{R}^N$.

4.1. Loopy Markov Chain

We test a toy example of the three-state loopy Markov chain, which was considered in [23,24]. The transition probability matrix for the four states $\{0, 1, 2, 3\}$ is as follows.

$$\tilde{K} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \epsilon & \frac{1-\epsilon}{3} & \frac{1-\epsilon}{3} & \frac{1-\epsilon}{3} \\ \epsilon & \frac{1-\epsilon}{3} & \frac{1-\epsilon}{3} & \frac{1-\epsilon}{3} \\ \epsilon & \frac{1-\epsilon}{3} & \frac{1-\epsilon}{3} & \frac{1-\epsilon}{3} \end{bmatrix}, \quad \epsilon \in (0, 1).$$

The state 0 is the absorbing state ∂ and $\mathcal{E} = \{1, 2, 3\}$. K is the sub-matrix of $\tilde{K}$ corresponding to the states $\{1, 2, 3\}$. With the probability ϵ, the process exits $\mathcal{E}$ directly from state 1, 2 or 3. The true quasi-stationary distribution of this example is the uniform distribution for any ϵ.

In order to show the advantage of our algorithm, we consider two cases: (1) $\epsilon = 0.1$ and (2) $\epsilon = 0.9$. For a larger ϵ, the original Markov chain is very easy to exit; thus, each iteration takes less time, but the convergence rate of Vanilla algorithm is slower.

In order to quantify the accuracy of the learned quasi-stationary distribution, we compute the L_2 norm of the error between the learned quasi-stationary distribution and the true values.

In Figure 1, we compute the QSD when $\epsilon = 0.1$. We set the initial value $\theta_0 = [-1, 1]$, $\psi_0 = [0, 0, 0], r_0 = 0$, the learning rate $\eta_n^\theta = \max\{1/n^{0.1}, 0.2\}, \eta_n^\psi = 0.0001, \eta_n^r = 0.0001$ and the batch size is 4. The step size for the Projection Algorithm is $\epsilon_n = n^{-0.99}$. Figure 2 is for the case when $\epsilon = 0.9$ We set the initial value $\theta_0 = [4, -2], \psi_0 = [0, 0, 0], r_0 = 0$, the learning rate $\eta_n^\theta = 0.04, \eta_n^\psi = 0.0001, \eta_n^r = 0.0001$ and the batch size is 32. The step size for the Projection Algorithm is $\epsilon_n = n^{-0.99}$.

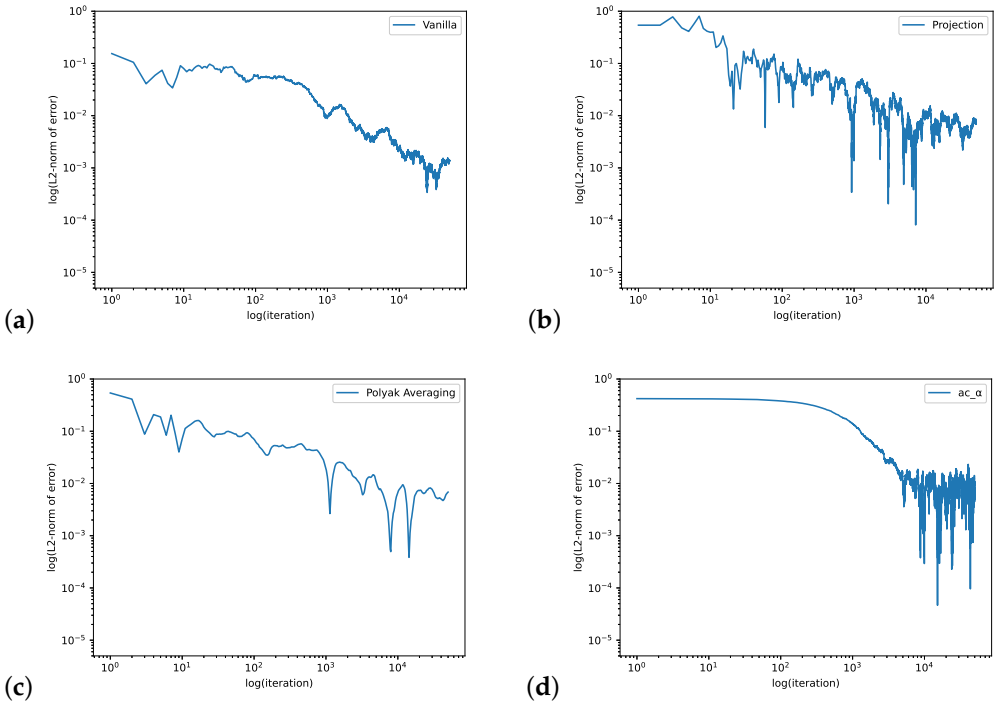

Figure 1. The loopy Markov chain example with $\epsilon = 0.1$. The figure shows the log–log plots of L_2-norm error of the Vanilla Algorithm (**a**), Projection Algorithm (**b**), Polyak Averaging Algorithm (**c**) and our actor-critic algorithm (**d**). The iteration for the actor-critic algorithm is defined as one step of gradient descent ("t" in Algorithm 1).

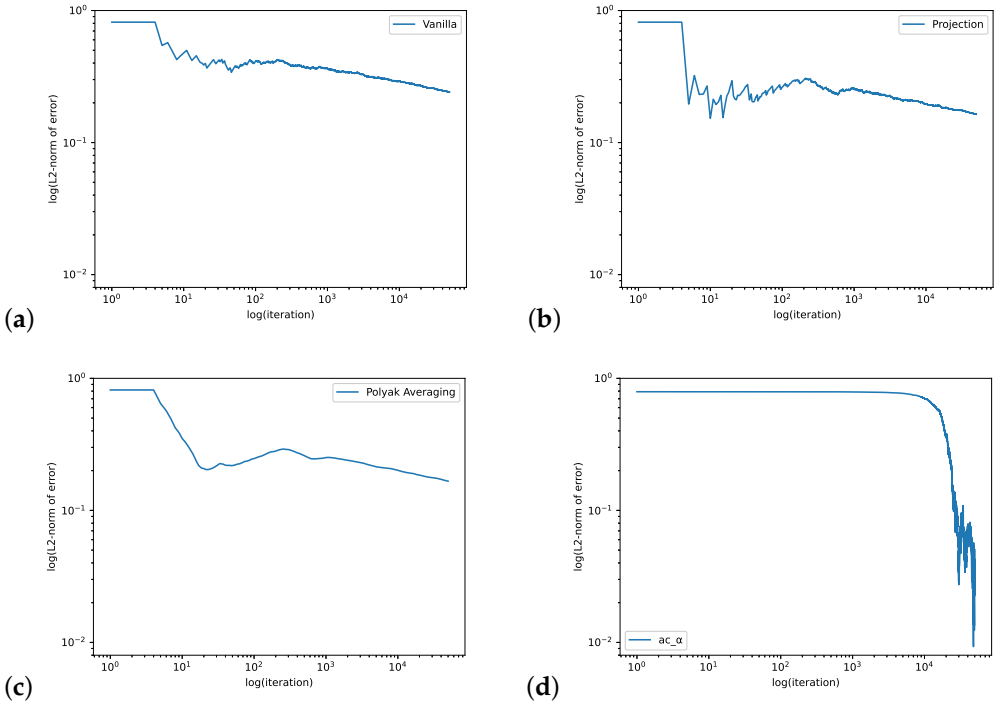

Figure 2. The loopy Markov chain example with $\epsilon = 0.9$. The figure shows the log–log plots of L_2-norm error of Vanilla Algorithm (**a**), Projection Algorithm (**b**), Polyak Averaging Algorithm (**c**) and our actor-critic algorithm (**d**).

4.2. M/M/1/N Queue with Finite Capacity and Absorption

Our second example is an M/M/1 queue with finite queue capacity. The 0 state has been set as an absorbing state. The transition probability matrix on $\{0,\dots,N\}$ takes the following form:

$$\tilde{K} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ \mu_1 & 0 & \lambda_1 & 0 & 0 & \dots & 0 & 0 \\ 0 & \mu_2 & 0 & \lambda_2 & 0 & \dots & 0 & 0 \\ 0 & 0 & \mu_3 & 0 & \lambda_3 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & & 0 & \lambda_{N-1} \\ 0 & 0 & 0 & 0 & 0 & \dots & 1 & 0 \end{bmatrix}$$

where $\lambda_i = \frac{\rho_i}{\rho_i+1}$, $\mu_i = \frac{1}{\rho_i+1}$, $i \in \{1,2,\cdots,N-1\}$. $\rho_i > 1$ means a higher chance to jump to the right than to the left. A larger ρ_i will have less probability of exiting $\mathcal{E}$. Note that $K(x,\mathcal{E}) = 1$ for $x \in \{2,\dots,N\}$. Thus, $K_\alpha(x,y) = K(x,y)$ for any α if $x \neq 1$ and $K_\alpha(1,y) = K(1,y) + \mu_1\alpha(y) = \begin{cases} \lambda_1 + \mu_1\alpha(1) & y = 1, \\ \mu_1\alpha(y) & 2 \le y \le N \end{cases}$. Then, $R_\theta(x,y) = -\ln \frac{K_{\alpha_\theta}(x,y)}{K_{\beta_\theta}(x,y)} = 0$ if $x \neq 1$ and by (20), the gradient is simplified as follows:

$$\nabla_\theta r(\theta) = \mathbf{E}_Y[\big(R_\theta(1,Y) - r(\theta) + V(Y) - V(1)\big)\nabla_\theta \ln K_{\alpha_\theta}(1,Y) + \nabla_\theta \ln K_{\beta_\theta}(1,Y)]$$

where Y follows distribution $K_\alpha(1,\cdot)$.

We consider two cases: (1) a constant $\rho_i = 1.25$ and (2) a state-dependent $\rho_i = 2 - \frac{3}{2N-4}(i-1)$. Note that $\rho_i = 1$ gives an equal probability of jumping to the left and to the right. Thus, in case (1), there is a boundary layer at the most right end and in case (2), we expect to see a peak of the QSD near $i \approx 2N/3$. Figure 3 shows the true QSD in both cases. We set $N = 500$.

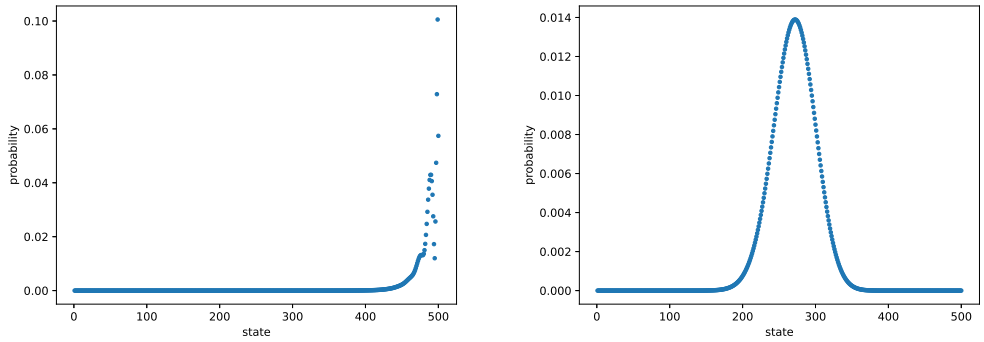

Figure 3. The QSD for M/M/1/500 queue with $\rho_i \equiv 1.25$ (**left**) and $\rho_i = 2 - \frac{3}{2N-4}(i-1)$ (**right**).

In Figure 4, we consider the case when $\rho_i = 1.25$ and compute L_2 errors. We set the initial value $\theta_0^i = -35 + \frac{35}{498}(i-1)$ for $i \in \{1,2,\dots,498\}$ and $\theta_0^{499} = 3$, $\psi_0 = [0,0,\dots,0]$, $r_0 = 0$ and the learning rate $\eta_n^\theta = 0.0003, \eta_n^\psi = 0.0001, \eta_n^r = 0.0001$ and the batch size is 64. The step size for Projection Algorithm is $\epsilon_n = n^{-0.95}$. Figure 5 plots the errors for the state-dependent $\rho_i = 2 - \frac{3}{2N-4}(i-1)$. We set the initial value $\theta_0^i = 8 + \frac{35}{250}(i-1)$ for $i \in \{1,2,\dots,250\}$, $\theta_0^{251} = 44$, $\theta_0^i = 43$ for $i \in \{252,\dots,305\}$, $\theta_0^{306} = 48$, $\theta_0^{307} = 42$ and $\theta_0^i = 43 - \frac{38}{293}(i-1)$ for $i \in \{308,309,\dots,499\}$, $\psi_0 = [0,0,\dots,0]$, $r_0 = 0$ and the learning rate is $\eta_n^\theta = 0.0002, \eta_n^\psi = 0.0001, \eta_n^R = 0.0001$ with the batch size as 128. The step size for the Projection Algorithm is $\epsilon_n = n^{-0.95}$. Both figures demonstrate that actor-critic algorithm performs quite well on this example.

In Table 1, we compared the CPU time of each algorithm in the M/M/1/500 queue when they obtain an accuracy at 2×10^{-1}. We found that our algorithm cost less time on this example.

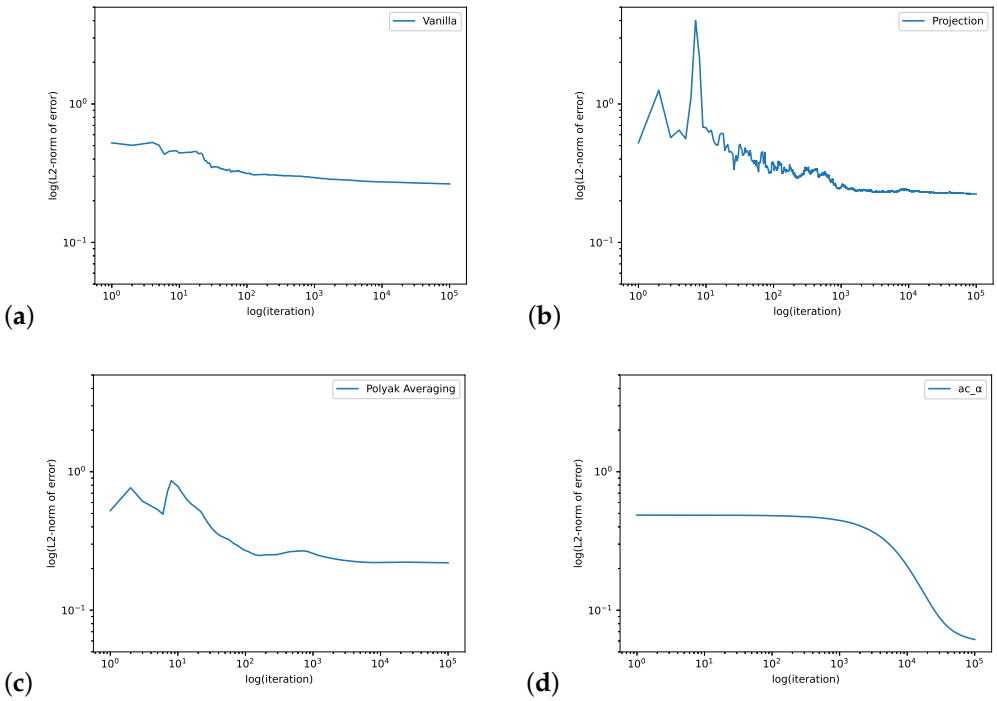

Figure 4. The M/M/1/500 queue with $\rho_i = 1.25$. The figure shows the log–log plots of L_2-norm error of Vanilla Algorithm (**a**), Projection Algorithm (**b**), Polyak Averaging Algorithm (**c**) and our actor-critic algorithm (**d**).

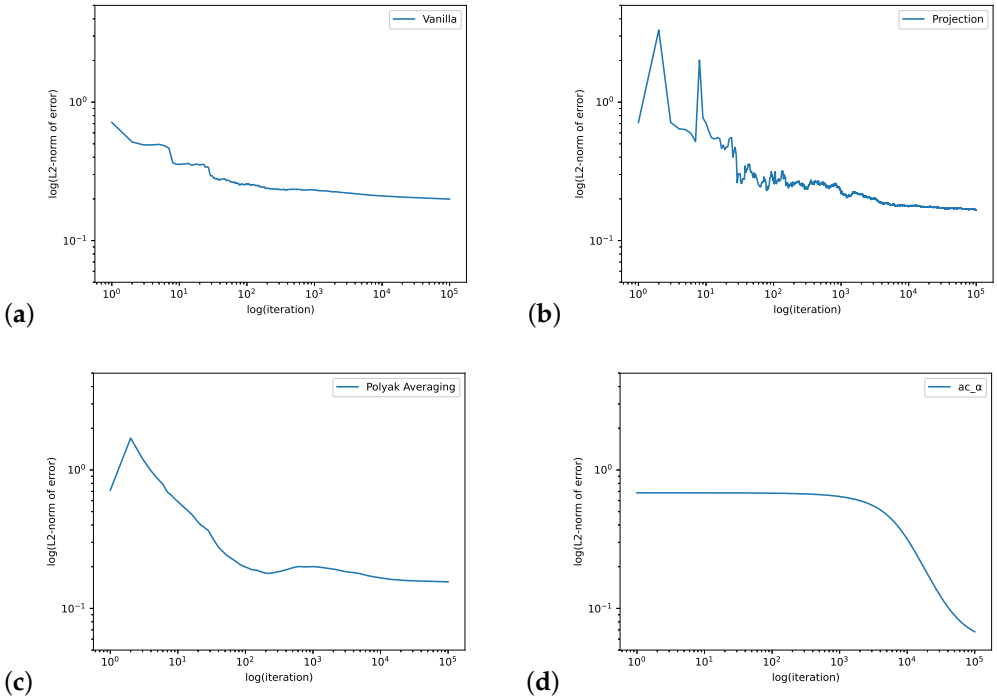

Figure 5. The M/M/1/500 queue with $\rho_i = 2 - \frac{3}{2N-4}(i-1)$. The figure shows the log–log plots of L_2-norm error of Vanilla Algorithm (**a**), Projection Algorithm (**b**), Polyak Averaging Algorithm (**c**) and our actor-critic algorithm (**d**).

Table 1. The CPU time of each algorithm in the M/M/1/500 queue when they obtain the accuracy at 2×10^{-1}.

Algorithm	Vanilla	Projection	Polyak Averaging	ac_α
Time (s)	1038.3279	429.6304	505.2299	186.9280
Time (s)	753.9503	259.0671	268.5476	251.5370

5. Summary and Conclusions

In this paper, we propose a reinforcement learning (RL) method for quasi-stationary distribution (QSD) in discrete time finite-state Markov chains. By minimizing the KL-divergence of two Markovian path distributions induced by the candidate distribution and the true target distribution, we introduce the formulation in terms of RL and derive the corresponding policy gradient theorem. We devise an actor-critic algorithm to learn the QSD in its parameterized form α_θ. This formulation of RL can receive benefit from the development of the RL method and the optimization theory. We illustrated our actor-critic methods on two numerical examples by using simple tabular parametrization and gradient descent optimization. It has been observed that the performance of our method is more prominent for large scale problems.

We only demonstrate the preliminary mechanism of the idea here, and there is much space left for improving the efficiency and extensions in future works. The generalization from the current consideration of finite-state Markov chain to the jump Markov process and the diffusion case is in consideration. More importantly, for very large or high dimensional state space, modern function approximation methods such as kernel methods or neural networks should be used for the distribution α_θ and the value function V_ψ. The recent tremendous advancement of optimization techniques for policy gradient in reinforcement learning could also contribute much to efficiency improvement of our current formulation.

Author Contributions: Conceptualization, Z.C. and L.L.; Investigation, Z.C.; Methodology, Z.C. and X.Z.; Software, Z.C.; Supervision, X.Z.; Writing—original draft, Z.C.; Writing—review & editing, L.L. and X.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Government of Hong Kong, Grant Number 11305318; NSFC Grant Number 11871486.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Acknowledgments: L.L. acknowledges the support of NSFC 11871486. X.Z. acknowledges the support of Hong Kong RGC GRF 11305318.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

In this appendix, we discuss the computation of the gradient of $\nabla_\theta \ln K_{\alpha_\theta}$ and $\nabla_\theta \ln K_{\beta_\theta}$. Note that $\nabla_\theta \alpha_\theta$ is straightforward since we model α in its parametrization form θ. By definition (4), we have the following:

$$\nabla_\theta \ln K_{\alpha_\theta}(X_t, X_{t+1}) = \frac{1 - K(X_t, \mathcal{E})}{K_{\alpha_\theta}(X_t, X_{t+1})} \nabla_\theta \alpha_\theta(X_{t+1})$$

and the following as well:

$$\nabla_\theta \ln K_{\beta_\theta}(X_t, X_{t+1}) = \frac{1 - K(X_t, \mathcal{E})}{K_{\beta_\theta}(X_t, X_{t+1})} \nabla \beta_\theta(X_{t+1}).$$

where $K_\beta(x,y) = K(x,y) + (1 - K(x,\mathcal{E}))\beta(y)$. The vector $K(x,\mathcal{E})$ for any x can be pre-computed and saved in tabular form.

By (19), the one-step distribution β is computed below.

$$\beta(X_{t+1}) = \sum_x \alpha(x)\Big[K(x,X_{t+1}) + (1 - K(x,\mathcal{E}))\alpha(X_{t+1})\Big] \approx \frac{1}{n}\sum_{i=1}^{n} K(Z_i,X_{t+1}) + (1 - K(Z_i,\mathcal{E}))\alpha(X_{t+1})$$

Here, the samples $Z_i \sim \alpha$ could be approximated by stationary distribution μ; thus, one may simply use the known sample X_t to replace Z_i with $n = 1$.

To find $\nabla_\theta \beta_\theta$, we use stochastic approximation again.

$$\begin{aligned}\nabla_\theta \beta_\theta(X_{t+1}) &= \sum_x \nabla\alpha_\theta(x)[K(x,X_{t+1}) + (1 - K(x,\mathcal{E}))\alpha_\theta(X_{t+1})] + \left[\sum_x \alpha_\theta(x)(1 - K(x,\mathcal{E}))\right]\nabla_\theta\alpha_\theta(y),\\ &\approx \frac{1}{n}\sum_{i=1}^{n} \nabla \ln \alpha_\theta(Z_i)[K(Z_i,X_{t+1}) + (1 - K(Z_i,\mathcal{E}))\alpha_\theta(X_{t+1})] + (1 - K(Z_i,\mathcal{E}))\nabla_\theta\alpha_\theta(X_{t+1}).\end{aligned}$$

References

1. Collet, P.; Martínez, S.; Martín, J.S. *Quasi-Stationary Distributions: Markov Chains, Diffusions and Dynamical Systems*; Springer Science & Business Media: Cham, Switzerlands, 2012.
2. Buckley, F.; Pollett, P. Analytical methods for a stochastic mainland–island metapopulation model. *Ecol. Model.* **2010**, *221*, 2526–2530. [CrossRef]
3. Lambert, A. Population dynamics and random genealogies. *Stoch. Model.* **2008**, *24*, 45–163. [CrossRef]
4. De Oliveira, M.M.; Dickman, R. Quasi-stationary distributions for models of heterogeneous catalysis. *Phys. Stat. Mech. Appl.* **2004**, *343*, 525–542. [CrossRef]
5. Dykman, M.I.; Horita, T.; Ross, J. Statistical distribution and stochastic resonance in a periodically driven chemical system. *J. Chem. Phys.* **1995**, *103*, 966–972. [CrossRef]
6. Artalejo, J.R.; Economou, A.; Lopez-Herrero, M.J. Stochastic epidemic models with random environment: Quasi-stationarity, extinction and final size. *J. Math. Biol.* **2013**, *67*, 799–831. [CrossRef] [PubMed]
7. Clancy, D.; Mendy, S.T. Approximating the quasi-stationary distribution of the sis model for endemic infection. *Methodol. Comput. Appl. Probab.* **2011**, *13*, 603–618. [CrossRef]
8. Sani, A.; Kroese, D.; Pollett, P. Stochastic models for the spread of hiv in a mobile heterosexual population. *Math. Biosci.* **2007**, *208*, 98–124. [CrossRef]
9. Chan, D.C.; Pollett, P.K.; Weinstein, M.C. Quantitative risk stratification in markov chains with limiting conditional distributions. *Med. Decis. Mak.* **2009**, *29*, 532–540. [CrossRef]
10. Berglund, N.; Landon, D. Mixed-mode oscillations and interspike interval statistics in the stochastic fitzhugh–nagumo model. *Nonlinearity* **2012**, *25*, 2303. [CrossRef]
11. Landon, D. Perturbation et Excitabilité Dans des Modeles Stochastiques de Transmission de l'Influx Nerveux. Ph.D. Thesis, Université d'Orléans, Orléans, France, 2012.
12. Gesù, G.D.; Lelièvre, T.; Peutrec, D.L.; Nectoux, B. Jump markov models and transition state theory: The quasi-stationary distribution approach. *Faraday Discuss.* **2017**, *195*, 469–495. [CrossRef]
13. Lelièvre, T.; Nier, F. Low temperature asymptotics for quasistationary distributions in a bounded domain. *Anal. PDE* **2015**, *8*, 561–628. [CrossRef]
14. Pollock, M.; Fearnhead, P.; Johansen, A.M.; Roberts, G.O. The scalable langevin exact algorithm: Bayesian inference for big data. *arXiv* **2016**, arXiv:1609.03436.
15. Wang, A.Q.; Roberts, G.O.; Steinsaltz, D. An approximation scheme for quasi-stationary distributions of killed diffusions. *Stoch. Process. Appl.* **2020**, *130*, 3193–3219. [CrossRef]
16. Watkins, D.S. *Fundamentals of Matrix Computations*; John Wiley & Sons: Hoboken, NJ, USA, 2004; Volume 64.
17. Bebbington, M. Parallel implementation of an aggregation/disaggregation method for evaluating quasi-stationary behavior in continuous-time markov chains. *Parallel Comput.* **1997**, *23*, 1545–1559. [CrossRef]
18. Pollett, P.; Stewart, D. An efficient procedure for computing quasi-stationary distributions of markov chains by sparse transition structure. *Adv. Appl. Probab.* **1994**, *26*, 68–79. [CrossRef]
19. Martinez, S.; Martin, J.S. Quasi-stationary distributions for a brownian motion with drift and associated limit laws. *J. Appl. Probab.* **1994**, *31*, 911–920. [CrossRef]
20. Aldous, D.; Flannery, B.; Palacios, J.L. Two applications of urn processes the fringe analysis of search trees and the simulation of quasi-stationary distributions of markov chains. *Probab. Eng. Inform. Sci.* **1988**, *2*, 293–307. [CrossRef]
21. Benaïm, M.; Cloez, B. A stochastic approximation approach to quasi-stationary distributions on finite spaces. *Electron. Commun. Probab.* **2015**, *20*, 1–13. [CrossRef]

22. De Oliveira, M.M.; Dickman, R. How to simulate the quasistationary state. *Phys. Rev. E* **2005**, *71*, 016129. [CrossRef]
23. Blanchet, J.; Glynn, P.; Zheng, S. Analysis of a stochastic approximation algorithm for computing quasi-stationary distributions. *Adv. Appl. Probab.* **2016**, *48*, 792–811. [CrossRef]
24. Zheng, S. Stochastic Approximation Algorithms in the Estimation of Quasi-Stationary Distribution of Finite and General State Space Markov Chains. Ph.D. Thesis, Columbia University, New York, NY, USA, 2014.
25. Kushner, H.; Yin, G.G. *Stochastic Approximation and Recursive Algorithms and Applications*; Springer Science & Business Media: Cham, Switzerlands, 2003; Volume 35.
26. Polyak, B.T.; Juditsky, A.B. Acceleration of stochastic approximation by averaging. *SIAM J. Control. Optim.* **1992**, *30*, 838–855. [CrossRef]
27. Blei, D.M.; Kucukelbir, A.; McAuliffe, J.D. Variational inference: A review for statisticians. *J. Am. Stat. Assoc.* **2017**, *112*, 859–877. [CrossRef]
28. Jordan, M.I.; Ghahramani, Z.; Jaakkola, T.S.; Saul, L.K. An Introduction to Variational Methods for Graphical Models. *Mach. Learn.* **1999**, *37*, 183–233. [CrossRef]
29. Liu, Q.; Wang, D. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *Advances in Neural Information Processing Systems*; Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2016; Volume 29.
30. Rezende, D.; Mohamed, S. Variational inference with normalizing flows. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 7–9 July 2015; Bach, F., Blei, D., Eds.; Volume 37, pp. 1530–1538.
31. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
32. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv* **2013**, arXiv:1312.5602.
33. Popova, M.; Isayev, O.; Tropsha, A. Deep reinforcement learning for de novo drug design. *Sci. Adv.* **2018**, *4*, eaap7885. [CrossRef] [PubMed]
34. Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science* **2018**, *362*, 1140–1144. [CrossRef] [PubMed]
35. Rose, D.C.; Mair, J.F.; Garrahan, J.P. A reinforcement learning approach to rare trajectory sampling. *New J. Phys.* **2021**, *23*, 013013. [CrossRef]
36. Méléard, S.; Villemonais, D. Quasi-stationary distributions and population processes. *Probab. Surv.* **2012**, *9*, 340–410. [CrossRef]
37. Blanchet, J.; Glynn, P.; Zheng, S. Empirical analysis of a stochastic approximation approach for computing quasi-stationary distributions. In *EVOLVE—A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation II*; Schütze, O., Coello, C.A.C., Tantar, A.-A., Tantar, E., Bouvry, P., Moral, P.D., Legrand, P., Eds.; Springer: Berlin/Heidelberg, Germany, 2013; pp. 19–37.
38. Boyd, S.; Boyd, S.P.; Vandenberghe, L. *Convex Optimization*; Cambridge University Press: Cambridge, UK, 2004.
39. Wang, W.; Carreira-Perpinán, M.A. Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application. *arXiv* **2013**, arXiv:1309.1541.