A Kernel-Based Approach to Non-Stationary Reinforcement Learning in Metric Spaces

Omar D. Domingues[1,2] Pierre Ménard[3] Matteo Pirotta[4]

Emilie Kaufmann[1,2,5] Michal Valko[1,6]
[1]Inria Lille [2]Université de Lille [3]OvGU [4]Facebook AI Research [5]CNRS [6]DeepMind Paris

Abstract

In this work, we propose KeRNS: an algorithm for episodic reinforcement learning in non-stationary Markov Decision Processes (MDPs) whose state-action set is endowed with a metric. Using a non-parametric model of the MDP built with time-dependent kernels, we prove a regret bound that scales with the covering dimension of the state-action space and the total variation of the MDP with time, which quantifies its level of non-stationarity. Our method generalizes previous approaches based on sliding windows and exponential discounting used to handle changing environments. We further propose a practical implementation of KeRNS, we analyze its regret and validate it experimentally.

1 Introduction

In reinforcement learning (RL), an agent interacts with an environment by sequentially taking actions, receiving rewards and observing state transitions. One of the main challenges in RL is the trade-off between exploration, the act of gathering information about the environment, and exploitation, the act of using the current knowledge to maximize the sum of rewards. In non-stationary environments, handling this trade-off becomes much harder: what has been learned in the past may no longer be valid in the present. Therefore, the agent needs to constantly re-explore previously known parts of the environment to discover possible changes.

Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130. Copyright 2021 by the author(s).

In this work, we propose KeRNS[1]: an algorithm that handles this problem by acting optimistically and by forgetting data that are far in the past, which naturally causes the agent to keep exploring to discover changes. KeRNS relies on non-parametric kernel estimators of the MDP, and the non-stationarity is handled by using time-dependent kernels.

The regret of an algorithm, defined as the difference between the rewards obtained by an optimal agent and the ones obtained by the algorithm, allows us to quantify how well an agent balances exploration and exploitation. We prove a regret bound for KeRNS that holds in a challenging setting, where the state-action space can be continuous and the environment can change in every episode, as long as the cumulative changes remain small when compared to the total number of episodes.

Related work Regret bounds for RL in stationary environments have been extensively studied in finite (tabular) MDPs (Jaksch et al., 2010; Azar et al., 2017; Dann et al., 2017; Jin et al., 2018; Zanette and Brunskill, 2019), and also in metric spaces under Lipschitz continuity assumptions (Ortner and Ryabko, 2012; Song and Sun, 2019; Sinclair et al., 2019; Domingues et al., 2020; Sinclair et al., 2020). Recent works provide algorithms with regret bounds for non-stationary RL in the tabular setting (Gajane et al., 2018; Ortner et al., 2019; Cheung et al., 2020). These algorithms estimate the transitions and the rewards in an episode k using the data observed up to episode $k-1$. However, since the MDP can change from one episode to another, these estimators are *biased*. If nothing is done to handle this bias, the algorithms will suffer a linear regret (Ortner et al., 2019) that depends on the magnitude of the bias. To deal with this issue, different approaches have been proposed: Gajane et al. (2018) and Cheung et al. (2020) use sliding windows to compute estimators that use

[1]meaning Kernel-based Reinforcement Learning in Non-Stationary environments.

only the most recently observed transitions, whereas Ortner et al. (2019) restart the algorithm periodically and, after each restart, new estimators are build and past data are discarded. In the multi-armed bandit literature, in addition to sliding windows, exponential discounting has also been used as a mean to give more importance to recent data (Kocsis and Szepesvári, 2006; Garivier and Moulines, 2011; Russac et al., 2019). In this paper, we study the *dynamic regret* of the algorithm, where, in each episode k, we compare the learner to the optimal policy of the MDP in episode k. A related approach consists in comparing the performance of the learner to the best stationary policy in hindsight, e.g., (Even-Dar et al., 2009; Yu and Mannor, 2009; Neu et al., 2013; Dick et al., 2014), which is less suited to non-stationary environments, since the performance of any fixed policy can be very bad. Non-stationary RL has also been studied outside the regret minimization framework, without, however, tackling the issue of exploration. For instance, Choi et al. (2000) propose a model where the MDP varies according to a sequence of tasks whose changes form a Markov chain. Szita et al. (2002) and Csáji and Monostori (2008) study the convergence of Q-learning when the environment changes but remain close to a fixed MDP. Assuming full knowledge of the MDP at each time step, but with unknown evolution, Lecarpentier and Rachelson (2019) introduce a risk-averse approach to planning in slowly changing environments. In a related setting, Lykouris et al. (2019) study episodic RL problems where the MDP can be corrupted by an adversary and provide regret bounds in this case.

Contributions We provide the first regret bound for non-stationary RL in continuous environments. More precisely, we show that the `Kernel-UCBVI` algorithm of Domingues et al. (2020), based on non-parametric kernel smoothing, can be modified to tackle non-stationary environments by using appropriate time- and space-dependent kernels. We analyze the resulting algorithm, `KeRNS`, under mild assumptions on the kernel, which in particular recover previously studied forgetting mechanisms to tackle non-stationarity in bandits and RL: sliding windows (Gajane et al., 2018) and exponential discounting (Kocsis and Szepesvári, 2006; Garivier and Moulines, 2011; Russac et al., 2019), and allow for combinations between those. On the practical side, kernel-based approaches can be very computationally demanding since their complexity grows with the number of data points. Building on the notion of representative states, promoted in previous work on practical kernel-based RL (Kveton and Theocharous, 2012; Barreto et al., 2016) we propose an efficient version of `KeRNS`, called `RS-KeRNS`, which has constant runtime per episode. We analyze the regret of `RS-KeRNS`, showing that it enables a trade-off between regret and runtime, and we validate this algorithm empirically.

2 Setting

Notation For any $n \in \mathbb{N}^*$, let $[n] \overset{\text{def}}{=} \{1, \ldots, n\}$. If μ and $P(\cdot|x,a)$ are measures for any (x,a) and f is an arbitrary function, we define $\mu f \overset{\text{def}}{=} \int f(y) \mathrm{d}\mu(y)$ and $Pf(x,a) \overset{\text{def}}{=} \int f(y) \mathrm{d}P(y|x,a)$.[2]

Non-stationary MDPs We consider an episodic RL setting where, in each episode $k \in [K]$, an agent interacts with the environment for $H \in \mathbb{N}^*$ time steps. The time is indexed by (k,h), where k represents an episode and h the time step within the episode. The environment is modeled as a non-stationary MDP, defined by the tuple $(\mathcal{X}, \mathcal{A}, r, \mathrm{P})$, where $\mathcal{X}$ is the state space, $\mathcal{A}$ is the action space, $r = \left\{r_h^k\right\}_{k,h}$ and $\mathrm{P} = \left\{\mathrm{P}_h^k\right\}_{k,h}$ are sets of reward functions and transition kernels, respectively. More precisely, when taking action a in state x at time (k,h), the agent observes a random reward $\widetilde{r}_h^k \in [0,1]$ with mean $r_h^k(x,a)$ and makes a transition to the next state according to the probability measure $\mathrm{P}_h^k(\cdot|x,a)$. A deterministic policy π is a mapping from $[H] \times \mathcal{X}$ to $\mathcal{A}$, and we denote by $\pi(h,x)$ the action chosen in state x at step h. The action-value function of a policy π in step h of episode k is defined as

$$\mathrm{Q}_{k,h}^{\pi}(x,a) \overset{\text{def}}{=} \mathbb{E}\left[\sum_{h'=h}^{H} r_{h'}^k(x_{h'}, a_{h'}) \middle| x_h = x, a_h = a\right]$$

where $x_{h'+1} \sim \mathrm{P}_{h'}^k(\cdot|x_{h'}, a_{h'})$, $a_{h'} = \pi(h', x)$, and its value function is defined by $\mathrm{V}_{k,h}^{\pi}(x) = \mathrm{Q}_{k,h}^{\pi}(x, \pi(h,x))$. The optimal value functions, $\mathrm{V}_{k,h}^*(x) \overset{\text{def}}{=} \sup_\pi \mathrm{V}_{k,h}^{\pi}(x)$ satisfy the Bellman equations (Puterman, 2014)

$$\mathrm{V}_{k,h}^*(x) = \max_{a \in \mathcal{A}} \mathrm{Q}_{k,h}^*(x,a), \text{ where}$$
$$\mathrm{Q}_{k,h}^*(x,a) \overset{\text{def}}{=} r_h^k(x,a) + \mathrm{P}_h^k \mathrm{V}_{k,h+1}^*(x,a)$$

and where $\mathrm{V}_{k,H+1}^* = 0$ by definition.

Dynamic regret The agent interacts with the environment in a sequence of episodes and, in each episode k, it uses a policy π_k that can be chosen based on its observations from previous episodes. We measure its performance by the dynamic regret, defined as the sum over all episodes of the difference between the optimal value function in episode k and the value of π_k:

$$\mathcal{R}(K) \overset{\text{def}}{=} \sum_{k=1}^{K} \left(\mathrm{V}_{k,1}^*(x_1^k) - \mathrm{V}_{k,1}^{\pi_k}(x_1^k) \right)$$

where x_1^k is the starting state in each episode, which is chosen arbitrarily and given to the learner.

[2]See also Table 2 in Appendix A summarizing the main notations used in the paper and in the proofs.

Assumptions Since regret lower bounds scale with the number of states and actions (Jaksch et al., 2010), structural assumptions are needed in order to enable learning in continuous MDPs. A common assumption is that rewards and transitions are Lipschitz continuous with respect to some known metric (Ortner and Ryabko, 2012; Song and Sun, 2019; Domingues et al., 2020; Sinclair et al., 2020), which is the approach that we follow in this work. We make no assumptions regarding how the MDP changes, and our regret bounds will be expressed in terms of its total variation over time.

Assumption 1. *The state-action space $\mathcal{X} \times \mathcal{A}$ is equipped with a metric $\rho : (\mathcal{X} \times \mathcal{A})^2 \to \mathbb{R}_+$, which is given to the learner. Also, we assume that there exists a metric $\rho_\mathcal{X}$ on $\mathcal{X}$ such that, for all (x, x', a), $\rho\left[(x, a), (x', a)\right] \leq \rho_\mathcal{X}(x, x')$.*[3]

Assumption 2. *The reward functions are L_r-Lipschitz and the transition kernels are L_p-Lipschitz with respect to the 1-Wasserstein distance: $\forall (x, a, x', a')$ and $\forall (k, h) \in [K] \times [H]$,*

$$\left| r_h^k(x,a) - r_h^k(x',a') \right| \leq L_\mathrm{r} \rho\left[(x,a),(x',a')\right], \text{ and}$$
$$\mathbb{W}_1\left(\mathrm{P}_h^k(\cdot|x,a), \mathrm{P}_h^k(\cdot|x',a')\right) \leq L_\mathrm{p} \rho\left[(x,a),(x',a')\right]$$

where, for two measures μ and ν, we have $\mathbb{W}_1(\mu,\nu) \overset{\text{def}}{=} \sup_{f:\mathrm{Lip}(f)\leq 1} \int_\mathcal{X} f(y)(\mathrm{d}\mu(y) - \mathrm{d}\nu(y))$ and where, for any Lipschitz function $f : \mathcal{X} \to \mathbb{R}$ with respect to $\rho_\mathcal{X}$, $\mathrm{Lip}(f)$ denotes its Lipschitz constant.

Assumption 3. *For any (k, h), the optimal Q-function $\mathrm{Q}^*_{k,h}$ is L-Lipschitz with respect to ρ. Assumptions 1 and 2 imply that $L \leq \sum_{h=1}^H L_\mathrm{r} L_\mathrm{p}{}^{H-h}$ (Lemma 26 in the Appendix).*

3 An Algorithm for Kernel-Based RL in Non-Stationary Environments

In this section, we introduce KeRNS, a model-based RL algorithm for learning in non-stationary MDPs. In each episode k, we estimate the transitions and the rewards using the data observed up to episode $k-1$. Using exploration bonuses that represent the uncertainty in the estimated model, KeRNS builds a Q-function Q_h^k, and plays the greedy policy with respect to it. KeRNS generalizes sliding-window and exponential discounting approaches by considering time-dependent kernel functions, which also allow us to handle exploration in continuous environments (Domingues et al., 2020).

[3] If $(\mathcal{A}, \rho_\mathcal{A})$ is also a metric space, we can take $\rho\left[(x,a),(x',a')\right] = \rho_\mathcal{X}(x,x') + \rho_\mathcal{A}(a,a')$, for instance. See Section 2.3 of Sinclair et al. (2019) for more examples and a discussion.

3.1 Kernel-Based Estimators for Changing MDPs

Let $\Gamma : \mathbb{N} \times (\mathcal{X} \times \mathcal{A})^2 \to [0,1]$ be a *non-stationary kernel function*, where $\Gamma(t, u, v)$ represents the similarity between two state action pairs u, v in $\mathcal{X} \times \mathcal{A}$ visited at an interval t.

Definition 1 (kernel weights)**.** *Let (x_h^s, a_h^s) be the state-action pair visited at time (s, h). For any $(x, a) \in \mathcal{X} \times \mathcal{A}$ and $s < k$, we define the weights and the normalized weights at time (k, h) as*

$$w_h^{k,s}(x,a) \overset{\text{def}}{=} \Gamma\left(k - s - 1, (x,a), (x_h^s, a_h^s)\right)$$

and $\widetilde{w}_h^{k,s}(x,a) \overset{\text{def}}{=} w_h^{k,s}(x,a)/\mathbf{C}_h^k(x,a)$, where $\mathbf{C}_h^k(x,a) \overset{\text{def}}{=} \beta + \sum_{s=1}^{k-1} w_h^{k,s}(x,a)$ and $\beta > 0$ is a regularization parameter.

Using the kernel function Γ and past data, KeRNS builds estimators $\widehat{r}_h^k$ of the reward function and $\widehat{P}_h^k$ of the transitions at time (k,h), which are defined below.

Definition 2 (empirical MDP)**.** *At time $(s,h) \in [K] \times [H]$, let $(x_h^s, a_h^s, x_{h+1}^s, \widetilde{r}_h^s)$ represent the state, the action, the next state and the reward observed by the algorithm. Before each episode k, KeRNS estimates the rewards and transitions using the data observed up to episode $k-1$:*

$$\widehat{r}_h^k(x,a) \overset{\text{def}}{=} \sum_{s=1}^{k-1} \widetilde{w}_h^{k,s}(x,a) \widetilde{r}_h^s,$$
$$\widehat{P}_h^k(y|x,a) \overset{\text{def}}{=} \sum_{s=1}^{k-1} \widetilde{w}_h^{k,s}(x,a) \delta_{x_{h+1}^s}(y)$$

where δ_x is the Dirac measure at x. Let $\widehat{\mathcal{M}}_k$ be the MDP whose rewards and transitions at step h are $\widehat{r}_h^k(x,a)$ and $\widehat{P}_h^k(y|x,a)$.[4]

The weights $w_h^{k,s}(x,a)$ measure the influence that the transitions and rewards observed at time (s,h) will have on the estimators for the state-action pair (x,a) at time (k,h). Their sum, $\mathbf{C}_h^k(x,a)$, is a proxy for the number of visits to (x,a). Intuitively, the kernel function Γ must be designed in order to ensure that $w_h^{k,s}(x,a)$ is small when (x,a) is very far from (x_h^s, a_h^s), with respect to the distance ρ. It must also be small when $k - s - 1$ is large, which means that the sample (x_h^s, a_h^s) was collected too far in the past and should have a small impact on the estimators. For our theoretical analysis, we will need the assumptions below on the kernel function Γ.

[4] Since the normalized weights do not sum to 1, $\widehat{P}_h^k$ is not a probability kernel. In this case, we suffer a bias of order β and the property that $\widehat{P}_h^k$ is a sub-probability measure is enough for the analysis.

Assumption 4 (kernel properties). *Let $\sigma > 0$, $\eta \in]0,1[$ and $W \in \mathbb{N}$ be the kernel parameters. For each set of parameters, we assume that we have access to a base kernel function $\overline{\Gamma}_{(\eta,W)} : \mathbb{N} \times \mathbb{R} \to [0,1]$ and we define, for any $t, u, v \in \mathbb{N}^* \times \mathcal{X} \times \mathcal{A}$,*

$$\Gamma(t,u,v) = \overline{\Gamma}_{(\eta,W)}\left(t, \rho\left[u,v\right]/\sigma\right).$$

We assume that $z \mapsto \overline{\Gamma}_{(\eta,W)}(t,z)$ is non-increasing for any $t \in \mathbb{N}$. Additionally, we assume that there exists positive constants C_1, C_2, a constant $C_3 \geq 0$ and an arbitrary function $G : \mathbb{R} \to \mathbb{R}_{\geq 0}$ that satisfies $G(4) > 0$ such that

(1) $\forall (t,z)$, $\overline{\Gamma}_{(\eta,W)}(t,z) \leq C_1 \exp\left(-z^2/2\right)$

(2) $\forall (t,y,z)$, $\left|\overline{\Gamma}_{(\eta,W)}(t,y) - \overline{\Gamma}_{(\eta,W)}(t,z)\right| \leq C_2 \left|y - z\right|$

(3) $\forall z$, $\overline{\Gamma}_{(\eta,W)}(t,z) \leq C_3 \eta^t$, *for all $t \geq W$*

(4) $\forall z$, $\overline{\Gamma}_{(\eta,W)}(t,z) \geq G(z)\eta^t$, *for all $t < W$.*

We now provide some justification for these conditions. (1) ensures that the bias due to kernel smoothing remains bounded by $\widetilde{\mathcal{O}}\left(\sigma\right)$ (Lemma 23); (2) ensures smoothness conditions that are needed to provide concentration inequalities for the rewards and transitions (Lemma 24); (3) and (4) allow us to control the bias and the variance due to non-stationarity, respectively (Lemmas 2 and 16). Intuitively, (3) says the algorithm should forget data further than W episodes in the past, and (4) says that recent data in the W most recent episodes must have a minimum weight. The condition $G(4) > 0$ is mostly technical: it is used to ensure that $\mathbf{C}_h^k(x,a)$ is not too small in a 4σ-neighborhood of (x,a) (see lemmas 15 and 16). The kernels in the example below satisfy our conditions, and show that they indeed generalize sliding-window and exponential discounting approaches:

Example 1 (sliding-window and exponential discount). *The kernels $\overline{\Gamma}_{(\eta,W)}(t,z) = \mathbb{I}\left\{t < W\right\}\exp(-\left|z\right|^p/2)$ (sliding-window) and $\overline{\Gamma}_{(\eta,W)}(t,z) = \eta^t \exp(-\left|z\right|^p/2)$ (exponential discount) satisfy Assumption 4 for $p \geq 2$.*

The conditions in Assumption 4 are needed to prove our regret bounds. However, if one has further knowledge about the MDP and its changes, this information can also be integrated to the kernel function Γ. For example, if the MDP only changes in certain region of the state-action space, the kernel can be designed to forget past data only in that region. Also, the kernel Γ can be designed to enforce restarts, as proposed by Ortner et al. (2019) for finite MDPs, by setting $\Gamma(t,u,v)$ to zero every time t exceeds a certain threshold. Although this would require a separate analysis, our proof could be combined to the one of (Ortner et al., 2019) to obtain a regret bound in this case.

3.2 Algorithm

KeRNS is presented in Algorithm 1. At time (k,h), let $\mathtt{B}_h^k(x,a)$ be the exploration bonus at (x,a) representing the uncertainty of $\widehat{\mathcal{M}}_k$ with respect to the true MDP:

$$\mathtt{B}_h^k(x,a) = \widetilde{\mathcal{O}}\left(\frac{H}{\sqrt{\mathbf{C}_h^k(x,a)}} + \frac{\beta H}{\mathbf{C}_h^k(x,a)} + L\sigma\right) \quad (1)$$

where $\widetilde{\mathcal{O}}\left(\cdot\right)$ hides logarithmic terms. The exact expression for the bonuses is given in Def. 5 in Appendix A. Before starting episode k, KeRNS computes, for all $h \in [H]$, the values Q_h^k by running backward induction on $\widehat{\mathcal{M}}_k$, with the bonus $\mathtt{B}_h^k(x,a)$ added to the rewards, followed by an interpolation step:

$$\widetilde{Q}_h^k(x,a) = \widehat{r}_h^k(x,a) + \widehat{P}_h^k V_{h+1}^k(x,a) + \mathtt{B}_h^k(x,a)$$
$$Q_h^k(x,a) = \min_{s \in [k-1]}\left(\widetilde{Q}_h^k(x_h^s, a_h^s) + L\rho\left[(x,a),(x_h^s,a_h^s)\right]\right)$$
$$V_h^k(x) = \min\left(H - h + 1, \max_a Q_h^k(x,a)\right)$$

where $V_{H+1}^k \overset{\text{def}}{=} 0$. The interpolation is needed to ensure that Q_h^k and V_h^k are L-Lipschitz. This procedure is defined in detail in Algorithm 3 in Appendix A, which is the same kind of backward induction used by `Kernel-UCBVI` (Domingues et al., 2020). Once Q_h^k is computed, KeRNS plays the greedy policy associated to it. Notice that, although $Q_h^k(x,a)$ and $V_h^k(x)$ are defined for all (x,a), they only need to be computed for the states and actions observed by the algorithm up to episode k.

Algorithm 1 KeRNS

```
1:  Input: $K$, $H$, $L$, $L_r$, $L_p$, $\beta$, $\delta$, $d$, $\sigma$, $\eta$, $W$.
2:  Initialize history: $\mathcal{T}_h = \emptyset$ for all $h \in [H]$.
3:  for episode $k = 1, \ldots, K$ do
4:      get initial state $x_1^k$
5:      // Run kernel backward induction
6:      compute $(Q_h^k)_h$ using $(\mathcal{T}_h)_h$ and Algorithm 3.
7:      for $h = 1, \ldots, H$ do
8:          execute $a_h^k = \mathrm{argmax}_a Q_h^k(x_h^k, a)$
9:          observe reward $\widetilde{r}_h^k$ and next state $x_{h+1}^k$
10:         store transition $\mathcal{T}_h = \mathcal{T}_h \cup \left\{x_h^k, a_h^k, x_{h+1}^k, \widetilde{r}_h^k\right\}$
11:     end for
12: end for
```

3.3 Theoretical guarantees

We introduce Δ, the total variation of the MDP in K episodes:

Definition 3 (MDP variation). *We define $\Delta = \Delta^{\mathrm{r}} +$*

$L\Delta^{\mathrm{p}}$, *where*

$$\Delta^{\mathrm{r}} \overset{\text{def}}{=} \sum_{i=1}^{K}\sum_{h=1}^{H} \sup_{x,a} \left| r_h^i(x,a) - r_h^{i+1}(x,a) \right| ,$$

$$\Delta^{\mathrm{p}} \overset{\text{def}}{=} \sum_{i=1}^{K}\sum_{h=1}^{H} \sup_{x,a} \mathbb{W}_1 \left(\mathrm{P}_h^i(\cdot|x,a), \mathrm{P}_h^{i+1}(\cdot|x,a) \right)$$

A similar notion has been introduced, for instance, by Ortner et al. (2019); Li and Li (2019) for MDPs and by Besbes et al. (2014) for multi-armed bandits. Here, the difference is that we use the Wasserstein distance to define the variation of the transitions, instead of the total variation (TV) distance $\|\mathrm{P}_h^i(\cdot|x,a) - \mathrm{P}_h^{i+1}(\cdot|x,a)\|_1$. This choice was made in order to take into account the metric ρ when measuring changes in the environment: our results would be analogous if we had chosen the TV distance.[5]

Using the same algorithm, we provide two regret bounds for `KeRNS`, which are given below. The notation $\lesssim$ omits constants and logarithmic terms (see Definition 4 in Appendix A).

Theorem 1. *The regret of* `KeRNS` *is bounded as* $\mathcal{R}^{\texttt{KeRNS}}(K) \lesssim \min\left(\mathcal{R}_1(K), \mathcal{R}_2(K)\right) + \mathbf{bias}(\sigma, \eta, W, \Delta)$, *where*

$$\mathcal{R}_1(K) = H^2 K \sqrt{\log\frac{1}{\eta}} \sqrt{|\mathcal{C}'_\sigma|\,|\mathcal{C}_\sigma|} + H^2\,|\mathcal{C}_\sigma|\,K \log\frac{1}{\eta}$$

$$\mathcal{R}_2(K) = H^2 K \sqrt{\log\frac{1}{\eta}} \sqrt{|\mathcal{C}_\sigma|} + H^3\,|\mathcal{C}_\sigma|\,|\mathcal{C}'_\sigma|\,K \log\frac{1}{\eta}$$

$$\mathbf{bias}(\sigma, \eta, W, \Delta) = W\Delta H + \frac{\eta^W}{1-\eta} K H^3 + LKH\sigma$$

with probability at least $1-\delta$. *Here,* $|\mathcal{C}'_\sigma|$ *and* $|\mathcal{C}_\sigma|$ *are the* σ*-covering numbers of* $(\mathcal{X}, \rho_{\mathcal{X}})$ *and* $(\mathcal{X}\times\mathcal{A}, \rho)$ *respectively,* (σ, η, W) *are the kernel parameters.*

Proof. This result comes from combining theorems 3 and 4 in Appendix F. See Section 5 for a proof outline.

As discussed below, after optimizing the kernel parameters (Table 1), the bound $\mathcal{R}_1$ has a worse dependence on K, and a better dependence on Δ. On the other hand, $\mathcal{R}_2$ is better with respect to K, but worse in Δ. Concretely, this trade-off may give hints on how to choose the kernel parameters according to the amount of variation that we expect to see in the environment. Technically, the difference comes from how we handle the concentration of the transitions in the proof. To obtain $\mathcal{R}_1$, we use concentration inequalities on the term $|(\widehat{P}_h^k - \mathrm{P}_h^k)f|$ for *all* functions f that are bounded and Lipschitz continuous. To obtain $\mathcal{R}_2$, the concentration is done only for $f = \mathrm{V}_{k,h+1}^*$, but this results in larger second-order terms, as in (Azar et al., 2017; Domingues et al., 2020).

Corollary 1. *Let* d *be the covering dimension of* $(\mathcal{X}\times\mathcal{A}, \rho)$*. By optimizing the kernel parameters, we obtain the regret bounds in Table 1. Table 3 in Appendix B.2 gives the values of* (σ, η, W) *that yield these bounds.*

Proof. Assuming that $|\mathcal{C}'_\sigma| \leq |\mathcal{C}_\sigma|$, we have that $|\mathcal{C}_\sigma|$ and $|\mathcal{C}'_\sigma|$ are $\mathcal{O}\left(1/\sigma^d\right)$. Then, the bounds follow from Theorem 1. The general case, handling separately the covering dimensions of $(\mathcal{X}\times\mathcal{A}, \rho)$ and $(\mathcal{X}, \rho_{\mathcal{X}})$, is stated in corollaries 6 and 9 in Appendix F.

Discussion We now discuss regret bounds for optimized kernel parameters, according to the covering dimension of $(\mathcal{X}\times\mathcal{A}, \rho)$, denoted by d. Roughly, the covering dimension is the smallest number $d \geq 0$ such that the σ-covering number $|\mathcal{C}_\sigma|$ is $\mathcal{O}\left(1/\sigma^d\right)$.[6] We consider two cases: the tabular (finite MDP) case, where the covering dimension of $(\mathcal{X}\times\mathcal{A}, \rho)$ is $d = 0$, and the continuous case, where $d > 0$.

Tabular case Let $X = |\mathcal{X}|$ and $A = |\mathcal{A}|$. By taking $\sigma = 0$, we have $|\mathcal{C}'_\sigma| = X$ and $|\mathcal{C}_\sigma| = XA$. As shown in Table 1, the $\mathcal{R}_1$ bound states that the regret of `KeRNS` is $\widetilde{\mathcal{O}}\left(H^2 X\sqrt{A}\Delta^{\frac{1}{3}}K^{\frac{2}{3}}\right)$. This bound matches the one proved by Ortner et al. (2019) for the average reward setting using restarts, up to a factor of $H^{\frac{2}{3}}$ coming from our episodic setting, where the transitions P_h^k depend on h. The $\mathcal{R}_2$ bound states that the regret of `KeRNS` can be improved to $\widetilde{\mathcal{O}}\left(H^2\sqrt{XA}\Delta^{\frac{1}{3}}K^{\frac{2}{3}}\right)$, up to second-order terms. In the bandit case ($H = 1$), these bounds are *optimal* in terms of K and Δ (Besbes et al., 2014).

Continuous case For $d > 0$, we prove the first dynamic regret bounds in our setting, which are of order $H^2\Delta^{\frac{1}{3}}K^{\frac{2d+2}{2d+3}}$ (better in Δ) or $H^2\Delta^{\frac{1}{2}}K^{\frac{2d+1}{2d+2}}$ (better in K) for two different tunings of the kernel. Deriving a lower bound in the non-stationary case for $d > 0$ is an open problem, even for multi-armed bandits. As a sanity-check, we note that in stationary MDPs, for which $\Delta = 0$, we recover the regret bound of `Kernel-UCBVI`[7] (Domingues et al., 2020) of $H^3K^{\frac{2d}{2d+1}}$ from the bound $\mathcal{R}_2$ with $\log(1/\eta) = 1/K$, $W \to \infty$ and $\sigma = K^{-\frac{1}{2d+1}}$, which is optimal for $d = 1$ in the (stationary) bandit case (Bubeck et al., 2011).

In tabular MDPs, we may achieve sub-linear regret as long as $\Delta < K$.[8] In the continuous case however,

[5]More precisely, in the proof of Corollary 2, the Wasserstein distance could be replaced by the TV distance.

[6]For more details about covering numbers and covering dimension, see Section 3 of Kleinberg et al. (2019) and Section 2.2 of Sinclair et al. (2019).

[7]Another choice of η might allow us to avoid the dependence on H^3 of `Kernel-UCBVI` and get H^2 instead.

[8]Notice that, if Δ scales linearly with the number of

our bounds show that we might need $\Delta < K^{\frac{3}{2d+3}}$ (for the $\mathcal{R}_1$ bound) or $\Delta < K^{\frac{1}{d+1}}$ (for the $\mathcal{R}_2$ bound) in order to avoid a linear regret, which is an immediate consequence of the bounds in Table 1.

Table 1: Regret for optimized kernel parameters.

	bound	regret
$d = 0$	$\mathcal{R}_1$	$H^2 X \sqrt{A} \Delta^{\frac{1}{3}} K^{\frac{2}{3}}$
	$\mathcal{R}_2$	$H^2 \sqrt{XA} \Delta^{\frac{1}{3}} K^{\frac{2}{3}} + H^3 X^2 A \Delta^{\frac{2}{3}} K^{\frac{1}{3}}$
$d > 0$	$\mathcal{R}_1$	$H^2 \Delta^{\frac{1}{3}} K^{\frac{2d+2}{2d+3}}$
	$\mathcal{R}_2$	$H^2 \Delta^{\frac{1}{2}} K^{\frac{2d+1}{2d+2}} + H^{\frac{3}{2}} \Delta^{\frac{1}{4}} K^{\frac{3}{4}}$

Knowledge of Δ To optimally choose the kernel parameters, KeRNS requires an upper bound on the variation Δ. Recent work has started to tackle this issue in bandit algorithms (Chen et al., 2019; Auer et al., 2019), and finite MDPs using sliding windows (Cheung et al., 2020). Their extension to continuous MDPs is left to future work.

4 Efficient Implementation

Since KeRNS uses non-parametric kernel estimators, its computational complexity scales with the number of observed transitions. Let τ_A be the time required to compute the maximum of $a \mapsto Q_h^k(x, a)$. Similarly to Kernel-UCBVI, its total space complexity is $\mathcal{O}(KH)$ and its time complexity per episode k is $\mathcal{O}\left(Hk^2 + H\tau_A k\right)$, resulting in a total runtime of $\mathcal{O}\left(HK^3 + H\tau_A K^2\right)$. This runtime is very prohibitive in practice, especially in changing environments, where we might need to run the algorithm for a very long time. Domingues et al. (2020) propose a version of Kernel-UCBVI with improved per-episode time complexity of $\mathcal{O}(H\tau_A k)$ based on real-time dynamic programming (RTDP) (Barto et al., 1995; Efroni et al., 2019). However, this requires the upper bounds V_h^k to be non-increasing, which is not the case in KeRNS, since V_h^k increases in regions that were not visited recently. This property is necessary to promote extra exploration and adapt to possible changes. Additionally, the RTDP-based approach of Domingues et al. (2020) still has a time complexity that scales with time, which can be a considerable issue in practice. Here, we propose an alternative to run KeRNS in *constant* time per episode, while controlling the impact of this speed-up on the regret.

episodes K, we cannot expect to learn. Indeed, according to the lower bound (Besbes et al., 2014), the regret is necessarily linear in this case.

4.1 Using Representative States and Actions

As proposed by Kveton and Theocharous (2012) and Barreto et al. (2016), we take an approach based on using *representative states* to construct an algorithm called RS-KeRNS (for KeRNS on Representative States). In each episode k, RS-KeRNS keeps and updates sets of representative states $\bar{\mathcal{X}}_h$, actions $\bar{\mathcal{A}}_h$ and next-states $\bar{\mathcal{Y}}_h$, for each h, whose cardinalities are denoted by $\bar{X}_h$, $\bar{A}_h$ and $\bar{Y}_h$, respectively. For simplicity, we omit the dependence on k of these sets and their cardinalities. Every time a new transition $\{x_h^k, a_h^k, x_{h+1}^k, \widetilde{r}_h^k\}$ is observed, the representative sets are updated using Algorithm 2, which ensures that any two representative state-action pairs are at a distance greater than ε from each other. Similarly, it ensures that any pair of representative next-states are at a distance greater than $\varepsilon_{\mathcal{X}}$ from each other. Then, (x_h^k, a_h^k) and x_{h+1}^k are mapped to their nearest neighbors in $\bar{\mathcal{X}}_h \times \bar{\mathcal{A}}_h$ and $\bar{\mathcal{Y}}_h$, respectively, and the estimators of the rewards and transitions are updated. Consequently, we build a finite MDP, denoted by $\breve{\mathcal{M}}_k$, with $\bar{X}_h$ states, $\bar{A}_h$ actions and $\bar{Y}_h$ next-states, *per stage* h. The rewards and transitions of $\breve{\mathcal{M}}_k$ can be stored in arrays of size $\bar{X}_h \bar{A}_h$ and $\bar{X}_h \bar{A}_h \bar{Y}_h$, for each h.

RS-KeRNS is described precisely in Algorithm 4 in Appendix G. It computes a Q-function for all $(\bar{x}, \bar{a}) \in \cup_h \bar{\mathcal{X}}_h \times \bar{\mathcal{A}}_h$ by running backward induction in $\breve{\mathcal{M}}_k$, which is then extended to any $(x, a) \in \mathcal{X} \times \mathcal{A}$ by performing an interpolation step, as in KeRNS. In Appendix G, we explain how the rewards and transitions estimators of $\breve{\mathcal{M}}_k$ can be updated online. Below, we provide regret and runtime guarantees for this efficient implementation.

Algorithm 2 Update Representative Sets

```
1: Input: $k$, $h$, $\bar{\mathcal{X}}_h$, $\bar{\mathcal{A}}_h$, $\bar{\mathcal{Y}}_h$, $\{x_h^k, a_h^k, x_{h+1}^k\}$, $\varepsilon$, $\varepsilon_{\mathcal{X}}$.
2: if $\min_{(\bar{x},\bar{a}) \in \bar{\mathcal{X}}_h \times \bar{\mathcal{A}}_h} \rho\left[(\bar{x}, \bar{a}), (x_h^k, a_h^k)\right] > \varepsilon$ then
3:     $\bar{\mathcal{X}}_h = \bar{\mathcal{X}}_h \cup \{x_h^k\}$,   $\bar{\mathcal{A}}_h = \bar{\mathcal{A}}_h \cup \{a_h^k\}$
4: end if
5: if $\min_{\bar{y} \in \bar{\mathcal{Y}}_h} \rho_{\mathcal{X}}\left(\bar{x}, x_{h+1}^k\right) > \varepsilon_{\mathcal{X}}$ then
6:     $\bar{\mathcal{Y}}_h = \bar{\mathcal{Y}}_h \cup \{x_{h+1}^k\}$
7: end if
```

4.2 Theoretical Guarantees & Runtime

Theorem 2 states that RS-KeRNS enjoys the same regret bounds as KeRNS plus a bias term that can be controlled by ε and $\varepsilon_{\mathcal{X}}$, as long as we use a Gaussian kernel.

Theorem 2 (regret of RS-KeRNS)**.** *Let* $\chi_{(\eta, W)} : \mathbb{N} \to [0, 1]$*,* $u, v \in \mathcal{X} \times \mathcal{A}$*, and consider the kernel*

$$\Gamma(t, u, v) = \chi_{(\eta, W)}(t) \exp\left(-\rho\left[u, v\right]^2 / (2\sigma^2)\right)$$

assumed to satisfy Assumption 4. With this choice of kernel, the regret of RS-KeRNS *is bounded by*

$$\mathcal{R}(K) \lesssim \mathcal{R}^{\texttt{KeRNS}}(K) + L(\varepsilon + \varepsilon_{\mathcal{X}})KH^2 + \frac{\varepsilon}{\sigma}KH^3$$

with probability at least $1-\delta$*, where* $\mathcal{R}^{\texttt{KeRNS}}$ *is regret bound of* KeRNS *given in Theorem 1.*

Proof. This result comes from theorems 5 and 6 in Appendix G. See Appendix B for a proof outline.

Lemma 1 (runtime of RS-KeRNS). *Consider the kernel defined in Theorem 2, and let* $(\alpha_i)_{i\geq 1}$ *be a sequence of real numbers in* $[0,1]$*. If we take* $\chi_{(\eta,W)}(t) = \prod_{i=1}^{t}\alpha_i$*, the per-episode runtime of* RS-KeRNS *is bounded by*

$$\mathcal{O}\left(H\min\left(k^2, |\mathcal{C}_\varepsilon|\left|\mathcal{C}'_{\varepsilon_{\mathcal{X}}}\right|\right) + H\min\left(k, \left|\mathcal{C}'_{\varepsilon_{\mathcal{X}}}\right|\right)\tau_A\right),$$

where $|\mathcal{C}_\varepsilon|$ *is the* ε*-covering number of* $(\mathcal{X}\times\mathcal{A},\rho)$*,* $\left|\mathcal{C}'_{\varepsilon_{\mathcal{X}}}\right|$ *is the* $\varepsilon_{\mathcal{X}}$*-covering number of* $(\mathcal{X},\rho)$*, and* τ_A *is the time required to compute the maximum of* $a \mapsto Q_h^k(x,a)$*. In particular, we can take* $\alpha_i = \eta$ *for all* i*, which gives an exponential-discount strategy for handling non-stationarity.*

Proof. By construction, in any episode k, we have $\bar{X}_h\bar{A}_h \leq \min(k, |\mathcal{C}_\varepsilon|)$ and $\bar{Y}_h \leq \min\left(k, \left|\mathcal{C}'_{\varepsilon_{\mathcal{X}}}\right|\right)$. Backward induction (Algorithm 5) is performed in $\mathcal{O}\left(\sum_h \bar{X}_h\bar{A}_h\bar{Y}_h + \tau_A\sum_h \bar{Y}_h\right)$ time, and the choice of $\chi_{(\eta,W)}(t)$ implies that the model updates can be done in $\mathcal{O}\left(\sum_h \bar{X}_h\bar{A}_h\bar{Y}_h\right)$ time, as detailed in Appendix G.2.

Consequently, the constants ε and $\varepsilon_{\mathcal{X}}$ provide a trade-off between regret and computational complexity. Since $|\mathcal{C}_\varepsilon| = \mathcal{O}\left(\varepsilon^{-d_1}\right)$ and $\left|\mathcal{C}'_{\varepsilon_{\mathcal{X}}}\right| = \mathcal{O}\left(\varepsilon^{-d_2}\right)$, increasing $(\varepsilon, \varepsilon_{\mathcal{X}})$ may reduce *exponentially* the runtime of RS-KeRNS, while having only a *linear* increase in its regret.

Kveton and Theocharous (2012) and Barreto et al. (2016) studied the idea of using representative states to accelerate kernel-based RL (KBRL), but we provide the first regret bounds in this setting. More precisely, our result improves previous work in the following aspects: (i) Kveton and Theocharous (2012) and Barreto et al. (2016) do not tackle exploration and do not have finite–time analyses: they provide approximate versions of the KBRL algorithm of Ormoneit and Sen (2002) which has asymptotic guarantees assuming that transitions are generated from *independent* samples; (ii) The error bounds of Kveton and Theocharous (2012) scale with $\exp(1/\sigma^2)$. In our online setting, σ can be chosen as a function of the horizon K, and their bound could result in an error that scales exponentially with K, instead of linearly, as ours. Our result comes from an improved analysis of the smoothness of kernel estimators, that leverages the regularization constant β (Lemma 25); (iii) Barreto et al. (2016) propose an algorithm that also builds a set of representative states in an online way. However, their theoretical guarantees only hold when this set is *fixed*, i.e., cannot be updated during exploration, whereas our bounds hold in this case; (iv) unlike (Kveton and Theocharous, 2012; Barreto et al., 2016), our theoretical results also hold in continuous action spaces.

4.3 Numerical Validation

To illustrate the behavior of RS-KeRNS, we consider a continuous MDP whose state-space is the unit ball in $\mathbb{R}^2$ with four actions, representing a move to the right, left, up or down. The agent starts at $(0,0)$. Let $b_i^k \in \{0, 0.25, 0.5, 0.75, 1\}$ and $x_i \in \{(0.8,0.0),(0.0,0.8),(-0.8,0.0),(0.0,-0.8)\}$. We consider the following mean reward function:

$$r_h^k(x,a) = \sum_{i=1}^{4} b_i^k \max\left(0, 1 - \frac{\|x-x_i\|_2}{0.5}\right)$$

which do not depend on h. Every N episodes, the coefficients b_i^k are changed, which impact the optimal policy.

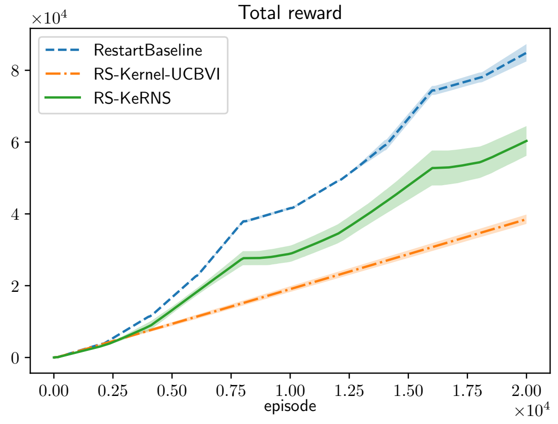

Figure 1: Performance of RS-KeRNS compared to baselines for $N = 2000$. Average over 4 runs.

Taking $\eta = \exp(-(1/N)^{2/3})$, we used the kernel $\Gamma(t,u,v) = \eta^t\exp\left(-(\rho[u,v]/\sigma)^4/2\right)$. We set $\sigma = 0.05$, $\varepsilon = \varepsilon_{\mathcal{X}} = 0.1$, $\beta = 0.01$, $H = 15$ and ran the algorithm for 2×10^4 episodes. KeRNS was compared to two baselines: (i) `Kernel-UCBVI` combined with representative states, that we call `RS-Kernel-UCBVI`, which is designed for stationary environments. This corresponds to RS-KeRNS with $\chi(t) = 1$, that is, $\eta = 1$; (ii) A restart-based algorithm, called `RestartBaseline`, which is implemented as `RS-Kernel-UCBVI`, but it has *full information* about when the environment changes, and, at every change, it restarts its reward estimator and its bonuses. We can see that, as expected, RS-KeRNS outperforms `RS-Kernel-UCBVI`, which was not designed for non-stationary environments, and is able to "track" the behavior of the restart-based algorithm which has full information about how the environment changes. In Appendix I, we give more details about the experimental setup and provide more experiments, varying the period N of changes in the MDP and the kernel function.

5 Proof Outline

We now outline the proof of Theorem 1 assuming, for simplicity, that the rewards are known.

Bias due to non-stationarity To bound the bias, we introduce an average MDP with transitions $\overline{P}_h^k$:

$$\overline{P}_h^k(y|x,a) \overset{\text{def}}{=} \sum_{s=1}^{k-1} \widetilde{w}_h^{k,s}(x,a)\mathrm{P}_h^s(y|x,a) + \frac{\beta\, \mathrm{P}_h^k(y|x,a)}{\mathbf{C}_h^k(x,a)},$$

where $(\mathrm{P}_h^s)_{s,h}$ are the true transitions at time (s,h). We prove that, for any L-Lipschitz function f bounded by H, (Corollary 2):

$$\left|\left(\mathrm{P}_h^k - \overline{P}_h^k\right) f(x,a)\right| \leq \mathbf{bias_p}(k,h)$$

where the term $\mathbf{bias_p}(k,h)$ is defined as

$$L \sum_{i=1\vee(k-W)}^{k-1} \sup_{x,a} \mathbb{W}_1\left(\mathrm{P}_h^i(\cdot|x,a), \mathrm{P}_h^{i+1}(\cdot|x,a)\right) + \frac{2C_3 H}{\beta}\frac{\eta^W}{1-\eta}\,.$$

Concentration Using concentration inequalities for weighted sums, we prove that $\widehat{P}_h^k$ is close to the average transition $\overline{P}_h^k$ using Hoeffding- and Bernstein-type inequalities (lemmas 5, 6, 7, and 8), and define an event $\mathcal{G}$ where our confidence sets hold (Lemma 9), such that $\mathbb{P}[\mathcal{G}] \geq 1-\delta/2$. For instance, Lemma 5 gives us

$$\left|(\widehat{P}_h^k - \overline{P}_h^k)\mathrm{V}_{k,h+1}^*(x,a)\right| \lesssim \sqrt{\frac{H^2}{\mathbf{C}_h^k(x,a)}} + \frac{\beta H}{\mathbf{C}_h^k(x,a)} + L\sigma\,,$$

which explains the form of the exploration bonuses.

Upper bound on the true value function On the event $\mathcal{G}$, we show that (Lemma 10):

$$Q_h^k(x,a) + \sum_{h'=h}^{H} \mathbf{bias}(k,h) \geq \mathrm{Q}_{k,h}^*(x,a)$$

where the term $\mathbf{bias}(k,h)$ is the sum of $\mathbf{bias_p}(k,h)$ defined above, and a similar term representing the bias in the reward estimation.

Regret bounds Let $(\widetilde{x}_h^k, \widetilde{a}_h^k)$ be the state-action pair among the previously visited ones that is the closest to (x_h^k, a_h^k):

$$(\widetilde{x}_h^k, \widetilde{a}_h^k) \overset{\text{def}}{=} \operatorname*{argmin}_{(x_h^s,a_h^s):s<k,h\in[H]} \rho\left[(x_h^k,a_h^k),(x_h^s,a_h^s)\right].$$

We show that (see proof of Lemma 11):

$$H\sum_{k=1}^{K}\sum_{h=1}^{H} \mathbb{I}\left\{\rho\left[(x_h^k,a_h^k),(\widetilde{x}_h^k,\widetilde{a}_h^k)\right] > 2\sigma\right\} \leq H^2\left|\mathcal{C}_\sigma\right|.$$

Thus, to simplify the outline, for all (k,h), we assume that $\rho\left[(x_h^k,a_h^k),(\widetilde{x}_h^k,\widetilde{a}_h^k)\right] \leq 2\sigma$ and add $H^2\left|\mathcal{C}_\sigma\right|$ to the final regret bound. On the event $\mathcal{G}$, we prove that the regret of KeRNS is bounded by (lemmas 11 and 12):

$$\mathcal{R}(K) \lesssim \sum_{k=1}^{K}\sum_{h=1}^{H}\left(\frac{H}{\sqrt{\mathbf{C}_h^k(\widetilde{x}_h^k,\widetilde{a}_h^k)}} + \frac{\beta H}{\mathbf{C}_h^k(\widetilde{x}_h^k,\widetilde{a}_h^k)}\right) + \sum_{k=1}^{K}\sum_{h=1}^{H}\mathbf{bias}(k,h) + LKH\sigma + H^2\left|\mathcal{C}_\sigma\right|$$

where we omitted factors involving $|\mathcal{C}_\sigma|$ and $|\mathcal{C}'_\sigma|$ (which depend on the type of bound considered, $\mathcal{R}_1$ or $\mathcal{R}_2$), and martingale terms (which are bounded by $\approx H^{3/2}\sqrt{K}$ with probability at least $1-\delta/2$).

Using the properties of the kernel Γ (Assumption 4), we prove that (Lemma 16):

$$\sum_{k=1}^{K}\sum_{h=1}^{H}\frac{1}{\sqrt{\mathbf{C}_h^k(\widetilde{x}_h^k,\widetilde{a}_h^k)}} \lesssim HK\log\frac{1}{\eta}\left(\left|\mathcal{C}_\sigma\right| + \sqrt{\frac{\left|\mathcal{C}_\sigma\right|}{\log(1/\eta)}}\right)$$

$$\sum_{k=1}^{K}\sum_{h=1}^{H}\frac{1}{\mathbf{C}_h^k(\widetilde{x}_h^k,\widetilde{a}_h^k)} \lesssim H\left|\mathcal{C}_\sigma\right| K\log\frac{1}{\eta}$$

Finally, in Corollary 5, we prove that the bias $\sum_{k=1}^{K}\sum_{h=1}^{H}\mathbf{bias}(h,k)$ is bounded by

$$2W\left(\Delta^{\mathrm{r}} + L\Delta^{\mathrm{p}}\right) + \frac{2C_3(H+1)KH}{\beta}\frac{\eta^W}{1-\eta}.$$

Putting these bounds together, we prove Theorem 1. The proof of Theorem 2 is outlined in Appendix B.

6 Conclusion

In this paper, we introduced and analyzed KeRNS, the first algorithm for continuous MDPs with dynamic regret guarantees in changing environments. Building upon previous work on using representative states for kernel-based RL, we proposed RS-KeRNS, a practical version of KeRNS that runs in constant time per episode. Moreover, we provide the first analysis that quantifies the trade-off between the regret and the computational complexity of this approach. In discrete environments, our regret bound matches the existing lower bound for multi-armed bandits in terms of the number of episodes and the variation of MDP, whereas finding a lower bound in continuous environments remains an open problem.

We believe that the ideas introduced in this paper might be useful for large-scale problems. Indeed, we provide stronger online guarantees for practical kernel-based RL, which has already been shown to be empirically successful in medium-scale environments ($d \approx 10$) (Kveton and Theocharous, 2012; Barreto et al., 2016), and we show that kernel-based RL is naturally suited to tackle non-stationarity. In larger dimension, kernel-based exploration bonuses have been recently shown to enhance exploration in RL for Atari games (Badia et al., 2020), and our approach might inspire the design of bonuses for high-dimensional non-stationary environments.

Acknowledgements

The research presented was supported by European CHIST-ERA project DELTA, French Ministry of Higher Education and Research, Nord-Pas-de-Calais Regional Council, French National Research Agency project BOLD (ANR19-CE23-0026-04), FMJH PGMO project 2018-0045, and the SFI Sachsen-Anhalt for the project RE-BCI ZS/2019/10/102024 by the Investitionsbank Sachsen-Anhalt.

References

Auer, P., Gajane, P., and Ortner, R. (2019). Adaptively tracking the best bandit arm with an unknown number of distribution changes. In *Conference on Learning Theory*, pages 138–158.

Azar, M. G., Osband, I., and Munos, R. (2017). Minimax regret bounds for reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 263–272. JMLR. org.

Badia, A. P., Sprechmann, P., Vitvitskyi, A., Guo, D., Piot, B., Kapturowski, S., Tieleman, O., Arjovsky, M., Pritzel, A., Bolt, A., and Blundell, C. (2020). Never give up: Learning directed exploration strategies. In *International Conference on Learning Representations*.

Barreto, A. M., Precup, D., and Pineau, J. (2016). Practical kernel-based reinforcement learning. *The Journal of Machine Learning Research*, 17(1):2372–2441.

Barto, A. G., Bradtke, S. J., and Singh, S. P. (1995). Learning to act using real-time dynamic programming. *Artificial intelligence*, 72(1-2):81–138.

Besbes, O., Gur, Y., and Zeevi, A. (2014). Stochastic multi-armed-bandit problem with non-stationary rewards. In *Advances in neural information processing systems*, pages 199–207.

Bubeck, S., Munos, R., Stoltz, G., and Szepesvári, C. (2011). X-armed bandits. *Journal of Machine Learning Research*, 12:1587–1627.

Chen, Y., Lee, C.-W., Luo, H., and Wei, C.-Y. (2019). A new algorithm for non-stationary contextual bandits: Efficient, optimal and parameter-free. In *Conference on Learning Theory*, pages 696–726. PMLR.

Cheung, W. C., Simchi-Levi, D., and Zhu, R. (2020). Reinforcement learning for non-stationary Markov decision processes: The blessing of (More) optimism. In *Proceedings of the 37th International Conference on Machine Learning*.

Choi, S. P., Yeung, D.-Y., and Zhang, N. L. (2000). Hidden-mode markov decision processes for nonstationary sequential decision making. In *Sequence Learning*, pages 264–287. Springer.

Csáji, B. C. and Monostori, L. (2008). Value function based reinforcement learning in changing markovian environments. *Journal of Machine Learning Research*, 9(Aug):1679–1709.

Dann, C., Lattimore, T., and Brunskill, E. (2017). Unifying pac and regret: Uniform pac bounds for episodic reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 5713–5723.

Dick, T., Gyorgy, A., and Szepesvari, C. (2014). Online learning in markov decision processes with changing cost sequences. In *International Conference on Machine Learning*, pages 512–520. PMLR.

Domingues, O. D., Ménard, P., Pirotta, M., Kaufmann, E., and Valko, M. (2020). Regret Bounds for Kernel-Based Reinforcement Learning. *arXiv e-prints*, page arXiv:2004.05599.

Efroni, Y., Merlis, N., Ghavamzadeh, M., and Mannor, S. (2019). Tight regret bounds for model-based reinforcement learning with greedy policies. In *Advances in Neural Information Processing Systems*, pages 12203–12213.

Even-Dar, E., Kakade, S. M., and Mansour, Y. (2009). Online markov decision processes. *Mathematics of Operations Research*, 34(3):726–736.

Gajane, P., Ortner, R., and Auer, P. (2018). A sliding-window algorithm for markov decision processes with arbitrarily changing rewards and transitions. *arXiv preprint arXiv:1805.10066*.

Garivier, A. and Moulines, E. (2011). On Upper-Confidence Bound Policies For Switching Bandit Problems. In *Algorithmic Learning Theory (ALT)*, pages 174–188. PMLR.

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

Jin, C., Allen-Zhu, Z., Bubeck, S., and Jordan, M. I. (2018). Is Q-learning provably efficient? In *Advances in Neural Information Processing Systems*, pages 4863–4873.

Kleinberg, R., Slivkins, A., and Upfal, E. (2019). Bandits and experts in metric spaces. *Journal of the ACM (JACM)*, 66(4):1–77.

Kocsis, L. and Szepesvári, C. (2006). Discounted UCB. In *2nd PASCAL Challenges Workshop*.

Kveton, B. and Theocharous, G. (2012). Kernel-based reinforcement learning on representative states. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Lecarpentier, E. and Rachelson, E. (2019). Non-stationary markov decision processes, a worst-case approach using model-based reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 7214–7223.

Li, Y. and Li, N. (2019). Online learning for markov decision processes in nonstationary environments: A dynamic regret analysis. In *2019 American Control Conference (ACC)*, pages 1232–1237. IEEE.

Lykouris, T., Simchowitz, M., Slivkins, A., and Sun, W. (2019). Corruption robust exploration in episodic reinforcement learning. *arXiv preprint arXiv:1911.08689*.

Neu, G., György, A., Szepesvari, C., and Antos, A. (2013). Online markov decision processes under bandit feedback. *IEEE Transactions on Automatic Control*, 59(3):676–691.

Ormoneit, D. and Sen, Ś. (2002). Kernel-based reinforcement learning. *Machine Learning*, 49(2):161–178.

Ortner, R., Gajane, P., and Auer, P. (2019). Variational regret bounds for reinforcement learning. In *Proceedings of the 35th Conference on Uncertainty in Artificial Intelligence*.

Ortner, R. and Ryabko, D. (2012). Online regret bounds for undiscounted continuous reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1763–1771.

Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Russac, Y., Vernade, C., and Cappé, O. (2019). Weighted linear bandits for non-stationary environments. In *Advances in Neural Information Processing Systems*, pages 12017–12026.

Sinclair, S., Wang, T., Jain, G., Banerjee, S., and Yu, C. (2020). Adaptive discretization for model-based reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 3858–3871.

Sinclair, S. R., Banerjee, S., and Yu, C. L. (2019). Adaptive discretization for episodic reinforcement learning in metric spaces. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 3(3):1–44.

Song, Z. and Sun, W. (2019). Efficient model-free reinforcement learning in metric spaces. *arXiv preprint arXiv:1905.00475*.

Szita, I., Takács, B., and Lörincz, A. (2002). ε-mdps: Learning in varying environments. *Journal of Machine Learning Research*, 3(Aug):145–174.

Yu, J. Y. and Mannor, S. (2009). Online learning in markov decision processes with arbitrarily changing rewards and transitions. In *2009 international conference on game theory for networks*, pages 314–322. IEEE.

Zanette, A. and Brunskill, E. (2019). Tighter problem-dependent regret bounds in reinforcement learning without domain knowledge using value function bounds. In *Proceedings of the 36th International Conference on Machine Learning*.