Kernel-Based Reinforcement Learning in Robust Markov Decision Processes

Shiau Hong Lim [1] **Arnaud Autef** [2]

Abstract

The robust Markov Decision Process (MDP) framework aims to address the problem of parameter uncertainty due to model mismatch, approximation errors or even adversarial behaviors. It is especially relevant when deploying the learned policies in real-world applications. Scaling up the robust MDP framework to large or continuous state space remains a challenging problem. The use of function approximation in this case is usually inevitable and this can only amplify the problem of model mismatch and parameter uncertainties. It has been previously shown that, in the case of MDPs with state aggregation, the robust policies enjoy a tighter performance bound compared to standard solutions due to its reduced sensitivity to approximation errors. We extend these results to the much larger class of kernel-based approximators and show, both analytically and empirically that the robust policies can significantly outperform the non-robust counterpart.

1. Introduction

We address the problem of learning a robust policy in Markov decision processes with large or continuous state spaces. Finding an exact solution is typically infeasible in such settings and some form of function approximation is needed. We focus on the approach of kernel-based reinforcement learning (Ormoneit & Sen, 2002; Barreto et al., 2016) where the reward and the transition functions are approximated through the use of kernel averaging on a set of training examples. Moreover, we employ a class of linear approximators called averagers (Gordon, 1995; Tsitsiklis & van Roy, 1996) to approximate the value function, which can be seen as an application of the stochastic-factorization trick (Barreto et al., 2016) in the kernel-based approach. This class includes kernel averaging, k-nearest-neighbor, weighted k-nearest neighbor, Bezier patches, linear interpolation etc. In particular, state aggregation is also a special case.

While the motivation of the robust MDP framework (Nilim & El Ghaoui, 2005; Iyengar, 2005; Wiesemann et al., 2013) is to deal with model mismatch or parameter uncertainties in the rewards and transitions, in this work we emphasize its use as a way to protect against the inflexibility or "mismatch" in the value function imposed by the chosen function approximators. It has been shown (Petrik & Subramanian, 2014) that for the class of approximators based on state aggregation, the robust solution enjoys a tighter performance bound compared to standard solutions, with a loss against the optimal policy that scales with $O(\frac{1}{1-\gamma})$ instead of $O(\frac{1}{(1-\gamma)^2})$, where $\gamma \in (0, 1)$ is the discount factor for the infinite-horizon discounted total reward. We extend and derive similar results for the much larger class of kernel-based approximators that includes state aggregation as a special case. Empirically, we demonstrate that the better performance bound does translate into solutions that perform better, especially when there is a model mismatch between the training and the testing environments.

Our contribution is twofold. Theoretically, we extend the existing work on the state-aggregation case (Petrik & Subramanian, 2014) to general kernel averagers. In the state-aggregation case, member states of the same aggregate state share the same action. This no longer holds in the general case and the extension involves the construction of a new robust MDP with an expanded action space that allows the derivation of a performance bound that subsumes the previous bound. Empirically, we present new results on the performance of robust policies in various benchmark tasks in reinforcement learning.

1.1. Related works

Until recently, the robust MDP framework has been analyzed under various finite-state settings (Nilim & El Ghaoui, 2005; Iyengar, 2005; Wiesemann et al., 2013). Tamar et al. (2014) proposed a robust approximate policy iteration algorithm with linear approximation of the value function. Additional assumptions need to be made on the exploration policy to ensure convergence. Roy et al. (2017) extended the

[1]IBM Research, Singapore [2]Applied Mathematics department, Ecole polytechnique, France. Work accomplished while working at IBM Research, Singapore. Correspondence to: Shiau Hong Lim <shonglim@sg.ibm.com>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

results to non-linear approximation and proposed stochastic gradient algorithms that are guaranteed to converge to a local minimum. As in standard results based on projected Bellman equation, the error bound for the final policy scales with $O(\frac{1}{(1-\gamma)^2})$ in both (Tamar et al., 2014) and (Roy et al., 2017).

Performance loss bound of $O(\frac{1}{1-\gamma})$ for state aggregation has been shown by Van Roy (2006), but it depends on the invariant distribution. Similar bound for linear approximation can be achieved by the method of DRADP (Petrik, 2012), but this approach results in NP-hard problems. It is interesting to note that a very different approach by Scherrer & Lesner (2012), which uses non-stationary policies, also results in $O(\frac{1}{1-\gamma})$ loss bound.

2. Problem setup

Let $M = (S, A, p, r, \gamma)$ be a discrete-time MDP with finite state space S and finite action set A. The finite-state assumption is for analytical convenience and our method applies without change to infinite or continuous-state MDPs. Transition probabilities are given by the function p where we define $p(i, a, i') = \Pr(s_{t+1} = i' | s_t = i, a_t = a)$ as the probability of transitioning to state i' when action a is taken in state i. The expected reward when taking a in i is given by $r(i, a)$. All rewards are assumed bounded and we focus on maximizing the total discounted reward where the discount factor is $\gamma \in (0, 1)$.

Given a stationary deterministic policy $\pi : S \to A$, its value at state i, $v^\pi(i)$ is defined as the expected infinite-horizon total discounted reward, when π is followed starting from state i. The value function (as a vector) satisfies:

$$v^\pi = r_\pi + \gamma P_\pi v^\pi = \sum_{t=0}^{\infty} (\gamma P_\pi)^t r_\pi,$$

where $r_\pi(i) = r(i, \pi(i))$ and the i-th row of the matrix P_π, denoted $P_\pi(i, \cdot)$ is the next-state distribution where $P_\pi(i, i') = p(i, \pi(i), i')$. It is a standard result of MDP (Puterman, 1994) that an optimal stationary deterministic policy π^* exists (regardless of starting state), and its value is given by

$$v^* := v^{\pi^*} = r_{\pi^*} + \gamma P_{\pi^*} v^{\pi^*} = \max_{\pi \in A^{|S|}} r_\pi + \gamma P_\pi v^{\pi^*}.$$

where the max function is assumed to be element-wise (i.e. state-wise). Let $\mathcal{T}$ be the Bellman operator on any value function v defined by

$$\mathcal{T} v := \max_{\pi \in A^{|S|}} r_\pi + \gamma P_\pi v. \tag{1}$$

Due to the standard result that $\mathcal{T}$ is a γ-contraction in the max-norm, one can find v^* by starting with an arbitraty v and repeatedly applying $\mathcal{T}$ until convergence.

In the robust MDP framework, instead of a fixed reward and transition function r and p, one defines an uncertainty set $\mathcal{U}_{i,a} \subseteq \mathcal{R}_{i,a} \times \mathcal{P}_{i,a}$ for each (i, a) where $\mathcal{R}_{i,a}$ is a set of possible expected rewards and $\mathcal{P}_{i,a}$ is a set of possible next-state distributions. We restrict ourselves to the SA-rectangular setting (Wiesemann et al., 2013) where for each policy π and each time step t, the "nature" is free to choose a pair $(r_t, P_t) \in \mathcal{U}_\pi$ such that $(r_t(i), P_t(i, \cdot)) \in \mathcal{U}_{i,\pi(i)}$ for all i. The immediate reward and next-state after taking action at time t will be drawn from r_t and P_t. We assume that all uncertainty sets are compact. The standard MDP is a robust MDP where $\mathcal{R}_{i,a}$ and $\mathcal{P}_{i,a}$ are singletons for all (i, a) and therefore $\mathcal{U}_\pi$ is a singleton $\{(r_\pi, P_\pi)\}$.

For robust MDPs, we define the robust value of a policy π as follows:

$$\tilde{v}^\pi := \min_{(r_t, P_t) \in \mathcal{U}_\pi} \sum_{t=0}^{\infty} (\gamma P_t)^t r_t.$$

Note that we use $\tilde{v}$ for the robust value and v for the standard value. It is a standard result of robust MDP (Nilim & El Ghaoui, 2005; Iyengar, 2005) that given a stationary deterministic policy π, there exists a particular choice of $(r^*_\pi, P^*_\pi) \in \mathcal{U}_\pi$ such that $\tilde{v}^\pi$ can be minimized by setting $(r_t, P_t) = (r^*_\pi, P^*_\pi)$ for all t. We define the robust Bellman operator $\widetilde{\mathcal{T}}$ as

$$\widetilde{\mathcal{T}} \tilde{v} := \max_{\pi \in A^{|S|}} \min_{(r,P) \in \mathcal{U}_\pi} r + \gamma P \tilde{v}.$$

Again, $\widetilde{\mathcal{T}}$ can be shown to be a γ-contraction and the optimal robust value satisfies $\tilde{v}^* = \widetilde{\mathcal{T}} \tilde{v}^*$. The optimal robust policy $\tilde{\pi}^*$ is then the greedy policy with respect to $\tilde{v}^*$ where

$$\tilde{\pi}^* := \arg \max_{\pi \in A^{|S|}} \min_{(r,P) \in \mathcal{U}_\pi} r + \gamma P \tilde{v}^*.$$

2.1. Kernel Averager

In this work we use kernel averagers for two related but different purposes. First, we use kernel averagers to approximate the reward and/or the transition functions based on a training set of example state-action outcomes. This is needed when the true reward and transition model is not available. Secondly, we use kernel averagers to approximate the value function. This addresses the practical issue of dealing with very large or continuous state spaces.

2.1.1. Approximating the reward and transition function

For a particular action a, when the true reward and transition function is not available, we approximate them based on a training set $D^a = \{(i^a_1, i'^a_1, r^a_1) \dots (i^a_{m_a}, i'^a_{m_a}, r^a_{m_a})\}$ as follows:

$$\hat{r}(i, a) := \frac{\sum_{k=1}^{m_a} \psi(i, i^a_k) r^a_k}{\sum_{k=1}^{m_a} \psi(i, i^a_k)}$$

and

$$\hat{p}(i,a,i') := \frac{\sum_{k=1}^{m_a} \psi(i,i_k^a)\mathbb{I}(i_k'^a = i')}{\sum_{k=1}^{m_a} \psi(i,i_k^a)}$$

where $\psi(i,i') \geq 0$ for all $i,i' \in S$ is a kernel function, and $\mathbb{I}$ is the indicator function. We define the normalized version

$$\hat{\psi}(i,i_k^a) = \frac{\psi(i,i_k^a)}{\sum_{k=1}^{m_a} \psi(i,i_k^a)}$$

and let $\hat{\psi}_a(i)$ be a vector whose k-th entry equals $\hat{\psi}(i,i_k^a)$. Let r^a be a vector whose k-th entry equals r_k^a. Then we have $\hat{r}(i,a) = \langle \hat{\psi}_a(i), r^a \rangle$.

One can solve the approximate MDP $(S,A,\hat{p},\hat{r},\gamma)$ using the approximate version of (1):

$$\hat{\mathcal{T}}v := \max_{\pi \in A^{|S|}} \hat{r}_\pi + \gamma \hat{P}_\pi v. \tag{2}$$

Let $\hat{S}^a = \{i_1'^a \dots i_{m_a}'^a\}$ and let $\hat{S} = \cup_{a\in A}\hat{S}^a$. Note that solving the above only requires keeping track of $v(i)$ for $i \in \hat{S}$ since $\hat{p}(\cdot,\cdot,i)$ will be zero for any $i \notin \hat{S}$.

2.1.2. Approximating the value function

When S is large, (1) is no longer feasible. Similarly, (2) can be too expensive to solve when the training set is large. We assume a linear approximation architecture where v is approximated by Φw where Φ is a $|S| \times m$ feature matrix, with $m \leq |S|$ (typically $m \ll |S|$) and w is an m-dimensional vector. The i-th row $\Phi(i,\cdot)$ corresponds to a feature vector for state i. The "standard" solution is to seek w that satisfies

$$\Phi w = \Pi_\Phi \mathcal{T} \Phi w \tag{3}$$

where Π_Φ is a projection onto the range of Φ. This can be done, for instance, with the method of least-squares. A greedy policy π_w with respect to w can then be derived via

$$\pi_w := \arg\max_\pi r_\pi + \gamma P_\pi \Phi w \tag{4}$$

when the model is available or

$$\hat{\pi}_w := \arg\max_\pi \hat{r}_\pi + \gamma \hat{P}_\pi \Phi w \tag{5}$$

when a training set is used.

We focus on the special case where Φ is a kernel averager. A feature matrix is a kernel averager if

$$\forall i,j,\ 0 \leq \Phi_{i,j} \leq 1 \quad \text{and} \quad \forall i,\ \sum_j \Phi_{i,j} = 1.$$

We have a simpler solution for w when Φ is a kernel averager. In this case, one can consider a subset $\tilde{S} \subset S$ of m representative states and w is the value function for these representative states. Consider a new finite MDP $(\tilde{S},A,p',r')$ with state space $\tilde{S} = \{1 \dots m\}$ where each $j \in \tilde{S}$ is associated with a corresponding state $i_j \in S$. Let the expected reward $r'(j,a) := r(i_j,a)$ and transition function

$$p'(j,a,j') := \sum_{i' \in S} p(i_j,a,i')\Phi_{i',j'}.$$

Note that each row in Φ is used as a next-state distribution. The Bellman operator

$$\mathcal{T}'w := \max_\pi r'_\pi + \gamma P'_\pi w \tag{6}$$

is a γ-contraction in this new m-state MDP and therefore $w = \mathcal{T}'w$ can be solved via repeated application of $\mathcal{T}'$. A greedy policy in the original MDP can then be derived via (4). Similarly, one can define $\hat{p}'$ and $\hat{r}'$ using the training set, solve the corresponding Bellman equation and obtain a greedy policy using (5).

3. Robust Solution

We can derive a performance bound for the policy derived via (4). In the case where Φ is a kernel averager with the additional condition that $\Phi_{i_j,j} = 1$ for all $j = 1 \dots m$, let w^* be the fixed-point of (6). It has been shown (Gordon, 1995) that

$$\|v^{\pi_{w^*}} - v^*\|_\infty \leq \frac{4\gamma\epsilon}{(1-\gamma)^2}$$

where $\epsilon = \min_w \|v^* - \Phi w\|_\infty$. This result is tight in the sense that for any $\gamma \in (0,1)$, there exists MDP and feature Φ such that the equality is achieved (Tsitsiklis & van Roy, 1996). Similar results, under slightly different condition for Φ are also available (Van Roy, 2006; Tsitsiklis & Van Roy, 1997). Since γ is typically close to 1, the performance bound can be very weak considering that the maximum value for any state scales with $\frac{1}{1-\gamma}$. For the case of state aggregation (i.e. Φ is binary), (Petrik & Subramanian, 2014) have shown that a robust solution can enjoy a better bound of $\frac{2\epsilon}{1-\gamma}$.

The robust solution by (Petrik & Subramanian, 2014) is obtained by constructing a corresponding robust MDP based on the original MDP and solving for the optimal robust policy. The construction relies on the fact that Φ is based on state aggregation. For general kernel averagers, we need a new construction, which we describe in this section.

Our robust MDP $\tilde{M}$ assumes that Φ is a kernel averager with m "abstract" states, each represented by a kernel. The state space of $\tilde{M}$ is therefore $\tilde{S} = \{1 \dots m\}$. We say that a state $i \in S$ is a member of abstract state $j \in \tilde{S}$ if $\Phi_{i,j} > 0$. We define $M(j) = \{i \in S : \Phi_{i,j} > 0\}$ as the set of member states of j. One can interpret $M(j)$ as the set of states that share a certain feature j.

The action space in $\tilde{M}$, instead of A, is expanded to $\tilde{A} = A^{|S|}$. This means that each "action" in $\tilde{M}$ corresponds to a

complete "policy" in the original MDP M. The execution of an "action" π in $\tilde{M}$ proceeds as follows:

1. Let $j \in \tilde{S}$ be the current state.
2. The agent chooses an action $\pi \in \tilde{A}$.
3. "Nature" chooses a member state $i \in M(j)$.
4. Action $\pi(i)$ is executed in i with reward $r(i, \pi(i))$ and next-state distribution $p(i, \pi(i))$ chosen according to M.
5. From the next state i', the next abstract state is chosen according to $\Phi_{i'}$.

Note that in the last step, the agent transitions to the next abstract state by treating each row of Φ as a distribution – which is valid since Φ is a kernel averager.

Given any MDP M and a kernel matrix Φ, we now have a corresponding robust MDP $\tilde{M}$. Similarly, any policy π in M has a corresponding policy in $\tilde{M}$, where the same "action" π is assigned to every abstract state $j \in \tilde{S}$. In the other direction, let $w \in \mathbb{R}^m$ be the (robust) value function of some policy in $\tilde{M}$. We can then derive a greedy policy π_w in M via (4). Let v^{π_w} be the (true) value of policy π_w in M, we show that the robust value Φw is always a lower bound to v^{π_w}:

Lemma 1. *Let w be the value of robust policy $\tilde{\pi}$ in $\tilde{M}$. Let π_w be the greedy policy with respect to w as defined in (4). Then*

$$v^{\pi_w} \geq \Phi w.$$

Proof. For each $i \in S$, we have

$$\begin{aligned}
&(\Phi w)(i) \\
&= \sum_{j \in \tilde{S}} \Phi_{i,j} w(j) \\
&\overset{(*)}{=} \sum_{j \in \tilde{S}} \Phi_{i,j} \left(\min_{i' \in M(j)} r(i', \tilde{\pi}_j(i')) + \gamma \langle p(i', \tilde{\pi}_j(i')), \Phi w \rangle \right) \\
&\overset{(**)}{\leq} \sum_{j \in \tilde{S}} \Phi_{i,j} \left(r(i, \tilde{\pi}_j(i)) + \gamma \langle p(i, \tilde{\pi}_j(i)), \Phi w \rangle \right) \\
&\leq \sum_{j \in \tilde{S}} \Phi_{i,j} \left(\max_a r(i,a) + \gamma \langle p(i,a), \Phi w \rangle \right) \\
&= \max_a r(i,a) + \gamma \langle p(i,a), \Phi w \rangle \\
&= r(i, \pi_w(i)) + \gamma \langle p(i, \pi_w(i)), \Phi w \rangle
\end{aligned}$$

where in (*) we apply the robust Bellman equation for the policy $\tilde{\pi}$ and in (**) either $\Phi_{i,j} = 0$ or $i \in M(j)$.

We therefore have that $\Phi w \leq \mathcal{T}_{\pi_w}(\Phi w)$ where $\mathcal{T}_{\pi_w}$ is the Bellman operator with respect to policy π_w. Theorem 6.2.2 of (Puterman, 1994) states that any v such that $v \leq \mathcal{T}_\pi v$ satisfies $v \leq v^\pi$. It follows that Φw is a lower bound of v^{π_w}. □

Our strategy is now clear. Since the robust value of every policy in $\tilde{M}$ is a lower bound for the corresponding greedy policy in M, we look for the optimal robust policy in $\tilde{M}$. This involves solving $\tilde{M}$ via the robust Bellman operator $\tilde{\mathcal{T}}$. Let $\tilde{\pi}^*$ be the optimal robust policy and w^* its robust value. We then derive a greedy policy π_{w^*}. We are now ready to derive a performance bound for $v^{\pi_{w^*}}$.

Theorem 1. *Let $\epsilon = \min_w \|v^* - \Phi w\|_\infty$ and $w_0 = \arg\min_w \|v^* - \Phi w\|$. Let $\tilde{\pi}^*$ be the optimal robust policy in $\tilde{M}$ and w^* its robust value. Let π_{w^*} be the greedy policy with respect to w^* via (4). Then*

$$\|v^{\pi_{w^*}} - v^*\|_\infty \leq \frac{2\epsilon + L_0}{1 - \gamma}$$

where

$$L_0 = \max_{\{(j,j') \in \tilde{S} | M(j) \cap M(j') \neq \emptyset\}} |w_0(j) - w_0(j')|.$$

Proof. We follow the strategy of (Petrik & Subramanian, 2014). First, let $\bar{\pi}$ be the corresponding robust policy in $\tilde{M}$ whose actions in all $j \in \tilde{S}$ equal the true optimal policy π^* in M. Let $w^{\bar{\pi}}$ be its robust value in $\tilde{M}$. Then,

$$\begin{aligned}
\forall i,\ v^*(i) - v^{\pi_{w^*}}(i) &\leq v^*(i) - (\Phi w^*)(i) \\
&\leq v^*(i) - (\Phi w^{\bar{\pi}})(i)
\end{aligned}$$

where the first inequality is by Lemma 1 and the second inquality is due to the fact that w^* is the value of the optimal robust policy.

Using the linear program definition of the robust value $w^{\bar{\pi}}$ (Marecki et al., 2013),

$$\begin{aligned}
\sum_j w^{\bar{\pi}}(j) = \max_w & \sum_j w(j) \\
\text{s.t.} \quad & \Phi^1 w \leq \Phi^2 r_{\pi^*} + \gamma \Phi^2 P_{\pi^*} \Phi w \qquad (7)
\end{aligned}$$

where Φ^1 and Φ^2 are defined as follows

- Φ^1 is a $(\sum_j |M(j)|) \times m$ matrix such that

$$\forall j, j' \in \tilde{S}, \forall i \in M(j), \Phi^1_{(j,i),j'} = \mathbb{I}(j = j')$$

- Φ^2 is a $(\sum_j |M(j)|) \times |S|$ matrix such that

$$\forall j \in \tilde{S}, \forall i \in M(j), \forall i' \in S, \Phi^1_{(j,i),i'} = \mathbb{I}(i = i')$$

Note that each row of Φ^1 and Φ^2 corresponds to a pair (j, i) of abstract state j and its member i. Any w that satisfies (7) is therefore a lower bound for $w^{\bar{\pi}}$.

Let $w_0 = \arg\min_w \|v^* - \Phi w\|_\infty$. Since

$$\forall j_0, \forall i_0 \in M(j_0), |v^*(i_0) - \sum_j \Phi_{i_0,j} w_0(j)| \leq \epsilon$$
$$|w_0(j_0) - \sum_j \Phi_{i_0,j} w_0(j)| \leq L_0$$

we have that

$$-(\epsilon + L_0)\mathbf{1} \leq \Phi^2 v^* - \Phi^1 w_0 \leq (\epsilon + L_0)\mathbf{1}.$$

Thus, for any δ, we have

$$(\delta - \epsilon - L_0)\mathbf{1} \leq \Phi^2 v^* - \Phi^1(w_0 - \delta\mathbf{1}) \leq (\epsilon + L_0 + \delta)\mathbf{1}. \quad (8)$$

By definition of w_0, we also have

$$-\epsilon\mathbf{1} \leq v^* - \Phi w_0 \leq \epsilon\mathbf{1}$$

and therefore

$$(\delta - \epsilon)\mathbf{1} \leq v^* - \Phi(w_0 - \delta\mathbf{1}) \leq (\epsilon + \delta)\mathbf{1}$$

hence

$$\gamma(\delta - \epsilon)\mathbf{1} \leq \gamma\Phi^2 P_{\pi^*} v^* - \gamma\Phi^2 P_{\pi^*}\Phi(w_0 - \delta\mathbf{1}) \leq \gamma(\epsilon + \delta)\mathbf{1}. \quad (9)$$

Combining (8) and (9) we obtain

$$\begin{aligned}
&\Phi^1(w_0 - \delta\mathbf{1}) - \gamma\Phi^2 P_{\pi^*}\Phi(w_0 - \delta\mathbf{1}) \\
&\leq \Phi^2 v^* + (\epsilon + L_0 - \delta)\mathbf{1} - \gamma\Phi^2 P_{\pi^*} v^* + \gamma(\epsilon + \delta)\mathbf{1} \\
&\leq \Phi^2 r_{\pi^*} + (\epsilon(1 + \gamma) + L_0 - (1 - \gamma)\delta)1
\end{aligned}$$

Consequently, if we take $\delta = \frac{1+\gamma}{1-\gamma}\epsilon + \frac{1}{1-\gamma}L_0$, then $w = w_0 - \delta\mathbf{1}$ is a vector that satisfies (7), and therefore $\Phi(w_0 - \delta\mathbf{1}) \leq \Phi w^{\bar{\pi}}$, hence

$$\forall i,\ v^*(i) - \Phi w^{\bar{\pi}}(i) \leq (\epsilon + \delta) = \frac{2\epsilon + L_0}{1 - \gamma}$$

which completes the proof. □

Theorem 1 involves L_0, which is a smoothness property of the best approximator w_0. In particular, it is the maximum gap in the robust values between any two abstract states that share some member states. In the special case of state aggregation, $L_0 = 0$ and we recover the performance bound in (Petrik & Subramanian, 2014). Note that depending on the kernels, both ϵ and L_0 can be small, which explains why the results on kernels can be significantly better than hard state aggregation. We provide a concrete example in the supplementary material. Empirically, this is illustrated in Section 5.2.1.

4. Algorithms

Note that Theorem 1 is based on solving the optimal robust policy in $\tilde{M}$. For large state space this can be a problem since the robust Bellman operator requires finding a minimum over all member states, for each abstract state. Secondly, since the true model is usually unknown, the reward and transition functions have to be estimated from training examples. We propose two approximate solutions, each adapting the kernel-based reinforcement learning algorithm to the robust MDP setting. Both methods assume that a training set D^a as defined in section 2.1.1 is given for each $a \in A$.

4.1. First Method

Following the construction in Section 3, our starting point is the following robust Bellman operator for each abstract state $j \in \{1 \dots m\}$,

$$(\widetilde{\mathcal{T}} w)(j) := \max_{\pi \in A^{|S|}} \min_{i \in \hat{M}(j)} \hat{r}(i, \pi(i)) + \gamma \langle \hat{p}(i, \pi(i)), \Phi w \rangle$$

where $\hat{r}$ and $\hat{p}$ are defined as in Section 2.1.1. Note that the minimum is taken over all states in $\hat{M}(j)$, which is defined as the set of all states $i \in M(j)$ that are present in the training set. Recall that each abstract state j corresponds to one "feature" and $M(j)$ contains all states that have non-zero value of feature j. Note also that even though the max operator is over the set of all policies, it needs only be computed for each member state $i \in \hat{M}(j)$ separately. The robust Bellman operator is a γ-contraction and therefore the algorithm is guaranteed to converge.

Under the right smoothness condition, kernel-based approximation of r and p are consistent (Ormoneit & Sen, 2002) and therefore in the limit of infinite number of training examples, the result of Theorem 1 applies to this method. This method is summarized in Algorithm 1.

Algorithm 1 Robust kernel-based value iteration

Input: D^a for each $a \in A$, ϵ

1. Initialize $t \leftarrow 0$, $w^0 \leftarrow \mathbf{0}$.
2. For each $j \in \{1 \dots m\}$,
$$w^{t+1}(j) \leftarrow \max_{\pi} \min_{i \in \hat{M}(j)} \hat{r}(i, \pi(i)) + \gamma \langle \hat{p}(i, \pi(i)), \Phi w^t \rangle$$
3. If $\|w^{t+1} - w^t\|_\infty < \epsilon$, stop and output w_{t+1}.
4. Set $t \leftarrow t + 1$. Go to step 2.

4.2. Second Method

When the training set is large, evaluating the min operator in Algorithm 1 can be expensive. It can also result in overly conservative policies especially when m is small. Our second method addresses this by defining a different uncertainty set for each abstract state j. We assume, in this case, that each j corresponds to a representative state $i_j \in S$ in the original MDP. For each a, the kernel-based approximation defines a probability distribution $\hat{\psi}_a(i_j)$ which is used to compute $\hat{r}$ and $\hat{p}$ (Section 2.1.1). The uncertainty set is then defined as a norm-ball around $\hat{\psi}_a(i_j)$. In particular, we use 1-norm ball since the minimum can be efficiently computed. We also define Φ^a as the (truncated) feature matrix where the k-th row corresponds to the feature vector of i'^a_k in D^a. Algorithm 2 shows the second method.

Algorithm 2 Robust kernel-based value iteration, II

Input: D^a for all $a \in A$, ϵ, β

1. Initialize $t \leftarrow 0$, $w^0 \leftarrow 0$.
2. For each $j \in \{1 \dots m\}$,
$$w^{t+1}(j) \leftarrow \max_a \min_{\|p - \hat{\psi}_a(i_j)\|_1 \leq \beta} \langle p, r^a + \gamma \Phi^a w^t \rangle$$
3. If $\|w^{t+1} - w^t\|_\infty < \epsilon$, stop and output w_{t+1}.
4. Set $t \leftarrow t + 1$. Go to step 2.

An additional parameter, β is required as input to describe the size of the norm-ball. Note that the support of the norm-ball should be restricted to only states i^a_k in the training set with non-zero $\hat{\psi}(i_j, i^a_k)$. For 1-norm, the minimum can be computed, for example, with the algorithm in Figure 2 of (Jaksch et al., 2010). The parameter β determines the level of robustness needed, where $\beta = 0$ corresponds to the non-robust version. Note that setting $\beta = 2$ is not equivalent to Algorithm 1. In Algorithm 1, the minimum makes use of $\hat{\psi}_a(i^a_k)$ for each $i^a_k \in M(j)$ whereas in Algorithm 2 only $\hat{\psi}_a(i_j)$ is used.[1]

5. Experiments

In all our experiments, we use the Gaussian kernel for both ψ and ϕ. In particular,

$$\psi(i, i') = \exp\left\{-\frac{1}{2}\left(\frac{\|i - i'\|}{\sigma_\psi}\right)^2\right\}$$

and

$$\phi(i, j) \propto \exp\left\{-\frac{1}{2}\left(\frac{\|i - j\|}{\sigma_\phi}\right)^2\right\}.$$

For computational efficiency, we always restrict ψ and ϕ such that only the 20 nearest neighbors of any given state have non-zero values. For the bandwidth parameters, we employ a wide range during training, from the set $\{\exp(-8), \exp(-7) \dots \exp(3)\}$. This results in 144 pairs of $(\sigma_\psi, \sigma_\phi)$, and we always choose the best-performing pair based on 30 independent test episodes.

We focus mainly on Algorithm 2 since it is the more practical and flexible version of the two. Our value iteration is stopped when $\|w^{t+1} - w^t\| < 0.001$ or after 100 iterations, whichever happens earlier. We use $\gamma = 0.99$ for all our tasks. Error bars in all our plots denote 95% confidence intervals for the mean.

The complete source code for the implementation of our algorithm as well as the task environments are provided in the supplementary material.

5.1. Puddle World

We begin with the toy problem Puddle World as described in (Sutton, 1996). This serves as a sanity check for the effect of the robust approach since the state space is two-dimensional and can be easily visualized. The agent has 4 actions, which correspond to moving in each axis direction, subject to random Gaussian noise. Each move receives a -1 reward. The goal is to reach the upper right corner ($[0.95, 1] \times [0.95, 1]$ box), which is a region of zero-reward absorbing states, while avoiding the two "puddles", since stepping on the puddle gives negative rewards whose magnitude depends on the distance to the middle of the puddle. We use a fixed initial state at $(0.25, 0.6)$.

We follow the strategy of (Barreto et al., 2016) in creating the training set D^a by running a random policy on 10 training episodes, each lasts until either the goal region is reached or 1000 steps. The representative states for ϕ are then created by running K-means on the training states. K here therefore determines the number of representative states. The best kernel parameters are then chosen as described in the beginning of Section 5. This process is repeated five times and the best-performing training set and kernel parameters are chosen and used to generate both the reference non-robust policy as well as the robust versions. The robust policies are generated using Algorithm 2 on a range of robustness parameters β.

For Puddle World, we notice that a near-optimal performance with a total reward of about -36 can be achieved with $K >= 80$. Figure 1 shows the results for policies generated by various β. We tested two cases. In the first case, the test environment is identical to the training envi-

[1] Empirically, setting $\beta = 2$ results in performance almost identical to that of Algorithm 1, which can be overly conservative. We show some additional results in the supplementary material.

ronment. In this case, we notice a slight performance hit with increasing β. In the second case, we enlarge the puddle size by 50% in the test environment. Here we see significant performance hit for the non-robust version ($\beta = 0$) while the robust policies suffer less.

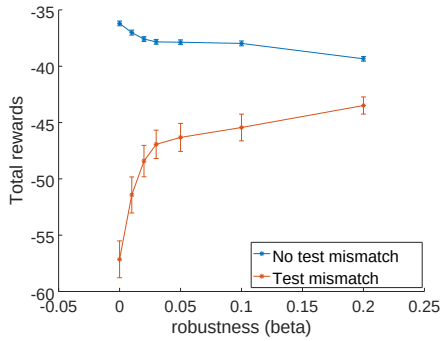

Figure 1. Puddle world performance across various robustness β.

Figures 2 and 3 illustrate the difference between a robust and a non-robust policy. In particular, the robust policy generates trajectories that move further around the puddle toward the goal, giving up some optimality for additional robustness.

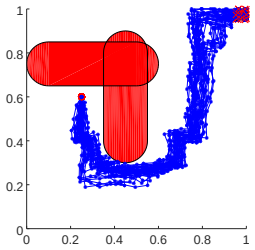

Figure 2. Test trajectories for non-robust policy ($\beta = 0$).

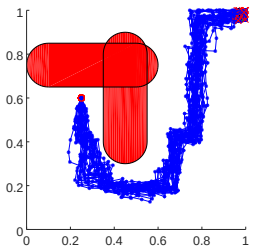

Figure 3. Test trajectories for robust policy ($\beta = 0.2$).

5.2. Acrobot

The acrobot (Sutton & Barto, 1998) is a control task where the objective is to swing the tip of a two-link, underactuated robot above a certain height by only exerting torque on the second joint – simulating a gymnast on a high bar bending at the waist. The state space is 4-dimensional. We simulate the system described by the equations in (Sutton & Barto, 1998) using the 4-th order Runge-Kutta method with a time step of 0.05 seconds and actions chosen every 0.2 seconds. The three actions are 3 torque levels of -1, 0 and 1 respectively. A reward of -1 is received in every step except in a goal state.

To add noise to the system, we introduce a noise parameter η, such that the action 1 would produce an actual torque that is uniformly random between $1 - \eta$ and 1, and likewise for the action -1. The training environment uses $\eta = 0.1$. Several observations can be made via the acrobot task, which we describe in the following subsections.

5.2.1. Kernel-based vs state aggregation

One of the motivation of this work is to generalize the results on state aggregation to the kernel-based setting. We implement state aggregation by simply forcing ϕ to always use only the nearest representative state. Figure 4 shows that indeed, the kernel-based approach performs better across various feature lengths (K). This can be observed in both robust and non-robust solutions.

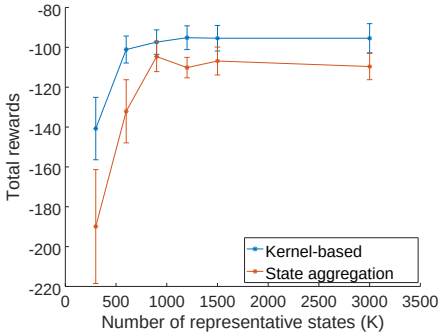

Figure 4. Kernel-based vs state aggregation performance on the acrobot.

5.2.2. Identical train/test environments

In finite-state MDPs without function approximation, it is clear that the optimal robust solution will not outperform the optimal non-robust solution if both the training and test environments are identical. In the case of function approximation, this is not always true since the space of representable value functions and policies can be very different between the robust and non-robust solutions. Figure 5 shows the performance of policies generated with various β, where $\beta = 0$ corresponds to the non-robust solution. We can observe that for certain range of β, the robust policies actually significantly outperform the non-robust policies, even though the training and test environments are identical. Also note that all kernel parameters are optimized for $\beta = 0$ and the same parameters (hence, the same kernel) are used for all other β. This supports our theoretical results where

better worst-case performance bound can be derived when we optimize the robust value, which is a lower bound to the true value.

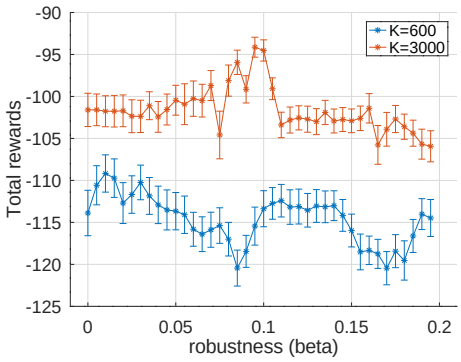

Figure 5. Performance on identical train/test environments with respect to beta.

5.2.3. Mismatched train/test environments

In the case of mismatched training and test environments, the robust solutions have better potential to outperform the non-robust solutions. Figure 6 illustrates this for test environments with $\eta = 0.1, 0.2$ and 0.5 while the training environment uses $\eta = 0.1$. An interesting observation is that for the range of β between 0.1 and 0.15, there is no drop in performance when the test environment has the larger $\eta = 0.2$.

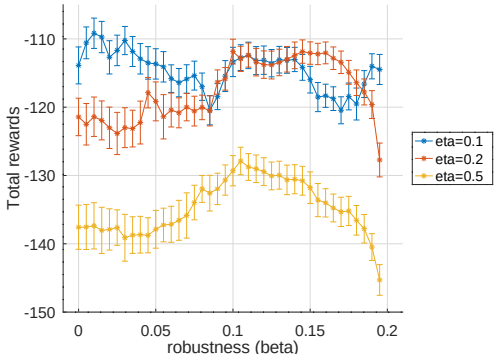

Figure 6. Performance on mismatched train/test environments with respect to beta (K=600).

5.3. Double pole-balancing

We further evaluate our approach on the pole-balancing task. In contrast to the puddle world and the acrobot, which are both goal-directed tasks, the objective in this task is to keep one or more poles hinged to a cart from falling over. The two actions correspond to applying forces 10 and -10 respectively to the cart, which are constrained to stay within a limited track. We focus on the double pole-balancing task, where two side-by-side poles on the cart need to be balanced simultaneously. The state space is 6 dimensional. We follow the exact simulation settings in (Gomez, 2003) and use the 4-th order Runge-Kutta method. A reward of 1 is received in each step. Each episode lasts 3000 steps unless a failed state is entered.

Again, we introduce noise to the actual forces applied by the action by adding a random decrease in magnitude up to τ. The training examples are generated without noise, based on 300 100-step episodes using a random policy. Due to the sensitivity of the performance of the resulting policies to the kernel parameters in this task, we optimize kernel parameters for the robust and non-robust solutions separately, but on the same training sets. The learned policies (with the best kernel parameters) are then tested on the test environments where noise is added.

Figure 7 shows the results for the case with and without noise in the test environment. Figure 8 shows how performance is affected by various noise levels in the test environments. From both figures, we can observe that the robust policies suffer smaller performance drop compared to the non-robust policy when noise is introduced in the test environment.

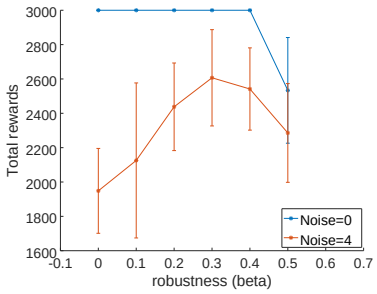

Figure 7. Performance on mismatched train/test environments with respect to beta (K=150).

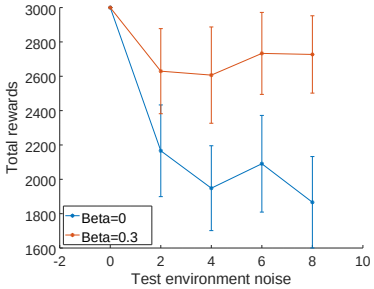

Figure 8. Performance on mismatched train/test environments with respect to noise level (K=150).

References

Barreto, A. M., Precup, D., and Pineau, J. Practical kernel-based reinforcement learning. *Journal of Machine Learning Research*, 17(67):1–70, 2016. URL http://jmlr.org/papers/v17/13-134.html.

Gomez, F. J. *Robust Non-linear Control Through Neuroevolution*. PhD thesis, 2003. AAI3116311.

Gordon, G. J. Stable function approximation in dynamic programming. Technical report, Pittsburgh, PA, USA, 1995.

Iyengar, G. N. Robust dynamic programming. *Math. Oper. Res.*, 30(2):257–280, 2005.

Jaksch, T., Ortner, R., and Auer, P. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11(Apr):1563–1600, 2010.

Marecki, J., Petrik, M., and Subramanian, D. Solution methods for constrained markov decision process with continuous probability modulation. In *Uncertainty in Artificial Intelligence*, pp. 518. Citeseer, 2013.

Nilim, A. and El Ghaoui, L. Robust control of Markov decision processes with uncertain transition matrices. *Oper. Res.*, 53(5):780–798, September 2005. ISSN 0030-364X.

Ormoneit, D. and Sen, Ś. Kernel-based reinforcement learning. *Machine Learning*, 49(2):161–178, Nov 2002. ISSN 1573-0565. doi: 10.1023/A:1017928328829. URL https://doi.org/10.1023/A:1017928328829.

Petrik, M. Approximate Dynamic Programming By Minimizing Distributionally Robust Bounds. In *Proceedings of the 29th International Conference on Machine Learning*, pp. 207. icml.cc / Omnipress, 2012.

Petrik, M. and Subramanian, D. Raam: The benefits of robustness in approximating aggregated mdps in reinforcement learning. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 27*, pp. 1979–1987. Curran Associates, Inc., 2014.

Puterman, M. L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience, April 1994. ISBN 0471619779.

Roy, A., Xu, H., and Pokutta, S. Reinforcement learning under model mismatch. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 3043–3052. Curran Associates, Inc., 2017.

Scherrer, B. and Lesner, B. On the use of non-stationary policies for stationary infinite-horizon markov decision processes. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 25*, pp. 1826–1834. Curran Associates, Inc., 2012.

Sutton, R. S. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems 8*, pp. 1038–1044. MIT Press, 1996.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998. URL citeseer.ist.psu.edu/sutton98reinforcement.html.

Tamar, A., Mannor, S., and Xu, H. Scaling up robust mdps using function approximation. In Xing, E. P. and Jebara, T. (eds.), *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pp. 181–189, Bejing, China, 22–24 Jun 2014. PMLR. URL http://proceedings.mlr.press/v32/tamar14.html.

Tsitsiklis, J. N. and van Roy, B. Feature-based methods for large scale dynamic programming. *Mach. Learn.*, 22(1-3):59–94, January 1996. ISSN 0885-6125. doi: 10.1007/BF00114724. URL http://dx.doi.org/10.1007/BF00114724.

Tsitsiklis, J. N. and Van Roy, B. Analysis of temporal-diffference learning with function approximation. In *Advances in neural information processing systems*, pp. 1075–1081, 1997.

Van Roy, B. Performance loss bounds for approximate value iteration with state aggregation. *Math. Oper. Res.*, 31(2):234–244, February 2006. ISSN 0364-765X. doi: 10.1287/moor.1060.0188. URL http://dx.doi.org/10.1287/moor.1060.0188.

Wiesemann, W., Kuhn, D., and Rustem, B. Robust markov decision processes. *Mathematics of Operations Research*, 38(1):153–183, 2013.