Efficient Exploration through Active Learning for Value Function Approximation in Reinforcement Learning

Takayuki Akiyama (akiyama@sg.cs.titech.ac.jp)
Tokyo Institute of Technology

Hirotaka Hachiya (hachiya@sg.cs.titech.ac.jp)
Tokyo Institute of Technology

Masashi Sugiyama (sugi@cs.titech.ac.jp)
Tokyo Institute of Technology
and
Japan Science and Technology Agency

Abstract

Appropriately designing sampling policies is highly important for obtaining better control policies in reinforcement learning. In this paper, we first show that the *least-squares policy iteration* (LSPI) framework allows us to employ statistical active learning methods for linear regression. Then we propose a design method of good sampling policies for efficient exploration, which is particularly useful when the sampling cost of immediate rewards is high. The effectiveness of the proposed method, which we call *active policy iteration* (API), is demonstrated through simulations with a batting robot.

Keywords

reinforcement learning, Markov decision process, least-squares policy iteration, active learning, batting robot

1 Introduction

Reinforcement learning (RL) is the problem of letting an agent learn intelligent behavior through trial-and-error interaction with unknown environment (Sutton & Barto, 1998). More specifically, the agent learns its control policy so that the amount of rewards it will receive in the future is maximized. Due to its potential possibilities, RL has attracted a great deal of attention recently in the machine learning community.

In practical RL tasks, it is often expensive to obtain immediate reward samples while state-action trajectory samples are readily available. For example, let us consider a robot-arm control task of hitting a ball by a bat and driving the ball as far away as possible (see Figure 9). Let us adopt the carry of the ball as the immediate reward. In this setting, obtaining state-action trajectory samples of the robot arm is easy and relatively cheap since we just need to control the robot arm and record its state-action trajectories over time. On the other hand, explicitly computing the carry of the ball from the state-action samples is hard due to friction and elasticity of links, air resistance, unpredictable disturbances such a current of air, and so on. Thus, in practice, we may have to put the robot in open space, let the robot really hit the ball, and measure the carry of the ball manually. Thus gathering immediate reward samples is much more expensive than the state-action trajectory samples.

When the sampling cost of immediate rewards is high, it is important to design the sampling policy appropriately so that a good control policy can be obtained from a small number of samples. So far, the problem of designing good sampling policies has been addressed in terms of the trade-off between *exploration* and *exploitation* (Kaelbling, Littman, & Moore, 1996). That is, an RL agent is required to determine either to explore new states for learning more about unknown environment or to exploit previously acquired knowledge for obtaining more rewards.

A simple framework of controlling the exploration-exploitation trade-off is the *ϵ-greedy policy* (Sutton & Barto, 1998)—with (small) probability ϵ, the agent chooses to explore unknown environment randomly; otherwise it follows the current control policy for exploitation. The choice of the parameter ϵ is critical in the ϵ-greedy policy. A standard and natural idea would be to decrease the probability ϵ as the learning process progresses, i.e., the environment is actively explored in the beginning and then the agent tends to be in the exploitation mode later. However, theoretically and practically sound methods for determining the value of ϵ seem to be still open research topics. Also, when the agent decides to explore unknown environment, merely choosing the next action randomly would be far from the best possible option.

An alternative strategy called *Explicit Explore or Exploit* (E^3) was proposed in Kearns & Singh (1998) and Kearns & Singh (2002). The basic idea of E^3 is to control the balance between exploration and exploitation so that the accuracy of environment model estimation is optimally improved. More specifically, when the number of known states is small, the agent actively explores unvisited (or less visited) states; as the number of known states increases, exploitation tends to be prioritized. The E^3 strategy is efficiently realized by an algorithm called *R-max* (Brafman & Tennenholtz, 2002; Strehl, Diuk, & Littman, 2007). R-max assigns the maximum 'value' to unknown states so that the unknown states are visited with high probability. An advantage of E^3 and R-max is that the polynomial-time convergence (with respect to the number of states) to a near-optimal policy is theoretically guaranteed. However, since the algorithms explicitly count the number of visits at every state, it is not straightforward to extend the idea to *continuous* state spaces (Li, Littman, & Mansley, 2008). This is a critical limitation in robotics applications since state spaces are usually spanned by continuous variables such as joint

angles and angular velocities.

In this paper, we address the problem of designing sampling policies from a different point of view—*active learning* (AL) for value function approximation. We adopt the framework of *least-squares policy iteration* (LSPI) (Lagoudakis & Parr, 2003) and show that statistical AL methods for linear regression (Fedorov, 1972; Cohn, Ghahramani, & Jordan, 1996; Wiens, 2000; Kanamori & Shimodaira, 2003; Sugiyama, 2006; Sugiyama & Nakajima, 2009) can be naturally employed. In the LSPI framework, the state-action value function is approximated by fitting a linear model with least-squares estimation. A traditional AL scheme (Fedorov, 1972; Cohn et al., 1996) is designed to find the input distribution such that the variance of the least-squares estimator is minimized. For justifying the use of the traditional AL scheme, the bias should be guaranteed not to increase when the variance is reduced, since the expectation of the squared approximation error of the value function is expressed as the sum of the squared bias and variance. To this end, we need to assume a strong condition that the linear model used for value function approximation is *correctly specified*, i.e., if the parameters are learned optimally, the true value function can be perfectly approximated.

However, such a correct model assumption may not be fulfilled in practical RL tasks since the profile of value functions may be highly complicated. To cope with this problem, a two-stage AL scheme has been proposed in Kanamori & Shimodaira (2003). The use of the two-stage AL scheme can be theoretically justified even when the model is *misspecified*, i.e., the true function is not included in the model. The key idea of this two-stage AL scheme is to use dummy samples gathered in the first stage for estimating the approximation error of the value function; then additional samples are chosen based on AL in the second stage. This two-stage scheme works well when a large number of dummy samples are used for estimating the approximation error in the first stage. However, due to high sampling costs in practical RL problems, the practical performance of the two-stage AL method in the RL scenarios would be limited.

To overcome the weakness of the two-stage AL method, single-shot AL methods have been developed (Wiens, 2000; Sugiyama, 2006). The use of the single-shot AL methods can be theoretically justified when the model is *approximately correct*. Since dummy samples are not necessary in the single-shot AL methods, good performance is expected even when the number of samples to be collected is not large. Moreover, the algorithms of the single-shot methods are very simple and computationally efficient. For this reason, we adopt the single-shot AL method proposed in Sugiyama (2006), and develop a new exploration scheme for the LSPI-based RL algorithm. The usefulness of the proposed approach, which we call *active policy iteration* (API), is demonstrated through batting-robot simulations.

The rest of this paper is organized as follows. In Section 2, we formulate the RL problem using Markov decision processes and review the LSPI framework. Then in Section 3, we show how a statistical AL method could be employed for optimizing the sampling policy in the context of value function approximation. In Section 4, we apply our AL strategy to the LSPI framework and show the entire procedure of the proposed API algorithm. In Section 5, we demonstrate the effectiveness of API through ball-batting simulations.

Finally, in Section 6, we conclude by summarizing our contributions and describing future work.

2 Formulation of Reinforcement Learning Problem

In this section, we formulate the RL problem as a Markov decision problem (MDP) following Sutton & Barto (1998), and review how it can be solved using a method of policy iteration following Lagoudakis & Parr (2003).

2.1 Markov Decision Problem

Let us consider an MDP specified by

$$(\mathcal{S}, \mathcal{A}, P_{\mathrm{T}}, R, \gamma), \tag{1}$$

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 conditional probability density of the agent's transition 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,
- γ $(\in (0,1])$ is the discount factor for future rewards.

Let $\pi(a|s)$ $(\in [0,1])$ be a stochastic policy which is a 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 denotes the expectation of the discounted sum of rewards the agent will receive when taking action a in state s and following policy π thereafter, i.e.,

$$Q^{\pi}(s,a) \equiv \mathop{\mathbb{E}}_{\{s_n,a_n\}_{n=2}^{\infty}} \left[\sum_{n=1}^{\infty} \gamma^{n-1} R(s_n, a_n, s_{n+1}) \,\middle|\, s_1 = s, a_1 = a \right], \tag{2}$$

where $\mathbb{E}_{\{s_n,a_n\}_{n=2}^{\infty}}$ denotes the expectation over trajectory $\{s_n,a_n\}_{n=2}^{\infty}$ following $P_{\mathrm{T}}(s_{n+1}|s_n,a_n)$ and $\pi(a_n|s_n)$.

The goal of RL is to obtain the policy such that the expectation of the discounted sum of future rewards is maximized. The optimal policy can be expressed as

$$\pi^*(a|s) \equiv \delta(a - \mathop{\mathrm{argmax}}_{a'} Q^*(s,a')), \tag{3}$$

where $\delta(\cdot)$ is Dirac's delta function and

$$Q^*(s,a) \equiv \max_{\pi} Q^{\pi}(s,a) \tag{4}$$

is the *optimal* state-action value function.

$Q^\pi(s,a)$ can be expressed by the following recurrent form called the *Bellman equation*:

$$Q^\pi(s,a) = R(s,a) + \gamma \mathop{\mathbb{E}}_{P_{\mathrm{T}}(s'|s,a)} \mathop{\mathbb{E}}_{\pi(a'|s')} \left[Q^\pi(s',a')\right], \quad \forall s \in \mathcal{S},\ \forall a \in \mathcal{A}, \tag{5}$$

where

$$R(s,a) \equiv \mathop{\mathbb{E}}_{P_{\mathrm{T}}(s'|s,a)} \left[R(s,a,s')\right] \tag{6}$$

is the expected reward when the agent takes action a in state 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, and $\mathbb{E}_{\pi(a'|s')}$ 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 may find a better policy $\pi'(a|s)$ by 'softmax' update:

$$\pi'(a|s) \propto \exp(Q^\pi(s,a)/\beta), \tag{7}$$

where β (>0) determines the randomness of the new policy π'; or by ϵ-greedy update:

$$\pi'(a|s) = \epsilon p_{\mathrm{u}}(a) + (1-\epsilon)\delta(a - \mathop{\mathrm{argmax}}_{a'} Q^\pi(s,a')), \tag{8}$$

where $p_{\mathrm{u}}(a)$ denotes the uniform probability density over actions and ϵ $(\in (0,1])$ determines how stochastic the new policy π' is. Updating π based on $Q^\pi(s,a)$ is called *policy improvement*. Repeating policy evaluation and policy improvement, we may find the optimal policy $\pi^*(a|s)$. This entire process is called *policy iteration* (Sutton & Barto, 1998).

2.3 Least-squares Framework for Value Function Approximation

Although policy iteration is a useful framework for solving an MDP problem, it is computationally expensive when the number of state-action pairs $|\mathcal{S}| \times |\mathcal{A}|$ is large. Furthermore, when the state space or action space is continuous, $|\mathcal{S}|$ or $|\mathcal{A}|$ becomes infinite and therefore it is no longer possible to directly implement policy iteration. To overcome this problem, we approximate the state-action value function $Q^\pi(s,a)$ using the following linear model:

$$\widehat{Q}^\pi(s,a;\boldsymbol{\theta}) \equiv \sum_{b=1}^{B} \theta_b \phi_b(s,a) = \boldsymbol{\theta}^\top \boldsymbol{\phi}(s,a), \tag{9}$$

where

$$\boldsymbol{\phi}(s,a) = (\phi_1(s,a), \phi_2(s,a), \ldots, \phi_B(s,a))^\top \tag{10}$$

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

$$\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_B)^\top \tag{11}$$

are model parameters to be learned. Note that B is usually chosen to be much smaller than $|\mathcal{S}| \times |\mathcal{A}|$ for computational efficiency.

For N-step transitions, we ideally want to learn the parameters $\boldsymbol{\theta}$ so that the squared Bellman residual $G(\boldsymbol{\theta})$ is minimized (Lagoudakis & Parr, 2003):

$$\boldsymbol{\theta}^* \equiv \operatorname*{argmin}_{\boldsymbol{\theta}} G(\boldsymbol{\theta}), \tag{12}$$

$$G(\boldsymbol{\theta}) \equiv \mathop{\mathbb{E}}_{P_\pi}\left[\frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{\theta}^\top \boldsymbol{\psi}(s_n, a_n) - R(s_n, a_n))^2\right], \tag{13}$$

$$\boldsymbol{\psi}(s, a) \equiv \boldsymbol{\phi}(s, a) - \gamma \mathop{\mathbb{E}}_{P_{\mathrm{T}}(s'|s,a)} \mathop{\mathbb{E}}_{\pi(a'|s')} [\boldsymbol{\phi}(s', a')]. \tag{14}$$

$\mathbb{E}_{P_\pi}$ denotes the expectation over the joint probability density function of an entire trajectory:

$$P_\pi(s_1, a_1, s_2, a_2, \ldots, s_N, a_N, s_{N+1}) \equiv P_{\mathrm{I}}(s_1)\prod_{n=1}^{N} P_{\mathrm{T}}(s_{n+1}|s_n, a_n)\pi(a_n|s_n), \tag{15}$$

where $P_{\mathrm{I}}(s)$ denotes the initial-state probability density function.

2.4 Value Function Approximation from Samples

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

$$\mathcal{D}^{\widetilde{\pi}} \equiv \{d_m^{\widetilde{\pi}}\}_{m=1}^{M}, \tag{16}$$

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

$$d_m^{\widetilde{\pi}} \equiv \{(s_{m,n}^{\widetilde{\pi}}, a_{m,n}^{\widetilde{\pi}}, r_{m,n}^{\widetilde{\pi}}, s_{m,n+1}^{\widetilde{\pi}})\}_{n=1}^{N}. \tag{17}$$

We use two types of policies for different purposes: the *sampling policy* $\widetilde{\pi}(a|s)$ for collecting data samples and the *evaluation policy* $\pi(a|s)$ for computing the value function $\widehat{Q}^\pi$. Minimizing the *importance-weighted* empirical generalization error $\widehat{G}(\boldsymbol{\theta})$, we can

obtain a *consistent* estimator of $\boldsymbol{\theta}^*$ as follows:

$$\widehat{\boldsymbol{\theta}} \equiv \operatorname*{argmin}_{\boldsymbol{\theta}} \widehat{G}(\boldsymbol{\theta}), \tag{18}$$

$$\widehat{G}(\boldsymbol{\theta}) \equiv \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} (\boldsymbol{\theta}^\top \widehat{\boldsymbol{\psi}}(s_{m,n}^{\widetilde{\pi}}, a_{m,n}^{\widetilde{\pi}}; \mathcal{D}^{\widetilde{\pi}}) - r_{m,n}^{\widetilde{\pi}})^2 w_{m,N}^{\widetilde{\pi}}, \tag{19}$$

$$\widehat{\boldsymbol{\psi}}(s, a; \mathcal{D}) \equiv \boldsymbol{\phi}(s, a) - \frac{\gamma}{|\mathcal{D}_{(s,a)}|} \sum_{s' \in \mathcal{D}_{(s,a)}} \mathop{\mathbb{E}}_{\pi(a'|s')} [\boldsymbol{\phi}(s', a')], \tag{20}$$

where $\mathcal{D}_{(s,a)}$ is a set of 4-tuple elements[1] containing state s and action a in the training data $\mathcal{D}$, $\sum_{s' \in \mathcal{D}_{(s,a)}}$ denotes the summation over s' in the set $\mathcal{D}_{(s,a)}$, and

$$w_{m,N}^{\widetilde{\pi}} \equiv \frac{\prod_{n'=1}^{N} \pi(a_{m,n'}^{\widetilde{\pi}} | s_{m,n'}^{\widetilde{\pi}})}{\prod_{n'=1}^{N} \widetilde{\pi}(a_{m,n'}^{\widetilde{\pi}} | s_{m,n'}^{\widetilde{\pi}})} \tag{21}$$

is called the *importance weight* (Sutton & Barto, 1998).

It is important to note that consistency of $\widehat{\boldsymbol{\theta}}$ can be maintained even if $w_{m,N}^{\widetilde{\pi}}$ is replaced by the *per-decision importance weight* $w_{m,n}^{\widetilde{\pi}}$ (Precup, Sutton, & Singh, 2000), which is computationally more efficient and stable. $\widehat{\boldsymbol{\theta}}$ can be analytically expressed with the matrices $\widehat{\boldsymbol{L}}$ ($\in \mathbb{R}^{B \times MN}$), $\widehat{\boldsymbol{X}}$ ($\in \mathbb{R}^{MN \times B}$), $\boldsymbol{W}$ ($\in \mathbb{R}^{MN \times MN}$), and the vector $\boldsymbol{r}^{\widetilde{\pi}}$ ($\in \mathbb{R}^{MN}$) as

$$\widehat{\boldsymbol{\theta}} = \widehat{\boldsymbol{L}} \boldsymbol{r}^{\widetilde{\pi}}, \tag{22}$$

$$\widehat{\boldsymbol{L}} \equiv (\widehat{\boldsymbol{X}}^\top \boldsymbol{W} \widehat{\boldsymbol{X}})^{-1} \widehat{\boldsymbol{X}}^\top \boldsymbol{W}, \tag{23}$$

$$\boldsymbol{r}_{N(m-1)+n}^{\widetilde{\pi}} \equiv r_{m,n}^{\widetilde{\pi}}, \tag{24}$$

$$\widehat{\boldsymbol{X}}_{N(m-1)+n,b} \equiv \widehat{\psi}_b(s_{m,n}^{\widetilde{\pi}}, a_{m,n}^{\widetilde{\pi}}; \mathcal{D}^{\widetilde{\pi}}), \tag{25}$$

$$\boldsymbol{W}_{N(m-1)+n,N(m'-1)+n'} \equiv w_{m,n}^{\widetilde{\pi}} I(m = m') I(n = n'), \tag{26}$$

where $I(c)$ denotes the indicator function:

$$I(c) = \begin{cases} 1 & \text{if the condition } c \text{ is true,} \\ 0 & \text{otherwise.} \end{cases} \tag{27}$$

When the matrix $\widehat{\boldsymbol{X}}^\top \boldsymbol{W} \widehat{\boldsymbol{X}}$ is ill-conditioned, it is hard to compute its inverse accurately. To cope with this problem, we may practically employ a *regularization* scheme (Tikhonov & Arsenin, 1977; Hoerl & Kennard, 1970; Poggio & Girosi, 1990):

$$(\widehat{\boldsymbol{X}}^\top \boldsymbol{W} \widehat{\boldsymbol{X}} + \lambda \boldsymbol{I})^{-1}, \tag{28}$$

where $\boldsymbol{I}$ ($\in \mathbb{R}^{B \times B}$) is the identity matrix and λ is a small positive scalar.

[1] When the state-action space is continuous, the set $\mathcal{D}_{(s_{m,n}^{\widetilde{\pi}}, a_{m,n}^{\widetilde{\pi}})}$ contains only a single sample $(s_{m,n}^{\widetilde{\pi}}, a_{m,n}^{\widetilde{\pi}}, r_{m,n}^{\widetilde{\pi}}, s_{m,n+1}^{\widetilde{\pi}})$ and then consistency of $\widehat{\boldsymbol{\theta}}$ may not be guaranteed. A possible measure for this would be to use several neighbor samples around $(s_{m,n}^{\widetilde{\pi}}, a_{m,n}^{\widetilde{\pi}})$. However, in our experiments, we decided to use the single-sample approximation since it performed reasonably well.

3 Efficient Exploration with Active Learning

The accuracy of the estimated value function depends on the training samples collected following sampling policy $\widetilde{\pi}(a|s)$. In this section, we propose a new method for designing a good sampling policy based on a statistical AL method proposed in Sugiyama (2006).

3.1 Preliminaries

Let us consider a situation where collecting state-action trajectory samples is easy and cheap, but gathering immediate reward samples is hard and expensive (for example, the batting robot explained in the introduction). In such a case, immediate reward samples are too expensive to be used for designing the sampling policy; only state-action trajectory samples may be used for sampling policy design.

The goal of AL in the current setup is to determine the sampling policy so that the expected generalization error is minimized. The generalization error is not accessible in practice since the expected reward function $R(s,a)$ and the transition probability $P_{\mathrm{T}}(s'|s,a)$ are unknown. Thus, for performing AL, the generalization error needs to be estimated from samples. A difficulty of estimating the generalization error in the context of AL is that its estimation needs to be carried out only from state-action trajectory samples *without* using immediate reward samples. This means that standard generalization error estimation techniques such as *cross-validation* (Hachiya, Akiyama, Sugiyama, & Peters, 2009) cannot be employed since they require both state-action and immediate reward samples. Below, we explain how the generalization error could be estimated under the AL setup (i.e., without the reward samples).

3.2 Decomposition of Generalization Error

The information we are allowed to use for estimating the generalization error is a set of roll-out samples *without* immediate rewards:

$$\overline{\mathcal{D}}^{\widetilde{\pi}} \equiv \{\overline{d}_m^{\widetilde{\pi}}\}_{m=1}^{M}, \tag{29}$$

$$\overline{d}_m^{\widetilde{\pi}} \equiv \{(s_{m,n}^{\widetilde{\pi}}, a_{m,n}^{\widetilde{\pi}}, s_{m,n+1}^{\widetilde{\pi}})\}_{n=1}^{N}. \tag{30}$$

Let us define the deviation of immediate rewards from the mean as

$$\epsilon_{m,n}^{\widetilde{\pi}} \equiv r_{m,n}^{\widetilde{\pi}} - R(s_{m,n}^{\widetilde{\pi}}, a_{m,n}^{\widetilde{\pi}}). \tag{31}$$

Note that $\epsilon_{m,n}^{\widetilde{\pi}}$ could be regarded as additive noise in the context of least-squares function fitting. By definition, $\epsilon_{m,n}^{\widetilde{\pi}}$ has mean zero and its variance generally depends on $s_{m,n}^{\widetilde{\pi}}$ and $a_{m,n}^{\widetilde{\pi}}$, i.e., *heteroscedastic* noise (Bishop, 2006). However, since estimating the variance of $\epsilon_{m,n}^{\widetilde{\pi}}$ without using reward samples is not generally possible, we ignore the dependence of the variance on $s_{m,n}^{\widetilde{\pi}}$ and $a_{m,n}^{\widetilde{\pi}}$. Let us denote the input-independent common variance by σ^2.

Now we would like to estimate the generalization error

$$\overline{G}(\widehat{\boldsymbol{\theta}}) \equiv \mathop{\mathbb{E}}_{P_\pi}\left[\frac{1}{N}\sum_{n=1}^{N}(\widehat{\boldsymbol{\theta}}^\top\widehat{\boldsymbol{\psi}}(s_n,a_n;\overline{\mathcal{D}}^{\widetilde{\pi}}) - R(s_n,a_n))^2\right] \tag{32}$$

from $\overline{\mathcal{D}}^{\widetilde{\pi}}$. Its expectation over 'noise' can be decomposed as follows (Sugiyama, 2006).

$$\mathop{\mathbb{E}}_{\boldsymbol{\epsilon}^{\widetilde{\pi}}}\left[\overline{G}(\widehat{\boldsymbol{\theta}})\right] = \text{Bias} + \text{Variance} + \text{ModelError}, \tag{33}$$

where $\mathbb{E}_{\boldsymbol{\epsilon}^{\widetilde{\pi}}}$ denotes the expectation over 'noise' $\{\epsilon^{\widetilde{\pi}}_{m,n}\}_{m=1,n=1}^{M,N}$. Bias, Variance, and ModelError are the *bias* term, the *variance* term, and the *model error* term defined by

$$\text{Bias} \equiv \mathop{\mathbb{E}}_{P_\pi}\left[\frac{1}{N}\sum_{n=1}^{N}\left\{(\mathop{\mathbb{E}}_{\boldsymbol{\epsilon}^{\widetilde{\pi}}}\left[\widehat{\boldsymbol{\theta}}\right] - \boldsymbol{\theta}^*)^\top\widehat{\boldsymbol{\psi}}(s_n,a_n;\overline{\mathcal{D}}^{\widetilde{\pi}})\right\}^2\right], \tag{34}$$

$$\text{Variance} \equiv \mathop{\mathbb{E}}_{P_\pi}\mathop{\mathbb{E}}_{\boldsymbol{\epsilon}^{\widetilde{\pi}}}\left[\frac{1}{N}\sum_{n=1}^{N}\left\{(\widehat{\boldsymbol{\theta}} - \mathop{\mathbb{E}}_{\boldsymbol{\epsilon}^{\widetilde{\pi}}}\left[\widehat{\boldsymbol{\theta}}\right])^\top\widehat{\boldsymbol{\psi}}(s_n,a_n;\overline{\mathcal{D}}^{\widetilde{\pi}})\right\}^2\right], \tag{35}$$

$$\text{ModelError} \equiv \mathop{\mathbb{E}}_{P_\pi}\left[\frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{\theta}^{*\top}\widehat{\boldsymbol{\psi}}(s_n,a_n;\overline{\mathcal{D}}^{\widetilde{\pi}}) - R(s_n,a_n))^2\right]. \tag{36}$$

$\boldsymbol{\theta}^*$ is the optimal parameter in the model, defined by Eq.(12). Note that the variance term can be expressed in a compact form as

$$\text{Variance} = \sigma^2\text{tr}(\boldsymbol{U}\widehat{\boldsymbol{L}}\widehat{\boldsymbol{L}}^\top), \tag{37}$$

where the matrix $\boldsymbol{U}$ $(\in \mathbb{R}^{B\times B})$ is defined as

$$\boldsymbol{U}_{b,b'} \equiv \mathop{\mathbb{E}}_{P_\pi}\left[\frac{1}{N}\sum_{n=1}^{N}\widehat{\psi}_b(s_n,a_n;\overline{\mathcal{D}}^{\widetilde{\pi}})\widehat{\psi}_{b'}(s_n,a_n;\overline{\mathcal{D}}^{\widetilde{\pi}})\right]. \tag{38}$$

3.3 Estimation of Generalization Error for AL

The model error is constant and thus can be safely ignored in generalization error estimation since we are interested in finding a minimizer of the generalization error with respect to $\widetilde{\pi}$. So we focus on the bias term and the variance term. However, the bias term includes the unknown optimal parameter $\boldsymbol{\theta}^*$, and thus it may not be possible to estimate the bias term without using reward samples; similarly, it may not be possible to estimate the 'noise' variance σ^2 included in the variance term without using reward samples.

It is known that the bias term is small enough to be neglected when the model is *approximately correct* (Sugiyama, 2006), i.e., $\boldsymbol{\theta}^{*\top}\widehat{\boldsymbol{\psi}}(s,a)$ approximately agrees with the true function $R(s,a)$. Then we have

$$\mathop{\mathbb{E}}_{\boldsymbol{\epsilon}^{\widetilde{\pi}}}\left[\overline{G}(\widehat{\boldsymbol{\theta}})\right] - \text{ModelError} - \text{Bias} \;\propto\; \text{tr}(\boldsymbol{U}\widehat{\boldsymbol{L}}\widehat{\boldsymbol{L}}^\top), \tag{39}$$

which does not require immediate reward samples for its computation. Since $\mathbb{E}_{P_\pi}$ included in $\boldsymbol{U}$ is not accessible (see Eq.(38)), we replace $\boldsymbol{U}$ by its consistent estimator $\widehat{\boldsymbol{U}}$:

$$\widehat{\boldsymbol{U}} \equiv \frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}\widehat{\boldsymbol{\psi}}(s^{\widetilde{\pi}}_{m,n}, a^{\widetilde{\pi}}_{m,n}; \overline{\mathcal{D}}^{\widetilde{\pi}})\widehat{\boldsymbol{\psi}}(s^{\widetilde{\pi}}_{m,n}, a^{\widetilde{\pi}}_{m,n}; \overline{\mathcal{D}}^{\widetilde{\pi}})^\top w^{\widetilde{\pi}}_{m,n}. \tag{40}$$

Consequently, we have the following generalization error estimator:

$$J \equiv \mathrm{tr}(\widehat{\boldsymbol{U}}\widehat{\boldsymbol{L}}\widehat{\boldsymbol{L}}^\top), \tag{41}$$

which can be computed only from $\overline{\mathcal{D}}^{\widetilde{\pi}}$ and thus can be employed in the AL scenarios. If it is possible to gather $\overline{\mathcal{D}}^{\widetilde{\pi}}$ multiple times, the above J may be computed multiple times and its average J' may be used as a generalization error estimator.

Note that the values of the generalization error estimator J and the true generalization error $\overline{G}$ are not directly comparable since irrelevant additive and multiplicative constants are ignored (see Eq.(39)). We expect that the estimator J has a similar *profile* to the true error $\overline{G}$ as a function of sampling policy $\widetilde{\pi}$ since the purpose of deriving a generalization error estimator in AL is not to approximate the true generalization error itself, but to approximate the *minimizer* of the true generalization error with respect to sampling policy $\widetilde{\pi}$. We will experimentally investigate this issue in Section 3.5.

3.4 Designing Sampling Policies

Based on the generalization error estimator derived above, we give an algorithm for designing a good sampling policy, which fully makes use of the roll-out samples without immediate rewards.

1. Prepare K candidates of sampling policy: $\{\widetilde{\pi}_k\}_{k=1}^K$.
2. Collect episodic samples without immediate rewards for each sampling-policy candidate: $\{\overline{\mathcal{D}}^{\widetilde{\pi}_k}\}_{k=1}^K$.
3. Estimate $\boldsymbol{U}$ using all samples $\{\overline{\mathcal{D}}^{\widetilde{\pi}_k}\}_{k=1}^K$:

$$\widetilde{\boldsymbol{U}} = \frac{1}{KMN}\sum_{k=1}^{K}\sum_{m=1}^{M}\sum_{n=1}^{N}\widehat{\boldsymbol{\psi}}(s^{\widetilde{\pi}_k}_{m,n}, a^{\widetilde{\pi}_k}_{m,n}; \{\overline{\mathcal{D}}^{\widetilde{\pi}_k}\}_{k=1}^K)\widehat{\boldsymbol{\psi}}(s^{\widetilde{\pi}_k}_{m,n}, a^{\widetilde{\pi}_k}_{m,n}; \{\overline{\mathcal{D}}^{\widetilde{\pi}_k}\}_{k=1}^K)^\top w^{\widetilde{\pi}_k}_{m,n}. \tag{42}$$

4. Estimate the generalization error for each k:

$$J_k \equiv \mathrm{tr}(\widetilde{\boldsymbol{U}}\widehat{\boldsymbol{L}}^{\widetilde{\pi}_k}\widehat{\boldsymbol{L}}^{\widetilde{\pi}_k\top}), \tag{43}$$

$$\widehat{\boldsymbol{L}}^{\widetilde{\pi}_k} \equiv (\widehat{\boldsymbol{X}}^{\widetilde{\pi}_k\top}\boldsymbol{W}^{\widetilde{\pi}_k}\widehat{\boldsymbol{X}}^{\widetilde{\pi}_k})^{-1}\widehat{\boldsymbol{X}}^{\widetilde{\pi}_k\top}\boldsymbol{W}^{\widetilde{\pi}_k}, \tag{44}$$

$$\widehat{\boldsymbol{X}}^{\widetilde{\pi}_k}_{N(m-1)+n,b} \equiv \widehat{\psi}_b(s^{\widetilde{\pi}_k}_{m,n}, a^{\widetilde{\pi}_k}_{m,n}; \{\overline{\mathcal{D}}^{\widetilde{\pi}_k}\}_{k=1}^K), \tag{45}$$

$$\boldsymbol{W}^{\widetilde{\pi}_k}_{N(m-1)+n,N(m'-1)+n'} \equiv w^{\widetilde{\pi}_k}_{m,n} I(m=m')I(n=n'). \tag{46}$$

5. (If possible) repeat 2. to 4. several times and calculate the average for each k: $\{J'_k\}_{k=1}^K$.
6. Determine the sampling policy: $\widetilde{\pi}_{\rm AL} \equiv {\rm argmin}_k \, J'_k$.
7. Collect training samples with immediate rewards following $\widetilde{\pi}_{\rm AL}$: $\mathcal{D}^{\widetilde{\pi}_{\rm AL}}$.
8. Learn the value function by LSPI using $\mathcal{D}^{\widetilde{\pi}_{\rm AL}}$.

3.5 Numerical Examples

Here we illustrate how the above AL method behaves in the 10-state chain-walk environment shown in Figure 1. The MDP consists of 10 states

$$\mathcal{S} = \{s^{(i)}\}_{i=1}^{10} = \{1, 2, \ldots, 10\} \tag{47}$$

and 2 actions

$$\mathcal{A} = \{a^{(i)}\}_{i=1}^{2} = \{\text{'L'}, \text{'R'}\}. \tag{48}$$

The immediate reward function is defined as

$$R(s, a, s') \equiv f(s'), \tag{49}$$

where the profile of the function $f(s')$ is illustrated in Figure 2.

The transition probability $P_{\rm T}(s'|s, a)$ is indicated by the numbers attached to the arrows in Figure 1; for example, $P_{\rm T}(s^{(2)}|s^{(1)}, \text{'R'}) = 0.8$ and $P_{\rm T}(s^{(1)}|s^{(1)}, \text{'R'}) = 0.2$. Thus the agent can successfully move to the intended direction with probability 0.8 (indicated by solid-filled arrows in the figure) and the action fails with probability 0.2 (indicated by dashed-filled arrows in the figure). The discount factor γ is set to 0.9. We use the 12 basis functions $\boldsymbol{\phi}(s, a)$ defined as

$$\phi_{2(i-1)+j}(s, a) = \begin{cases} I(a = a^{(j)}) \exp\left(-\dfrac{(s - c_i)^2}{2\tau^2}\right) & \text{for } i = 1, 2, \ldots, 5 \text{ and } j = 1, 2 \\ I(a = a^{(j)}) & \text{for } i = 6 \text{ and } j = 1, 2, \end{cases} \tag{50}$$

where $c_1 = 1$, $c_2 = 3$, $c_3 = 5$, $c_4 = 7$, $c_5 = 9$, and $\tau = 1.5$.

For illustration purposes, we evaluate the selection of sampling policies only in one-step policy evaluation; evaluation through iterations will be addressed in the next section. Sampling policies and evaluation policies are constructed as follows. First, we prepare a deterministic 'base' policy $\overline{\pi}$, e.g., 'LLLLLRRRRR', where the i-th letter denotes the action taken at $s^{(i)}$. Let $\overline{\pi}^\epsilon$ be the 'ϵ-greedy' version of the base policy $\overline{\pi}$, i.e., the intended action can be successfully chosen with probability $1 - \epsilon/2$ and the other action is chosen with probability $\epsilon/2$. We perform experiments for three different evaluation policies:

$$\overline{\pi}_1 : \text{'RRRRRRRRRR'}, \tag{51}$$

$$\overline{\pi}_2 : \text{'RRLLLLLRRR'}, \tag{52}$$

$$\overline{\pi}_3 : \text{'LLLLLRRRRR'} \tag{53}$$

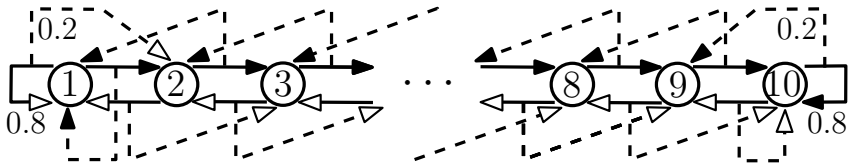


Figure 1: 10-state chain walk. Filled/unfilled arrows indicate the transitions when taking action 'R'/'L' and solid/dashed lines indicate the successful/failed transitions.

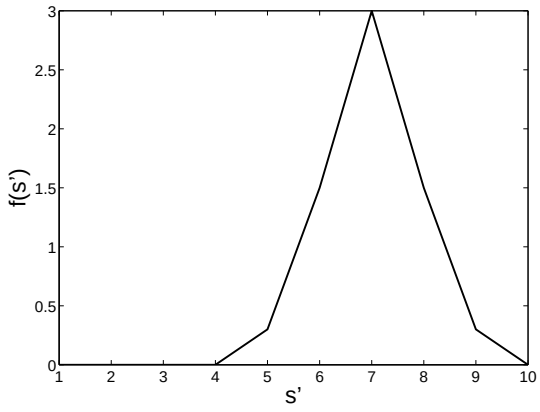


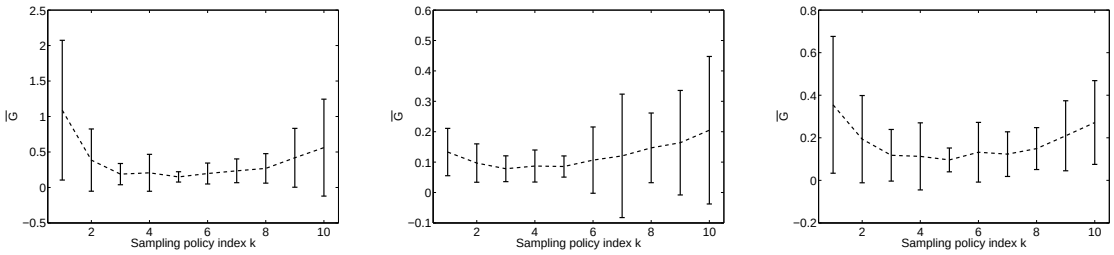
Figure 2: Profile of the function $f(s')$.

with $\epsilon = 0.1$. For each evaluation policy $\overline{\pi}_i^{0.1}$ $(i = 1, 2, 3)$, we prepare 10 candidates of the sampling policy $\{\widetilde{\pi}_i^{(k)}\}_{k=1}^{10}$, where the k-th sampling policy $\widetilde{\pi}_i^{(k)}$ is defined as $\overline{\pi}_i^{k/10}$. Note that $\widetilde{\pi}_i^{(1)}$ is equivalent to the evaluation policy $\overline{\pi}_i^{0.1}$.

For each sampling policy, we calculate the J-value 5 times and take the average. The numbers of episodes and steps are set to $M = 10$ and $N = 10$, respectively. The initial-state probability $P_{\mathrm{I}}(s)$ is set to be uniform. The regularization parameter is set to $\lambda = 10^{-3}$ for avoiding matrix singularity. This experiment is repeated 100 times with different random seeds and the mean and standard deviation of the true generalization error and its estimate are evaluated.

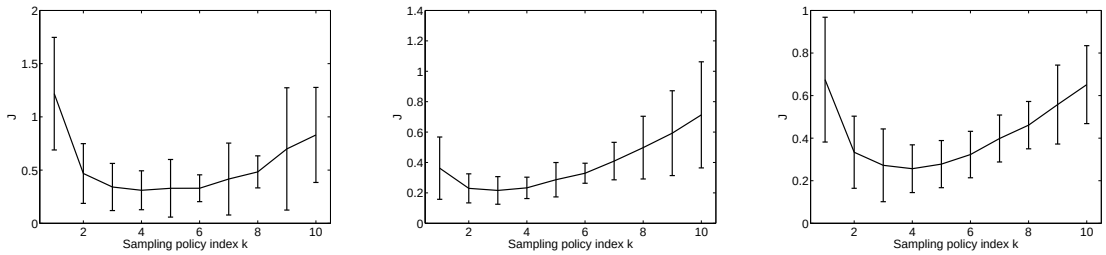
The results are depicted in Figure 3 (the true generalization error) and Figure 4 (its estimate) as functions of the index k of the sampling policies. Note that in these figures, we ignored irrelevant additive and multiplicative constants when deriving the generalization error estimator (see Eq.(39)). Thus, directly comparing the values of the true generalization error and its estimate is meaningless. The graphs show that the proposed generalization error estimator overall captures the trend of the true generalization error well for all three cases.

For active learning purposes, we are interested in choosing the value of k so that the true generalization error is minimized. Next, we investigate the values of the obtained generalization error $\overline{G}$ when k is chosen so that J is minimized (active learning; AL), the evaluation policy $(k = 1)$ is used for sampling (passive learning; PL), and k is chosen



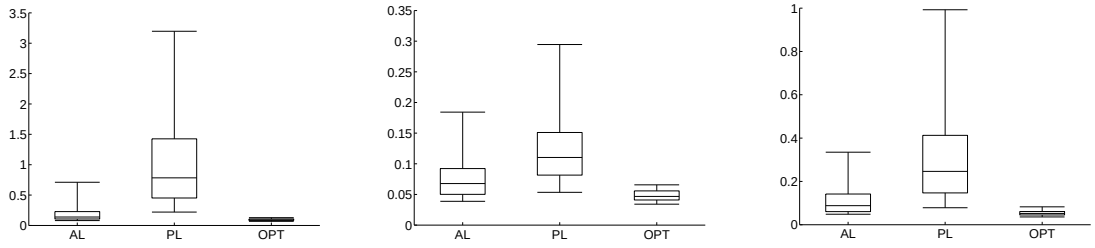
(a) $\overline{\pi}_1^{0.1}$ (b) $\overline{\pi}_2^{0.1}$ (c) $\overline{\pi}_3^{0.1}$

Figure 3: The mean and standard deviation of the true generalization error over 100 trials.



(a) $\overline{\pi}_1^{0.1}$ (b) $\overline{\pi}_2^{0.1}$ (c) $\overline{\pi}_3^{0.1}$

Figure 4: The mean and standard deviation of the estimated generalization error J over 100 trials.



(a) $\overline{\pi}_1^{0.1}$ (b) $\overline{\pi}_2^{0.1}$ (c) $\overline{\pi}_3^{0.1}$

Figure 5: The box-plots of the values of the obtained generalization error $\overline{G}$ over 100 trials when k is chosen so that J is minimized (active learning; AL), the evaluation policy ($k = 1$) is used for sampling (passive learning; PL), and k is chosen optimally so that the true generalization error is minimized (optimal; OPT). The box-plot notation indicates the 5%-quantile, 25%-quantile, 50%-quantile (i.e., median), 75%-quantile, and 95%-quantile from bottom to top.

optimally so that the true generalization error is minimized (optimal; OPT). Figure 5 depicts the box-plots of the generalization error values for AL, PL, and OPT over 100 trials, where the box-plot notation indicates the 5%-quantile, 25%-quantile, 50%-quantile (i.e., median), 75%-quantile, and 95%-quantile from bottom to top. The graphs show that the proposed AL method compares favorably with PL and performs well for reducing the generalization error.

We will continue the performance evaluation of the proposed AL method through iterations in Section 4.2.

4 Active Learning in Policy Iteration

In Section 3, we have shown that the unknown generalization error could be accurately estimated without using immediate reward samples in one-step policy evaluation. In this section, we extend the idea to the full policy-iteration setup.

4.1 Sample Reuse Policy Iteration with Active Learning

Sample reuse policy iteration (SRPI) (Hachiya et al., 2009) is a recently-proposed framework of *off-policy RL* (Sutton & Barto, 1998; Precup et al., 2000), which allows us to reuse previously-collected samples effectively. Let us denote the evaluation policy at the l-th iteration by π_l and the maximum number of iterations by L.

In the policy iteration framework, new data samples $\mathcal{D}^{\pi_l}$ are collected following the new policy π_l for the next policy evaluation step. In ordinary policy-iteration methods, only the new samples $\mathcal{D}^{\pi_l}$ are used for policy evaluation. Thus the previously-collected data samples $\{\mathcal{D}^{\pi_1}, \mathcal{D}^{\pi_2}, \ldots, \mathcal{D}^{\pi_{l-1}}\}$ are not utilized:

$$\pi_1 \overset{\mathrm{E}:\{\mathcal{D}^{\pi_1}\}}{\to} \widehat{Q}^{\pi_1} \overset{\mathrm{I}}{\to} \pi_2 \overset{\mathrm{E}:\{\mathcal{D}^{\pi_2}\}}{\to} \widehat{Q}^{\pi_2} \overset{\mathrm{I}}{\to} \pi_3 \overset{\mathrm{E}:\{\mathcal{D}^{\pi_3}\}}{\to} \cdots \overset{\mathrm{I}}{\to} \pi_{L+1}, \tag{54}$$

where 'E : $\{\mathcal{D}\}$' indicates policy evaluation using the data sample $\mathcal{D}$ and 'I' denotes policy improvement. On the other hand, in SRPI, all previously-collected data samples are reused for policy evaluation as

$$\pi_1 \overset{\mathrm{E}:\{\mathcal{D}^{\pi_1}\}}{\to} \widehat{Q}^{\pi_1} \overset{\mathrm{I}}{\to} \pi_2 \overset{\mathrm{E}:\{\mathcal{D}^{\pi_1},\mathcal{D}^{\pi_2}\}}{\to} \widehat{Q}^{\pi_2} \overset{\mathrm{I}}{\to} \pi_3 \overset{\mathrm{E}:\{\mathcal{D}^{\pi_1},\mathcal{D}^{\pi_2},\mathcal{D}^{\pi_3}\}}{\to} \cdots \overset{\mathrm{I}}{\to} \pi_{L+1}, \tag{55}$$

where appropriate importance weights are applied to each set of previously-collected samples in the policy evaluation step.

Here, we apply the AL technique proposed in the previous section to the SRPI framework. More specifically, we optimize the sampling policy at each iteration. Then the iteration process becomes

$$\pi_1 \overset{\mathrm{E}:\{\mathcal{D}^{\widetilde{\pi}_1}\}}{\to} \widehat{Q}^{\pi_1} \overset{\mathrm{I}}{\to} \pi_2 \overset{\mathrm{E}:\{\mathcal{D}^{\widetilde{\pi}_1},\mathcal{D}^{\widetilde{\pi}_2}\}}{\to} \widehat{Q}^{\pi_2} \overset{\mathrm{I}}{\to} \pi_3 \overset{\mathrm{E}:\{\mathcal{D}^{\widetilde{\pi}_1},\mathcal{D}^{\widetilde{\pi}_2},\mathcal{D}^{\widetilde{\pi}_3}\}}{\to} \cdots \overset{\mathrm{I}}{\to} \pi_{L+1}. \tag{56}$$

Thus, we do not gather samples following the current evaluation policy π_l, but following the sampling policy $\widetilde{\pi}_l$ optimized based on the AL method given in the previous section.

We call this framework *active policy iteration* (API). Figure 6 and Figure 7 show the pseudo code of the API algorithm. Note that the previously-collected samples are used not only for value function approximation, but also for sampling-policy selection. Thus API fully makes use of the samples.

4.2 Numerical Examples

Here we illustrate how the API method behaves using the same 10-state chain-walk problem as Section 3.5 (see Figure 1).

The initial evaluation policy π_1 is set as

$$\pi_1(a|s) \equiv 0.15p_{\mathrm{u}}(a) + 0.85I(a = \operatorname*{argmax}_{a'} \widehat{Q}_0(s,a')), \tag{57}$$

where

$$\widehat{Q}_0(s,a) \equiv \sum_{b=1}^{12} \phi_b(s,a). \tag{58}$$

Policies are updated in the l-th iteration using the ϵ-greedy rule with $\epsilon = 0.15/l$. In the sampling-policy selection step of the l-th iteration, we prepare the four sampling-policy candidates

$$\{\widetilde{\pi}_l^{(1)}, \widetilde{\pi}_l^{(2)}, \widetilde{\pi}_l^{(3)}, \widetilde{\pi}_l^{(4)}\} \equiv \{\overline{\pi}_l^{0.15/l}, \overline{\pi}_l^{0.15/l+0.15}, \overline{\pi}_l^{0.15/l+0.5}, \overline{\pi}_l^{0.15/l+0.85}\}, \tag{59}$$

where $\overline{\pi}_l$ denotes the policy obtained by greedy update using $\widehat{Q}^{\pi_{l-1}}$. The number of iterations to learn the policy is set to $L = 7$, the number of steps is set to $N = 10$, and the number M of episodes is different in each iteration and defined as

$$\{M_1, M_2, M_3, M_4, M_5, M_6, M_7\}, \tag{60}$$

where M_l $(l = 1, 2, \ldots, 7)$ denotes the number of episodes collected in the l-th iteration. In this experiment, we compare two types of scheduling: $\{5, 5, 3, 3, 3, 1, 1\}$ and $\{3, 3, 3, 3, 3, 3, 3\}$, which we refer to as the 'decreasing M' strategy and the 'fixed M' strategy, respectively. The J-value calculation is repeated 5 times for AL. In order to avoid matrix singularity, the regularization parameter is set to $\lambda = 10^{-3}$. The performance of learned policy π_{L+1} is measured by the discounted sum of immediate rewards for test samples $\{r_{m,n}^{\pi_{L+1}}\}_{m,n=1}^{50}$ (50 episodes with 50 steps collected following π_{L+1}):

$$\text{Performance} = \frac{1}{50} \sum_{m=1}^{50} \sum_{n=1}^{50} \gamma^{n-1} r_{m,n}^{\pi_{L+1}}, \tag{61}$$

where the discount factor γ is set to 0.9.

We compare the performance of passive learning (PL; the current policy is used as the sampling policy in each iteration) and the proposed AL method (the best sampling policy is chosen from the policy candidates prepared in each iteration). We repeat the same

Algorithm 1: *ActivePolicyIteration*$(\boldsymbol{\phi}, \pi_1, \lambda, Z)$

$//\boldsymbol{\phi}$ Basis functions, $\boldsymbol{\phi}(s,a) = (\phi_1(s,a), \phi_2(s,a), \ldots, \phi_B(s,a))^\top$
$//\pi_1$ Initial policy, $\pi_1(a|s) \in [0,1]$
$//\lambda$ Regularization parameter, $\lambda > 0$
$//Z$ The number of J-value calculations to take the average J', $Z \in \mathbb{N}$

$l \leftarrow 1$

for $l \leftarrow 1, 2, \ldots, L$

// Determine sampling policy $\widetilde{\pi}_l$ by the active learning method
$\widetilde{\pi}_l \leftarrow$ *SamplingPolicySelection*$(\{\mathcal{D}^{\widetilde{\pi}_{l'}}\}_{l'=1}^{l-1}, \boldsymbol{\phi}, \pi_l, \lambda, Z)$

// Collect episodic samples using policy $\widetilde{\pi}_l$
$\mathcal{D}^{\widetilde{\pi}_l} \leftarrow$ *DataSampling*$(\widetilde{\pi}_l)$

do // Learn the value function Q^{π_l} from the samples $\{\mathcal{D}^{\widetilde{\pi}_{l'}}\}_{l'=1}^{l}$

$$\boldsymbol{A} \leftarrow \frac{1}{lMN} \sum_{l'=1}^{l} \sum_{m=1}^{M} \sum_{n=1}^{N} \widehat{\boldsymbol{\psi}}(s_{m,n}^{\widetilde{\pi}_{l'}}, a_{m,n}^{\widetilde{\pi}_{l'}}; \{\mathcal{D}^{\widetilde{\pi}_{l'}}\}_{l'=1}^{l}) \widehat{\boldsymbol{\psi}}(s_{m,n}^{\widetilde{\pi}_{l'}}, a_{m,n}^{\widetilde{\pi}_{l'}}; \{\mathcal{D}^{\widetilde{\pi}_{l'}}\}_{l'=1}^{l})^\top w_{m,n}^{\widetilde{\pi}_{l'}}$$

$$\boldsymbol{B} \leftarrow \frac{1}{lMN} \sum_{l'=1}^{l} \sum_{m=1}^{M} \sum_{n=1}^{N} \widehat{\boldsymbol{\psi}}(s_{m,n}^{\widetilde{\pi}_{l'}}, a_{m,n}^{\widetilde{\pi}_{l'}}; \{\mathcal{D}^{\widetilde{\pi}_{l'}}\}_{l'=1}^{l}) r_{m,n}^{\widetilde{\pi}_{l'}}$$

$\widehat{\boldsymbol{\theta}}_l \leftarrow (\boldsymbol{A} + \lambda \boldsymbol{I})^{-1} \boldsymbol{B}$

// Update π_l using $\widehat{Q}^{\pi_l}$
$\pi_{l+1} \leftarrow$ *PolicyImprovement*$(\widehat{\boldsymbol{\theta}}_l, \boldsymbol{\phi})$

return (π_{L+1})

Figure 6: The pseudo code of *ActivePolicyIteration*. By the *DataSampling* function, episodic samples (M episodes and N steps) are collected using the input policy. By the *PolicyImprovement* function, the current policy is updated with policy improvement such as ϵ-greedy update or softmax update. The pseudo code of *SamplingPolicySelection* is shown in Algorithm 2 in Figure 7.

Algorithm 2: *SamplingPolicySelection*$(\{\mathcal{D}^{\widetilde{\pi}_{l'}}\}_{l'=1}^{l-1}, \boldsymbol{\phi}, \pi_l, \lambda, Z)$

$//\{\mathcal{D}^{\widetilde{\pi}_{l'}}\}_{l'=1}^{l-1}$ The previously-collected training samples up to $(l-1)$-th iteration
$//\boldsymbol{\phi}$ Basis functions, $\boldsymbol{\phi}(s,a) = (\phi_1(s,a), \phi_2(s,a), \ldots, \phi_B(s,a))^\top$
$//\pi_l$ The evaluation policy in the l-th iteration, $\pi_l(a|s)$ $(\in [0,1])$
$//\lambda$ Regularization parameter, λ (> 0)
$//Z$ The number of J-value calculations to compute the average J', Z $(\in \mathbb{N})$

for $z \leftarrow 1, 2, \ldots, Z$

do

- **for** $k \leftarrow 1, 2, \ldots, K$
  - **do**
    - //Generate sampling policy candidate $\widetilde{\pi}_k^{(l)}$ and collect episodic samples
    - //without immediate rewards using $\widetilde{\pi}_k^{(l)}$
    - $\overline{\mathcal{D}}^{\widetilde{\pi}_k^{(l)}} \leftarrow$ *RewardlessDataSampling*$(\widetilde{\pi}_k^{(l)})$
- //Estimate matrix $\widetilde{\boldsymbol{U}}$
- $//\overline{\mathcal{D}}_0 \equiv \{\overline{\mathcal{D}}^{\widetilde{\pi}_k^{(l)}}\}_{k=1}^K \cup \{\overline{\mathcal{D}}^{\widetilde{\pi}_{l'}}\}_{l'=1}^{l-1}$, $\Pi_0 \equiv \{\widetilde{\pi}_k^{(l)}\}_{k=1}^K \cup \{\widetilde{\pi}_{l'}\}_{l'=1}^{l-1}$
- $\widetilde{\boldsymbol{U}} \leftarrow \frac{1}{(K+l-1)MN} \sum_{\pi \in \Pi_0} \sum_{m=1}^{M} \sum_{n=1}^{N} \widehat{\boldsymbol{\psi}}(s_{m,n}^\pi, a_{m,n}^\pi; \overline{\mathcal{D}}_0) \widehat{\boldsymbol{\psi}}(s_{m,n}^\pi, a_{m,n}^\pi; \overline{\mathcal{D}}_0)^\top w_{m,n}^\pi$

do

- **for** $k \leftarrow 1, 2, \ldots, K$
  - **do**
    - //Calculate J_k^z
    - $//\Pi_k \equiv \{\widetilde{\pi}_k^{(l)}\} \cup \{\widetilde{\pi}_{l'}\}_{l'=1}^{l-1}$, $\boldsymbol{h}_{m,n}^\pi \equiv w_{m,n}^\pi \boldsymbol{e}^{(N(m-1)+n)}$ $(\in \mathbb{R}^{MN})$,
    - $//\boldsymbol{e}^{(i)}$ $(\in \mathbb{R}^{MN})$ is the standard basis vector: $\boldsymbol{e}_j^{(i)} \equiv I(i=j)$
    - $\boldsymbol{A} \leftarrow \frac{1}{lMN} \sum_{\pi \in \Pi_k} \sum_{m=1}^{M} \sum_{n=1}^{N} \widehat{\boldsymbol{\psi}}(s_{m,n}^\pi, a_{m,n}^\pi; \overline{\mathcal{D}}_0) \widehat{\boldsymbol{\psi}}(s_{m,n}^\pi, a_{m,n}^\pi; \overline{\mathcal{D}}_0)^\top w_{m,n}^\pi$
    - $\boldsymbol{B} \leftarrow \frac{1}{lMN} \sum_{\pi \in \Pi_k} \sum_{m=1}^{M} \sum_{n=1}^{N} \widehat{\boldsymbol{\psi}}(s_{m,n}^\pi, a_{m,n}^\pi; \overline{\mathcal{D}}_0) \boldsymbol{h}_{m,n}^{\pi\top}$
    - $\widehat{\boldsymbol{L}_k} \leftarrow (\boldsymbol{A} + \lambda \boldsymbol{I})^{-1} \boldsymbol{B}$
    - $J_k^z \leftarrow \mathrm{tr}(\widetilde{\boldsymbol{U}} \widehat{\boldsymbol{L}_k} \widehat{\boldsymbol{L}_k}^\top)$

//Choose the policy $\widetilde{\pi}_{\mathrm{AL}}$ which minimizes $J_k' = \frac{1}{Z} \sum_{z=1}^{Z} J_k^z$ $(k = 1, 2, \ldots, K)$
$\widetilde{\pi}_{\mathrm{AL}} \leftarrow \mathrm{argmin}_{\pi_k'} J_k'$
return $(\widetilde{\pi}_{\mathrm{AL}})$

Figure 7: The pseudo code of *SamplingPolicySelection*. In the function *RewardlessDataSampling*, episodic samples without immediate rewards (M episodes and N steps) are collected. Previously-collected training samples $\{\mathcal{D}^{\widetilde{\pi}_{l'}}\}_{l'=1}^{l}$ are used for the calculation of matrices $\widetilde{\boldsymbol{U}}$, $\boldsymbol{A}$, and $\boldsymbol{B}$ in J-value calculation.

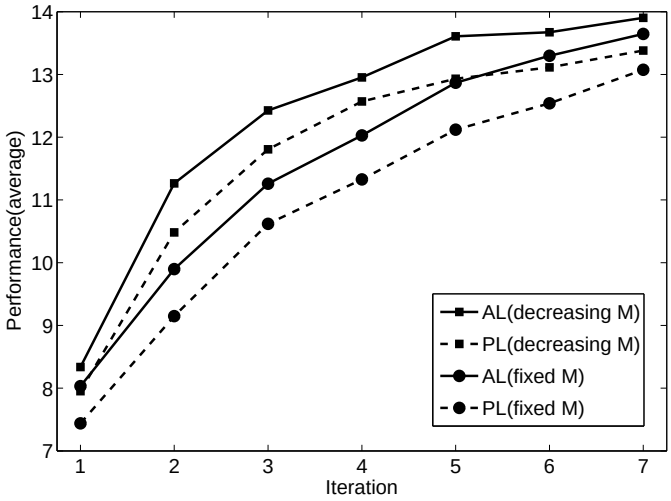


Figure 8: The mean performance over 1000 trials in the 10-state chain-walk experiment. The dotted lines denote the performance of passive learning (PL) and the solid lines denote the performance of the proposed active learning (AL) method. The error bars are omitted for clear visibility. For both the 'decreasing M' and 'fixed M' strategies, the performance of AL after the 7-th iteration is significantly better than that of PL according to the two-tailed paired Student t-test at the significance level 1% applied to the error values at the 7-th iteration.

experiment 1000 times with different random seeds and evaluate the average performance of each learning method. The results are depicted in Figure 8, showing that the proposed AL method works better than PL in both types of episode scheduling with statistical significance by the two-tailed paired *Student t-test* at the significance level 1% (Henkel, 1979) for the error values obtained at the 7-th iteration. Furthermore, the 'decreasing M' strategy outperforms the 'fixed M' strategy for both PL and AL, showing the usefulness of the 'decreasing M' strategy.

5 Experiments

Finally, we evaluate the performance of the proposed API method using a ball-batting robot illustrated in Figure 9, which consists of two links and two joints. The goal of the ball-batting task is to control the robot arm so that it drives the ball as far away as possible. The state space $\mathcal{S}$ is continuous and consists of angles φ_1[rad] ($\in [0, \pi/4]$) and φ_2[rad] ($\in [-\pi/4, \pi/4]$) and angular velocities $\dot{\varphi}_1$[rad/s] and $\dot{\varphi}_2$[rad/s]. Thus a state s ($\in \mathcal{S}$) is described by a four-dimensional vector:

$$s = (\varphi_1, \dot{\varphi}_1, \varphi_2, \dot{\varphi}_2)^\top. \tag{62}$$

The action space $\mathcal{A}$ is discrete and contains two elements:

$$\mathcal{A} = \{a^{(i)}\}_{i=1}^{2} = \{(50, -35)^\top, (-50, 10)^\top\}, \tag{63}$$

where the i-th element ($i = 1, 2$) of each vector corresponds to the torque [N $\cdot$ m] added to joint i.

We use the *Open Dynamics Engine* ('http://ode.org/') for physical calculations including the update of the angles and angular velocities, and collision detection between the robot arm, ball, and pin. The simulation time-step is set to 7.5 [ms] and the next state is observed after 10 time-steps. The action chosen in the current state is kept taken for 10 time-steps. To make the experiments realistic, we add noise to actions: if action $(f_1, f_2)^\top$ is taken, the actual torques applied to the joints are $f_1 + \varepsilon_1$ and $f_2 + \varepsilon_2$, where ε_1 and ε_2 are drawn independently from the Gaussian distribution with mean 0 and variance 3.

The immediate reward is defined as the carry of the ball. This reward is given only when the robot arm collides with the ball for the first time at state s' after taking action a at current state s. For value function approximation, we use the 110 basis functions defined as

$$\phi_{2(i-1)+j} = \begin{cases} I(a = a^{(j)}) \exp\left(-\dfrac{\| s - c_i \|^2}{2\tau^2}\right) & \text{for } i = 1, 2, \ldots, 54 \text{ and } j = 1, 2, \\ I(a = a^{(j)}) & \text{for } i = 55 \text{ and } j = 1, 2, \end{cases} \tag{64}$$

where τ is set to $3\pi/2$ and the Gaussian centers c_i ($i = 1, 2, \ldots, 54$) are located on the regular grid

$$\{0, \pi/4\} \times \{-\pi, 0, \pi\} \times \{-\pi/4, 0, \pi/4\} \times \{-\pi, 0, \pi\}. \tag{65}$$

We set $L = 7$ and $N = 10$. We again compare the 'decreasing M' strategy and the 'fixed M' strategy. The 'decreasing M' strategy is defined by $\{10, 10, 7, 7, 7, 4, 4\}$ and the 'fixed M' strategy is defined by $\{7, 7, 7, 7, 7, 7, 7\}$. The initial state is always set to $s = (\pi/4, 0, 0, 0)^\top$. The regularization parameter is set to $\lambda = 10^{-3}$ and the number of J-calculations in the AL method is set to 5. The initial evaluation policy π_1 is set to the ϵ-greedy policy defined as

$$\pi_1(a|s) \equiv 0.15 p_{\mathrm{u}}(a) + 0.85 I(a = \operatorname*{argmax}_{a'} \widehat{Q}_0(s, a')), \tag{66}$$

$$\widehat{Q}_0(s, a) \equiv \sum_{b=1}^{110} \phi_b(s, a). \tag{67}$$

Policies are updated in the l-th iteration using the ϵ-greedy rule with $\epsilon = 0.15/l$. The way we prepare sampling-policy candidates is the same as the chain-walk experiment in Section 4.2.

The discount factor γ is set to 1 and the performance of learned policy π_{L+1} is measured by the discounted sum of immediate rewards for test samples $\{r_{m,n}^{\pi_{L+1}}\}_{m=1,n=1}^{20,10}$ (20 episodes

with 10 steps collected following π_{L+1}):

$$\text{Performance} = \sum_{m=1}^{M} \sum_{n=1}^{N} r_{m,n}^{\pi_{L+1}}. \tag{68}$$

The experiment is repeated 500 times with different random seeds and the average performance of each learning method is evaluated. The results are depicted in Figure 10, showing that the proposed API method outperforms the PL strategy; for the 'decreasing M' strategy, the performance difference is statistically significant by the two-tailed paired Student t-test at the significance level 1% for the error values at the 7-th iteration.

Based on the experimental evaluation, we conclude that the proposed sampling-policy design method, API, is useful for improving the RL performance. Moreover, the 'decreasing M' strategy is shown to be a useful heuristic to further enhance the performance of API.

6 Conclusions and Future Work

When we cannot afford to collect many training samples due to high sampling costs, it is crucial to choose the most 'informative' samples for efficiently learning the value function. In this paper, we proposed a new data sampling strategy for reinforcement learning based on a statistical active learning method proposed by Sugiyama (2006). The proposed procedure called *active policy iteration* (API)—which effectively combines the framework of sample-reuse policy iteration (Hachiya et al., 2009) with active sampling-policy selection—was shown to perform well in simulations with chain-walk and batting robot control.

Our active learning strategy is a batch method and does not require previously collected reward samples. However, in the proposed API framework, reward samples are available from the previous iterations. A naive extension would be to include those previous samples in the generalization error estimator, for example, following the two-stage active learning scheme proposed by Kanamori & Shimodaira (2003), in which both the bias and variance terms are estimated using the labeled samples. However, such a bias-and-variance approach was shown to perform poorly compared with the variance-only approach (which we used in the current paper) (Sugiyama, 2006). Thus, developing an active learning strategy which can effectively make use of previously collected samples is an important future work.

For the case where the number of episodic samples to be gathered is fixed, we gathered many samples in earlier iterations, rather than gathering samples evenly in each iteration. Although this strategy was shown to perform well in the experiments, so far we do not have strong justification for this heuristic yet. Thus theoretical analysis would be necessary for understanding the mechanism of this approach and further improving the performance.

In the proposed method, the basis function $\boldsymbol{\psi}(s,a)$ defined by Eq.(14) was approximated by $\widehat{\boldsymbol{\psi}}(s,a,\mathcal{D})$ defined by Eq.(20) using samples. When the state-action space is continuous, this is theoretically problematic since only a single sample is available for

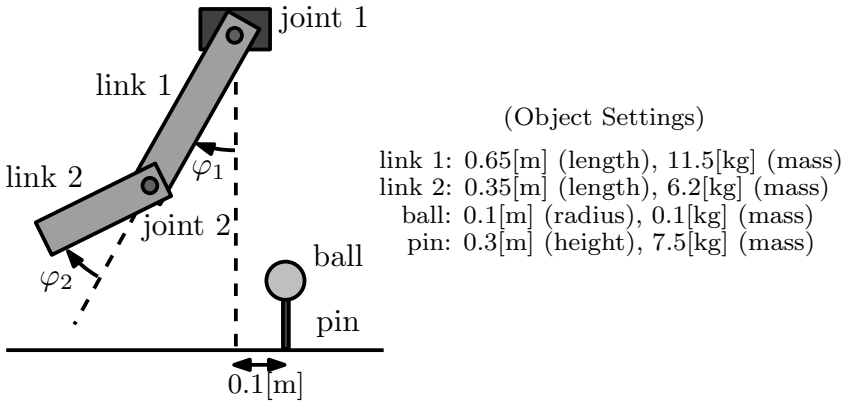


Figure 9: A ball-batting robot.

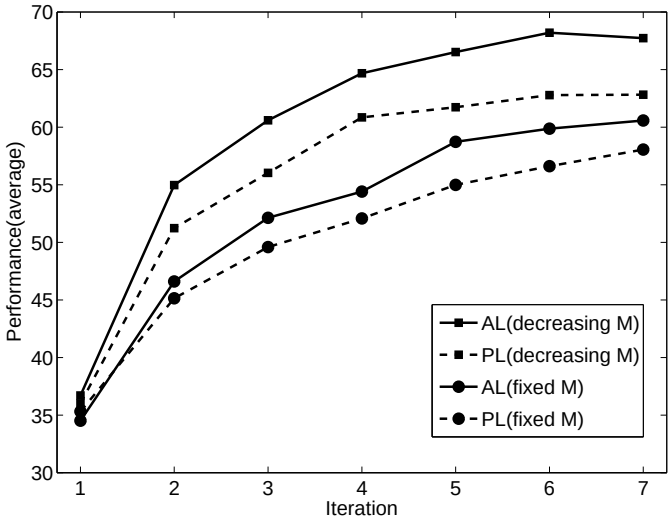


Figure 10: The mean performance over 500 trials in the ball-batting experiment. The dotted lines denote the performance of passive learning (PL) and the solid lines denote the performance of the proposed active learning (AL) method. The error bars are omitted for clear visibility. For the 'decreasing M' strategy, the performance of AL after the 7-th iteration is significantly better than that of PL according to the two-tailed paired Student t-test at the significance level 1% for the error values at the 7-th iteration.

approximation and thus consistency may not be guaranteed. Although we experimentally confirmed that the single-sample approximation gave reasonably good performance, it is important to theoretically investigate the convergence issue in the future work.

The R-max strategy (Brafman & Tennenholtz, 2002) is an approach to controlling the trade-off between exploration and exploitation in the model-based RL framework. The *LSPI R-max* method (Strehl et al., 2007; Li, Littman, & Mansley, 2009) is an application

of the R-max idea to the LSPI framework. It is therefore interesting to investigate the relation between the LSPI R-max method and the proposed method. Moreover, exploring alternative active learning strategies in the model-based RL formulation would be a promising research direction in the future.

Acknowledgments

We thank fruitful comments from anonymous reviewers. MS was supported by Grant-in-Aid for Young Scientists (A), 20680007, AOARD, and SCAT.

References

Bishop, C. M. (2006). *Pattern recognition and machine learning*. New York: Springer.

Brafman, R. I., & Tennenholtz, M. (2002). R-max—a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, *3*, 213–231.

Cohn, D. A., Ghahramani, Z., & Jordan, M. I. (1996). Active learning with statistical models. *Journal of Artificial Intelligence Research*, *4*, 129–145.

Fedorov, V. V. (1972). *Theory of optimal experiments*. New York: Academic Press.

Hachiya, H., Akiyama, T., Sugiyama, M., & Peters, J. (2009). Adaptive importance sampling for value function approximation in off-policy reinforcement learning. *Neural Networks*, *22*(10), 1399–1410.

Henkel, R. E. (1979). *Tests of significance*. Beverly Hills: SAGE Publication.

Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, *12*(3), 55–67.

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, *4*, 237–285.

Kanamori, T., & Shimodaira, H. (2003). Active learning algorithm using the maximum weighted log-likelihood estimator. *Journal of Statistical Planning and Inference*, *116*(1), 149–162.

Kearns, M., & Singh, S. (1998). Near-optimal reinforcement learning in polynomial time. In *Proceedings of international conference on machine learning* (pp. 260–268).

Kearns, M., & Singh, S. (2002). Near-optimal reinforcement learning in polynomial time. *Machine Learning*, *49*, 209–232.

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

Li, L., Littman, M. L., & Mansley, C. R. (2008). *Exploration in least-squares policy iteration* (Tech. Rep. No. DCS-TR-641). Rutgers, The State University of New Jersey.

Li, L., Littman, M. L., & Mansley, C. R. (2009). Online exploration in least-squares policy iteration. In *Proceedings of autonomous agents and multiagent systems.*

Poggio, T., & Girosi, F. (1990). Networks for approximation and learning. *Proceedings of the IEEE*, *78*(9), 1481–1497.

Precup, D., Sutton, R. S., & Singh, S. (2000). Eligibility traces for off-policy policy evaluation. In *Proceedings of the seventeenth international conference on machine learning* (pp. 759–766). Morgan Kaufmann.

Strehl, A. L., Diuk, C., & Littman, M. L. (2007). Efficient structure learning in factored-state mdps. In *Proceedings of the twenty-second national conference on artificial intelligence* (pp. 645–650).

Sugiyama, M. (2006). Active learning in approximately linear regression based on conditional expectation of generalization error. *Journal of Machine Learning Research*, *7*, 141–166.

Sugiyama, M., & Nakajima, S. (2009). Pool-based active learning in approximate linear regression. *Machine Learning*, *75*(3), 249-274.

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

Tikhonov, A. N., & Arsenin, V. Y. (1977). *Solutions of ill-posed problems*. Washington DC: V. H. Winston.

Wiens, D. P. (2000). Robust weights and designs for biased regression models: Least squares and generalized M-estimation. *Journal of Statistical Planning and Inference*, *83*(2), 395–412.