Adaptive Importance Sampling for Value Function Approximation in Off-policy Reinforcement Learning

Abstract

Off-policy reinforcement learning is aimed at efficiently using data samples gathered from a policy that is different from the currently optimized policy. A common approach is to use importance sampling techniques for compensating for the bias of value function estimators caused by the difference between the data-sampling policy and the target policy. However, existing off-policy methods often do not take the variance of the value function estimators explicitly into account and therefore their performance tends to be unstable. To cope with this problem, we propose using an adaptive importance sampling technique which allows us to actively control the trade-off between bias and variance. We further provide a method for optimally determining the trade-off parameter based on a variant of cross-validation. We demonstrate the usefulness of the proposed approach through simulations.

1 Introduction

Policy iteration is a reinforcement learning setup where the optimal policy is obtained by iteratively performing policy evaluation and improvement steps (Sutton & Barto, 1998; Bertsekas & Tsitsiklis, 1996). When policies are updated, many popular policy iteration methods require the user to gather new data samples following the updated policy, and the new samples are used for *value function approximation.* However, this approach is inefficient particularly when the sampling cost is high and it would be more cost-efficient if we could reuse the data collected in the past. A situation where the sampling policy (a policy used for gathering data samples) and the current policy are different is called *off-policy* reinforcement learning (Sutton & Barto, 1998).

In the off-policy setup, simply employing a standard policy iteration method (such as *least-squares* policy iteration (Lagoudakis & Parr, 2003)) does not lead to the optimal policy as the sampling policy can introduce bias into value function approximation. This distribution mismatch problem can be eased by the use of *importance sampling* techniques (**?**), which cancel the bias asymptotically. However, the approximation error is not necessarily small when the bias is reduced by importance sampling; the variance of estimators also needs to be taken into account since the approximation error is the sum of squared bias and variance. Due to large variance, existing importance sampling techniques tend to be unstable (Sutton & Barto, 1998; Precup et al., 2000).

To overcome the instability problem, we propose using an *adaptive importance sampling* technique used in statistics (**?**). The proposed adaptive method, which smoothly bridges the ordinary estimator and importance-weighted estimator, allows us to control the trade-off between bias and variance. Thus, given that the trade-off parameter is determined carefully, the optimal performance can be achieved in terms of both bias and variance. However, the optimal value of the trade-off parameter is heavily dependent on data samples and policies, and therefore using a pre-determined parameter value may not be always effective in practice.

For optimally choosing the value of the trade-off parameter, we propose using an automatic model selection method based on a variant of cross-validation (?). The method called *importance-weighted cross-validation* enables us to estimate the approximation error of value functions in an almost unbiased manner even under off-policy situations. Thus we can adaptively choose the trade-off parameter based on data samples at hand. We demonstrate the usefulness of the proposed approach through simulations.

2 Background and Notation

In this section, we review how Markov decision problems can be solved using policy iteration based on value functions.

2.1 Markov Decision Problems

Let us consider a Markov decision problem (MDP) specified by

$$(\mathcal{S}, \mathcal{A}, P_{\mathrm{T}}, R, \gamma),$$

where

- $\mathcal{S}$ is a set of states,
- $\mathcal{A}$ is a set of actions,
- $P_{\mathrm{T}}(s'|s,a)$ $(\in [0,1])$ is the transition probability-density from state s to next state s' when action a is taken,
- $R(s,a,s')$ $(\in \mathbb{R})$ is a reward for transition from s to s' by taking action a,
- $\gamma \in (0,1]$ is the discount factor for future rewards.

Let $\pi(a|s) \in [0,1]$ be a stochastic policy which is the conditional probability density of taking action a given state s. The state-action value function $Q^\pi(s,a) \in \mathbb{R}$ for policy π is the expected discounted sum of rewards the agent will receive when taking action a in state s and following policy π thereafter, i.e.,

$$Q^\pi(s,a) \equiv \mathop{\mathbb{E}}_{\pi, P_{\mathrm{T}}} \left[\sum_{n=1}^{\infty} \gamma^{n-1} R(s_n, a_n, s_{n+1}) \Big| s_1 = s, a_1 = a \right],$$

where $\mathbb{E}_{\pi,P_{\mathrm{T}}}$ denotes the expectation over $\{s_n, a_n\}_{n=1}^{\infty}$ following $\pi(a_n|s_n)$ and $P_{\mathrm{T}}(s_{n+1}|s_n,a_n)$.

The goal of reinforcement learning is to obtain the policy which maximizes the sum of future rewards; the optimal policy can be expressed[1] as

$$\pi^*(a|s) \equiv \delta(a - \mathop{\arg\max}_{a'} Q^*(s,a')),$$

where $\delta(\cdot)$ is the Dirac delta function and $Q^*(s,a)$ is the *optimal* state-action value function defined by

$$Q^*(s,a) \equiv \max_{\pi} Q^\pi(s,a).$$

$Q^\pi(s,a)$ can be expressed as the following recurrent form called the *Bellman equation* (Sutton & Barto, 1998):

$$Q^\pi(s,a) = R(s,a) + \gamma \mathop{\mathbb{E}}_{P_{\mathrm{T}}(s'|s,a)} \mathop{\mathbb{E}}_{\pi(a'|s')} \left[Q^\pi(s',a') \right], \forall s \in \mathcal{S}, \forall a \in \mathcal{A}, \tag{1}$$

[1]We assume that given state s there is only one action maximizing the optimal value function $Q^*(s,a)$.

where $R(s,a)$ is the expected reward function when the agent takes action a in state s:

$$R(s,a) \equiv \mathop{\mathbb{E}}_{P_{\mathrm{T}}(s'|s,a)} [R(s,a,s')] .$$

$\mathbb{E}_{P_{\mathrm{T}}(s'|s,a)}$ denotes the conditional expectation of s' over $P_{\mathrm{T}}(s'|s,a)$ given s and a. $\mathbb{E}_{\pi(a'|s')}$ denotes the conditional expectation of a' over $\pi(a'|s')$ given s'.

2.2 Policy Iteration

Computing the value function $Q^{\pi}(s,a)$ is called *policy evaluation.* Using $Q^{\pi}(s,a)$, we can find a better policy $\pi'(a|s)$ by

$$\pi'(a|s) = \delta(a - \mathop{\arg\max}_{a'} Q^{\pi}(s,a')).$$

This is called (greedy) *policy improvement.* It is known that repeating policy evaluation and policy improvement results in the optimal policy $\pi^*(a|s)$ (Sutton & Barto, 1998). This entire process is called *policy iteration*:

$$\pi_1 \xrightarrow{E} Q^{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} Q^{\pi_2} \xrightarrow{I} \pi_3 \xrightarrow{E} \cdots \xrightarrow{I} \pi^*,$$

where π_1 is an initial policy, E and I indicate policy *evaluation* and *improvement* steps respectively. For technical reasons, we assume that all policies are strictly positive (i.e., all actions have non-zero probability densities). In order to guarantee this, we use policy improvement which generates explorative policies such as the *Gibbs policy* and the *ϵ-greedy policy.* In the case of the Gibbs policy,

$$\pi'(a|s) = \frac{\exp(Q^{\pi}(s,a)/\tau)}{\int_{\mathcal{A}} \exp(Q^{\pi}(s,a')/\tau)\, da'}, \tag{2}$$

where τ is a positive parameter which determines the randomness of the new policy π'. In the case of the ϵ-greedy policy,

$$\pi'(a|s) = \begin{cases} 1 - \epsilon + \epsilon/|\mathcal{A}| & \text{if } a = a^*, \\ \epsilon/|\mathcal{A}| & \text{otherwise,} \end{cases} \tag{3}$$

where

$$a^* = \mathop{\arg\max}_{a} Q^{\pi}(s,a),$$

and $\epsilon \in (0,1]$ determines how stochastic the new policy π' is.

2.3 Value Function Approximation

Although policy iteration is guaranteed to produce the optimal policy, it is often computationally intractable since the number of state-action pairs $|\mathcal{S}| \times |\mathcal{A}|$ is very large; $|\mathcal{S}|$ or $|\mathcal{A}|$ becomes infinite when the state space or action space is continuous. To overcome this problem, the authors of the references (Sutton & Barto, 1998; Precup et al., 2001; Lagoudakis & Parr, 2003) proposed to approximate the state-action value function $Q^{\pi}(s,a)$ using the following linear model:

$$\widehat{Q}^{\pi}(s,a;\boldsymbol{\theta}) \equiv \sum_{b=1}^{B} \theta_b \phi_b(s,a) = \boldsymbol{\theta}^{\top} \boldsymbol{\phi}(s,a),$$

where

$$\boldsymbol{\phi}(s,a) = (\phi_1(s,a), \phi_2(s,a), \ldots, \phi_B(s,a))^{\top}$$

are the fixed basis functions, $\top$ denotes the transpose, B is the number of basis functions, and

$$\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_B)^{\top}$$

are model parameters. Note that B is usually chosen to be much smaller than $|\mathcal{S}| \times |\mathcal{A}|$. For N-step transitions, we ideally want to learn the parameters $\boldsymbol{\theta}$ so that the approximation error is minimized:

$$\min_{\boldsymbol{\theta}} \mathop{\mathbb{E}}_{P_{\mathrm{I}},\pi,P_{\mathrm{T}}} \left[\frac{1}{N} \sum_{n=1}^{N} \left(\widehat{Q}^{\pi}(s_n, a_n; \boldsymbol{\theta}) - Q^{\pi}(s_n, a_n) \right)^2 \right],$$

where $\mathbb{E}_{P_{\mathrm{I}},\pi,P_{\mathrm{T}}}$ denotes the expectation over $\{s_n, a_n\}_{n=1}^{N}$ following the initial-state probability density $P_{\mathrm{I}}(s_1)$, the policy $\pi(a_n|s_n)$, and the transition probability density $P_{\mathrm{T}}(s_{n+1}|s_n, a_n)$.

A fundamental problem of the above formulation is that the target function $Q^{\pi}(s, a)$ cannot be observed directly. To cope with this problem, we use the square of the Bellman residual (Schoknecht, 2003; Lagoudakis & Parr, 2003) as

$$\begin{aligned} \boldsymbol{\theta}^* &\equiv \mathop{\arg\min}_{\boldsymbol{\theta}} G, \\ G &\equiv \mathop{\mathbb{E}}_{P_{\mathrm{I}},\pi,P_{\mathrm{T}}} \left[\frac{1}{N} \sum_{n=1}^{N} g(s_n, a_n; \boldsymbol{\theta}) \right], \\ g(s, a; \boldsymbol{\theta}) &\equiv \left(\widehat{Q}^{\pi}(s, a; \boldsymbol{\theta}) - R(s, a) - \gamma \mathop{\mathbb{E}}_{P_{\mathrm{T}}(s'|s,a)} \mathop{\mathbb{E}}_{\pi(a'|s)} \left[\widehat{Q}^{\pi}(s', a'; \boldsymbol{\theta}) \right] \right)^2, \end{aligned} \tag{4}$$

where $g(s, a; \boldsymbol{\theta})$ is the approximation error for one step (s, a) derived from the Bellman equation[2] (1).

2.4 On-policy vs. Off-policy

We suppose that a dataset consisting of M episodes of N steps is available. The agent initially starts from a randomly selected state s_1 following the initial-state probability density $P_{\mathrm{I}}(s)$ and chooses an action based on a *sampling policy* $\widetilde{\pi}(a_n|s_n)$. Then the agent makes a transition following $P_{\mathrm{T}}(s_{n+1}|s_n, a_n)$ and receives a reward r_n $(= R(s_n, a_n, s_{n+1}))$. This is repeated for N steps—thus the training data $\mathcal{D}^{\widetilde{\pi}}$ is expressed as

$$\mathcal{D}^{\widetilde{\pi}} \equiv \{d_m^{\widetilde{\pi}}\}_{m=1}^{M},$$

where each episodic sample $d_m^{\widetilde{\pi}}$ consists of a set of 4-tuple elements as

$$d_m^{\widetilde{\pi}} \equiv \{(s_{m,n}^{\widetilde{\pi}}, a_{m,n}^{\widetilde{\pi}}, r_{m,n}^{\widetilde{\pi}}, s_{m,n+1}^{\widetilde{\pi}})\}_{n=1}^{N}.$$

We use two types of policies which have different purposes: the *sampling policy* $\widetilde{\pi}(a|s)$ for collecting data samples and the *current policy* $\pi(a|s)$ for computing the value function $\widehat{Q}^{\pi}$. When $\widetilde{\pi}(a|s)$ is equal to $\pi(a|s)$ (the situation called *on-policy*), just replacing the expectation contained in the error G by sample averages gives a *consistent* estimator (i.e., the estimated parameter converges to the optimal value as the number M of episodes goes to infinity):

$$\begin{aligned} \widehat{\boldsymbol{\theta}}_{\mathrm{NIW}} &\equiv \mathop{\arg\min}_{\boldsymbol{\theta}} \widehat{G}_{\mathrm{NIW}}, \\ \widehat{G}_{\mathrm{NIW}} &\equiv \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} \widehat{g}_{m,n}, \\ \widehat{g}_{m,n} &\equiv \widehat{g}(s_{m,n}^{\widetilde{\pi}}, a_{m,n}^{\widetilde{\pi}}; \boldsymbol{\theta}, \mathcal{D}^{\widetilde{\pi}}), \\ \widehat{g}(s, a; \boldsymbol{\theta}, \mathcal{D}) &\equiv \left(\widehat{Q}^{\pi}(s, a; \boldsymbol{\theta}) - \frac{1}{|\mathcal{D}_{(s,a)}|} \sum_{r \in \mathcal{D}_{(s,a)}} r - \frac{\gamma}{|\mathcal{D}_{(s,a)}|} \sum_{s' \in \mathcal{D}_{(s,a)}} \mathop{\mathbb{E}}_{\pi(a'|s')} [\widehat{Q}^{\pi}(s', a'; \boldsymbol{\theta})] \right)^2, \end{aligned}$$

[2]We note that $g(s, a; \boldsymbol{\theta})$ with a reward observation r instead of the expected reward $R(s, a)$ corresponds to the square of the TD error. That is,

$$g_{\mathrm{TD}}(s, a, r; \boldsymbol{\theta}) \equiv \left(\widehat{Q}^{\pi}(s, a, \boldsymbol{\theta}) - r - \gamma \mathop{\mathbb{E}}_{P_{\mathrm{T}}(s'|s,a)} \mathop{\mathbb{E}}_{\pi(a'|s)} \left[\widehat{Q}^{\pi}(s', a'; \boldsymbol{\theta}) \right] \right)^2.$$

In this paper, the Bellman residual is used to measure the approximation error but we can easily replace it with the TD error.

where $\mathcal{D}_{(s,a)}$ is a set of 4-tuple elements containing state s and action a in the training data $\mathcal{D}$, and $\sum_{r\in\mathcal{D}_{(s,a)}}$ and $\sum_{s'\in\mathcal{D}_{(s,a)}}$ denote the summation over r and s' in the set $\mathcal{D}_{(s,a)}$, respectively. Note that 'NIW' stands for 'No Importance Weight' (explained later).

However, $\widetilde{\pi}(a|s)$ is usually different from $\pi(a|s)$ in reality since the current policy is updated in policy iteration. The situation where $\widetilde{\pi}(a|s)$ is different from $\pi(a|s)$ is called *off-policy.* In the off-policy setup, $\widehat{\boldsymbol{\theta}}_{\mathrm{NIW}}$ is no longer consistent. This inconsistency problem could be avoided by gathering new samples, i.e., when the current policy is updated, new samples are gathered following the updated policy and the new samples are used for policy evaluation. However, when the data sampling cost is high, this is not cost-efficient—it would be more efficient if we could reuse the previously gathered samples.

In the following sections, we address the issue of sample reuse in the off-policy setup.

3 Importance-weighting Techniques

In this section, we review existing off-policy reinforcement learning techniques.

3.1 Importance Sampling

Importance sampling is a general technique for dealing with the off-policy situation. Suppose we have i.i.d. (*independent and identically distributed*) samples $\{x_m\}_{m=1}^{M}$ from a strictly positive probability density function $\widetilde{P}(x)$. Using these samples, we would like to compute the expectation of a function $g(x)$ over another probability density function $P(x)$. A consistent approximation of the expectation is given by the *importance-weighted* average as follows:

$$\begin{aligned}\frac{1}{M}\sum_{m=1}^{M} g(x_m)\frac{P(x_m)}{\widetilde{P}(x_m)} &\overset{M\to\infty}{\longrightarrow} \mathop{\mathbb{E}}_{\widetilde{P}(x)}\left[g(x)\frac{P(x)}{\widetilde{P}(x)}\right]\\ &= \int g(x)\frac{P(x)}{\widetilde{P}(x)}\widetilde{P}(x)dx = \int g(x)P(x)dx = \mathop{\mathbb{E}}_{P(x)}\left[g(x)\right].\end{aligned}$$

However, applying the importance sampling technique in off-policy reinforcement learning is not that straightforward since our training samples of state s and action a are not i.i.d. due to the sequential nature of MDPs. Below, we review existing importance-weighting techniques in MDPs.

3.2 Episodic Importance-weights

In the reference (Sutton & Barto, 1998), the *episodic importance-weight* (EIW) method was proposed which utilizes the independence between episodes:

$$P(d,d') = P(d)P(d') = P(s_1,a_1,\ldots,s_N,a_N,s_{N+1})P(s'_1,a'_1,\ldots,s'_N,a'_N,s'_{N+1}).$$

Based on the independence between episodes, the error G defined by Eq.(4) can be rewritten as

$$G = \mathop{\mathbb{E}}_{P_{\mathrm{I}},\widetilde{\pi},P_{\mathrm{T}}}\left[\frac{1}{N}\sum_{n=1}^{N} g(s_n,a_n;\boldsymbol{\theta})w_N\right],$$

where

$$w_N \equiv \frac{P_{\pi}(d)}{P_{\widetilde{\pi}}(d)}.$$

$P_\pi(d)$ and $P_{\widetilde{\pi}}(d)$ are the probability densities of observing episodic data d under policy π and $\widetilde{\pi}$:

$$P_\pi(d) \equiv P_{\rm I}(s_1) \prod_{n=1}^{N} \pi(a_n|s_n) P_{\rm T}(s_{n+1}|s_n, a_n),$$
$$P_{\widetilde{\pi}}(d) \equiv P_{\rm I}(s_1) \prod_{n=1}^{N} \widetilde{\pi}(a_n|s_n) P_{\rm T}(s_{n+1}|s_n, a_n).$$

We note that the importance weights can be computed without explicitly knowing $P_{\rm I}$ and $P_{\rm T}$ since they are canceled out:

$$w_N = \frac{\prod_{n=1}^{N} \pi(a_n|s_n)}{\prod_{n=1}^{N} \widetilde{\pi}(a_n|s_n)}.$$

Using the training data $\mathcal{D}^{\widetilde{\pi}}$, we can construct a consistent estimator of G as

$$\widehat{G}_{\rm EIW} \equiv \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} \widehat{g}_{m,n} \widehat{w}_{m,N}, \tag{5}$$

where

$$\widehat{w}_{m,N} \equiv \frac{\prod_{n'=1}^{N} \pi(a^{\widetilde{\pi}}_{m,n'}|s^{\widetilde{\pi}}_{m,n'})}{\prod_{n'=1}^{N} \widetilde{\pi}(a^{\widetilde{\pi}}_{m,n'}|s^{\widetilde{\pi}}_{m,n'})}.$$

Based on this, the parameter $\boldsymbol{\theta}$ is estimated by

$$\widehat{\boldsymbol{\theta}}_{\rm EIW} \equiv \arg\min_{\boldsymbol{\theta}} \widehat{G}_{\rm EIW}.$$

3.3 Per-decision Importance-weights

In the reference (Precup et al., 2000), a more efficient importance sampling technique called the *per-decision importance-weight* (PDIW) method was proposed. A crucial observation in PDIW is that the error at the n-th step does not depend on the samples after the n-th step, i.e., the error G can be rewritten as

$$G = \mathop{\mathbb{E}}_{P_{\rm I}, \widetilde{\pi}, P_{\rm T}} \left[\frac{1}{N} \sum_{n=1}^{N} g(s_n, a_n; \boldsymbol{\theta}) w_n \right].$$

Using the training data $\mathcal{D}^{\widetilde{\pi}}$, we can construct a consistent estimator as follows (cf. Eq.(5))

$$\widehat{G}_{\rm PDIW} \equiv \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} \widehat{g}_{m,n} \widehat{w}_{m,n}. \tag{6}$$

$\widehat{w}_{m,n}$ in Eq.(6) *only* contains the relevant terms up to the n-th step, while $\widehat{w}_{m,N}$ in Eq.(5) includes all the terms until the end of the episode.

Based on this, the parameter $\boldsymbol{\theta}$ is estimated by

$$\widehat{\boldsymbol{\theta}}_{\rm PDIW} \equiv \arg\min_{\boldsymbol{\theta}} \widehat{G}_{\rm PDIW}.$$

4 Adaptive Per-decision Importance-weights

The importance-weighted estimator $\widehat{\boldsymbol{\theta}}_{\rm PDIW}$ (also $\widehat{\boldsymbol{\theta}}_{\rm EIW}$) is guaranteed to be consistent. However, both are not *efficient* in the statistical sense (?), i.e., they do not have the smallest admissible variance[3]. For this reason, $\widehat{\boldsymbol{\theta}}_{\rm PDIW}$ can have large variance in finite sample cases and therefore learning with PDIW could be unstable in practice. In this section, we propose a new importance-weighting method that is more stable than the existing methods.

[3]More precisely, an estimator $\widehat{\boldsymbol{\theta}}$ is said to be efficient if it is unbiased and achieves the Cramér-Rao lower-bound (?).

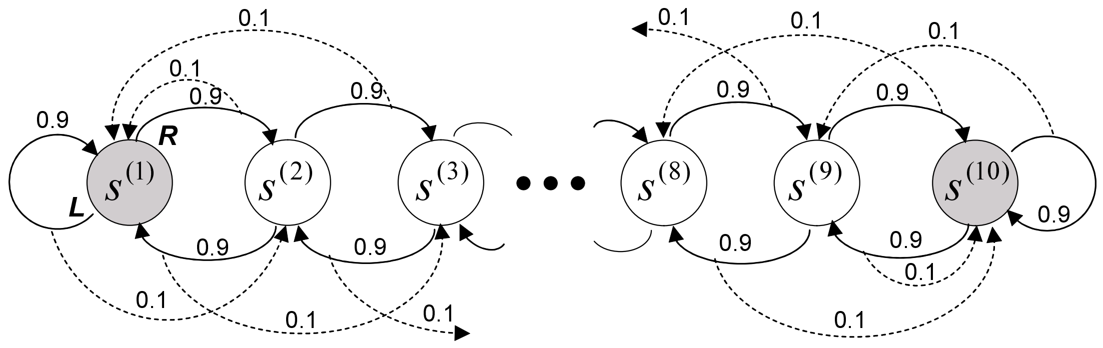


Figure 1: 10-state chain-walk MDP

4.1 Definition

In order to improve the estimation accuracy, it is important to control the trade-off between consistency and efficiency (or similarly bias and variance) based on the training data. Here, we introduce a *flattening parameter* ν ($\in [0,1]$) to control the trade-off by slightly 'flattening' the importance weights (?; ?):

$$\widehat{G}_{\text{APDIW}} \equiv \frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}\widehat{g}_{m,n}(\widehat{w}_{m,n})^{\nu}, \tag{7}$$

where APDIW means Adaptive PDIW. Based on this, the parameter $\boldsymbol{\theta}$ is estimated as follows:

$$\widehat{\boldsymbol{\theta}}_{\text{APDIW}} \equiv \operatorname*{arg\,min}_{\boldsymbol{\theta}} \widehat{G}_{\text{APDIW}}.$$

When $\nu = 0$, $\widehat{\boldsymbol{\theta}}_{\text{APDIW}}$ is reduced to the ordinary estimator $\widehat{\boldsymbol{\theta}}_{\text{NIW}}$. Therefore, it has large bias but has relatively small variance. On the other hand, when $\nu = 1$, $\widehat{\boldsymbol{\theta}}_{\text{APDIW}}$ is reduced to the importance-weighted estimator $\widehat{\boldsymbol{\theta}}_{\text{PDIW}}$. Therefore, it has small bias but has relatively large variance. In practice, an intermediate ν will yield the best performance.

Regarding the computation of $\widehat{\boldsymbol{\theta}}_{\text{APDIW}}$, we have the following lemma.

Lemma 1 *The solution $\widehat{\boldsymbol{\theta}}_{\text{APDIW}}$ can be computed analytically as follows:*

$$\begin{aligned}\widehat{\boldsymbol{\theta}}_{\text{APDIW}} = &\left(\sum_{m=1}^{M}\sum_{n=1}^{N}\widehat{\boldsymbol{\psi}}(s^{\widetilde{\pi}}_{m,n}, a^{\widetilde{\pi}}_{m,n}; \mathcal{D}^{\widetilde{\pi}})\widehat{\boldsymbol{\psi}}(s^{\widetilde{\pi}}_{m,n}, a^{\widetilde{\pi}}_{m,n}; \mathcal{D}^{\widetilde{\pi}})^{\top}(\widehat{w}_{m,n})^{\nu}\right)^{-1} \\ &\times\left(\sum_{m=1}^{M}\sum_{n=1}^{N}\left(\frac{1}{|\mathcal{D}_{(s^{\widetilde{\pi}}_{m,n}, a^{\widetilde{\pi}}_{m,n})}|}\sum_{r\in\mathcal{D}_{(s^{\widetilde{\pi}}_{m,n}, a^{\widetilde{\pi}}_{m,n})}} r\right)\widehat{\boldsymbol{\psi}}(s^{\widetilde{\pi}}_{m,n}, a^{\widetilde{\pi}}_{m,n}; \mathcal{D}^{\widetilde{\pi}})(\widehat{w}_{m,n})^{\nu}\right),\end{aligned} \tag{8}$$

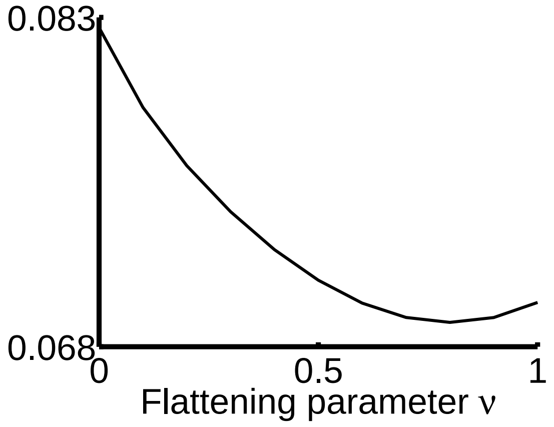
where $\widehat{\boldsymbol{\psi}}(s, a; \mathcal{D})$ is a B-dimensional column vector defined by

$$\widehat{\boldsymbol{\psi}}(s, a; \mathcal{D}) \equiv \boldsymbol{\phi}(s, a) - \frac{\gamma}{|\mathcal{D}_{(s,a)}|}\sum_{s'\in\mathcal{D}_{(s,a)}}\mathop{\mathbb{E}}_{\pi(a'|s')}\left[\boldsymbol{\phi}(s', a')\right].$$

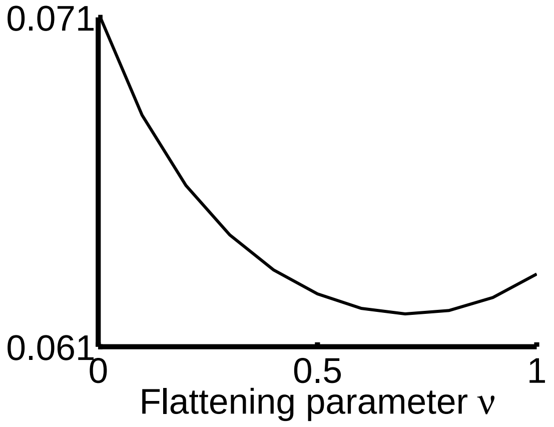
The proof of Lemma 1 is given in Appendix. This lemma implies that the cost for computing $\widehat{\boldsymbol{\theta}}_{\text{APDIW}}$ is essentially the same as $\widehat{\boldsymbol{\theta}}_{\text{NIW}}$ and $\widehat{\boldsymbol{\theta}}_{\text{PDIW}}$.

4.2 Numerical Examples

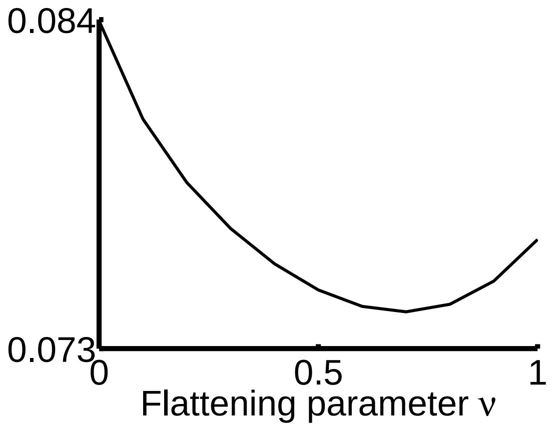
In order to illustrate how the flattening parameter ν influences the estimator $\widehat{\boldsymbol{\theta}}_{\text{APDIW}}$, we perform policy evaluation in the *chain-walk MDP* illustrated in Fig.1.



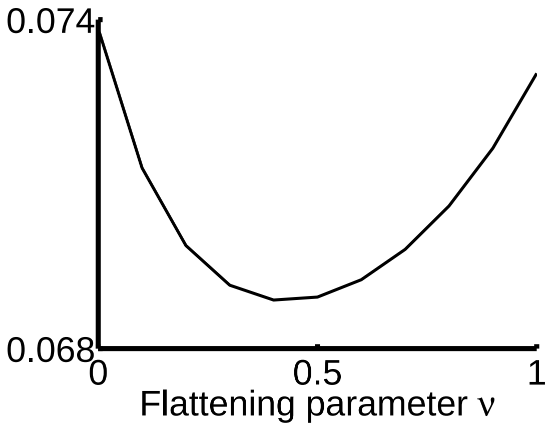
(a) 50 episodes



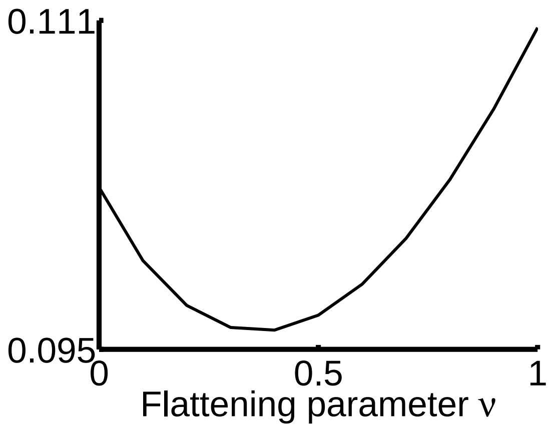
(a) 50 episodes



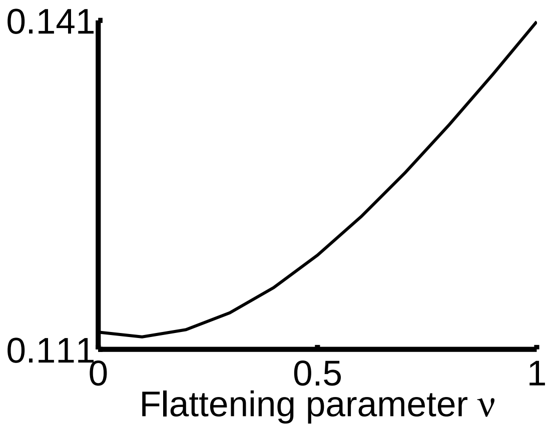
(b) 30 episodes



(b) 30 episodes



(c) 10 episodes



(c) 10 episodes

Figure 2: True approximation error G averaged over 50 trials as a function of the flattening parameter ν in the 10-state chain-walk problem. The number of steps is fixed at $N = 10$. The trend of G differs depending on the number of episodes.

Figure 3: Average approximation error estimated by 5-fold IWCV ($\widehat{G}_{\mathrm{IWCV}}$) as a function of the flattening parameter ν in the 10-state chain-walk problem. The number of steps is fixed at $N = 10$. IWCV nicely captures the trend of the true approximation error G (cf. Fig.2).

The MDP consists of 10 states

$$\mathcal{S} = \{s^{(i)}\}_{i=1}^{10}$$

and two actions

$$\mathcal{A} = \{a^{(i)}\}_{i=1}^{2} = \{L, R\}.$$

The reward +1 is given when visiting $s^{(1)}$ and $s^{(10)}$. The transition probability P_{T} is indicated by the numbers attached to the arrows in the figure; for example, $P_{\mathrm{T}}(s^{(2)}|s^{(1)}, R) = 0.9$ and $P_{\mathrm{T}}(s^{(1)}|s^{(1)}, R) = 0.1$, meaning that the agent can successfully move to the right node with probability 0.9 (indicated by solid arrows in the figure) and the action fails with probability 0.1 (indicated by dashed arrows in the figure). We use 6 Gaussian kernels with standard deviation $\sigma = 10$ as basis functions and locate kernel centers at $s^{(1)}$, $s^{(5)}$ and $s^{(10)}$. More specifically, the basis functions $\boldsymbol{\phi}(s,a) = (\phi_1(s,a), \ldots, \phi_6(s,a))$ are defined by

$$\phi_{3(i-1)+j}(s,a) = I(a = a^{(i)})\exp\left(-\frac{(s-c_j)^2}{2\sigma^2}\right), \tag{9}$$

for $i = 1, 2$ and $j = 1, 2, 3$, where

$$c_1 = 1, c_2 = 5, c_3 = 10,$$

and

$$I(x) = \begin{cases} 1 & \text{if } x \text{ is true,} \\ 0 & \text{if } x \text{ is not true,} \end{cases}$$

We ran the experiments 50 times; the sampling policy $\widetilde{\pi}(a|s)$ and the current policy $\pi(a|s)$ were chosen randomly at every trial such that $\widetilde{\pi} \neq \pi$. The discount factor was set at $\gamma = 0.9$. The model parameter $\widehat{\boldsymbol{\theta}}_{\mathrm{APDIW}}$ was learned from the training samples $\mathcal{D}_{\widetilde{\pi}}$ and its approximation error was computed from the test samples $\mathcal{D}_{\pi}$.

Fig.2 depicts the true approximation error G averaged over 50 trials as a function of the flattening parameter ν for $M = 10, 30, 50$. Fig.2(a) shows that when the number of episodes is large ($M = 50$), the approximation error tends to decrease as the flattening parameter increases. This would be a natural result due to the consistency of $\widehat{\boldsymbol{\theta}}_{\mathrm{APDIW}}$ when $\nu = 1$. On the other hand, Fig.2(b) shows that when the number of episodes is not large ($M = 30$), $\nu = 1$ performs rather poorly. This implies that the consistent estimator tends to be unstable when the number of episodes is not large enough—$\nu = 0.7$ works the best in this case. Fig.2(c) shows the results when the number of episodes is further reduced ($M = 10$). This illustrates that the consistent estimator with $\nu = 1$ is even worse than the ordinary estimator ($\nu = 0$) because the bias is dominated by large variance. In this case, the best ν is even smaller and is achieved at $\nu = 0.4$. The above results show that APDIW can outperform PDIW particularly when only a small number of training samples are available, provided that the flattening parameter ν is chosen appropriately.

5 Automatic Selection of the Flattening Parameter

Generally, the best ν tends to be large (small) when the number of training samples is large (small). However, this general trend is not sufficient to fine-tune the flattening parameter since the best value of ν depends on training samples, policies, the model of value functions etc. In this section, we discuss how we perform *model selection* in order to choose the best flattening parameter ν automatically from the training data and policies.

5.1 Importance-weighted Cross-validation

As shown in Section 4, the performance of APDIW depends on the choice of the flattening parameter ν. Ideally, we set ν so that the approximation error G is minimized, but true G is inaccessible in practice. To cope with this problem, we estimate the approximation error G using *importance-weighted cross-validation* (IWCV) (?). The basic idea of IWCV is to divide the training data $\mathcal{D}^{\widetilde{\pi}}$ into a 'training part' and a 'validation part'. Then the parameter $\boldsymbol{\theta}$ is learned from the training part and its approximation error is estimated using the validation part. Below we explain in more detail how we apply IWCV to the selection of the flattening parameter ν in the current context.

Let us divide a training dataset $\mathcal{D}^{\widetilde{\pi}}$ containing M episodes into K subsets $\{\mathcal{D}^{\widetilde{\pi}}_k\}_{k=1}^K$ of approximately the same size (typically $K = 5$). For simplicity, we assume that M is divisible by K. Let $\widehat{\boldsymbol{\theta}}^k_{\mathrm{APDIW}}$ be the parameter learned from $\{\mathcal{D}^{\widetilde{\pi}}_{k'}\}_{k' \neq k}$ with APDIW (cf. Eq.(7)). Then, the approximation error is estimated by

$$\widehat{G}_{\mathrm{IWCV}} = \frac{1}{K} \sum_{k=1}^{K} \widehat{G}^k_{\mathrm{IWCV}},$$

where

$$\widehat{G}^k_{\mathrm{IWCV}} = \frac{K}{MN} \sum_{d^{\widetilde{\pi}}_m \in \mathcal{D}^{\widetilde{\pi}}_k} \sum_{n=1}^{N} \widehat{g}(s^{\widetilde{\pi}}_{m,n}, a^{\widetilde{\pi}}_{m,n}, r^{\widetilde{\pi}}_{m,n}; \widehat{\boldsymbol{\theta}}^k_{\mathrm{APDIW}}, \mathcal{D}^{\widetilde{\pi}}_k)\widehat{w}_{m,n}.$$

We estimate the approximation error by the above K-fold IWCV method for all candidate models (in the current setting, a candidate model corresponds to a different value of the flattening parameter ν) and choose the one that minimizes the estimated error:

$$\widehat{\nu}_{\mathrm{IWCV}} = \arg\min_{\nu} \widehat{G}_{\mathrm{IWCV}}.$$

One may think that, for model selection, $\widehat{G}_{\mathrm{APDIW}}$ could be directly used, instead of $\widehat{G}_{\mathrm{IWCV}}$. However, it can be proven that $\widehat{G}_{\mathrm{APDIW}}$ is heavily biased (or in other words, *over-fitted*) since the same training samples are used twice for learning parameters and estimating the approximation error (?). On the other hand, we can prove that $\widehat{G}_{\mathrm{IWCV}}$ is an almost unbiased estimator of G, where 'almost' comes from the fact that the number of training samples is reduced due to data splitting in the cross-validation procedure (?). Note that ordinary CV (without importance weight) is heavily biased due to the off-policy setup.

In general, the use of IWCV is computationally rather expensive since $\widehat{\boldsymbol{\theta}}^k_{\mathrm{APDIW}}$ and $\widehat{G}^k_{\mathrm{IWCV}}$ need to be computed many times. For example, when performing 5-fold IWCV for 11 candidates of the flattening parameter $\nu \in \{0.0, 0.1, \ldots, 0.9, 1.0\}$, we need to compute $\widehat{\boldsymbol{\theta}}^k_{\mathrm{APDIW}}$ and $\widehat{G}^k_{\mathrm{IWCV}}$ 55 times. However, we argue that this is acceptable in practice due to the following two reasons. First, sensible model selection via IWCV allows us to obtain a much better solution with a small number of samples. Thus, in total, the computation time may not grow that much. The second reason is that cross-validation is suitable for paralell computing since error estimation for different flattening parameters and different folds are independent of each other. For instance, when performing 5-fold IWCV for 11 candidates of the flattening parameter, we can compute $\widehat{G}_{\mathrm{IWCV}}$ for all candidates at once in parallel using 55 CPUs; this is highly realistic in the current computing environment.

5.2 Numerical Examples

In order to illustrate how IWCV works, we use the same numerical examples as Section 4.2. Fig.3 depicts the approximation error estimated by 5-fold IWCV averaged over 50 trials as a function of the flattening parameter ν. The graphs show that IWCV nicely captures the trend of the true approximation error for all three cases (cf. Fig.2).

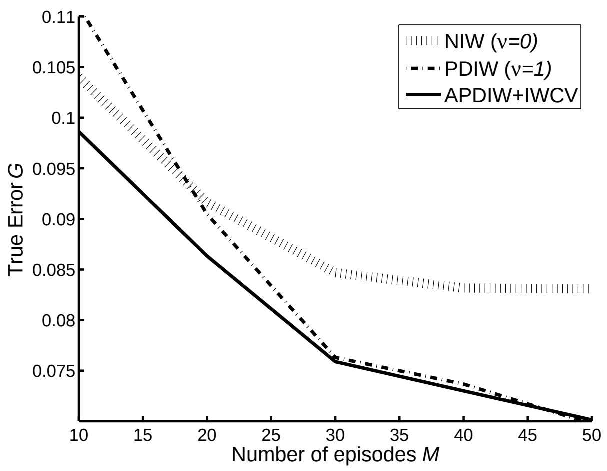


Figure 4: Average true approximation error G over 50 trials obtained by NIW ($\nu = 0$), PDIW ($\nu = 1$), APDIW+IWCV (ν is chosen by IWCV) in the 10-state chain-walk MDP.

Fig.4 describes, as a function of the number M of episodes, the average true approximation error obtained by NIW (APDIW with $\nu = 0$), PDIW (APDIW with $\nu = 1$), APDIW+IWCV ($\nu \in \{0.0, 0.1, \ldots, 0.9, 1.0\}$ is selected in each trial using 5-fold IWCV). This result shows that the improvement of the performance by NIW saturates when $M \geq 30$, implying that the bias caused by NIW is not negligible. The performance of PDIW is worse than NIW when $M \leq 20$, which is caused by the large variance of PDIW. On the other hand, APDIW+IWCV consistently gives good performance for all M, illustrating the strong adaptation ability of the proposed method.

6 Sample-reuse Policy Iteration (SRPI)

So far, we have only used our proposed APDIW+IWCV method in the context of policy evaluation. In this section, we extend the method to the full policy-iteration setup.

6.1 Algorithm

Let us denote the policy at the l-th iteration by π_l and the maximum number of iterations by L. In general policy-iteration methods, new data samples $\mathcal{D}^{\pi_l}$ are collected following the new policy π_l during the policy evaluation step. Thus, previously-collected data samples $\{\mathcal{D}^{\pi_1}, \mathcal{D}^{\pi_2}, ..., \mathcal{D}^{\pi_{l-1}}\}$ are not used:

$$\pi_1 \xrightarrow{E:\{\mathcal{D}^{\pi_1}\}} \widehat{Q}^{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E:\{\mathcal{D}^{\pi_2}\}} \widehat{Q}^{\pi_2} \xrightarrow{I} \pi_3 \xrightarrow{E:\{\mathcal{D}^{\pi_3}\}} \cdots \xrightarrow{I} \pi_L,$$

where $E : \{\mathcal{D}\}$ indicates policy evaluation using the data sample $\mathcal{D}$. It would be more cost-efficient if we could reuse all previously-collected data samples to perform policy evaluation with a growing dataset as:

$$\pi_1 \xrightarrow{E:\{\mathcal{D}^{\pi_1}\}} \widehat{Q}^{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E:\{\mathcal{D}^{\pi_1},\mathcal{D}^{\pi_2}\}} \widehat{Q}^{\pi_2} \xrightarrow{I} \pi_3 \xrightarrow{E:\{\mathcal{D}^{\pi_1},\mathcal{D}^{\pi_2},\mathcal{D}^{\pi_3}\}} \cdots \xrightarrow{I} \pi_L.$$

Reusing previously collected data samples turns this into an *off-policy* scenario as the previous policies and the current policy are different unless the current policy has converged to the optimal one. Here, we propose using APDIW+IWCV in policy iteration. For this purpose, we extend the definition of $\widehat{G}_{\mathrm{APDIW}}$

```
Algorithm 1: SampleReusePolicyIteration($\boldsymbol{\phi}, \pi_1$)

  //$\boldsymbol{\phi}$   Basis functions, $\boldsymbol{\phi}(s,a) = (\phi_1(s,a), \phi_2(s,a), \ldots, \phi_B(s,a))^\top$
  //$\pi_1$   Initial policy function, $\pi_1(a|s) \in [0,1]$

 $l \leftarrow 1$

 for $l \leftarrow 1, 2, \ldots, L$
   do {
     // Collect data samples using current policy $\pi_l$
     $\mathcal{D}^{\pi_l} \leftarrow$ DataSampling($\pi_l$)

     // Choose flattening parameter $\widehat{\nu}$ that minimizes $\widehat{G}_{\mathrm{IWCV}}$
     for $\nu \leftarrow 0, 0.1, \ldots, 1$
       do { $\widehat{G}_{\mathrm{IWCV}}[\nu] \leftarrow$ ApproximationErrorEstimation($\{\mathcal{D}^{\pi_{l'}}\}_{l'=1}^{l}, \{\pi_{l'}\}_{l'=1}^{l}, \nu, \boldsymbol{\phi}$)
     $\widehat{\nu} \leftarrow \arg\min_{\nu} \widehat{G}_{\mathrm{IWCV}}[\nu]$

     // Learn parameter $\boldsymbol{\theta}$ for policy evaluation $\widehat{Q}^{\pi_l}$
     $\widehat{\boldsymbol{\theta}} \leftarrow$ PolicyEvaluation($\{\mathcal{D}^{\pi_{l'}}\}_{l'=1}^{l}, \{\pi_{l'}\}_{l'=1}^{l}, \widehat{\nu}, \boldsymbol{\phi}$)

     // Update policy function using $\widehat{Q}^{\pi_l}$
     $\pi_{l+1} \leftarrow$ PolicyImprovement($\widehat{\boldsymbol{\theta}}, \boldsymbol{\phi}$)
 return ($\pi_L$)
```

Figure 5: Pseudo code of SRPI. L is the maximum number of policy iteration. By the *DataSampling* function, data samples (M episodes and N steps) are collected following the current policy π_l. By the *PolicyImprovement* function, the current policy is updated based on the value function $\widehat{Q}^{\pi_l}$ with a policy-improvement method such as the Gibbs policy (2) and the ϵ-greedy policy (3). The pseudo codes of the *ApproximationErrorEstimation* and *PolicyEvaluation* functions are described in Figs.6 and 7, respectively.

so that multiple sampling policies $\{\pi_1, \pi_2, \ldots, \pi_l\}$ are taken into account:

$$\widehat{\boldsymbol{\theta}}^{l}_{\mathrm{APDIW}} \equiv \arg\min_{\boldsymbol{\theta}} \widehat{G}^{l}_{\mathrm{APDIW}},$$

$$\widehat{G}^{l}_{\mathrm{APDIW}} \equiv \frac{1}{lMN} \sum_{l'=1}^{l} \sum_{m=1}^{M} \sum_{n=1}^{N} \widehat{g}(s^{\pi_{l'}}_{m,n}, a^{\pi_{l'}}_{m,n}; \boldsymbol{\theta}, \{\mathcal{D}^{\pi_{l'}}\}_{l'=1}^{l}) \left(\frac{\prod_{n'=1}^{n} \pi_l(a^{\pi_{l'}}_{m,n'} | s^{\pi_{l'}}_{m,n'})}{\prod_{n'=1}^{n} \pi_{l'}(a^{\pi_{l'}}_{m,n'} | s^{\pi_{l'}}_{m,n'})} \right)^{\nu_l}, \tag{10}$$

where $\widehat{G}^{l}_{\mathrm{APDIW}}$ is the approximation error estimated at the l-th policy evaluation using APDIW. The flattening parameter ν_l is chosen based on IWCV before performing policy evaluation. We call this *sample-reuse policy iteration* (SRPI). The SRPI algorithm is given in pseudo code in Figs.5-7.

6.2 Numerical Examples

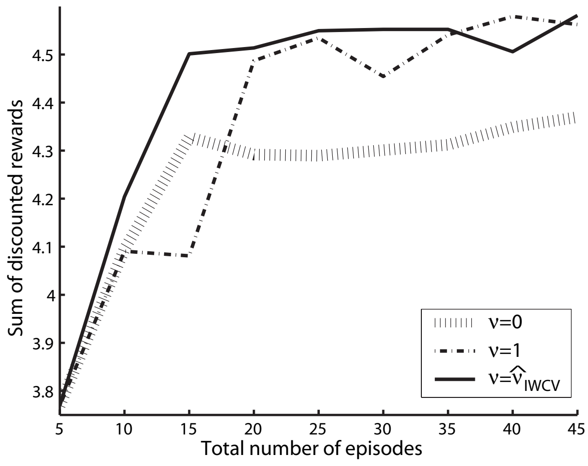
We will use the system from the numerical examples in Section 5.2 also in the policy iteration context in order to illustrate how SRPI works. We consider three scenarios: ν is fixed at 0, ν is fixed at 1, and SRPI where ν is chosen by IWCV. The agent collects samples $\mathcal{D}^{\pi_l}$ at every policy iteration following the current policy π_l, and computes $\widehat{\boldsymbol{\theta}}^{l}_{\mathrm{APDIW}}$ from all collected samples $\{\mathcal{D}^{\pi_1}, \mathcal{D}^{\pi_2}, \ldots, \mathcal{D}^{\pi_l}\}$ using Eq.(10).

```
Algorithm 2: ApproximationErrorEstimation($\{\mathcal{D}^{\pi_{l'}}\}_{l'=1}^{l}, \{\pi_{l'}\}_{l'=1}^{l}, \nu, \boldsymbol{\phi}$)

$\{\mathcal{D}_{k'}\}_{k'=1}^{K} \leftarrow \{\mathcal{D}^{\pi_{l'}}\}_{l'=1}^{l}$   // Divide $M$ episodic samples into $K$ groups
$M' \leftarrow \frac{M}{K}$   // Number of episodes in each group. Assume $M/K$ is integer.

for $k \leftarrow 1, 2, \ldots, K$
do
    // Learn parameter $\boldsymbol{\theta}$ from $K-1$ groups of data samples $\{\mathcal{D}_{k'}\}_{k' \neq k}$
    $\widehat{\boldsymbol{\theta}}_{\mathrm{APDIW}}^{k} \leftarrow$ PolicyEvaluation($\{\mathcal{D}_{k'}\}_{k' \neq k}, \{\pi_{l'}\}_{l'=1}^{l}, \nu, \boldsymbol{\phi}$)

    // Estimate approximation error of $\widehat{\boldsymbol{\theta}}_{\mathrm{APDIW}}^{k}$ from $k$-th group of data samples $\{\mathcal{D}_{k'}\}_{k'=k}$
    $\widehat{G}_{\mathrm{IWCV}}^{k} \leftarrow \frac{1}{lM'N} \sum_{l'=1}^{l} \sum_{m=1}^{M'} \sum_{n=1}^{N} \widehat{g}(s_{m,n}^{\pi_{l'}}, a_{m,n}^{\pi_{l'}}; \widehat{\boldsymbol{\theta}}_{\mathrm{APDIW}}^{k}, \mathcal{D}_k) \left( \frac{\prod_{n'=1}^{n} \pi_l(a_{m,n'}^{\pi_{l'}} | s_{m,n'}^{\pi_{l'}})}{\prod_{n'=1}^{n} \pi_{l'}(a_{m,n'}^{\pi_{l'}} | s_{m,n'}^{\pi_{l'}})} \right)$
// Compute the mean of approximation errors
$\widehat{G}_{\mathrm{IWCV}} \leftarrow \frac{1}{K} \sum_{k=1}^{K} \widehat{G}_{\mathrm{IWCV}}^{k}$
return ($\widehat{G}_{\mathrm{IWCV}}$)
```

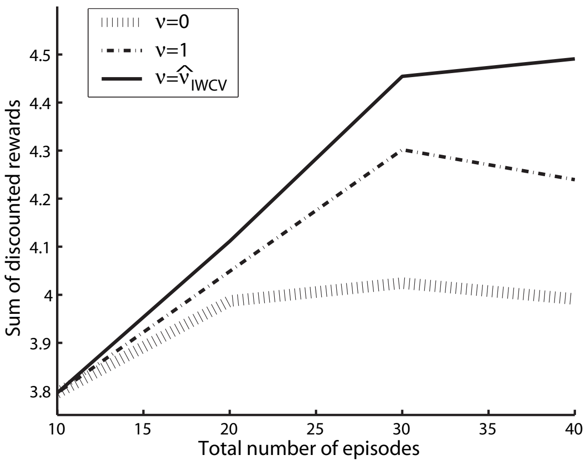
Figure 6: Pseudo code of ApproximationErrorEstimation. ν is the flattening parameter, $\boldsymbol{\phi}$ is a vector of basis functions, K is the number of folds, M is the number of episodes, and N is the number of steps.

```
Algorithm 3: PolicyEvaluation($\{\mathcal{D}^{\pi_{l'}}\}_{l'=1}^{l}, \{\pi_{l'}\}_{l'=1}^{l}, \nu, \boldsymbol{\phi}$)

$\widehat{\boldsymbol{A}} \leftarrow 0$   // $(B \times B)$ matrix, corresponding to the left term of $Eq.(8)$
$\widehat{\boldsymbol{b}} \leftarrow 0$   //$B$-dimensional vector, corresponding to the right term of $Eq.(8)$

for $l' \leftarrow 1, 2, \ldots, l$
do
    $w \leftarrow 1$  //Importance weight
    for each $(s, a) \in \mathcal{D}^{\pi_{l'}}$
    do
        // Estimate the expectation of $\boldsymbol{\phi}(s', a')$ under $\pi_l$ and $P_{\mathrm{T}}$
        $\widehat{\boldsymbol{\psi}}(s, a) \leftarrow \boldsymbol{\phi}(s, a) - \frac{\gamma}{|\mathcal{D}_{(s,a)}|} \sum_{s' \in \mathcal{D}_{(s,a)}} \mathop{\mathbb{E}}_{\pi_l(a'|s')} [\boldsymbol{\phi}(s', a')]$

        // Compute the expected reward
        $\widehat{R}(s, a) \leftarrow \frac{1}{|\mathcal{D}_{(s,a)}|} \sum_{r \in \mathcal{D}_{(s,a)}} r$

        // Update importance weight
        $w \leftarrow w \cdot \frac{\pi_l(a|s)}{\pi_{l'}(a|s)}$

        // Increment $\boldsymbol{A}$ and $\boldsymbol{b}$
        $\widehat{\boldsymbol{A}} \leftarrow \widehat{\boldsymbol{A}} + \widehat{\boldsymbol{\psi}}(s, a) \widehat{\boldsymbol{\psi}}(s, a)^{\top} w^{\nu}$
        $\widehat{\boldsymbol{b}} \leftarrow \widehat{\boldsymbol{b}} + \widehat{R}(s, a) \widehat{\boldsymbol{\psi}}(s, a)^{\top} w^{\nu}$
// Compute $\widehat{\boldsymbol{\theta}}_{\mathrm{APDIW}}$
$\widehat{\boldsymbol{\theta}}_{\mathrm{APDIW}} \leftarrow \widehat{\boldsymbol{A}}^{-1} \widehat{\boldsymbol{b}}$
return ($\widehat{\boldsymbol{\theta}}_{\mathrm{APDIW}}$)
```

Figure 7: Pseudo code of PolicyEvaluation. ν is the flattening parameter and $\boldsymbol{\phi}$ is a vector of basis functions.

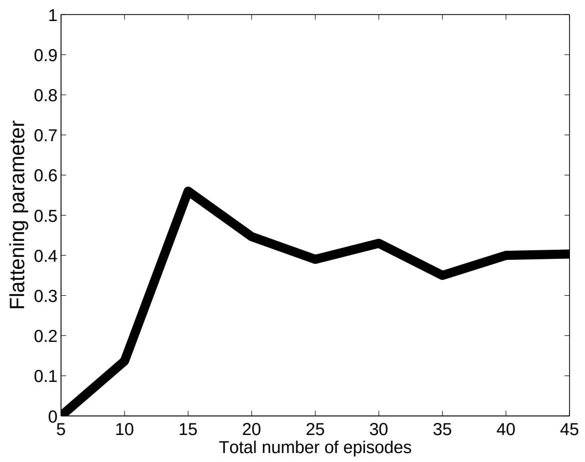


(a) $M = 5$

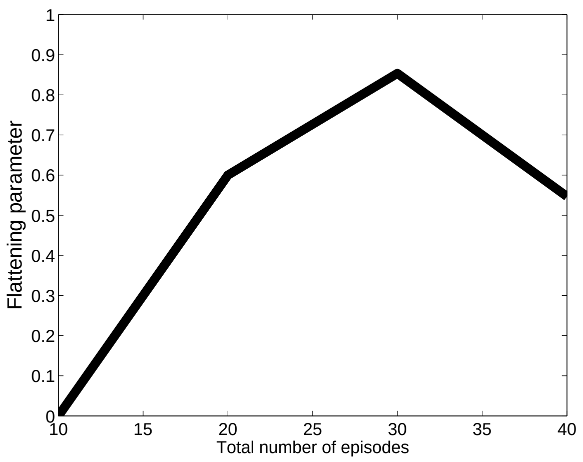


(b) $M = 10$

Figure 8: The performance of policies learned in three scenarios: $\nu = 0$, $\nu = 1$, and SRPI (ν is chosen by IWCV) in the 10-state chain-walk problem. The performance is measured by the average sum of discounted rewards computed from test samples over 30 trials. The agent collects training sample $\mathcal{D}^{\pi_l}$ ($M = 5$ or 10 with $N = 10$) at every iteration and performs policy evaluation using all collected samples $\{\mathcal{D}^{\pi_1}, \mathcal{D}^{\pi_2}, \ldots, \mathcal{D}^{\pi_l}\}$. The total number of episodes means the number of training episodes ($M \times l$) collected by the agent in policy iteration.



(a) $M = 5$



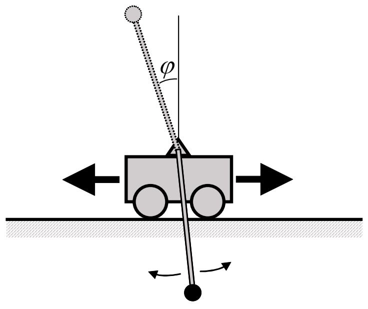
(b) $M = 10$

Figure 9: Average flattening parameter used by SRPI over 30 trials as a function of total number of episodes in the 10-state chain-walk problem.

We use Gaussian kernels defined by Eq.(9) except that kernel centers $\{c_1, c_2, c_3\}$ are randomly selected from the state space $\mathcal{S}$ every trial. The initial policy π_1 is chosen randomly and policy improvement is carried out by the Gibbs policy (2) with $\tau = 2l$.

Fig.8 depicts the average sum of discounted rewards over 30 trials when $M = 5, 10$ with a fixed number of steps ($N = 10$). The graphs show that SRPI provides stable and fast learning of policies while the performance improvement of policies learned with $\nu = 0$ saturates in early iterations. The method with $\nu = 1$ can improve policies well but its progress tends to be behind SRPI.

Fig.9 depicts the average value of the flattening parameter used in SRPI as a function of the total number of episodes. The graphs show that the value of the flattening parameter chosen by IWCV tends to rise in the beginning and go down later. At first sight, this does not agree with the general trend

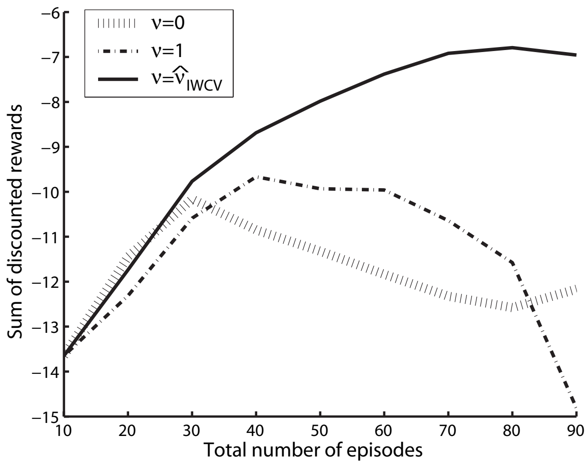


(a) Ilustration

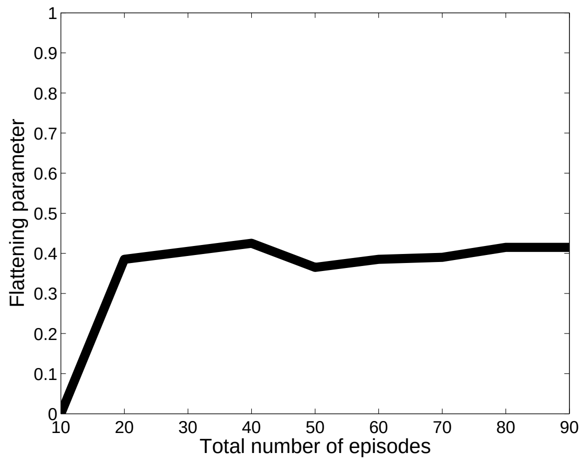
Parameter	Value
Mass of the cart, W	$8[kg]$
Mass of the rod, w	$2[kg]$
Length of the rod, d	$0.5[m]$
Simulation time step, Δt	$0.1[s]$

(b) Parameter setting

Figure 10: Illustration of the inverted pendulum task and parameters used in the simulation.



(a) Performance of policy



(b) Average flattening parameter

Figure 11: The results of sample-reuse policy iteration in the task of swing-up inverted-pendulum. The agent collects training sample $\mathcal{D}^{\pi_l}$ ($M = 10$ and $N = 100$) at every iteration and policy evaluation is performed using all collected samples $\{\mathcal{D}^{\pi_1}, \mathcal{D}^{\pi_2}, \ldots, \mathcal{D}^{\pi_l}\}$. (a) The performance of policies learned with $\nu = 0$, $\nu = 1$, and SRPI. The performance is measured by the average sum of discounted rewards computed from test samples over 20 trials. The total number of episodes means the number of training episodes ($M \times l$) collected by the agent in policy iteration. (b) Average flattening parameter used by SRPI over 20 trials.

of preferring a low-variance estimator in early stages and preferring a low-bias estimator later—but his result is still consistent with the general trend. When the sum of discounted rewards increase rapidly (the total numbers of episodes is up to 15 when $M = 5$ and 30 when $M = 10$ in Fig.9), the value of the flattening parameter increases (see Fig.8). After that, the sum of discounted rewards does not increase anymore (see Fig.8) since the policy iteration has already been converged. Then it is natural to prefer a small flattening parameter (Fig.9) since the sample selection bias becomes mild after convergence.

These results show that SRPI can effectively reuse previously-collected samples by appropriately tuning the flattening parameter according to the condition of data samples, policies etc.

7 Experiments

In this section, we evaluate the performance of our proposed method SRPI in more complex tasks, i.e., swing-up of an inverted pendulum and a mountain-car problem.

7.1 Swing-up Inverted Pendulum

We consider the task of *swing-up inverted pendulum* (?) illustrated in Fig.10(a), consisting of a rod hinged at the top of a cart. The goal of the task is to swing the rod up by moving the cart. We have three actions: applying positive force $+50$ $[kg \cdot m/s^2]$ to the cart to move right, negative force -50 to move left, and zero force to just coast. That is, the action space $\mathcal{A}$ is discrete and described by

$$\mathcal{A} = \{50, -50, 0\} \ \ [kg \cdot m/s^2].$$

We note that the force itself is not strong enough to swing the rod up; so the cart needs to be moved back and forth several times to swing the rod up. The state space $\mathcal{S}$ is continuous and consists of the angle φ $[rad]$ ($\in [0, 2\pi]$) and the angular velocity $\dot{\varphi}$ $[rad/s]$ ($\in [-\pi, \pi]$)—thus, a state s is described by two-dimensional vector $s = (\varphi, \dot{\varphi})^\top$. Fig.10(b) shows the parameter setting used in the simulation. The angle φ and angular velocity $\dot{\varphi}$ are updated (?) as follows:

$$\begin{aligned}\varphi_{t+1} =& \varphi_t + \dot{\varphi}_{t+1}\Delta t,\\ \dot{\varphi}_{t+1} =& \dot{\varphi}_t + \frac{9.8\sin(\varphi_t) - \alpha w d(\dot{\varphi}_t)^2 \sin(2\varphi_t)/2 + \alpha\cos(\varphi_t)a_t}{4l/3 - \alpha w d\cos^2(\varphi_t)}\Delta t,\end{aligned}$$

where $\alpha = 1/(W + w)$ and a_t is the action ($\in \mathcal{A}$) chosen at time t. We define the reward function $R(s, a, s')$ as

$$R(s, a, s') = \cos(\varphi_{s'}),$$

where $\varphi_{s'}$ denotes the angle φ of state s'.

We use 48 Gaussian kernels with standard deviation $\sigma = \pi$ as basis functions, and arrange kernel centers over the following grid points:

$$\{0, 2/3\pi, 4/3\pi, 2\pi\} \times \{-3\pi, -\pi, \pi, 3\pi\}.$$

That is, the basis functions $\boldsymbol{\phi}(s, a) = \{\phi_1(s, a), \ldots, \phi_{16}(s, a)\}$ are set as

$$\phi_{16(i-1)+j}(s, a) = I(a = a^{(i)})\exp\left(-\frac{\|s - c_j\|^2}{2\sigma^2}\right),$$

for $i = 1, 2, 3$ and $j = 1, 2, \ldots, 16$, where

$$c_1 = (0, -3\pi)^\top, c_2 = (0, -\pi)^\top, \ldots, c_{12} = (2\pi, 3\pi)^\top.$$

The initial policy $\pi_1(a|s)$ is chosen randomly, and the initial-state probability density $P_{\mathrm{I}}(s)$ is set to be uniform. The agent collects data samples $\mathcal{D}^{\pi_l}$ ($M = 10$ and $N = 100$) at each policy iteration following the current policy π_l. The discounted factor is set at $\gamma = 0.95$ and the policy is improved by the Gibbs policy (2) with $\tau = l$.

Fig.11(a) describes the performance of learned policies. The graph shows that SRPI nicely improves the performance throughout entire policy iteration. On the other hand, the performance when the flattening parameter is fixed at $\nu = 0$ or $\nu = 1$ is not properly improved after the middle of iterations. The average flattening parameter depicted in Fig.11(b) shows that the parameter tends to increase quickly in the beginning and then is kept at medium values. These results indicate that the flattening parameter is well-adjusted to reuse the previously-collected samples effectively for policy evaluation, and thus SRPI can outperform the other methods.

7.2 Mountain Car

We evaluate our proposed method in another task—the *mountain car task* (Sutton & Barto, 1998) illustrated in Fig.12. The task consists of a car and two hills whose landscape is described by $\sin(3x)$; the top of the right hill is the goal to which we want to guide the car. We have three actions

$$\{+0.2, -0.2, 0\},$$

which are the values of the force applied to the car. We note that the force of the car is not strong enough to climb up the slope to reach the goal. The state space $\mathcal{S}$ is described by the horizontal position x $[m]$ ($\in [-1.2, 0.5]$) and the velocity $\dot{x}$ $[m/s]$ ($\in [-1.5, 1.5]$):

$$s = (x, \dot{x})^{\top}.$$

Fig.12 shows the parameter setting used in the simulation. The position x and velocity $\dot{x}$ are updated by

$$\begin{aligned} x_{t+1} =& x_t + \dot{x}_{t+1}\Delta t, \\ \dot{x}_{t+1} =& \dot{x}_t + \Big(-9.8w\cos(3x_t) + \frac{a_t}{w} - k\dot{x}_t \Big)\Delta t, \end{aligned}$$

where a_t is the action ($\in \mathcal{A}$) chosen at the time t. We define the reward function $R(s, a, s')$ as

$$R(s,a,s') = \begin{cases} 1 & \text{if } x_{s'} \geq 0.5, \\ -0.01 & \text{otherwise,} \end{cases}$$

where $x_{s'}$ denotes the horizontal position x of state s'.

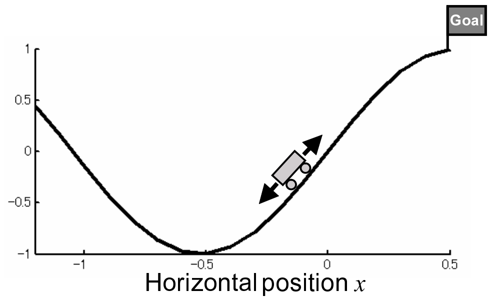
We use the same experimental setup as that of the swing-up inverted pendulum task except that the number of Gaussian kernels is 36, the standard deviation is $\sigma = 1$, and the kernel centers are allocated over the follwoing grid points:

$$\{-1.2, 0.35, 0.5\} \times \{-1.5, -0.5, 0.5, 1.5\}.$$

Fig.13(a) describes the performance of learned policies measured by the average sum of discounted rewards computed from the test samples. The graph shows similar tendencies to the swing-up inverted pendulum task for SRPI and $\nu = 1$, while the method with $\nu = 0$ performs relatively well this time. This would imply that the bias in previously-collected data samples does not affect so much on estimation of value functions—it may happen if the basis function represents the value function quite well (in other words, the model is almost correctly speficied) or the policy is not significantly changed over iterations. The average flattening parameter (cf. Fig.13(b)) shows that the parameter value decreases soon after the increase in the beginning, and then the smaller values tend to be chosen. This indicates that SRPI tends to use low-variance estimators in this task. These results show that SRPI can perform stable and faster learning by effectively reusing previously-collected data.

8 Conclusions and Outlook

Instability has been one of the critical limitations of importance-sampling techniques, which often makes off-policy methods impractical. To overcome this weakness, we introduced an *adaptive* importance-sampling technique for controlling the trade-off between consistency and stability in value function approximation. We further provided an automatic model selection method for actively choosing the trade-off parameter. We also proposed using the adaptive importance-sampling technique in policy iteration for efficiently reusing previously-collected data samples. The experimental results showed that the proposed method compares favorably with existing approaches.

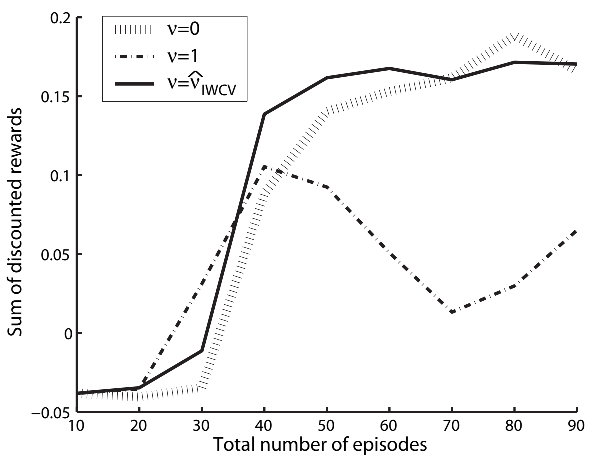


(a) Ilustration

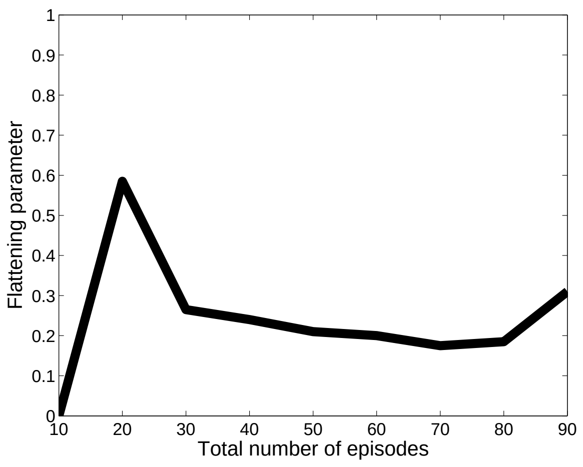
Parameter	Value
Mass of the car, w	$0.2[kg]$
Friction coefficient, k	0.3
Simulation time step, Δt	$0.1[s]$

(b) Parameter setting

Figure 12: Illustration of the mountain car task and parameters used in the simulation.



(a) Performance of policy



(b) Average flattening parameter

Figure 13: The results of sample-reuse policy iteration in the mountain-car task. The agent collects training sample $\mathcal{D}^{\pi_l}$ ($M = 10$ and $N = 100$) at every iteration and policy evaluation is performed using all collected samples $\{\mathcal{D}^{\pi_1}, \mathcal{D}^{\pi_2}, \ldots, \mathcal{D}^{\pi_l}\}$. (a) The performance of policies learned with $\nu = 0$, $\nu = 1$, and SRPI. The performance is measured by the average sum of discounted rewards computed from test samples over 20 trials. The total number of episodes means the number of training episodes ($M \times l$) collected by the agent in policy iteration. (b) Average flattening parameter used by SRPI over 20 trials.

We have assumed that the length of episodes is the same for all samples. However, in practice, the length may vary depending on trials; then the domain of the importance weight also varies. Thus it is important to extend the current approach so that such variable-length epsodic data could be handled.

The method presented in this paper may be extended to policy gradient methods where the need for the sample reuse is even more urgent as small gradient-descent steps result into an under utilization of the data. While importance-sampling has been applied in the settings of policy gradient methods (Shelton, 2001; Peshkin, 2002), policy gradient methods tend to be unstable when used with standard importance sampling methods (Kakade, 2002)—the proposed methods would offer an interesting alternative.

Appendix: Proof of Lemma 1

Taking the partial derivative of $\widehat{G}_{\mathrm{APDIW}}$ with respect to $\boldsymbol{\theta}$ and equating it to zero, we have

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\theta}} \widehat{G}_{\mathrm{APDIW}} &= \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} \frac{\partial}{\partial \boldsymbol{\theta}} \widehat{g}_{m,n} (\widehat{w}_{m,n})^{\nu} \\
&= \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} \frac{\partial}{\partial \boldsymbol{\theta}} \Bigg(\boldsymbol{\theta}^{\top} \boldsymbol{\phi}(s^{\widetilde{\pi}}_{m,n}, a^{\widetilde{\pi}}_{m,n}) - \frac{1}{|\mathcal{D}_{(s^{\widetilde{\pi}}_{m,n}, a^{\widetilde{\pi}}_{m,n})}|} \sum_{r \in \mathcal{D}_{(s^{\widetilde{\pi}}_{m,n}, a^{\widetilde{\pi}}_{m,n})}} r \\
&\qquad\qquad - \frac{\gamma}{|\mathcal{D}_{(s^{\widetilde{\pi}}_{m,n}, a^{\widetilde{\pi}}_{m,n})}|} \sum_{s' \in \mathcal{D}_{(s^{\widetilde{\pi}}_{m,n}, a^{\widetilde{\pi}}_{m,n})}} \mathrm{E}_{\widetilde{\pi}(a'|s')} \Big[\boldsymbol{\theta}^{\top} \boldsymbol{\phi}(s', a') \Big] \Bigg)^{2} (\widehat{w}_{m,n})^{\nu} \\
&= \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} \frac{\partial}{\partial \boldsymbol{\theta}} \Bigg(\boldsymbol{\theta}^{\top} \widehat{\boldsymbol{\psi}}(s^{\widetilde{\pi}}_{m,n}, a^{\widetilde{\pi}}_{m,n}; \mathcal{D}^{\pi}) - \frac{1}{|\mathcal{D}_{(s^{\widetilde{\pi}}_{m,n}, a^{\widetilde{\pi}}_{m,n})}|} \sum_{r \in \mathcal{D}_{(s^{\widetilde{\pi}}_{m,n}, a^{\widetilde{\pi}}_{m,n})}} r \Bigg)^{2} (\widehat{w}_{m,n})^{\nu} \\
&= \frac{2}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} \Bigg(\boldsymbol{\theta}^{\top} \widehat{\boldsymbol{\psi}}(s^{\widetilde{\pi}}_{m,n}, a^{\widetilde{\pi}}_{m,n}; \mathcal{D}^{\pi}) - \frac{1}{|\mathcal{D}_{(s^{\widetilde{\pi}}_{m,n}, a^{\widetilde{\pi}}_{m,n})}|} \sum_{r \in \mathcal{D}_{(s^{\widetilde{\pi}}_{m,n}, a^{\widetilde{\pi}}_{m,n})}} r \Bigg) \\
&\qquad\qquad \times \widehat{\boldsymbol{\psi}}(s^{\widetilde{\pi}}_{m,n}, a^{\widetilde{\pi}}_{m,n}; \mathcal{D}^{\pi})^{\top} (\widehat{w}_{m,n})^{\nu} \\
&= \mathbf{0}.
\end{aligned}$$

Then, the parameter $\widehat{\boldsymbol{\theta}}_{\mathrm{APDIW}}$ is obtained by

$$\begin{aligned}
\widehat{\boldsymbol{\theta}}_{\mathrm{APDIW}} &= \Bigg(\sum_{m=1}^{M} \sum_{n=1}^{N} \widehat{\boldsymbol{\psi}}(s^{\widetilde{\pi}}_{m,n}, a^{\widetilde{\pi}}_{m,n}; \mathcal{D}^{\widetilde{\pi}}) \widehat{\boldsymbol{\psi}}(s^{\widetilde{\pi}}_{m,n}, a^{\widetilde{\pi}}_{m,n}; \mathcal{D}^{\widetilde{\pi}})^{\top} (\widehat{w}_{m,n})^{\nu} \Bigg)^{-1} \\
&\qquad \times \Bigg(\sum_{m=1}^{M} \sum_{n=1}^{N} \Big(\frac{1}{|\mathcal{D}_{(s^{\widetilde{\pi}}_{m,n}, a^{\widetilde{\pi}}_{m,n})}|} \sum_{r \in \mathcal{D}_{(s^{\widetilde{\pi}}_{m,n}, a^{\widetilde{\pi}}_{m,n})}} r \Big) \widehat{\boldsymbol{\psi}}(s^{\widetilde{\pi}}_{m,n}, a^{\widetilde{\pi}}_{m,n}; \mathcal{D}^{\widetilde{\pi}})^{\top} (\widehat{w}_{m,n})^{\nu} \Bigg).
\end{aligned}$$

(Q.E.D.)

References

Bertsekas, P. D., & Tsitsiklis, J. (1996). *Neuro-dynamic programming.* NH, USA: Athena Scientific.

Bugeja, M. (2003). Non-linear swing-up and stabilizing control of an inverted pendulum system. *Proceedings of IEEE Region 8 EUROCON* (pp. 437–441).

Fishman, G. S. (1996). *Monte carlo: Concepts, algorithms, and applications.* Berlin: Springer-Verlag.

Kakade, S. (2002). A natural policy gradient. *Neural Information Processing Systems 14* (pp. 1531–1538).

Lagoudakis, M. G., & Parr, R. (2003). Least-squares policy iteration. *Journal of Machine Learning Research*, *4(Dec)*, 1107–1149.

Peshkin, L. Shelton, C. R. (2002). Learning from scarce experience. *Proceedings of International Conference on Machine Learning* (pp. 498–505).

Precup, D., Sutton, R. S., & Dasgupta, S. (2001). Off-policy temporal-difference learning with function approximation. *Proceedings of International Conference on Machine Learning* (pp. 417–424).

Precup, D., Sutton, R. S., & Singh, S. (2000). Eligibility traces for off-policy policy evaluation. *Proceedings of International Conference on Machine Learning* (pp. 759–766).

Rao, C. R. (1973). *Linear statistical inference and its applications.* New York: Wiley.

Schoknecht, R. (2003). Optimality of reinforcement learning algorithms with linear function approximation. *Neural Information Processing Systems 15* (pp. 1555–1562).

Shelton, C. R. (2001). Policy improvement for pomdps using normalized importance sampling. *Proceedings of Uncertainty in Artificial Intelligence* (pp. 496–503).

Shimodaira, H. (2000). Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, *90*, 227–244.

Sugiyama, M., Krauledat, M., & Müller, K.-R. (2007). Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, *8(May)*, 985–1005.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction.* MA, USA: The MIT Press.

Wang, H. O., Tanaka, K., & Griffin, M. F. (1996). An approach to fuzzy control of nonlinear systems: Stability and design issues. *IEEE Transactions on Fuzzy Systems*, 14–23.