Active Policy Iteration: Efficient Exploration through Active Learning for Value Function Approximation in Reinforcement Learning

Takayuki Akiyama, Hirotaka Hachiya, and Masashi Sugiyama
Department of Computer Science, Tokyo Institute of Technology
{akiyama@sg., hachiya@sg., sugi@}cs.titech.ac.jp

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. We demonstrate the usefulness of the proposed method, named *active policy iteration* (API), through simulations with a batting robot.

1 Introduction

In practical reinforcement learning (RL), 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 drive the ball as far as possible (see Figure 5). Let us use the carry distance 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 and 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, and unpredictable disturbances such a current of air. Thus, in practice, we may have to put the robot in a large place, 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. In this paper, we first show that the least-squares policy iteration (LSPI) framework [Lagoudakis and Parr, 2003] allows us to use statistical active learning (AL) methods for linear regression [Cohn *et al.*, 1996; Sugiyama, 2006].

In the LSPI framework, the state-action value function is approximated by fitting a linear model with least-squares estimation. A traditional AL scheme [Cohn *et al.*, 1996] is designed to find the input distribution[1] that minimizes the variance of the least-squares estimator. Since the expected approximation error of the value function is expressed as the sum of the (squared) bias and variance, the bias needs to be zero for justifying the use of the traditional AL scheme. To this end, we need to assume that the linear model used for approximating the value function 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, an importance-sampling based AL method has been developed recently [Sugiyama, 2006]. This AL algorithm is valid even when the model is misspecified, i.e., even when the true value function is not included in the model—which would be a usual case in practice—a good input distribution can be designed.

In this paper, we develop a new exploration scheme for LSPI based on the importance-sampling based AL idea. The proposed method combined with LSPI is called *active policy iteration* (API). Through batting-robot simulations, the usefulness of API is demonstrated.

2 Formulation of RL Problem

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

MDPs: Let us consider an 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 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, and γ $(\in (0,1])$ 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

[1] When approximating the state-action value function, the input distribution corresponds to the stationary distribution of states and actions.

discounted sum of rewards the agent will receive when taking action a in state s and following policy π thereafter, i.e.,

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

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 RL is to obtain the policy which maximizes the discounted sum of future rewards; the optimal policy can be expressed as $\pi^*(a|s) \equiv \delta(a - \mathrm{argmax}_{a'} Q^*(s,a'))$, where $\delta(\cdot)$ is Dirac's delta function and $Q^*(s,a) \equiv \max_{\pi} Q^{\pi}(s,a)$ is the *optimal* state-action value function.

$Q^{\pi}(s,a)$ can be expressed as the following recurrent form called the *Bellman equation*: $\forall s \in \mathcal{S}, \forall a \in \mathcal{A}$,

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

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

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:

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

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

$$\pi'(a|s) = \epsilon p_{\mathrm{u}}(a) + (1-\epsilon) I(a = \mathop{\arg\max}_{a'} Q^{\pi}(s,a')),$$

where $I(c)$ is the indicator function (1 if c is true and 0 otherwise), p_{u} is the uniform probability density over actions, and ϵ $(\in (0,1])$ determines how deterministic 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 and Barto, 1998].

Least-squares Framework for Value Function Approximation: Although policy iteration is useful, 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, we approximate the state-action value function $Q^{\pi}(s,a)$ using the following linear model:

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

where $\boldsymbol{\phi}(s,a) = (\phi_1(s,a), \phi_2(s,a), \ldots, \phi_B(s,a))^{\top}$ are the fixed linearly independent basis functions, $\top$ denotes the transpose, B is the number of basis functions, and $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_B)^{\top}$ 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 squared Bellman residual $G(\boldsymbol{\theta})$ is minimized [Lagoudakis and Parr, 2003]:

$$\boldsymbol{\theta}^* \equiv \mathop{\arg\min}_{\boldsymbol{\theta}} G(\boldsymbol{\theta}),$$

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

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

$\mathbb{E}_{P_{\pi}}$ denotes the expectation over the joint probability density function $P_{\pi}(s_1,a_1,s_2,a_2,\ldots,s_N,a_N,s_{N+1}) \equiv P_{\mathrm{I}}(s_1)\prod_{n=1}^{N}P_{\mathrm{T}}(s_{n+1}|s_n,a_n)\pi(a_n|s_n)$, where $P_{\mathrm{I}}(s)$ denotes the initial-state probability density function.

Value Function Learning from Samples: Suppose that roll-out data samples consisting of M episodes with N steps are available as training data. 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 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

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

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 *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:

$$\widehat{\boldsymbol{\theta}} \equiv \mathop{\mathrm{argmin}}_{\boldsymbol{\theta}} \widehat{G}(\boldsymbol{\theta}),$$

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

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

where $\mathcal{D}_{(s,a)}$ is a set of 4-tuple elements 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

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

is called the *importance weight* [Sutton and 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 *et al.*, 2000], which is more efficient

to calculate. $\widehat{\boldsymbol{\theta}}$ 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 $\boldsymbol{r}^{\widetilde{\pi}}(\in \mathbb{R}^{MN\times 1})$ as

$$\widehat{\boldsymbol{\theta}} = \widehat{\boldsymbol{L}}\boldsymbol{r}^{\widetilde{\pi}}, \qquad \widehat{\boldsymbol{L}} \equiv (\widehat{\boldsymbol{X}}^\top \boldsymbol{W}\widehat{\boldsymbol{X}})^{-1}\widehat{\boldsymbol{X}}^\top \boldsymbol{W},$$

$$\boldsymbol{r}^{\widetilde{\pi}}_{N(m-1)+n} \equiv r^{\widetilde{\pi}}_{m,n}, \quad \widehat{\boldsymbol{X}}_{N(m-1)+n,b} \equiv \widehat{\psi}_b(s^{\widetilde{\pi}}_{m,n}, a^{\widetilde{\pi}}_{m,n}; \mathcal{D}^{\widetilde{\pi}}),$$

$$\boldsymbol{W}_{N(m-1)+n,N(m'-1)+n'} \equiv \begin{cases} w^{\widetilde{\pi}}_{m,n} & \text{if } (m,n) = (m',n'), \\ 0 & \text{if } (m,n) \neq (m',n'). \end{cases}$$

3 Efficient Exploration with Active Learning

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

Preliminaries: Here we consider the case where collecting state-action trajectory samples is easy and cheap, but gathering immediate reward samples is hard and expensive (examples include the batting robot explained in the introduction). In such a case, immediate reward samples cannot be used for designing the sampling policy, but only state-action trajectory samples are available.

The goal of active learning 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, so the generalization error needs to be estimated from samples. A difficulty of estimating the generalization error in the context of active learning is that its estimation needs to be carried out only from state-action trajectory samples *without* using immediate reward samples; thus standard techniques such as *cross-validation* [Hachiya *et al.*, 2008] cannot be employed since it requires both state-action and immediate reward samples. Below, we explain how the generalization error could be estimated under the active learning setup.

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:

$$\overline{\mathcal{D}}^{\widetilde{\pi}} \equiv \{\overline{d}^{\widetilde{\pi}}_m\}_{m=1}^M, \quad \overline{d}^{\widetilde{\pi}}_m \equiv \{(s^{\widetilde{\pi}}_{m,n}, a^{\widetilde{\pi}}_{m,n}, s^{\widetilde{\pi}}_{m,n+1})\}_{n=1}^N.$$

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

$$\epsilon^{\widetilde{\pi}}_{m,n} \equiv r^{\widetilde{\pi}}_{m,n} - R(s^{\widetilde{\pi}}_{m,n}, a^{\widetilde{\pi}}_{m,n}).$$

Note that $\epsilon^{\widetilde{\pi}}_{m,n}$ could be regarded as additive noise in least-squares function fitting. By definition, $\epsilon^{\widetilde{\pi}}_{m,n}$ has mean zero and the variance may depend on $s^{\widetilde{\pi}}_{m,n}$ and $a^{\widetilde{\pi}}_{m,n}$, i.e., *heteroscedastic* noise. However, since estimating the variance of $\epsilon^{\widetilde{\pi}}_{m,n}$ without using reward samples is not possible, we assume that the variance does not depend on $s^{\widetilde{\pi}}_{m,n}$ and $a^{\widetilde{\pi}}_{m,n}$—let us denote the common variance by σ^2.

Now we would like to estimate the generalization error

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

from $\overline{\mathcal{D}}^{\widetilde{\pi}}$. Its expectation can be decomposed as

$$\mathop{\mathbb{E}}_{\boldsymbol{\epsilon}^{\widetilde{\pi}}} \overline{G}(\widehat{\boldsymbol{\theta}}) = B + V + C,$$

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

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

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

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

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

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

Note that the variance term V can be expressed as

$$V = \sigma^2 \mathrm{tr}(\boldsymbol{U}\widehat{\boldsymbol{L}}\widehat{\boldsymbol{L}}^\top).$$

Estimation of Generalization Error for AL: The model error C is constant and can be safely ignored in generalization error estimation. So we only need to estimate the bias term B and the variance term V. However, B includes the unknown optimal parameter $\boldsymbol{\theta}^*$ and therefore it may not be possible to estimate B without using reward samples; similarly, it may not be possible to estimate the 'noise' variance σ^2 included in the variance term V without using reward samples.

It is known that the bias term B 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

$$\mathop{\mathbb{E}}_{\boldsymbol{\epsilon}^{\widetilde{\pi}}} \overline{G}(\widehat{\boldsymbol{\theta}}) - C - B \propto \mathrm{tr}(\boldsymbol{U}\widehat{\boldsymbol{L}}\widehat{\boldsymbol{L}}^\top),$$

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

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

Consequently, we have the following generalization error estimator:

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

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 $\overline{J}$ may be used as a generalization error estimator.

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 $\{\overline{\mathcal{D}}^{\widetilde{\pi}_k}\}_{k=1}^K$:

$$\widetilde{\boldsymbol{U}} = \frac{1}{KMN}\sum_{k=1}^{K}\sum_{m=1}^{M}\sum_{n=1}^{N}\widehat{\boldsymbol{\psi}}(s_{m,n}^{\widetilde{\pi}_k}, a_{m,n}^{\widetilde{\pi}_k}; \{\overline{\mathcal{D}}^{\widetilde{\pi}_k}\}_{k=1}^K) \times \widehat{\boldsymbol{\psi}}(s_{m,n}^{\widetilde{\pi}_k}, a_{m,n}^{\widetilde{\pi}_k}; \{\overline{\mathcal{D}}^{\widetilde{\pi}_k}\}_{k=1}^K)^\top w_{m,n}^{\widetilde{\pi}_k}.$$

4. Estimate the generalization error for each k:

$$J_k \equiv \mathrm{tr}(\widetilde{\boldsymbol{U}}\widehat{\boldsymbol{L}}^{\widetilde{\pi}_k}\widehat{\boldsymbol{L}}^{\widetilde{\pi}_k\top}),$$
$$\widehat{\boldsymbol{L}}^{\widetilde{\pi}_k} \equiv (\widehat{\boldsymbol{X}}^{\widetilde{\pi}_k\top}\boldsymbol{W}^{\widetilde{\pi}_k}\widehat{\boldsymbol{X}}^{\widetilde{\pi}_k})^{-1}\widehat{\boldsymbol{X}}^{\widetilde{\pi}_k\top}\boldsymbol{W}^{\widetilde{\pi}_k},$$
$$\widehat{\boldsymbol{X}}_{N(m-1)+n,b}^{\widetilde{\pi}_k} \equiv \widehat{\psi}_b(s_{m,n}^{\widetilde{\pi}_k}, a_{m,n}^{\widetilde{\pi}_k}; \{\overline{\mathcal{D}}^{\widetilde{\pi}_k}\}_{k=1}^K),$$
$$\boldsymbol{W}_{N(m-1)+n,N(m'-1)+n'}^{\widetilde{\pi}_k} \equiv \begin{cases} w_{m,n}^{\widetilde{\pi}_k} & \text{if } (m,n)=(m',n'), \\ 0 & \text{if } (m,n)\neq(m',n'). \end{cases}$$

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

Numerical Examples: Here we illustrate how the proposed method behaves in the 10-state chain-walk environment shown in Figure 1. The MDP consists of 10 states $\mathcal{S} = \{s^{(i)}\}_{i=1}^{10} = \{1, 2, \ldots, 10\}$ and 2 actions $\mathcal{A} = \{a^{(i)}\}_{i=1}^{2} = \{\text{'L'}, \text{'R'}\}$. The immediate reward function $R(s, a, s')$ is defined by $R(s, a, s') \equiv 0.3\times(7-|s'-7|)$. The transition probability $P_{\mathrm{T}}(s'|s, a)$ is indicated by the numbers attached to the arrows in Figure 1; for example, $P_{\mathrm{T}}(s^{(2)}|s^{(1)}, \text{'R'}) = 0.9$ and $P_{\mathrm{T}}(s^{(1)}|s^{(1)}, \text{'R'}) = 0.1$. Thus the agent can successfully move to the intended direction 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). The discount factor γ is set to 0.9. We use the 12 basis functions $\boldsymbol{\phi}(s, a)$ defined as

$$\phi_{2(i-1)+j}(s,a) = \begin{cases} I(a=a^{(j)})\exp(-(s-c_i)^2/(2\sigma^2)) & \text{for } 1\le i\le 5, 1\le j\le 2, \\ I(a=a^{(j)}) & \text{for } i=6, 1\le j\le 2, \end{cases}$$

where $c_1 = 1$, $c_2 = 3$, $c_3 = 5$, $c_4 = 7$, $c_5 = 9$, and $\sigma = 3.0$.

For illustration purposes, we evaluate the selection of sampling policies only in one-step policy evaluation; evaluation over iteration will be addressed in the next section. Sampling policies and evaluation policies are constructed as follows.

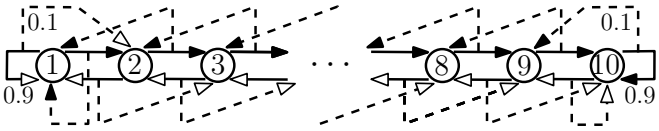


Figure 1: 10-state chain walk. Filled/unfilled arrows indicate the transitions when taking action 'R'/'L' and solid/dashed lines indicate the success/failed transitions.

Table 1: The parameters of sampling policy candidates $\{\widetilde{\pi}_k\}_{k=1}^{10}$ for the 10-state chain-walk simulation.

k	1	2	3	4	5	6	7	8	9	10
ϵ_k	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
α_k	R	R	R	R	R	L	L	L	L	L
μ_k	0.5	0.4	0.3	0.2	0.1	0.1	0.2	0.3	0.4	0.5

First, we prepare a deterministic 'base' policy, e.g., 'LLLLLRRRRR', where the i-th letter denotes the action taken at $s^{(i)}$. Let π_ϵ be the 'ϵ-greedy' version of the base policy, 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 use $\pi_{0.1}$ as the evaluation policy π. 10 candidates of the sampling-policy $\{\widetilde{\pi}_k\}_{k=1}^{10}$ are constructed as follows: with probability $(1-\mu_k)$, the agent chooses an action following π_{ϵ_k}; with probability μ_k, the agent takes action α_k. The settings of the sampling-policy parameters $\{\epsilon_k, \alpha_k, \mu_k\}_{k=1}^{10}$ are summarized in Table 1.

For each sampling policy, we calculate the J-value 5 times and take the average. The numbers of episodes M and steps N are set to 10 and 5, respectively; the initial-state probability $P_{\mathrm{I}}(s)$ is set to be uniform. This experiment is repeated 100 times and we evaluate the mean and standard deviation of the true generalization error and its estimate.

The results are depicted in Figure 2 (the true generalization error) and Figure 3 (its estimate) as functions of the index k of the sampling policies. We are interested in choosing k such that the true generalization error is minimized. The results show that the proposed generalization error estimator overall captures the trend of the true generalization error well. Thus the proposed generalization error estimator would be useful for choosing a good sampling policy.

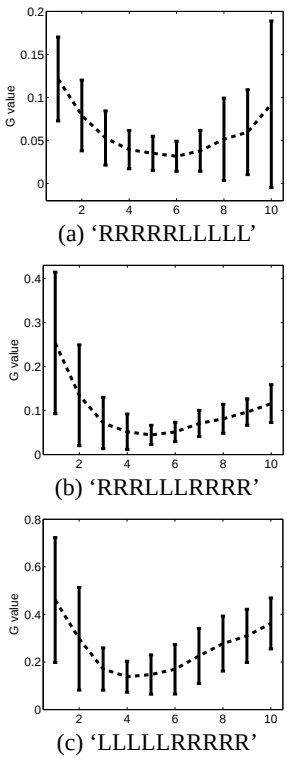
4 Active Learning in Policy Iteration

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.

Sample-reuse Policy Iteration (SRPI) with Active Learning: SRPI is a policy-iteration framework which allows us to reuse previously-collected samples efficiently [Hachiya *et al.*, 2008]. Let us denote the evaluation policy at the l-th iteration by π_l and the maximum number of iterations by L.

In ordinary 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:

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

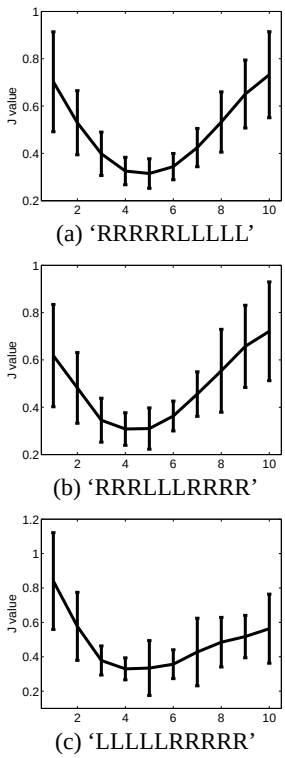


(a) ‘RRRRRLLLLL’

(b) ‘RRRLLLRRRR’

(c) ‘LLLLLRRRRR’

Figure 2: The mean and standard deviation of the true generalization error over 100 trials.



(a) ‘RRRRRLLLLL’

(b) ‘RRRLLLRRRR’

(c) ‘LLLLLRRRRR’

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

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 performing policy evaluation as:

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

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

Here, we apply the active learning 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

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

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 active learning method given in the previous section. We call this framework *active policy iteration* (API).

Numerical Examples: Here we illustrate how the API method behaves using the same 10-state chain-walk problem (see Figure 1). The initial evaluation policy π_1 is set to be uniform and policies are updated in the l-th iteration using the ϵ-greedy rule with $\epsilon = 2^{-l}$. In the sampling-policy selection step of the l-th iteration, we prepare the two sampling-policy candidates $\{\widetilde{\pi}_k^{(l)}\}_{k=1}^2$ with $(\epsilon_1, \alpha_1, \mu_1) = (0.4, \text{‘R’}, 0.8)$, $(\epsilon_2, \alpha_2, \mu_2) = (0.4, \text{‘L’}, 0.8)$. The number M of episodes and the number N of steps are both set to 5, and J-value calculation is repeated 5 times for active learning. The performance of the 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}):

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

where the discount factor γ is set to 0.9.

We compare the performance of fixed sampling policies $\{\widetilde{\pi}_k\}_{k=1}^2$ and active learning of choosing the best sampling policy from $\{\widetilde{\pi}_k\}_{k=1}^2$. The results are depicted in Figure 4, showing that the proposed method works very well. Actually, the proposed method outperforms the best fixed strategy ($k = 2$); this can happen since the optimal sampling policy is not always $k = 2$ and it varies in each trial depending on randomness of the training dataset. Thus, the results show that the proposed active learning scheme can *adaptively* choose a good policy based on the training dataset at hand.

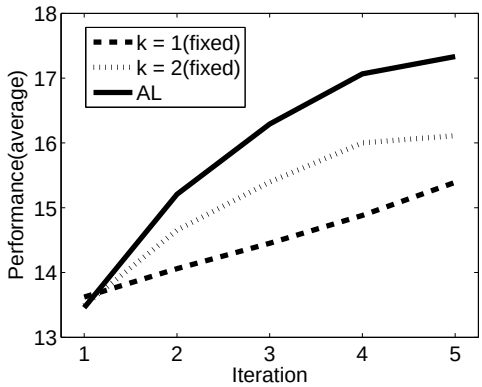


Figure 4: The mean performance over 150 trials in the 10-state chain-walk experiment. The dotted lines denote the performance when fixed sampling policies are used and the solid line denotes the performance when the sampling policies are optimized by the proposed AL method. The error bars are omitted for clear visibility; but they were all reasonably small.

5 Experiments

Finally, we evaluate our proposed method using a ball-batting robot illustrated in Figure 5, 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 as possible. The state space $\mathcal{S}$ is continuous and consists of the angles $\varphi_1[\mathrm{rad}]$ ($\in [0, \pi/4]$) and $\varphi_2[\mathrm{rad}]$ ($\in [-\pi/4, \pi/4]$) and the angular velocities $\dot{\varphi}_1[\mathrm{rad/s}]$ and $\dot{\varphi}_2[\mathrm{rad/s}]$. Thus a state s ($\in \mathcal{S}$) is described by a four-dimensional vector: $s = (\varphi_1, \dot{\varphi}_1, \varphi_2, \dot{\varphi}_2)^\top$. The action space $\mathcal{A}$ is discrete and contains two elements: $\mathcal{A} = \{a^{(i)}\}_{i=1}^2 = \{(50, -35)^\top, (-50, 10)^\top\}$, where the i-th

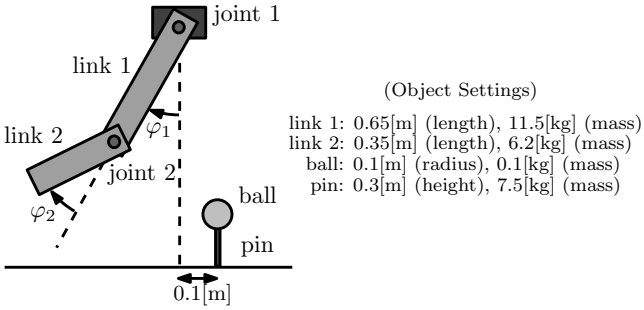


Figure 5: A ball-batting robot.

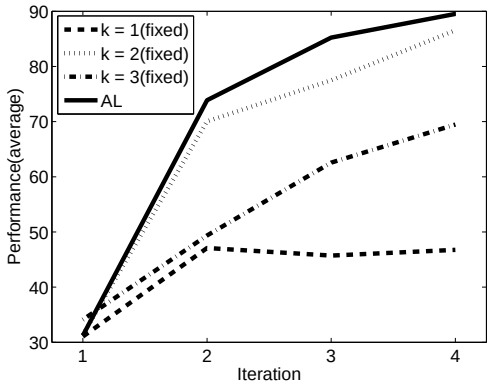


Figure 6: The mean performance over 100 trials in the ball-batting experiment. The dotted lines denote the performance when fixed sampling policies are used and the solid line denotes the performance when the sampling policies are optimized by the proposed AL method. The error bars are omitted for clear visibility.

element $(i = 1, 2)$ of each vector corresponds to the torque $[\mathrm{N} \cdot \mathrm{m}]$ 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 $f_1 + \delta_1$ and $f_2 + \delta_2$, where $\delta_i (i = 1, 2)$ follows the standard normal distribution independently.

The immediate reward is defined as the horizontal carry of the ball; this reward is given only when the robot arm collides with the ball for the first time at the state s' after taking action a at the current state s.

We use the 110 basis functions defined as

$$\phi_{2(i-1)+j} = \begin{cases} I(a = a^{(j)}) \exp(-\| s - c_j \|^2 / (2\sigma^2)) \\ \qquad\qquad \text{for } 1 \le i \le 54, 1 \le j \le 2, \\ I(a = a^{(j)}) \quad \text{for } i = 55, 1 \le j \le 2, \end{cases}$$

where σ is set to $3\pi/2$ and c_j $(j = 1, 2, \ldots, 54)$ are located on the regular grid $\{-\pi/4, 0\} \times \{-\pi, 0, \pi\} \times \{-\pi/4, 0, \pi/4\} \times \{-\pi, 0, \pi\}$. We set $M = 20$ and $N = 12$ and the initial state is always set to $s = (\pi/4, 0, 0, 0)^\top$. The initial evaluation policy is set to the ϵ-greedy version of the base policy with $\epsilon = 0.5$; the base policy is defined by the greedy update using the 'constant' $\widehat{Q}$ function with $\widehat{\theta}_i = 0.5$ $(1 \le i \le B)$. Policies are updated in the l-th iteration using the ϵ-greedy rule with $\epsilon = 2^{-(1+l)}$. The set of sampling-policy candidates $\{\pi_k^{(l)}\}_{k=1}^3$ in the l-th iteration is defined as $(\epsilon_1, \alpha_1, \mu_1) = (0.1, (50, -35)^\top, 0.7)$, $(\epsilon_2, \alpha_2, \mu_2) = (0.4, *, 0.0)$, and $(\epsilon_3, \alpha_3, \mu_3) = (0.1, (-50, 10)^\top, 0.25)$, where the symbol '$*$' means "don't care" since the value of μ is zero. The discount factor γ is set to 0.95 and the performance of the learned policy π_{L+1} is measured by the discounted sum of immediate rewards for test samples $\{r_{m,n}^{\pi_{L+1}}\}_{m=1,n=1}^{20,12}$ (20 episodes with 12 steps collected following π_{L+1}):

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

The results are depicted in Figure 6, showing that the proposed method works very well and it is comparable to or slightly better than the best fixed strategy ($k = 2$). Based on the experimental evaluation, we conclude that the proposed sampling-policy design method, API, is useful in improving the RL performance.

6 Conclusions

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

References

[Cohn *et al.*, 1996] D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

[Hachiya *et al.*, 2008] H. Hachiya, T. Akiyama, M. Sugiyama, and J. Peters. Adaptive importance sampling with automatic model selection in value function approximation. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI2008)*, pages 1351–1356, Chicago, USA, Jul. 13–17 2008. The AAAI Press.

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

[Precup *et al.*, 2000] D. Precup, R. S. Sutton, and S. Singh. Eligibility traces for off-policy policy evaluation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 759–766, Morgan Kaufmann, 2000.

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

[Sutton and Barto, 1998] R. S. Sutton and G. A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.