Reinforcement Learning in Feature Space: Matrix Bandit, Kernels, and Regret Bound

Lin F. Yang 1 Mengdi Wang 2

Abstract

Exploration in reinforcement learning (RL) suffers from the curse of dimensionality when the state-action space is large. A common practice is to parameterize the high-dimensional value and policy functions using given features. However, existing methods either have no theoretical guarantee or suffer a regret that is exponential in the planning horizon \( H \). In this paper, we propose an online RL algorithm, namely the MatrixRL, that leverages ideas from linear bandit to learn a low-dimensional representation of the probability transition model while carefully balancing the exploitation-exploration tradeoff. We show that MatrixRL achieves a regret bound \( O(H^2 \tilde{d} \log T \sqrt{T}) \) where \( \tilde{d} \) is the number of features, independent with the number of state-action pairs. MatrixRL has an equivalent kernelized version, which is able to work with an arbitrary kernel Hilbert space without using explicit features. In this case, the kernelized MatrixRL satisfies a regret bound \( O(H^2 \tilde{d} \log T \sqrt{T}) \), where \( \tilde{d} \) is the effective dimension of the kernel space.

1. Introduction

Reinforcement learning (RL) is about learning to make sequential decisions in an unknown environment through trial and error. It finds wide applications in robotics (Kober et al., 2013), autonomous driving (Shalev-Shwartz et al., 2016), game AI (Silver et al., 2017) and beyond. We consider a basic RL model - the Markov decision process (MDP). In the MDP, an agent at a state \( s \in S \) is able to play an action \( a \in A \), where \( S \) and \( A \) are the state and action spaces. Then the system transitions to another state \( s' \in S \) according to an unknown probability \( P(s' \mid s, a) \), while returning an immediate reward \( r(s, a) \in [0, 1] \). The goal of the agent is to obtain the maximal possible return after playing for a period of time - even though she has no knowledge about the transition probabilities at the beginning.

The performance of a learning algorithm is measured by “regret”. Regret is the difference between the cumulative reward obtained using the best possible policy and the cumulative reward obtained by the learning algorithm. Any algorithm that is capable of learning requires to have a regret sublinear in \( T \), the number of time steps. In the tabular setting where \( S \) and \( A \) are finite sets, there exist algorithms that achieve asymptotic regret \( \sim \text{poly}(|S|, |A|) \cdot \sqrt{T} \) (e.g., (Jaksch et al., 2010; Dann & Brunskill, 2015; Osband & Van Roy, 2016; Osband et al., 2017; Agrawal & Jia, 2017; Azar et al., 2017; Dann et al., 2018; Jin et al., 2018)). However, the aforementioned regret bound depends polynomially on \(|S|\) and \(|A|\), sizes of the state and action space, which can be very large or even infinite. For instance, the game of Go has \( 3^{361} \) unique states, and a robotic arm has infinitely many continuous-valued states. In the most general sense, the regret \( \text{poly}(|S|, |A|) \cdot \sqrt{T} \) is nonimprovable in the worst case (Jaksch et al., 2010). This issue is more generally known as the “curse of dimensionality” of control and dynamic programming (Bellman, 1966).

To tackle the dimensionality, a common practice is to use features to parameterize high-dimensional value and policy functions in compact presentations, with the hope that the features can capture leading structures of the MDP. In fact, there are phenomenal empirical successes of reinforcement learning using explicit features and/or neural networks as implicit features (see e.g., (Mnih et al., 2015)). However, there is a lack of theoretical understanding about using features for exploration in RL and its learning complexity. In this paper, we are interested in the following theoretical question:

How to use features for provably efficient exploration in reinforcement learning?

Furthermore, we consider online RL in a reproducing ker-
Reinforcement Learning in Feature Space

Kernel methods are well known to be powerful to capture nonlinearity and high dimensionality in many machine learning tasks (Shawe-Taylor et al., 2004). We are interested in using kernel methods to capture nonlinearity in the state-transition dynamics of MDP. A kernel space may consist of infinitely many implicit feature functions. We study the following questions: How to use kernels in online reinforcement learning? Can one achieve low regret even though the kernel space is infinite-dimensional? The goal of this paper is to answer the aforementioned questions affirmatively. In particular, we would like to design algorithms that take advantages of given features and kernels to achieve efficient exploration.

1.1. Our Approach and Main Results

Consider episodic reinforcement learning in finite-horizon MDP. The agent learns through episodes, and each episode consists of $H$ time steps. Here $H$ is also called the planning horizon. Let $\phi(\cdot) \in \mathbb{R}^d, \psi(\cdot) \in \mathbb{R}^d$ be given feature functions, where we usually assume $d' = O(d)$. We focus on the case that the probability transition model $P(\cdot | \cdot)$ can be fully embedded in the feature space (Assumption 1), i.e., there exists some core matrix $M^*$ such that

$$ P(\cdot | \cdot) = \phi(\cdot)^T M^* \psi(\cdot). $$

In the kernel setting, this condition is equivalent to that the transition probability model $P$ belongs to the product space of the reproducing kernel spaces. This condition is essentially equivalent to using the features $\phi$ to represent value functions (Parr et al., 2008). When the probability transition model $P$ cannot be fully embedded using $\phi$, then value function approximation using $\phi$ may lead to arbitrarily large Bellman error (Yang & Wang, 2019). Note that the core matrix model is related to many statistic models, e.g., latent state model (Singh et al., 1995), Lumpyable Markov Chain (Weinan et al., 2008), Markov process with rich observations (Azizzadenesheli et al., 2016), and so on. For completeness, we list their relationships to the core matrix model in Section A. Moreover, the importance of similar models have also been emphasized in (Yang & Wang (2019) and (Jin et al., 2019).

We propose an algorithm, which is referred to as MatrixRL, that actively explores the state-action space by estimating the core matrix via ridge regression. The algorithm balances the exploitation-exploration tradeoff by constructing a confidence ball of core matrix for optimistic dynamic programming. It can be thought of as a “matrix bandit” algorithm which generalizes the idea of linear bandit (e.g. (Dani et al., 2008; Li et al., 2010; Chu et al., 2011)). It is proved to achieve the regret bound either

$$ \tilde{O}(H^2 \delta^{3/2} \sqrt{T}) \quad \text{or} \quad \tilde{O}(H^2 d \sqrt{T}). \quad \text{\footnote{\tilde{O}() hides poly log factors of the input.}} $$

depending on regularity properties of the features. MatrixRL can be implemented efficiently in space $O(d^2)$. Each step can be carried out in closed form. Next we extend the MatrixRL to work with the kernel spaces with $k_\phi(\cdot, \cdot) = \langle \phi(\cdot), \phi(\cdot) \rangle$ and $k_\psi(\cdot, \cdot) = \langle \psi(\cdot), \psi(\cdot) \rangle$, and show that it admits a kernelized version. The kernelized MatrixRL achieves a regret bound of

$$ \tilde{O}(H^2 \tilde{d} \sqrt{T}) $$

where $\tilde{d}$ is the effective dimension of kernel space, even if there may be infinitely many features. The regret bounds using features or kernels do not depend on sizes of the state and action spaces, making efficient exploration possible in high dimensions. Very recently, (Du et al., 2019a) show that even if the value-function of the MDP admits an approximate linear representation with features, there is still an $\Omega(2^H)$ regret lower bound. Our results complement this lower bound by showing that feature representations of the transition model effectively reduces the learning complexity.

Note that for linear bandit, the regret lower bound is known to be $\Omega(d \sqrt{T})$ (Dani et al., 2008). Since linear bandit is a special case of RL, our regret bounds match the lower bound up to polylog factors in $d$ and $T$. To our best knowledge, for reinforcement learning using features/kernels, our result gives the first regret bound that is simultaneously near-optimal in time $T$, polynomial in the planning horizon $H$, and the feature dimension $d$.

1.2. Related Literature

In the tabular case where there are finitely many states and actions without any structural knowledge, complexity and regret for RL has been extensively studied. For $H$-horizon episodic RL, efficient methods typically achieve regret that scale asymptotically as $O(\sqrt{HSAT})$ (see for examples (Jaksch et al., 2010; Osband & Van Roy, 2016; Osband et al., 2017; Agrawal & Jia, 2017; Azar et al., 2017; Dann et al., 2018; Jin et al., 2018)). In particular, (Jaksch et al., 2010) provided a regret lower bound $\Omega(\sqrt{HSAT})$ for $H$-horizon MDP. There is also a line of works studying the sample complexity of obtaining a value or policy that is at most $\epsilon$-suboptimal (Kakade, 2003; Strehl et al., 2006; 2009; Szita & Szepesvári, 2010; Lattimore & Hutter, 2014; Azar et al., 2013; Dann & Brunskill, 2015; Sidford et al., 2018). The optimal sample complexity for finding an $\epsilon$-optimal policy is $O(\frac{1}{(1-\gamma)^2 \epsilon^2})$ (Sidford et al., 2018) for a discounted MDP with discount factor $\gamma$. The optimal lower bound has been proven in (Azar et al., 2013).

There is also a line of works on solving MDPs with a function approximation. For instance (Baird, 1995; Tsitsiklis & Van Roy, 1997; Parr et al., 2008; Mnih et al., 2013; 2015; Silver et al., 2017; Yang & Wang, 2019). There
Reinforcement Learning in Feature Space

are also phenomenal empirical successes in deep reinforcement learning as well (e.g., (Silver et al., 2017)). However there are not many works on the regret analysis of RL with function approximators. Very recently, (Azizzadenesheli et al., 2018) studied the regret bound for linear function approximator. However their bound have that can be exponential in $H$. (Chowdhury & Gopalan, 2019) considers the regret bound for kernelized MDP. However, they need a Gaussian process prior and assumes that the reward function $r$ is known. This is in fact without intermediary reward $a$ are not necessarily finite. At any state $s$ there is a set of states $S$ from which the regret bound is linearly depending on the number of states. More recently, there are several works released after the first arXiv version of this paper, e.g. (Du et al., 2019b; Jin et al., 2019; Zanette et al., 2020). They study similar models as ours but with different algorithms and from different perspectives. In particular, (Jin et al., 2019; Zanette et al., 2020) are considering a similar low-rank MDP model. The major distinction of our results to theirs is that ours is a model-based one whereas theirs are model-free. Algorithm-wise, these results together form the major distinction of our results to theirs is that ours is a model-based one whereas theirs are.

In an episodic Markov decision process (MDP for short), there is a set of states $S$ and a set of actions $A$, which are not necessarily finite. At any state $s \in S$, an agent is allowed to play an action $a \in A$. She receives an immediate reward $r(s, a) \in [0, 1]$ after playing $a$ at $s$, the process will transition to the next state $s' \in S$ with probability $P(s' | s, a)$, where $P$ is the collection of transition distributions. After $H$ time steps, the system restarts at a prespecified state $s_0$. The full instance of an MDP can be described by the tuple $M = (S, A, P, r, s_0, H)$. The agent would like to find a policy $\pi : \mathcal{S} \times [H] \rightarrow A$ that maximizes the long-term expected reward starting from every state $s$ and every stage $h \in [H]$, i.e.,

$$V^*_{R}(s) := \mathbb{E} \left[ \sum_{h=0}^{H} r(s^h, \pi_h(s^h)) | s^0 = s \right].$$

We call $V^* : \mathcal{S} \times [H] \rightarrow \mathbb{R}$ the value function of policy $\pi$. A policy $\pi^*$ is said to be optimal if it attains the maximal possible value at every state-stage pair $(s, h)$. We denote $V^*$ as the optimal value function. We also denote the optimal action-value function (or $Q$-function) as

$$Q^*_{R}(s, a) = r(s, a) + \mathbb{E}[\max_{|s, a|} V^*_{R+1}],$$

and $Q^*_{H}(s, a) = r(s, a)$.

In the online RL setting, the learning algorithm interacts with the environment episodically. Each episode starts from state $s_0$ takes $H$ steps to finish. We let $n$ denote the current number of episodes and denote $t = (n - 1)H + h$ the current time step. We equalize $t$ and $(n, h)$ and may switch between the two nations. We use the following definition of regret.

**Definition 1.** Suppose we run algorithm $K$ in the online environment of an MDP $M = (S, A, P, r, s_0, H)$ for $T = NH$ steps. We define the regret for algorithm $K$ as

$$\text{Regret}_K(T) = \mathbb{E}_K \left[ \sum_{n=1}^{N} \left( V^*(s_0) - \sum_{h=1}^{H} r(s_h, a_h) \right) \right],$$

(1)

where $\mathbb{E}_K$ is taken over the random path of states under the control of algorithm $K$.

Throughout this paper, we focus on RL problems where the probability transition kernel $P(\cdot | \cdot)$ can be fully embedded in a given feature space.

**Assumption 1 (Feature Embedding of Transition Model).** For each $(s, a) \in S \times A$, $\tilde{s} \in S$, feature vectors $\phi(s, a) \in \mathbb{R}^d$, $\psi(\tilde{s}) \in \mathbb{R}^d$ are given as a priori. There exists an unknown matrix $M^* \in \mathbb{R}^{d \times d}$ such that

$$P(\tilde{s} | s, a) = \phi(s, a)^\top M^* \psi(\tilde{s}).$$

Here, we call the matrix $M^*$ as a transition core.

Note that when $\phi$, $\psi$ are features associated with two reproducing kernel spaces $H_1$ and $H_2$, this assumption requires that $P$ belong to their product kernel space $H_1 \times H_2$.

For simplicity of representation, we assume throughout that the reward function $r$ is known. This is in fact without
loss of generality because learning about the environment $P$ is much harder than learning about $r$. In the case if $r$ is unknown but satisfies $\mathbb{E}[r(s, a) \mid s, a] = \phi(s, a) \theta^*$ for some unknown $\theta^* \in \mathbb{R}^d$, we can extend our algorithm by adding a step of optimistic reward estimation like in LinUCB (Dani et al., 2008; Chu et al., 2011). This would generate an extra $O(d\sqrt{T})$ regret, which is a low order term compared to our current regret bounds.

3. RL Exploration in Feature Space

In this section, we study the near optimal way to balance exploration and exploitation in RL using a given set of features. We aim to develop an online RL algorithm with regret that depends only on the feature size $d$ but not on the size of the state-action space. Our algorithm is inspired by the LinUCB algorithm (Chu et al., 2011) and its variants (Dani et al., 2008) and can be viewed as a “matrix bandit” method.

3.1. The MatrixRL Algorithm

The high level idea of the algorithm is to approximate the unknown transition core $M^*$ using data that has been collected so far. Suppose at the time step $t = (n, h)$ (i.e. episode $n \leq N$ and stage $h \leq H$), we obtain the following state-action-state transition triplet: $(s_t, a_t, s_{t+1})$, where $s_t := s_{t+1}$. For simplicity, we denote the associated features by

$$\phi_t := \phi(s_t, a_t) \in \mathbb{R}^d \quad \text{and} \quad \psi_t := \psi(s_t) \in \mathbb{R}^d.$$  

**Estimating the core matrix.** Let $K_{\psi} := \sum_{s \in S} \psi(s)\psi(s)^T$, which we assume is given along with the features $\psi$. We then construct our estimator of $M^*$ as:

$$M_n = [A_n]^{-1} \sum_{n' < n, h \leq H} \phi_{n', h} \psi_{n', h}^T K_{\psi}^{-1},  \quad (2)$$

where

$$A_n = I + \sum_{n' < n, h \leq H} \phi_{n', h} \phi_{n', h}^T.$$  

Let us explain the intuition of $M_n$. Note that

$$\mathbb{E} \left[ \phi_{n, h} \psi_{n, h}^T K_{\psi}^{-1} \mid s_n, a_n \right] = \sum_{\tilde{s}} \phi_{n, h} P(\tilde{s} \mid s_n, a_n) \psi(\tilde{s})^T K_{\psi}^{-1}$$

$$= \sum_{\tilde{s}} \phi_{n, h} \phi_{n, h}^T M^* \psi(\tilde{s}) \psi(\tilde{s})^T K_{\psi}^{-1}$$

$$= \phi_{n, h} \phi_{n, h}^T M^*.$$  

Therefore $M_n$ is the approximate solution to the following ridge regression problem:

$$M_n = \arg \min_{M} \sum_{n' < n, h \leq H} \left\| \psi_{n', h}^T K_{\psi}^{-1} - \phi_{n', h}^T M \right\|^2_2 + \| M \|^2_2.  \quad (3)$$

**Upper confidence RL using a matrix ball.** In online RL, a critical step is to estimate future value of the current state and action use dynamic programming. To better balance exploitation and exploration, we use a matrix ball to construct optimistic value function estimator. At episode $n$:

$$w_{n, h+1} = 0 \in \mathbb{R}^d \quad \text{and} \quad \forall h \in [H]: \quad w_{n, h} = \alpha_\psi \cdot \sum_{j \in [m_w]} \frac{\psi(s_j)}{\|\psi(s_j)\|_1} \cdot V_{n, h}(s_j)$$  

(4)

where $\alpha_\psi = \sum_s \|\psi(s)\|_1$ is also given along with $\psi$, $m_w$ is a number to be determined, and $s_j$ are picking from the distribution generated by $p(s) = \|\psi(s)\|_1/\alpha_\psi$, which is also given with $\psi$, and, for all $(s, a)$, we define,

$$V_{n, h}(s) := \Pi[0, H] \left[ \max_a Q_{n, h}(s, a) \right],$$

$$Q_{n, h}(s, a) := r(s, a) + \max_{M \in B_n} \phi(s, a)^T M w_{n, h+1}. \quad (5)$$

Here the matrix ball $B_n = B^{(1)}_n$ is constructed as

$$B^{(1)}_n := \left\{ M \in \mathbb{R}^{d \times d} : \| (A_n)^{1/2} (M - M_n) \|_{2,1} \leq \sqrt{d \beta_n} \right\}$$  

(6)

where $\beta_n$ is a parameter to be determined later, and $\| Y \|_{2,1} = \sum_i \sqrt{\sum_j Y(i, j)^2}$. At time $(n, h)$, suppose the current state is $s_n, h$, we play the optimistic action $a_{n, h} = \arg \max_a Q_{n, h}(s_n, a)$.

**Remark 1** (Computation in Continuous Space). Later we will show that $m_w$ is polynomial in the number of episodes. Consequently, our algorithm computes $w_{n, h}$ efficiently in each round. In the continuous space, the only issue is to solve the max problem in (4). In this case, we assume there exists an optimization oracle that solves $\sup_a Q_{n, h}(s, a)$ in time $O(1)$. Note that the time complexity of the oracle depends strictly on the feature space structure, and there is no general way of expressing it analytically.

The full algorithm is given in Algorithm 1.

3.2. Regret Bounds for MatrixRL

Let $\Psi = [\psi(s_1), \psi(s_2), \ldots, \psi(s_{|S|})]^T \in \mathbb{R}^{S \times d'}$ be the matrix of all $\psi$ features. We first introduce some regularity conditions of the features space.
Reinforcement Learning in Feature Space

Algorithm 1 Upper Confidence Matrix Reinforcement Learning (UC-MatrixRL)

1: **Input:** An episodic MDP environment \( M = (S, A, P, r, s_0, H) \);
2: Features \( \phi: S \times A \rightarrow \mathbb{R}^d \) and \( \psi: S \rightarrow \mathbb{R}^d \);
3: Total number of episodes \( N \);
4: **Initialize:** \( A_1 \leftarrow I \in \mathbb{R}^{d \times d} \), \( M_1 \leftarrow 0 \in \mathbb{R}^{d \times d} \);
5: **for** episode \( n = 1, 2, \ldots, N \) **do**
6: Let \( \{Q_{n,h}\} \) be given in (4) using \( M_n, \beta_n \);
7: **for** stage \( h = 1, 2, \ldots, H \) **do**
8: Let the current state be \( s_{n,h} \);
9: Play action \( a_{n,h} = \arg \max_{a \in A} Q_{n,h}(s_{n,h}, a) \)
10: Record the next state \( s_{n,h+1} \);
11: **end for**
12: \( A_{n+1} \leftarrow A_n + \sum_{h \leq H} \theta_{n,h} \phi_{n,h}^\top \);
13: Compute \( M_{n+1} \) using (2);
14: **end for**

**Assumption 2** (Feature Regularity). Let \( C_M, C_\phi, C_\psi \) and \( C'_\psi \) be positive parameters.

1. \( \|M^*\|_F^2 \leq C_M \cdot d \);
2. \( \forall (s, a) \in S \times A : \|\phi(s, a)\|_2^2 \leq C_\phi d \);
3. \( \forall v \in \mathbb{R}^S : \|\Psi^\top v\|_\infty \leq C_\psi \|v\|_\infty \), and \( \|\Psi K_\psi^{-1}\|_{2,\infty} \leq C'_\psi \).

With these conditions we are ready to provide the regret bound.

**Theorem 1.** Suppose Assumption 1 and Assumption 2 hold, and \( m_w = \text{poly}(T) \). Then after \( T = NH \) steps, Algorithm 1 achieves regret bound:

\[
\text{Regret}(T) \leq O \left[ \sqrt{C_\psi(C_M + C_\psi^2) \ln(C_\phi T)} \right] \cdot H^2 \cdot \sqrt{d^2d'},
\]

if we let \( \beta_n = c \cdot (C_M + C_\psi^2) \cdot \ln(NHC_\phi) \cdot d \), for some absolute constant \( c > 0 \). Moreover, for each new episode, Algorithm 1 takes time \( \text{poly}(T) \) to complete.

Consider the case where \( C_M, C_\phi, C_\psi \) and \( C'_\psi \) are absolute constants. For example, we may let \( \Psi, \Phi, M^* \) be probability matrices. Then Assumption 2 automatically holds with \( C_M = C_\psi = C'_\psi = 1 \). In this case, our regret bound is simply \( O(d^{3/2}H^2 \log(T) \sqrt{T}) \). The \( d \)-dependence in such a regret bound is consistent with the regret of the \( \ell_1 \)-ball algorithm for linear bandit (Dani et al., 2008).

Further, if the feature space \( \Psi \) admits a tighter bound for value function in this space, we can slightly modify our algorithm to achieve sharper regret bound. To do this, we need to slightly change our Assumption 2 to Assumption 2'.

**Assumption 2'** (Stronger Feature Regularity). Let \( C_M, C_\phi, C_\psi \) and \( C'_\psi \) be positive parameters.

1. \( \|M^*\|_F^2 \leq C_M \cdot d \);
2. \( \forall (s, a) \in S \times A : \|\phi(s, a)\|_2^2 \leq C_\phi d \);
3. \( \forall v \in \mathbb{R}^S : \|\Psi^\top v\|_2 \leq C_\psi \|v\|_\infty \), and \( \|\Psi K_\psi^{-1}\|_{2,\infty} \leq C'_\psi \).

We modify the algorithm slightly by using a Frobenious-norm matrix ball instead of the 2-1 norm and computing sharper confidence bounds. Let \( B_n = B_n^{(2)} \) in (4), where

\[
B_n^{(2)} := \left\{ M \in \mathbb{R}^{d \times d'} : \|(A_n)^{1/2}(M - M_n)\|_F \leq \sqrt{\beta_n} \right\}
\]

Then a sharper regret bound can be established.

**Theorem 2.** Suppose Assumption 1 and Assumption 2' hold, and \( m_w = \text{poly}(T) \). Then after \( T = NH \) steps, Algorithm 1, with \( B_n = B_n^{(2)} \) applied in (4), achieves regret

\[
\text{Regret}(T) \leq O \left[ \sqrt{C_\psi(C_M + C_\psi^2) \ln(C_\phi T)} \right] \cdot dH^2 \cdot \sqrt{T},
\]

provided \( \beta_n = c \cdot (C_M + C_\psi^2) \cdot \ln(NHC_\phi) \cdot d \), for some absolute constant \( c > 0 \). Moreover, for each new episode, Algorithm 1 takes time \( \text{poly}(T) \) to complete.

The only stronger condition needed by Assumption 2' is \( \|\Psi^\top v\|_2 \leq C_\psi \|v\|_\infty \). It can be satisfied if \( \Psi \) is a set of sparse features, or if \( \Psi \) is a set of highly concentrated features. Note that a key difference of our analysis to that of (Jin et al., 2019) is that they apply an \( \epsilon \)-net argument to bound the regret. Such an union bound gives a regret bound proportional to \( \sqrt{d^3} \). In contrast, we use a model-based method to decouple the dependence between samples and apply a martingale concentration argument, which saves an additional \( d \) factor under Assumption 2'.

We remark that in Theorem 1 and Theorem 2, we need to know the value \( N \) in \( \beta_n \) before the algorithm runs. In the
Reinforcement Learning in Feature Space

case when \( N \) is unknown, one can use the doubling trick to learn \( N \) adaptively: first we run the algorithm by picking \( N = 2 \), then for \( N = 4, 8, \ldots, 2^m \) until the true \( N \) is reached. It is standard knowledge that this trick increase the overall regret by only a constant factor (e.g. (Besson & Kaufmann, 2018)).

**Proof Sketch.** The proof consists of two parts. We show that when the core matrix \( M^* \) belongs to the sequence of constructed balls \( \{ B_n \} \), the estimated Q-functions provide optimistic estimates of the optimal values, therefore the algorithm’s regret can be bounded using the sum of confidence bounds on the sample path. The second part constructs a martingale difference sequence by decomposing a matrix into an iterative sum and uses a probabilistic concentration argument to show that the “good” event happens with sufficiently high probability. In the next section, we present the full proof for Theorems 1. The full proof of Theorems 2 is deferred to the appendix.

**Near optimality of regret bounds?** The regret bound in Theorem 2 matches the optimal regret bound \( \tilde{O}(d\sqrt{T}) \) for linear bandit (Dani et al., 2008). In fact, linear bandit is a special case of RL: the planning horizon \( H \) is 1. However the hard instance for linear bandit may not be necessarily constructable using the core matrix model.

**Closed-form confidence bounds.** Equation (4) requires maximization over a matrix ball. However, it is not necessary to solve this maximization problem explicitly. The algorithm only requires an optimistic Q value. In fact, we can use a closed-form confidence bound instead of searching for the optimal \( M \) in the confidence ball. It can be verified that Theorem 1 still holds (by following the same proofs of the theorem) if we replace the second equation of (4) as the following equation (see the proof of Theorem 2)

\[
\forall h \in [H] : Q_{n,h}(s,a) = r(s,a) + \phi(s,a)^\top M_n \Psi V_{n,h+1} + 2C_V \sqrt{d/\beta_n \cdot w_{n,h}},
\]

(8)

where \( w_{n,h} := \sqrt{\phi_{n,h}^\top (A_n)^{-1} \phi_{n,h}} \). Similarly, Theorem 2 still holds if we replace the second equation of 4 with

\[
\forall h \in [H] : Q_{n,h}(s,a) = r(s,a) + \phi(s,a)^\top M_n \Psi V_{n,h+1} + 2C_V \sqrt{\beta_n \cdot w_{n,h}},
\]

(9)

Equations (8) and (9) can be computed easily. They can be viewed as the “dualization” of (4).

**3.3. Analysis and Proofs**

In this section we will focus on proving Theorem 1. In the proof we will also establish all the necessary analytical tools for proving Theorem 2 and Theorem 9. We provide the proofs of the last two theorems in the appendix.

To begin, we first investigate the Monte-Carlo step used in (4). For each step, we let \( \bar{w}_{n,h} = \Psi^\top V_{n,h} \). For \( w_{n,h} \), we have the following guarantee, which is a simple consequence of Hoeffding's inequality.

**Proposition 1.** With probability at least \( 1 - 1/\delta \), we have

\[
\|\bar{w}_{n,h} - w_{n,h}\|_\infty \leq C \cdot \sqrt{\log \frac{1}{\delta} / m_w}
\]

for some generic constant \( C \) depends on the properties of \( \Psi \).

Notice that the randomness used in computing the above equation is independent with that from other parts of the algorithm, and can the error can be made arbitrarily small by picking \( m_w = \text{poly}(T) \). The regret incurred by this Monte-Carlo step can be at most

\[
T \cdot \frac{1}{\sqrt{m_w}} \ll \sqrt{T}.
\]

For the sake of presentation, we assume \( w_{n,H} = \bar{w}_{n,h} \) in the rest of the proof.

The proof of Theorem 1 consists of two steps: (a) We first show that if the true transition core \( M^* \) is always in the confidence ball \( B_n \), defined in Equation 6, we can then achieve the desired regret bound; (b) We then show that with high probability, the event required by (a) happens. We formalize the event required by step (a) as follows.

**Definition 2 (Good Estimator Event).** For all \( n \in [N] \), we denote \( E_n = 1 \) if \( M^* \in B_n \), for all \( n' \in [n] \) and otherwise \( E_n = 0 \).

Note that \( E_n \) is completely determined by the game history up to episode \( n \). In Section B.2, we show that \( E_n = 1 \) happens with high probability for all \( n \). In the next subsection, we show (a).

**3.4. Regret Under Good Event**

To better investigate the regret formulation (1), we rewrite it according to Algorithm 1. Note that conditioning on the history before episode \( n \), the algorithm plays a fixed policy \( \pi_n \) for episode \( n \). Therefore, we have

\[
\text{Regret}(NH) = \sum_{n=1}^{N} \mathbb{E}[R(n)],
\]

(10)

where \( R(n) = V_1^*(s_0) - V_1^{\pi_n}(s_0) \). We now show that the algorithm always plays an optimistic action (an action with value estimated greater than the optimal value of the state).

All missing proofs can be found in Section B.1.
Lemma 3 (Optimism). Suppose for \( n \in [N] \), we have the good estimator event, \( E_n = 1 \), happens. Then for \( h \in [H] \) and \((s, a) \in S \times A\), we have
\[
Q^*_h(s, a) \leq Q_{n,h}(s, a).
\]

Next we show that the confidence ball \( B_n \) actually gives a strong upper bound for the estimation error: the estimation error is “along” the direction of the exploration.

Lemma 4. For any \( M \in B_n \), we have
\[
\|\phi_{s,a}^T(M - M_n)\|_1 \leq \sqrt{d' \beta_n \cdot \phi(s, a)\sp{T}(A_n)^{-1} \phi(s, a)}.
\]

Proof.
\[
\|\phi_{s,a}^T(M - M_n)\|_1 = \|\phi_{s,a}^T(A_n)^{-1/2}(A_n)^{1/2}(M - M_n)\|_1 \\
\leq \|\phi_{s,a}^T(A_n)^{-1/2}\|_2 \cdot \|(A_n)^{1/2}(M - M_n)\|_2,1 \\
\leq \sqrt{d' \beta_n} \cdot \|\phi_{s,a}^T(A_n)^{-1/2}\|_2
\]
as desired.

We are now ready to show the regret per-step does not introduce too much error.

Lemma 5. Suppose for \( n \in [N] \), \( E_n = 1 \). Then for \( h \in [H] \), we have
\[
Q_{n,h}(s_{n,h}, a_{n,h}) - \left[r(s_{n,h}, a_{n,h}) + P(\cdot | s_{n,h}, a_{n,h})^T V_{n,h+1}\right] \\
\leq 2C_\psi H \sqrt{d' \beta_n} \cdot w_{n,h}
\]
where
\[
w_{n,h} := \sqrt{\phi_{n,h}^T(A_n)^{-1} \phi_{n,h}}. \quad (11)
\]

We are now ready to show the regret bound, whose proof is presented in the appendix.

Lemma 6. Suppose Assumption 2 holds, \( 1 \leq \beta_1 \leq \beta_2 \leq \ldots \beta_N \), then,
\[
\text{Regret}(N H) \leq 2C_\psi H \sqrt{d' \beta_N} \\
\cdot \mathbb{E} \left[ \sum_{n=1}^{N} \sum_{h=1}^{H} \sqrt{\min(1, w_{n,h}^2)} \right] + \sum_{n=1}^{N} H \mathbb{P}[E_n \neq 1]
\]
It remains to bound
\[
\sum_{n=1}^{N} \sum_{h=1}^{H} \sqrt{\min(1, w_{n,h}^2)} \leq \sqrt{H N} \sum_{n=1}^{N} \sum_{h=1}^{H} \min(1, w_{n,h}^2).
\]

Lemma 7.
\[
\sum_{n=1}^{N} \sum_{h=1}^{H} \min(1, w_{n,h}^2) \leq 2H \ln \det(A_{N+1}) \\
\leq 2H d \ln(NHC_\phi + 1).
\]

The proof of this lemma is rather technical and requires some new notations, we postpone it to Section B.3. We are now ready to state the regret bound.

Lemma 8. Suppose \( 1 \leq \beta_1 \leq \beta_2 \leq \ldots \beta_N \) and \( C_\psi \geq 1 \), then
\[
\text{Regret}(T) \leq 2C_\psi H \sqrt{d' \beta_N} \\
\cdot \sqrt{H N} \cdot 2H d \cdot \ln[NHC_\phi + 1] \\
+ \sum_{n=1}^{N} H \mathbb{P}[E_n = 0].
\]

We are now ready to prove Theorem 1.

Proof of Theorem 1. We let \( \delta \leq 1/(NH) \). By Lemma 13 and 15, we pick
\[
\beta_n \geq cd \cdot [C_M + C_\psi^2 \cdot \ln(nHC_\phi) + c \cdot C_\psi^2 \cdot \ln(nHC_\phi/d)] \\
= \Theta(C_M + C_\psi^2 \cdot \ln(nHC_\phi) \cdot d).
\]
Then the following is guaranteed,
\[
\Pr[\forall n \leq N : E_n = 1] \geq 1 - \delta.
\]

Then by Lemma 8, we have,
\[
\text{Regret}(T) \leq 2C_\psi H \sqrt{d' \beta_N} \\
\cdot \sqrt{H N} \cdot 2H d \cdot \ln[NHC_\phi + 1] + O(1) \\
= O\left[C_\psi \sqrt{C_M + C_\psi^2 \cdot \ln(NHC_\phi)} \cdot \sqrt{d' d'' H T} \right].
\]

4. RL Exploration in Kernel Space

In this section, we transform MatrixRL to work with kernels instead of explicitly given features. Suppose we are given two reproducing kernel Hilbert spaces \( \mathcal{H}_\phi, \mathcal{H}_\psi \) with kernel functions \( k_\phi : (S \times A) \times (S \times A) \to \mathbb{R} \) and \( k_\psi : S \times S \to \mathbb{R} \), respectively. There exist implicit features \( \phi, \psi \) such that \( k_\phi(x, y) = \phi(x)^T \phi(y) \) and \( k_\psi(x, y) = \psi(x)^T \psi(y) \), but the learning algorithm can access the kernel functions only.

4.1. Kernelization of MatrixRL

The high level idea of Kernelized MatrixRL is to represent all the features used in Algorithm 1 with their corresponding kernel representations. We first introduce some notations. For episode \( n \) and \( T = nH \), we denote \( K_{\phi_n} \in \mathbb{R}^{S \times S \times A \times A} \).
Reinforcement Learning in Feature Space

\( \mathbb{R}^{n \times T} \) and \( \mathbf{K}_{\psi n} \in \mathbb{R}^{n \times T} \) as the Gram matrix, respectively, i.e., for all \( t_1 = (n_1, \psi_1), t_2 = (n_2, \psi_2) \in [n] \times [H] \),

\[
\begin{align*}
\mathbf{K}_{\phi n}[t_1, t_2] &= k_{\phi}([s_{t_1}, a_{t_1}], [s_{t_2}, a_{t_2}]), \\
\mathbf{K}_{\psi n}[t_1, t_2] &= k_{\psi}(\tilde{s}_{t_1}, \tilde{s}_{t_2}),
\end{align*}
\]

where \( \tilde{s}_t := s_{t+1} \). We denote \( \mathbf{K}_{\psi n} \in \mathbb{R}^{T \times |S|} \) and \( k_{\psi n, s, a} \in \mathbb{R}^T \) by

\[
\begin{align*}
\mathbf{K}_{\psi n}[t, s] &= k_{\psi}(\tilde{s}_{t}, s), \\
k_{\psi n, s, a}[t] &= k_{\phi}([s_t, a_t], (s, a)).
\end{align*}
\]

We are now ready to kernelize Algorithm 1. The full algorithm of Kernelized MatrixRL is given in Algorithm 2. Note that the new Q function estimator (12) is the dualization form of (4). Therefore Algorithm 2 is more general but essentially equivalent to Algorithm 1 if we let \( k_{\phi}(x, y) := \phi(x)\phi(y) \) and \( k_{\psi}(x, y) := \psi(x)^T \psi(y) \).

Note that the computation of \( (\lambda I + \mathbf{K}_{\phi n-1}^{1\dagger} \mathbf{K}_{\psi n-1}^{1\dagger})^{-1} \mathbf{K}_{\psi n} \) can again be performed using Monte-Carlo efficiently (as in (4)). For the sake of representation, we assume we have an exact computation. See Section C for the proof.

4.2. Regret Bound for Kernelized MatrixRL

We define the effective dimension of the kernel space \( \mathcal{H}_\phi \) as

\[
\tilde{d} = \sup_{t \leq H} \sup_{X \subseteq S \times A} \log |X| \cdot \log(1 + t),
\]

where \( X = \{x_j\}_{j \in [t]} \), \( \mathbf{K}_X \in \mathbb{R}^{t \times t} \) with \( \mathbf{K}_X[i, j] = k_{\phi}(x_i, x_j) \) is the Gram matrix over data set \( X \). Note that \( \tilde{d} \) captures the effective dimension of the space spanned by the features of state-action pairs. Consider the case when \( \phi \) are \( d \)-dimensional unit vectors, then \( \mathcal{H}_\phi \) has dimension at most \( d \). It can be verified that \( \tilde{d} \leq d \). A similar notion of effective dimension was introduced by (Valko et al., 2013) for analyzing kernelized contextual bandit.

Further, we need regularity assumptions for the kernel space.

**Assumption 3.** Let \( \mathcal{H}_\psi \) be generated by orthonormal basis on \( S \), i.e., there exists \( \psi \) such that \( \sum_{s \in S} \psi(s)\psi(s)^T = I \) and \( k_\psi(s, s') = \psi(s)^T \psi(s') \). There exists a constant \( C_\psi \) such that for all \( v \in \mathcal{H}_\psi \), \( \|v\|_{\mathcal{H}_\psi} \leq C_\psi \|v\|_{\mathcal{H}_\psi} \), where \( \| \cdot \|_{\mathcal{H}_\psi} \) denotes the Hilbert space norm.

The formal guarantee of Kernelized MatrixRL is presented as follows.

**Theorem 9.** Suppose the probability transition kernel \( P(\cdot | \cdot) \) belongs to the product Hilbert spaces, i.e., \( P \in \mathcal{H}_\phi \times \mathcal{H}_\psi \). Let Assumption 3 hold. Then after \( T = NH \) time steps, the regret of Algorithm 2 satisfies

\[
\text{Regret}(T) \leq O\left(C_\psi \cdot P\|\mathcal{H}_\phi \times \mathcal{H}_\psi\| \cdot \log(T) \cdot \tilde{d} \cdot H^2 \cdot \sqrt{T}\right)
\]

provided \( \eta_n = 2C_\psi H \sqrt{T \ln N} \) and \( \beta_n = \Theta(P\|\mathcal{H}_\phi \times \mathcal{H}_\psi \| \cdot \ln(NH) \cdot \tilde{d}) \).

Note that in Assumption 3, we can additionally relax the assumption on the orthogonality of \( \psi \). Similar regret bound can be proved with Assumption 2'. The proof of Theorem 9 is very similar to that of Theorem 2. Although Kernelized MatrixRL does not access the features, the proof is based on the underlying features and the equivalence between kernel representation and feature representation. We postpone it to Section C.

**Remark.** Similar as MatrixRL, Kernelized MatrixRL can be generalized to deal with unknown reward function by using the Kernelized Bandit (Valko et al., 2013). Again, since linear bandit problems are special cases of kernel RL with \( H = 1 \), our results match the linear bandit bound on \( \tilde{d} \) and \( \sqrt{T} \). The computation time of Kernelized MatrixRL scales with time \( T \) as \( T^2 \) (by applying randomized algorithms, e.g. (Dani et al., 2008), in dealing with \( \mathbf{K}_\psi \) matrices), still polynomial in \( T \). We can apply the random features or sketching techniques for kernel to additionally accelerate the computation (e.g. (Rahimi & Recht, 2008; Yang et al., 2017)).

5. Summary

This paper provided the algorithm MatrixRL for episodic reinforcement learning in high dimensions. It also provides the first regret bounds that are near-optimal in time \( T \) and feature dimension \( d \) and polynomial in the planning horizon \( H \). MatrixRL uses given features (or kernels) to estimate a core transition matrix and its confidence ball, which is used to compute optimistic Q-functions for balancing the exploitation-exploration tradeoff. We prove that the regret of MatrixRL is bounded by \( O(H^2d \log T \sqrt{T}) \) where \( d \) is the number of features, provided that the feature space satisfies some regularity conditions. MatrixRL has an equivalent kernel version, which does not require explicit features. The kernelized MatrixRL satisfies a regret bound \( O(H^2d \log T \sqrt{T}) \), where \( \tilde{d} \) is the effective dimension of the kernel space. For future work, it remains open if the regularity condition can be relaxed and if there is a more efficient way for constructing confidence balls in order to further reduce the regret.

**Acknowledgment**

Mengdi Wang gratefully acknowledges funding from the U.S. National Science Foundation (NSF) grant CMMI-1653435, Air Force Office of Scientific Research (AFOSR) grant FA9550-19-1-020, and C3.ai DTI.
Algorithm 2 KernelMatrixRL: Reinforcement Learning with Kernels

1: Input: An episodic MDP environment $M = (\mathcal{S}, \mathcal{A}, P, s_0, r, H)$, kernel functions $k_\phi, k_\psi$;
2: Total number of episodes $N$;
3: Initialize: empty reply buffer $B = \{\}$;
4: for episode $n = 1, 2, \ldots, N$ do
5: For $(s, a) \in \mathcal{S} \times \mathcal{A}$, let
$$w_n(s, a) := \sqrt{k_\phi[(s, a), (s, a)] - k_\phi^T \Phi_n^{-1} s, a - k_\phi^T \Phi_n^{-1} I + K \Phi_n^{-1} s, a};$$
$$x_n(s, a) := k_{\phi_n}^T (I + K \Phi_n^{-1} \lambda I + K \Psi_n^{-1})^{-1} K \Psi_n^{-1};$$
where $\lambda > 0$ is a parameter to be adjusted;
6: Let $\{Q_{n,h}\}$ be defined as follows:
$$\forall (s, a) \in \mathcal{S} \times \mathcal{A}: \quad Q_{n,H+1}(s, a) := 0 \quad \text{and} \quad \forall h \in [H]: \quad Q_{n,h}(s, a) := r(s, a) + x_n(s, a)^T V_{n,h+1} + \eta_n w_n(s, a),$$
where
$$V_{n,h}(s) = \Pi_{[0, H]}[\max_a Q_{n,h}(s, a)] \quad \forall s, a, n, h;$$
and $\eta_n$ is a parameter to be determined;
7: for stage $h = 1, 2, \ldots, H$ do
8: Let the current state be $s_{n,h}$;
9: Play action $a_{n,h} = \arg \max_{a \in \mathcal{A}} Q_{n,h}(s_{n,h}, a);$
10: Record the next state $s_{n,h+1}$: $B \leftarrow B \cup \{s_{n,h}, a_{n,h}, s_{n,h+1}\};$
11: end for
12: end for
References


Reinforcement Learning in Feature Space


Valko, M., Korda, N., Munos, R., Flaounas, I., and Cristianini, N. Finite-time analysis of kernelised contextual...