IEICE Transactions on Information and Systems,
vol.E93-D, no.9, pp.2555–2565, 2010.

Least Absolute Policy Iteration—A Robust Approach to Value Function Approximation

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

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

Hisashi Kashima (kashima@mist.i.u-tokyo.ac.jp)
Department of Mathematical Informatics, the University of Tokyo

Tetsuro Morimura (tetsuro@jp.ibm.com)
IBM Research - Tokyo

Abstract

Least-squares policy iteration is a useful reinforcement learning method in robotics due to its computational efficiency. However, it tends to be sensitive to outliers in observed rewards. In this paper, we propose an alternative method that employs the absolute loss for enhancing robustness and reliability. The proposed method is formulated as a linear programming problem which can be solved efficiently by standard optimization software, so the computational advantage is not sacrificed for gaining robustness and reliability. We demonstrate the usefulness of the proposed approach through a simulated robot-control task.

Keywords

Reinforcement learning, value function approximation, least-squares policy iteration, outlier, ℓ_1-loss function, linear programming

1 Introduction

One of the popular reinforcement learning frameworks for obtaining the optimal policy is *policy iteration*, which performs policy evaluation and improvement steps iteratively [28, 4]. The computational cost of naive implementation of policy iteration is dominated by the number of states and actions, so it is not scalable to real-world robotics problems with large state/action space. To cope with this problem, an alternative method called *least-squares policy iteration (LSPI)* has been proposed [16]. In LSPI, value functions of policies are approximated using linear architecture, so its computational cost is governed by the number of parameters in the linear model. Thus, if the number of parameters is kept reasonably small, LSPI is applicable to large-scale robot-control tasks.

A basic idea of LSPI is to learn the parameters of the linear model so that the *temporal-difference (TD)* error is minimized under the squared loss. On the other hand, in this paper, we propose to minimize the TD error under the absolute loss (see Figure 1). This is just a replacement of the loss function, but we argue that this modification brings about highly useful advantages in practical robotics problems. More specifically, the rationale behind the use of the absolute loss lies in *robustness* and *reliability*.

1.1 Robustness

In many robotics applications, immediate rewards are obtained through measurement such as distance sensors or computer vision. Due to intrinsic measurement noise or recognition error, the obtained rewards often deviate from the true value; in particular, the rewards occasionally contain *outliers*, which are significantly different from regular values.

Residual minimization under the squared loss amounts to obtaining the *mean* of samples $\{x_i\}_{i=1}^m$:

$$\operatorname*{argmin}_c \left[\sum_{i=1}^m (x_i - c)^2\right] = \mathrm{mean}(\{x_i\}_{i=1}^m) = \frac{1}{m}\sum_{i=1}^m x_i.$$

If one of the values is very large, the mean would be strongly affected by this outlier sample. Thus *all* the values $\{x_i\}_{i=1}^m$ are responsible for the mean, and therefore even a *single* outlier observation can significantly damage the learned result.

On the other hand, residual minimization under the absolute loss amounts to obtaining the *median*.

$$\operatorname*{argmin}_c \left[\sum_{i=1}^{2n+1} |x_i - c|\right] = \mathrm{median}(\{x_i\}_{i=1}^{2n+1}) = x_{n+1},$$

where $x_1 \leq x_2 \leq \cdots \leq x_{2n+1}$. The median is influenced not by the *magnitude* of the values $\{x_i\}_{i=1}^{2n+1}$ but only by their *order*. Thus, as long as the order is kept unchanged, the median is not affected by outliers—in fact, the median is known to be the most robust estimator in the light of *breakdown-point analysis* [13, 26].

Therefore, the use of the absolute loss would remedy the problem of robustness in policy iteration.

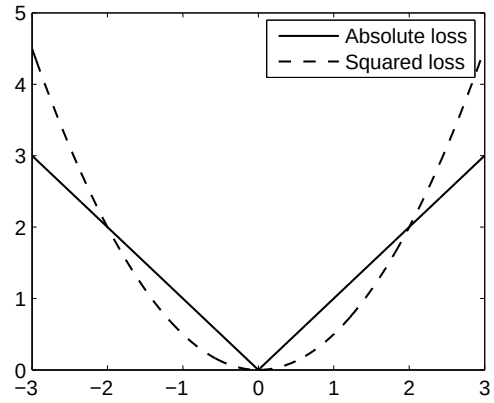


Figure 1: The absolute and squared loss functions for reducing the temporal-difference error.

1.2 Reliability

In practical robot-control tasks, we often want to attain a stable performance, rather than to achieve a "dream" performance with a little chance of success; for example, in the acquisition of a humanoid gait, we may want the robot to walk forward in a stable manner with high probability of success, rather than to rush very fast in a chance level.

On the other hand, we do not want to be too conservative when training robots—if we are overly concerned with unrealistic failure, no practically useful control policy could be obtained. For example, any robots can be broken in principle if activated for long time. However, if we fear this fact too much, we may end up in a control policy that does not activate the robots at all—obviously this is non-sense in practice.

Since the squared loss solution is not robust against outliers, it is sensitive to rare events with either positively or negatively very large immediate rewards. Consequently, the squared loss prefers an extraordinarily successful motion even if the success probability is very low; similarly, it dislikes an unrealistic trouble even if such a terrible event may not happen in practice. On the other hand, the absolute loss solution is not easily affected by such rare events due to robustness. Therefore, the use of the absolute loss would produce a reliable control policy even in the presence of such extreme events.

1.3 Goal of This Paper

As shown above, the use of the absolute loss in value function approximation would bring about robustness and reliability, which are preferable properties in real-world robotics problems. This modification is very simple, but to the best of our knowledge, such an idea has never been incorporated in value function approximation.

Another important advantage of the proposed approach is that scalability to massive data is not sacrificed for enhancing robustness and reliability. Indeed, the absolute-loss solution can be obtained by solving a *linear programming* problem; this can be carried out very efficiently by a standard optimization software. We demonstrate the usefulness of the absolute-loss approach through robotics simulations.

1.4 Related Work

In the seminal paper [10], the *α-value criterion* was introduced as an alternative to the expected discounted reward. This criterion is essentially identical to the *value-at-risk* of the discounted reward, which is a popular risk measure in finance [25]. However, the resulting optimization problem is not convex, and therefore it is difficult to obtain a good solution efficiently [3]. A *soft risk aversion method* [19] emphasizes actions whose rewards are less than expected. Although the idea of this approach is intuitive, it is rather heuristic and does not have a clear interpretation as risk minimization. In the paper [27], an approach that optimizes a linear combination of the mean and the variance of discounted rewards was proposed. This approach is based on the *mean-variance model*, which is popular modeling also in finance [18]. The assumption behind the mean-variance model is that the discounted rewards follow the Gaussian distribution, which may not be true in practice. On the other hand, the method proposed in this paper has a clear interpretation as *median* risk minimization and no strong assumption is imposed on the rewards. Furthermore, the resulting optimization problem is a linear program, which is convex and can be solved efficiently using standard optimization software.

In the area of optimal control, *robust control theory* was used to design stable controllers [15, 20, 2]. Although this approach is also sometimes referred to as *robust* reinforcement learning, its aim is different from the current paper—robust control aims at enhancing robustness against *uncertainties in the environment*, while our goal is to enhance robustness against *outliers in the rewards*.

2 Problem Formulation

In this section, we formulate the reinforcement learning problem using a Markov decision process (MDP), and briefly review the core ideas of policy iteration and value function approximation.

2.1 Markov Decision Process

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 transition probability-density from state s to next state s' when action a is taken, $R(s,a,s')$ ($\in \mathbb{R}$) is a reward for transition from s to s' by taking action a, 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 $Q(s,a)$ ($\in \mathbb{R}$) for policy π is the expected discounted sum of rewards the agent will receive when taking action a

in state s and following policy π thereafter, i.e.,

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

where $\mathbb{E}_{\pi,P_{\mathrm{T}}}$ denotes the expectation over trajectory $\{s_n,a_n\}_{n=1}^{\infty}$ following $\pi(a_n|s_n)$ and $P_{\mathrm{T}}(s_{n+1}|s_n,a_n)$. The goal of reinforcement learning is to obtain the policy that maximizes the discounted sum of future rewards.

Computing the value function $Q(s,a)$ is called *policy evaluation* since this corresponds to evaluating the value of policy π. Using $Q(s,a)$, we may find a better policy as

$$\pi(a|s) \leftarrow \delta(a-\mathop{\mathrm{argmax}}_{a'} Q(s,a')),$$

where $\delta(\cdot)$ is Dirac's delta function. This is called *policy improvement*. It is known that repeating policy evaluation and policy improvement leads to the optimal policy under some condition [28]. This entire process is called *policy iteration*.

2.2 Value Function Approximation

Although policy iteration is guaranteed to produce the optimal policy, it is computationally intractable when the number of state-action pairs $|\mathcal{S}|\times|\mathcal{A}|$ is very large; $|\mathcal{S}|$ or $|\mathcal{A}|$ becomes infinity when the state space or action space is continuous. To overcome this problem, the state-action value function $Q(s,a)$ may be approximated using the following linear model:

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

where

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

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

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

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

Suppose we have an N-step data sample, i.e., the agent initially starts from a randomly selected state s_1 following the initial-state probability density $P_{\mathrm{I}}(s_1)$ and chooses an action based on the current policy $\pi(a_n|s_n)$. Then the agent makes a transition following $P_{\mathrm{T}}(s_{n+1}|s_n,a_n)$ and receives an immediate reward r_n $(=R(s_n,a_n,s_{n+1}))$—thus the training dataset $\mathcal{D}$ is expressed as

$$\mathcal{D} \equiv \{(s_n,a_n,r_n,s_{n+1})\}_{n=1}^{N}.$$

The *temporal-difference (TD)* error for the n-th sample is defined by

$$\boldsymbol{\theta}^{\top}\widehat{\boldsymbol{\psi}}(s_n,a_n) - r_n, \tag{2}$$

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

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

$\mathcal{D}_{(s,a)}$ is a set of 4-tuple elements (s,a,r,s') 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 $\mathbb{E}_{\pi(a'|s')}$ denotes the conditional expectation with respect to a' over $\pi(a'|s')$ given s'.

The issue we would like to address in this paper is the choice of the loss function when evaluating the TD error (2); more specifically, we argue that the use of the absolute loss is more advantageous than the popular squared loss [28, 23, 16]. We note that our results could be easily extended to various settings—for example, multiple sequences of episodic training samples can be employed without essentially changing the framework. The *off-policy* scenarios where the sampling policy is different from the evaluation policy can also be incorporated by applying *importance-weighting* techniques [28, 24, 9]. However, we do not go into the details of such generalization for keeping the presentation of the current paper simple.

3 Loss Functions for TD-error Minimization

In this section, we first review a squared-loss method for TD-error minimization, and then introduce an absolute-loss method.

3.1 Least-Squares Policy Iteration (LSPI)

A standard choice of the loss function for minimizing the residual error would be the squared loss [28, 23, 16]. The least-squares TD-error solution $\widehat{\boldsymbol{\theta}}_{\mathrm{LS}}$ is defined as

$$\widehat{\boldsymbol{\theta}}_{\mathrm{LS}} \equiv \mathop{\mathrm{argmin}}_{\boldsymbol{\theta}} \left[\frac{1}{2} \sum_{n=1}^{N} \left(\boldsymbol{\theta}^\top \widehat{\boldsymbol{\psi}}(s_n,a_n) - r_n \right)^2 \right].$$

The solution $\widehat{\boldsymbol{\theta}}_{\mathrm{LS}}$ can be analytically computed as

$$\widehat{\boldsymbol{\theta}}_{\mathrm{LS}} = \left(\sum_{n=1}^{N} \widehat{\boldsymbol{\psi}}(s_n,a_n) \widehat{\boldsymbol{\psi}}(s_n,a_n)^\top \right)^{-1} \sum_{n=1}^{N} r_n \widehat{\boldsymbol{\psi}}(s_n,a_n).$$

The value-function approximation method based on the above least-squares formulation is called *least-squares TDQ (LSTDQ)* of the Bellman residual and the policy iteration method based on LSTDQ is called *least-squares policy iteration (LSPI)* [16].

3.2 Least Absolute Policy Iteration (LAPI)

As explained in the introduction, LSPI suffers from excessive sensitivity to outliers and less reliability. Here, we introduce an alternative approach to value function approximation, which we refer to as *least absolute TDQ (LATDQ)*—we propose to employ the absolute loss instead of the squared loss (Figure 1):

$$\widehat{\boldsymbol{\theta}}_{\mathrm{LA}} \equiv \operatorname*{argmin}_{\boldsymbol{\theta}} \left[\sum_{n=1}^{N} \left| \boldsymbol{\theta}^\top \widehat{\boldsymbol{\psi}}(s_n, a_n) - r_n \right| \right]. \tag{3}$$

This minimization problem looks cumbersome due to the absolute value operator which is non-differentiable, but the following mathematical trick mitigates this issue.

Proposition 1 *[5]*

$$|x| = \min_{b} \ b \quad \text{subject to } -b \leq x \leq b. \tag{4}$$

Since Eq.(4) is a linear programming problem, it can be solved efficiently using a standard optimization software. Using this proposition, we can reduce the minimization problem (3) to the following linear program:

$$\begin{cases} \min\limits_{\boldsymbol{\theta}, \{b_n\}_{n=1}^N} & \sum\limits_{n=1}^{N} b_n \\ \text{subject to} & -b_n \leq \boldsymbol{\theta}^\top \widehat{\boldsymbol{\psi}}(s_n, a_n) - r_n \leq b_n, \ \forall n. \end{cases}$$

The number of constraints is N in the above linear program. When N is large, we may employ sophisticated optimization techniques such as *column generation* [7] for efficiently solving the linear programming problem. Alternatively, an approximate solution can be obtained by gradient descent or the (quasi)-Newton methods if the absolute loss is approximated by a smooth loss (see e.g., Section 5.2).

We refer to the policy iteration method based on LATDQ as *least absolute policy iteration (LAPI)*.

3.3 Numerical Examples of LATDQ

For illustration purposes, let us consider the 4-state MDP problem described in Figure 2. The agent is initially located at state $s^{(0)}$ and the actions the agent is allowed to take are moving to the left or right state. If the left-movement action is chosen, the agent always receives small positive reward $+0.1$ at $s^{(L)}$. On the other hand, if the right-movement action is chosen, the agent receives negative reward -1 with probability 0.9999 at $s^{(R1)}$ or it receives very large positive reward $+20000$ with probability 0.0001 at $s^{(R2)}$. The mean and median rewards for left movement are both $+0.1$, while the mean and median rewards for right movement are $+1.0001$ and -1, respectively.

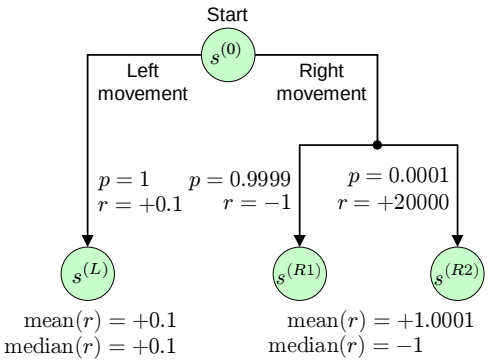


Figure 2: Illustrative MDP problem.

If $Q(s^{(0)}, \text{'Left'})$ and $Q(s^{(0)}, \text{'Right'})$ are approximated by LSTDQ, it returns the mean rewards, i.e., $+0.1$ and $+1.0001$, respectively. Thus, LSTDQ prefers right movement, which is a 'gambling' policy that negative reward -1 is almost always obtained at $s^{(R1)}$, but it is possible to obtain very high reward $+20000$ with a very small probability at $s^{(R2)}$. On the other hand, if $Q(s^{(0)}, \text{'Left'})$ and $Q(s^{(0)}, \text{'Right'})$ are approximated by LATDQ, it returns the median rewards, i.e., $+0.1$ and -1, respectively. Thus LATDQ prefers left movement, which is a stable policy that the agent can always receive small positive reward $+0.1$ at $s^{(L)}$.

If all the rewards in Figure 2 are negated, the value functions are also negated and we obtain different interpretation: LSTDQ is afraid of the risk of receiving very large negative reward -20000 at $s^{(R2)}$ with a very low probability, and consequently it ends up in a very conservative policy that the agent always receives negative reward -0.1 at $s^{(L)}$. On the other hand, LATDQ tries to receive positive reward $+1$ at $s^{(R1)}$ without being afraid of visiting $s^{(R2)}$ too much.

As illustrated above, LATDQ tends to provide qualitatively different solutions from LSTDQ. We argue that the robust and reliable behavior of LATDQ would be more preferable in practical robotics tasks, as discussed in the introduction. In the next section, we experimentally show the usefulness of the proposed method in robot-control tasks.

3.4 Properties of LATDQ

We have illustrated a robust property of LATDQ above. Here, we investigate its properties when the model $\widehat{Q}(s,a)$ is *correctly specified*, i.e., there exists a parameter $\boldsymbol{\theta}^*$ such that

$$\widehat{Q}(s,a) = Q(s,a) \quad \text{for all } s \text{ and } a.$$

Under the correct model assumption, when the number of samples N tends to infinity, the LATDQ solution $\widehat{\boldsymbol{\theta}}$ would satisfy the following equation [14]:

$$\widehat{\boldsymbol{\theta}}^\top \boldsymbol{\psi}(s,a) = \underset{P_{\mathrm{T}}(s'|s,a)}{\mathbb{M}} [R(s,a,s')] \quad \text{for all } s \text{ and } a, \tag{5}$$

where $\mathbb{M}_{P_{\mathrm{T}}(s'|s,a)}$ denotes the conditional median of s' over $P_{\mathrm{T}}(s'|s,a)$ given s and a. $\boldsymbol{\psi}(s,a)$ is defined by

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

where $\mathbb{E}_{P_{\mathrm{T}}(s'|s,a)}$ 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'.

From Eq.(5), we can obtain the following Bellman-like recursive expression:

$$\widehat{Q}(s,a) = \mathop{\mathbb{M}}_{P_{\mathrm{T}}(s'|s,a)} [R(s,a,s')] + \gamma \mathop{\mathbb{E}}_{P_{\mathrm{T}}(s'|s,a)} \mathop{\mathbb{E}}_{\pi(a'|s')} \left[\widehat{Q}(s',a')\right]. \tag{6}$$

Note that in the case of LSTDQ where

$$\widehat{\boldsymbol{\theta}}^{\top} \boldsymbol{\psi}(s,a) = \mathop{\mathbb{E}}_{P_{\mathrm{T}}(s'|s,a)} [R(s,a,s')]$$

is satisfied in the limit under the correct model assumption, we have

$$\widehat{Q}(s,a) = \mathop{\mathbb{E}}_{P_{\mathrm{T}}(s'|s,a)} [R(s,a,s')] + \gamma \mathop{\mathbb{E}}_{P_{\mathrm{T}}(s'|s,a)} \mathop{\mathbb{E}}_{\pi(a'|s')} \left[\widehat{Q}(s',a')\right]. \tag{7}$$

This is the ordinary *Bellman equation* [28]. Thus, Eq.(6) could be regarded as an extension of the Bellman equation to the absolute loss.

From the ordinary Bellman equation (7), we can recover Eq.(1), the original definition of the state-value function $Q(s,a)$ [28]. In contrast, from the absolute-loss Bellman equation (6), we have

$$Q'(s,a) \equiv \mathop{\mathbb{E}}_{\pi,P_{\mathrm{T}}} \left[\sum_{n=1}^{\infty} \gamma^{n-1} \mathop{\mathbb{M}}_{P_{\mathrm{T}}(s_{n+1}|s_n,a_n)} [R(s_n,a_n,s_{n+1})] \,\middle|\, s_1 = s, a_1 = a \right],$$

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 $P_{\mathrm{T}}(s_{n+1}|s_n,a_n)$. This is the value function LATDQ is trying to approximate, which is different from the ordinary value function. Since the discounted sum of *median* rewards—not the expected rewards—is maximized, LATDQ would be less sensitive to outliers than LSTDQ.

4 Experimental Evaluation

In this section, we apply LAPI to simulated robot-control problems and evaluate its practical performance.

Here, we use an *acrobot* illustrated in Figure 3. The acrobot is an under-actuated system and consists of two links, two joints, and an end effector. The length of each link is 0.3 $[m]$, and the diameter of each joint is 0.15 $[m]$. The diameter of the end effector is 0.10 $[m]$ and the height of the horizontal bar is 1.2 $[m]$. The first joint connects the first link to the horizontal bar and is *not* controllable. The second joint connects the first

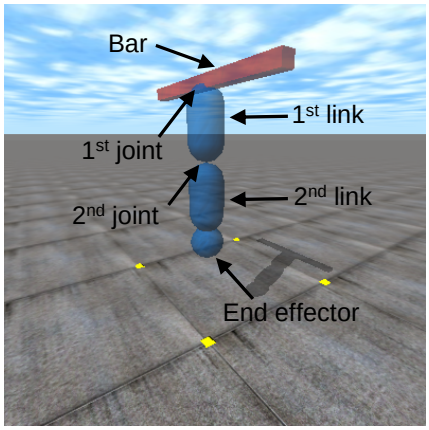


Figure 3: Illustration of the acrobot. The goal is to swing up the end effector by only controlling the second joint.

link to the second link and is controllable. The end effector is attached to the tip of the second link. The control command (action) we can choose is applying positive torque $+50\ [N \cdot m]$, no torque $0\ [N \cdot m]$, or negative torque $-50\ [N \cdot m]$ to the second joint. Note that the acrobot moves only within a plane orthogonal to the horizontal bar.

The goal is to acquire a control policy such that the end effector is swung up as high as possible. The state space consists of the angle $\theta_i\ [rad]$ and angular velocity $\dot{\theta}_i\ [rad/s]$ of the first and second joints $(i = 1, 2)$. The immediate reward is given according to the height h of the center of the end effector as follows:

$$R(s, a, s') = \begin{cases} 10 & \text{if } h > 1.75, \\ \exp\left(-\frac{(h-1.85)^2}{2(0.2)^2}\right) & \text{if } 1.5 < h \leq 1.75, \\ 0.001 & \text{otherwise.} \end{cases}$$

Note that $0.55 \leq h \leq 1.85$ in the current setting.

Here, we suppose that the length of the links is unknown; thus the height h cannot be directly computed from state information. The height of the end effector is supposed to be estimated from an image taken by a camera—the end effector is detected in the image and then its vertical coordinate is computed. Due to recognition error, the estimated height is highly noisy and could contain outliers.

In each policy iteration step, 20 episodic training samples of length 150 are gathered. The performance of the obtained policy is evaluated using 50 episodic test samples of length 300. Note that the test samples are not used for learning policies; they are used only for the purpose of evaluating learned policies. The policies are updated in a *soft-max* manner:

$$\pi(a|s) \longleftarrow \frac{\exp(Q(s, a)/\eta)}{\sum_{a' \in \mathcal{A}} \exp(Q(s, a')/\eta)}, \tag{8}$$

where $\eta = 10 - t + 1$ with t being the iteration number. The discounted factor is set to $\gamma = 1$, i.e., no discount. As basis functions for value function approximation, we use the

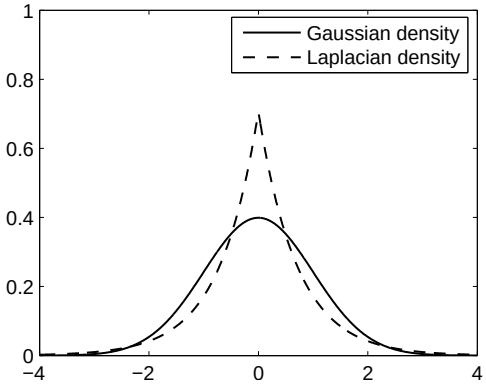


Figure 4: Probability density functions of Gaussian and Laplacian distributions.

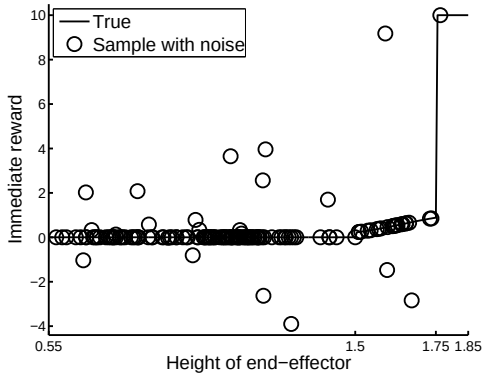


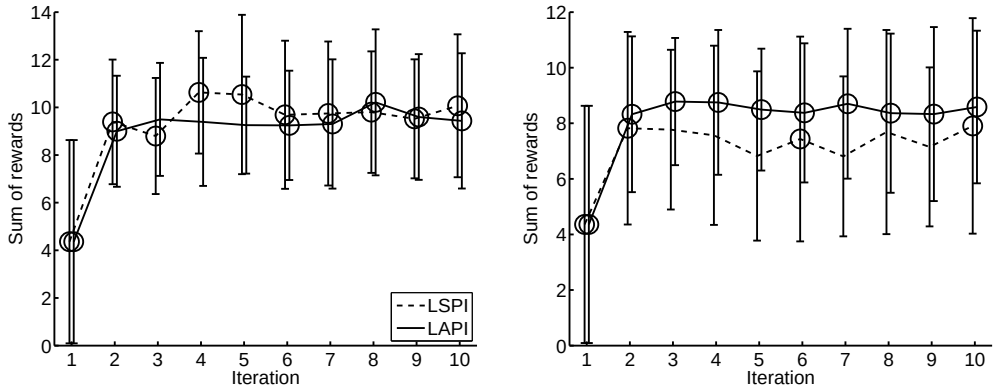
Figure 5: An example of training samples with Laplace noise. The horizontal axis is the height of the end effector. The solid line denotes the noiseless immediate reward and the '$\circ$' denotes a noisy training sample.

Gaussian kernel with standard deviation π—the Gaussian centers are located at

$$(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2) \in \{-\pi, -\tfrac{\pi}{2}, 0, \tfrac{\pi}{2}, \pi\} \times \{-\pi, 0, \pi\} \times \{-\pi, 0, \pi\} \times \{-\pi, 0, \pi\}.$$

The above 135 ($= 5 \times 3 \times 3 \times 3$) Gaussian kernels are defined for each of the three actions; thus 405 ($= 135 \times 3$) kernels are used in total.

We consider two noise environments; one is the case where no noise is added to the rewards and the other case is where Laplacian noise with mean zero and standard deviation 2 is added to the rewards with probability 0.1. Note that the tail of the Laplacian density is heavier than that of the Gaussian density (see Figure 4), implying that a small number of outliers tend to be included in the Laplacian noise environment. An example of the noisy training samples is shown in Figure 5. For each noise environment, the experiment is repeated 50 times with different random seeds and the average of the sum of rewards obtained by LAPI and LSPI are summarized in Figure 6. The best method in terms of the mean value and comparable methods according to the *t-test* [11] at the significance level 5% are specified by '$\circ$'.



(a) No noise (b) Laplacian noise

Figure 6: Average and standard deviation of the sum of rewards over 50 runs for the acrobot swinging-up simulation. The best method in terms of the mean value and comparable methods according to the *t-test* at the significance level 5% are specified by '$\circ$'.

In the noise-less case (see Figure 6(a)), both LAPI and LSPI improve the performance over iterations in a comparable way. On the other hand, in the noisy case (see Figure 6(b)), the performance of LSPI is not improved much due to outliers, while LAPI still produces a good control policy.

Figures 7 and 8 depict motion examples of the acrobot learned by LAPI and LSPI in the Laplacian-noise environment. A demo movie is available from

'`http://sugiyama-www.cs.titech.ac.jp/~sugi/2010/LAPIvsLSPI.mp4`'.

When LSPI is used (Figure 7), the second joint is swung hard in order to lift the end effector. However, the end effector tends to stay below the horizontal bar, and therefore only a small amount of rewards can be obtained by LSPI. This would be due to the existence of outliers. On the other hand, when LAPI is used (Figure 8), the end effector goes beyond the bar, and therefore a large amount of rewards can be obtained even in the presence of outliers.

5 Possible Extension and Variation

In this section, we show possible extension and variation of the proposed approach.

5.1 Regularization

When the number of training samples is small, approximated functions tend to overfit. To alleviate this problem, it is common to *regularize* the solution, i.e., imposing a penalty on the solution when it is too 'large' [12, 32, 8]. Here we introduce regularized variants of LSTDQ and LATDQ which do not sacrifice computational efficiency.

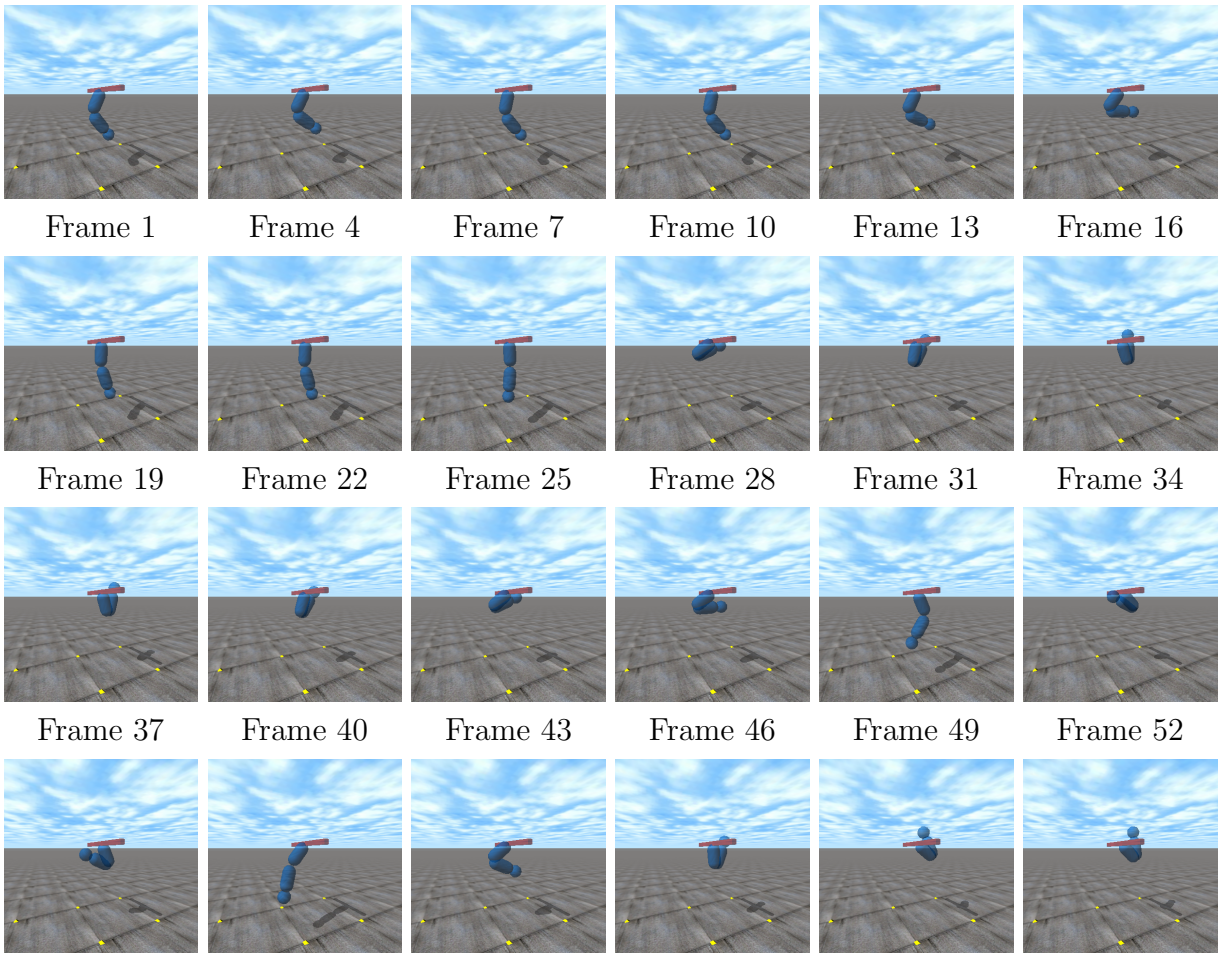


Figure 7: A motion example of the acrobot learned by LSPI in the Laplacian-noise environment. The frame rate is 50 $[ms/frame]$.

Given that the LSTDQ objective function is in a quadratic form, it is convenient to use the squared penalty term as follows [12, 22]:

$$\widehat{\boldsymbol{\theta}}_{\mathrm{LS}} \equiv \operatorname*{argmin}_{\boldsymbol{\theta}} \left[\frac{1}{2} \sum_{n=1}^{N} \left(\boldsymbol{\theta}^\top \widehat{\boldsymbol{\psi}}(s_n, a_n) - r_n \right)^2 + \frac{\kappa}{2} \sum_{b=1}^{B} \theta_b^2 \right],$$

where κ (≥ 0) the regularization parameter that controls the strength of regularization. The above solution is still given analytically as

$$\widehat{\boldsymbol{\theta}}_{\mathrm{LS}} = \left(\sum_{n=1}^{N} \widehat{\boldsymbol{\psi}}(s_n, a_n) \widehat{\boldsymbol{\psi}}(s_n, a_n)^\top + \kappa \boldsymbol{I} \right)^{-1} \sum_{n=1}^{N} r_n \widehat{\boldsymbol{\psi}}(s_n, a_n),$$

where $\boldsymbol{I}$ is the identity matrix.

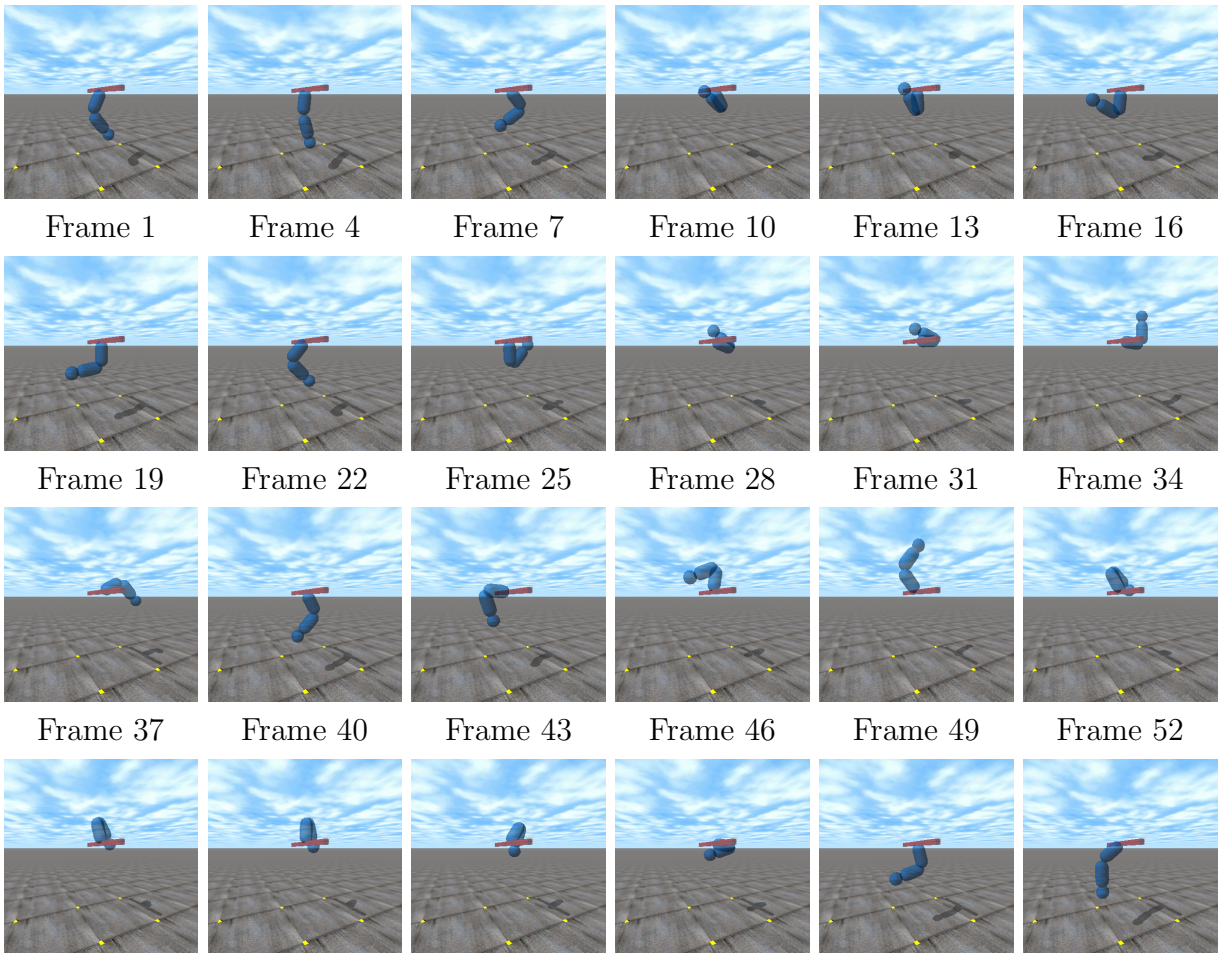


Figure 8: A motion example of the acrobot learned by LAPI in the Laplacian-noise environment. The frame rate is 50 $[ms/frame]$.

Since LATDQ is formulated as a linear programming problem, it is convenient to use the absolute penalty as follows [33, 30, 6]:

$$\widehat{\boldsymbol{\theta}}_{\mathrm{LA}} \equiv \operatorname*{argmin}_{\boldsymbol{\theta}} \left[\sum_{n=1}^{N} \left| \boldsymbol{\theta}^\top \widehat{\boldsymbol{\psi}}(s_n, a_n) - r_n \right| + \kappa \sum_{b=1}^{B} |\theta_b| \right].$$

The solution $\widehat{\boldsymbol{\theta}}_{\mathrm{LA}}$ can be obtained by solving the following optimization problem, which is still a linear program:

$$\begin{cases} \min\limits_{\boldsymbol{\theta}, \{b_n\}_{n=1}^N, \{c_b\}_{b=1}^B} & \sum\limits_{n=1}^{N} b_n + \kappa \sum\limits_{b=1}^{B} c_b \\ \text{subject to} & -b_n \leq \boldsymbol{\theta}^\top \widehat{\boldsymbol{\psi}}(s_n, a_n) - r_n \leq b_n, \ \forall n. \\ & -c_b \leq \theta_b \leq c_b, \ \forall b. \end{cases}$$

An additional advantage of using the absolute penalty is that the solution tends to be *sparse*, i.e., most of $\{\theta_b\}_{b=1}^B$ become zero. This highly contributes to reducing the computation time in the test phase.

5.2 Huber Loss

Minimizing the TD error under the *Huber loss* corresponds to making a compromise between the squared and absolute loss functions [13]:

$$\widehat{\boldsymbol{\theta}}_{\mathrm{HB}} \equiv \operatorname*{argmin}_{\boldsymbol{\theta}} \left[\sum_{n=1}^{N} \rho_t^{\mathrm{HB}} \left(\boldsymbol{\theta}^\top \widehat{\boldsymbol{\psi}}(s_n, a_n) - r_n \right) \right],$$

where t (≥ 0) is a threshold parameter and ρ_t^{HB} is the Huber loss defined as follows (see Figure 9):

$$\rho_t^{\mathrm{HB}}(x) \equiv \begin{cases} \frac{1}{2}x^2 & \text{if } |x| \leq t, \\ t|x| - \frac{1}{2}t^2 & \text{if } |x| > t. \end{cases}$$

The Huber loss converges to the absolute loss as t tends to zero, and it converges to the squared loss as t tends to infinity.

The Huber loss function is rather intricate, but the solution $\widehat{\boldsymbol{\theta}}_{\mathrm{HB}}$ can be obtained by solving the following convex quadratic program [17]:

$$\begin{cases} \min\limits_{\boldsymbol{\theta}, \{b_n, c_n\}_{n=1}^N} & \dfrac{1}{2}\displaystyle\sum_{n=1}^{N} b_n^2 + t\sum_{n=1}^{N} c_n \\ \text{subject to} & -c_n \leq \boldsymbol{\theta}^\top \widehat{\boldsymbol{\psi}}(s_n, a_n) - r_n - b_n \leq c_n, \ \forall n. \end{cases}$$

Another way to obtain the solution $\widehat{\boldsymbol{\theta}}_{\mathrm{HB}}$ is to use a *gradient descent method*—the parameter $\boldsymbol{\theta}$ is updated as follows until convergence:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \varepsilon \sum_{n=1}^{N} \Delta\rho_t^{\mathrm{HB}}(\boldsymbol{\theta}^\top \widehat{\boldsymbol{\psi}}(s_n, a_n) - r_n)\widehat{\boldsymbol{\psi}}(s_n, a_n),$$

where ε (> 0) is a learning-rate parameter and $\Delta\rho_t^{\mathrm{HB}}$ is the derivative of ρ_t^{HB} given by

$$\Delta\rho_t^{\mathrm{HB}}(x) = \begin{cases} x & \text{if } |x| \leq t, \\ t & \text{if } x > t, \\ -t & \text{if } x < -t. \end{cases}$$

In practice, the following *stochastic gradient method* [1] would be more convenient—for a randomly chosen index n ($1 \leq n \leq N$) in each iteration, repeat the following update until convergence:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \varepsilon\Delta\rho_t^{\mathrm{HB}}(\boldsymbol{\theta}^\top \widehat{\boldsymbol{\psi}}(s_n, a_n) - r_n)\widehat{\boldsymbol{\psi}}(s_n, a_n).$$

The plain/stochastic gradient methods also come in handy when approximating the LATDQ solution—the Huber loss function with small t could be regarded as a smooth approximation to the absolute loss.

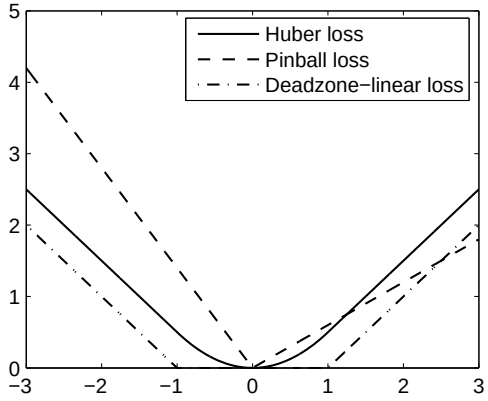


Figure 9: The Huber loss function (with $t = 1$), the pinball loss function (with $\tau = 0.3$), and the deadzone-linear loss function (with $\epsilon = 1$).

5.3 Pinball Loss

We have seen that the absolute loss induces the median, which corresponds to the 50-percentile point. A similar discussion is also possible for an arbitrary percentile 100τ ($0 \leq \tau \leq 1$) based on the *pinball* loss [14]:

$$\widehat{\boldsymbol{\theta}}_{\mathrm{PB}} \equiv \operatorname*{argmin}_{\boldsymbol{\theta}} \left[\sum_{n=1}^{N} \rho_{\tau}^{\mathrm{PB}}(\boldsymbol{\theta}^{\top} \widehat{\boldsymbol{\psi}}(s_n, a_n) - r_n) \right],$$

where $\rho_{\tau}^{\mathrm{PB}}(x)$ is the pinball loss defined by

$$\rho_{\tau}^{\mathrm{PB}}(x) \equiv \begin{cases} 2\tau x & \text{if } x \geq 0, \\ 2(\tau - 1)x & \text{if } x < 0. \end{cases}$$

The profile of the pinball loss is depicted in Figure 9. When $\tau = 0.5$, the pinball loss is reduced to the absolute loss.

The solution $\widehat{\boldsymbol{\theta}}_{\mathrm{PB}}$ can be obtained by solving the following linear program:

$$\begin{cases} \min\limits_{\boldsymbol{\theta}, \{b_n\}_{n=1}^{N}} & \sum\limits_{n=1}^{N} b_n \\ \text{subject to} & \dfrac{b_n}{2(\tau - 1)} \leq \boldsymbol{\theta}^{\top} \widehat{\boldsymbol{\psi}}(s_n, a_n) - r_n \leq \dfrac{b_n}{2\tau}, \ \forall n. \end{cases}$$

5.4 Deadzone-linear Loss

Another variant of the absolute loss is the *deadzone-linear loss* (see Figure 9):

$$\widehat{\boldsymbol{\theta}}_{\mathrm{DL}} \equiv \operatorname*{argmin}_{\boldsymbol{\theta}} \left[\sum_{n=1}^{N} \rho_{\epsilon}^{\mathrm{DL}}(\boldsymbol{\theta}^{\top} \widehat{\boldsymbol{\psi}}(s_n, a_n) - r_n) \right],$$

where $\rho_\epsilon^{\mathrm{DL}}(x)$ is the deadzone-linear loss defined by

$$\rho_\epsilon^{\mathrm{DL}}(x) \equiv \begin{cases} 0 & \text{if } |x| \leq \epsilon, \\ |x| - \epsilon & \text{if } |x| > \epsilon. \end{cases}$$

That is, if the magnitude of the TD error is less than ϵ, no error is assessed. This loss is also called the *ϵ-insensitive loss* and used in *support vector regression* [31].

When $\epsilon = 0$, the deadzone-linear loss is reduced to the absolute loss. Thus the deadzone-linear loss and the absolute loss are highly related to each other. However, the effect of the deadzone-linear loss is completely opposite to the absolute loss when $\epsilon > 0$—the influence of 'good' samples (with small TD-error) are deemphasized in the deadzone-linear loss, while the absolute loss tends to suppress the influence of 'bad' samples (with large TD-error) compared with the squared loss.

The solution $\widehat{\boldsymbol{\theta}}_{\mathrm{DL}}$ can be obtained by solving the following linear program [5]:

$$\begin{cases} \min\limits_{\boldsymbol{\theta}, \{b_n\}_{n=1}^N} & \sum\limits_{n=1}^N b_n \\ \text{subject to} & -b_n - \epsilon \leq \boldsymbol{\theta}^\top \widehat{\boldsymbol{\psi}}(s_n, a_n) - r_n \leq b_n + \epsilon, \\ & b_n \geq 0, \ \forall n. \end{cases}$$

5.5 Chebyshev Approximation

The *Chebyshev approximation* minimizes the error for the 'worst' sample:

$$\widehat{\boldsymbol{\theta}}_{\mathrm{CS}} \equiv \operatorname*{argmin}_{\boldsymbol{\theta}} \left[\max_n |\boldsymbol{\theta}^\top \widehat{\boldsymbol{\psi}}(s_n, a_n) - r_n| \right].$$

This is also called the *minimax approximation*.

The solution $\widehat{\boldsymbol{\theta}}_{\mathrm{CS}}$ can be obtained by solving the following linear program [5]:

$$\begin{cases} \min\limits_{\boldsymbol{\theta}, b} & b \\ \text{subject to} & -b \leq \boldsymbol{\theta}^\top \widehat{\boldsymbol{\psi}}(s_n, a_n) - r_n \leq b, \ \forall n. \end{cases}$$

5.6 Conditional Value-at-Risk

In the area of finance, the *conditional value-at-risk (CVaR)* is a popular risk measure [25]. The CVaR corresponds to the mean of the error for a set of 'bad' samples (see Figure 10).

More specifically, let us consider the distribution of the absolute TD-error over all training samples $\{(s_n, a_n, r_n)\}_{n=1}^N$:

$$\Phi(\alpha|\boldsymbol{\theta}) \equiv P\{(s_n, a_n, r_n) \ : \ |\boldsymbol{\theta}^\top \widehat{\boldsymbol{\psi}}(s_n, a_n) - r_n| \leq \alpha\}.$$

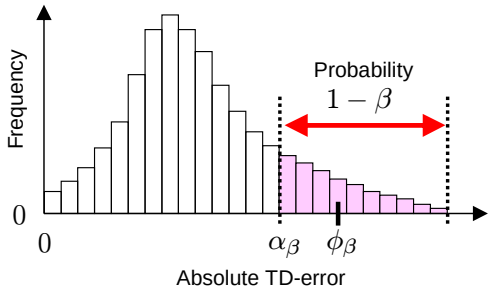


Figure 10: The conditional value-at-risk (CVaR).

For $\beta \in [0, 1)$, let $\alpha_\beta(\boldsymbol{\theta})$ be the 100β-percentile of the absolute TD-error distribution:

$$\alpha_\beta(\boldsymbol{\theta}) \equiv \min\{\alpha \mid \Phi(\alpha|\boldsymbol{\theta}) \geq \beta\}.$$

Thus only the fraction $(1-\beta)$ of the absolute TD-error $|\boldsymbol{\theta}^\top \widehat{\boldsymbol{\psi}}(s_n, a_n) - r_n|$ exceeds the threshold $\alpha_\beta(\boldsymbol{\theta})$. $\alpha_\beta(\boldsymbol{\theta})$ is also referred to as the *value-at-risk* (VaR).

Let us consider the β-tail distribution of the absolute TD-error:

$$\Phi_\beta(\alpha|\boldsymbol{\theta}) \equiv \begin{cases} 0 & \text{if } \alpha < \alpha_\beta(\boldsymbol{\theta}), \\ \dfrac{\Phi(\alpha|\boldsymbol{\theta}) - \beta}{1-\beta} & \text{if } \alpha \geq \alpha_\beta(\boldsymbol{\theta}). \end{cases}$$

Let $\phi_\beta(\boldsymbol{\theta})$ be the mean of the β-tail distribution of the absolute TD-error:

$$\phi_\beta(\boldsymbol{\theta}) \equiv \mathbb{E}_{\Phi_\beta}\left[|\boldsymbol{\theta}^\top \widehat{\boldsymbol{\psi}}(s_n, a_n) - r_n|\right],$$

where $\mathbb{E}_{\Phi_\beta}$ denotes the expectation over the distribution Φ_β. $\phi_\beta(\boldsymbol{\theta})$ is called the CVaR. By definition, the CVaR of the absolute TD-error is reduced to the mean absolute TD-error if $\beta = 0$ and it converges to the worst absolute TD-error as β tends to 1. Thus the CVaR smoothly bridges the proposed least-absolute approach and the Chebyshev approximation method. CVaR is also referred to as the *expected shortfall*.

The CVaR minimization problem in the current context is formulated as

$$\widehat{\boldsymbol{\theta}}_{\mathrm{CV}} \equiv \operatorname*{argmin}_{\boldsymbol{\theta}} \left[\mathbb{E}_{\Phi_\beta}\left[|\boldsymbol{\theta}^\top \widehat{\boldsymbol{\psi}}(s_n, a_n) - r_n|\right]\right].$$

This optimization problem looks complicated, but the solution $\widehat{\boldsymbol{\theta}}_{\mathrm{CV}}$ can be obtained by solving the following linear program [25]:

$$\begin{cases} \min\limits_{\boldsymbol{\theta}, \{b_n\}_{n=1}^N, \{c_n\}_{n=1}^N, \alpha} & N(1-\beta)\alpha + \sum\limits_{n=1}^N c_n \\ \text{subject to} & -b_n \leq \boldsymbol{\theta}^\top \widehat{\boldsymbol{\psi}}(s_n, a_n) - r_n \leq b_n, \\ & c_n \geq b_n - \alpha, \quad c_n \geq 0, \ \forall n. \end{cases}$$

Note that if the definition of the absolute TD-error is slightly changed, the CVaR minimization method amounts to minimizing the deadzone-linear loss [29]:

6 Conclusions and Future Work

In this paper, we proposed using the absolute loss in value function approximation for enhancing robustness and reliability. The change of loss functions resulted in a linear programming formulation which can be solved efficiently by a standard optimization software. We experimentally investigated the usefulness of the proposed method, LAPI, in a simulated robot-control task, and confirmed the advantages of LAPI; the good performance of the existing method, LSPI, is maintained in the noise-less case and higher tolerance to outliers than LSPI is exhibited in the noisy case.

The state-action value function $Q(s, a)$ is defined as the *expectation* of the discounted sum of rewards (see Eq. (1)), which is to be maximized in the standard reinforcement learning framework. On the other hand, one may want to be more risk-sensitive, and maximize other quantities such as the *median* or a *quantile* of the discounted sum of rewards. However, such risk-sensitive reinforcement learning is not straightforward since the Bellman-like simple recursive expression is not available for quantiles of rewards. In the paper [21], it was shown that a Bellman-like recursive equation holds for the *distribution* of the discounted sum of rewards. Based on this preliminary theoretical result, it may be possible to derive a TDQ algorithm that directly optimizes the median or a quantile of the discounted sum of rewards. However, this seems to be an open research issue currently, and would be a promising future direction to pursue.

Another important issue to be further discussed along the current line of research would be the statistical efficiency of the LATDQ estimator. The least-absolute estimator was shown to be more robust against outliers than the least-squares estimator, e.g., under *breakdown point analysis* [13]. However, the least-absolute estimator may be statistically less efficient (i.e., having a larger variance) than the least-squares estimator under the Gaussian noise assumption. Thus loss of efficiency would be the price we have to pay for gaining robustness in LATDQ. We believe that robustness is more important in practice than (asymptotic) efficiency since we seldom have so many data samples that asymptotics matter and the noise distribution may not be Gaussian. Nevertheless, the trade-off between robustness and efficiency in the context of value function approximation or policy iteration would be an important theoretical research issue to be investigated.

Finally, we provided various possibilities for further extending the proposed method in Section 5. Experimentally evaluating these variations is left open as future work.

Acknowledgments

We thank fruitful comments from Akira Ohgawara anonymous reviewers.

References

[1] S. Amari, "Theory of adaptive pattern classifiers," IEEE Transactions on Electronic Computers, vol.EC-16, no.3, pp.299–307, 1967.

[2] C.W. Anderson, P.M. Young, J.N. Buehner, M. R. Knight, H.A. Bush, and D.C. Hittle, "Robust reinforcement learning control using integral quadratic constraints for recurrent neural networks," IEEE Transactions on Neural Networks, vol.18, no.4, pp.993–1002, 2007.

[3] P. Artzner, F. Delbaen, J.M. Eber, and D. Heath, "Coherent measures of risk," Mathematical Finance, vol.9, no.3, pp.203–228, 1999.

[4] P.D. Bertsekas and J. Tsitsiklis, Neuro-Dynamic Programming, Athena Scientific, NH, USA, 1996.

[5] S. Boyd and L. Vandenberghe, Convex Optimization, Cambridge University Press, Cambridge, UK, 2004.

[6] S.S. Chen, D.L. Donoho, and M.A. Saunders, "Atomic decomposition by basis pursuit," SIAM Journal on Scientific Computing, vol.20, no.1, pp.33–61, 1998.

[7] A. Demiriz, K.P. Bennett, and J. Shawe-Taylor, "Linear programming boosting via column generation," Machine Learning, vol.46, no.1/3, p.225, 2002.

[8] T. Evgeniou, M. Pontil, and T. Poggio, "Regularization networks and support vector machines," Advances in Computational Mathematics, vol.13, no.1, pp.1–50, 2000.

[9] H. Hachiya, T. Akiyama, M. Sugiyama, and J. Peters, "Adaptive importance sampling for value function approximation in off-policy reinforcement learning," Neural Networks, vol.22, no.10, pp.1399–1410, 2009.

[10] M. Heger, "Considering of risk in reinforcement learning," Proceedings of the 11th International Conference on Machine Learning, pp.105–111, 1994.

[11] R.E. Henkel, Tests of Significance, SAGE Publication, Beverly Hills, CA, USA., 1979.

[12] A.E. Hoerl and R.W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," Technometrics, vol.12, no.3, pp.55–67, 1970.

[13] P.J. Huber, Robust Statistics, Wiley, New York, 1981.

[14] R. Koenker, Quantile Regression, Cambridge University Press, Cambridge, 2005.

[15] R.M. Kretchmar, P.M. Young, C.W. Anderson, D.C. Hittle, M.L. Anderson, and C.C. Delnero, "Robust reinforcement learning control with static and dynamic stability," International Journal of Robust and Nonlinear Control, vol.11, no.15, pp.1469–1500, 2001.

[16] M.G. Lagoudakis and R. Parr, "Least-squares policy iteration," Journal of Machine Learning Research, vol.4, pp.1107–1149, 2003.

[17] O.L. Mangasarian and D.R. Musicant, "Robust linear and support vector regression," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.22, no.9, pp.950–955, 2000.

[18] H.M. Markovitz, "Portfolio selection," Journal of Finance, vol.7, no.1, pp.77–91, 1952.

[19] O. Mihatsch and R. Neuneier, "Risk sensitive reinforcement learning," Machine Learning, vol.49, no.2-3, pp.267–290, 2002.

[20] J. Morimoto and K. Doya, "Robust reinforcement learning," Neural Computation, vol.17, no.2, pp.335–359, 2005.

[21] H. Nakata and T. Tanaka, "Evaluation of return distributions in Markov decision processes," Proceedings of 2006 Workshop on Information-Based Induction Sciences (IBIS2006), Osaka, Japan, pp.113–117, Oct. 31–Nov. 2 2006. (In Japanese).

[22] T. Poggio and F. Girosi, "Networks for approximation and learning," Proceedings of the IEEE, vol.78, no.9, pp.1481–1497, 1990.

[23] D. Precup, R.S. Sutton, and S. Dasgupta, "Off-policy temporal-difference learning with function approximation," Proceedings of International Conference on Machine Learning, pp.417–424, 2001.

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

[25] R.T. Rockafellar and S. Uryasev, "Conditional value-at-risk for general loss distributions," Journal of Banking & Finance, vol.26, no.7, pp.1443–1472, 2002.

[26] P.J. Rousseeuw and A.M. Leroy, Robust Regression and Outlier Detection, Wiley, New York, 1987.

[27] M. Sato, H. Kimura, and S. Kobayashi, "TD algorithm for the variance of return and mean-variance reinforcement learning," Journal of Japanese Society of Artificial Intelligence, vol.16, no.3, pp.353–362, 2001. (In Japanese).

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

[29] A. Takeda, "Support vector machine based on conditional value-at-risk minimization," Tech. Rep. B-439, Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, Mar. 2007.

[30] R. Tibshirani, "Regression shrinkage and selection via the lasso," Journal of the Royal Statistical Society, Series B, vol.58, no.1, pp.267–288, 1996.

[31] V.N. Vapnik, Statistical Learning Theory, Wiley, New York, NY, USA, 1998.

[32] G. Wahba, Spline Models for Observational Data, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1990.

[33] P.M. Williams, "Bayesian regularization and pruning using a Laplace prior," Neural Computation, vol.7, no.1, pp.117–143, 1995.