A Tale of Two Efficient Value Iteration Algorithms for Solving Linear MDPs with Large Action Space

Zhaozhuo Xu
Rice University

Zhao Song
Adobe Research

Anshumali Shrivastava
Rice University

Abstract

Markov Decision Process (MDP) with large action space naturally occurs in many applications such as language processing, information retrieval, and recommendation system. There have been various approaches to solve these MDPs through value iteration (VI). Unfortunately, all VI algorithms require expensive linear scans over the entire action space for value function estimation during each iteration. To this end, we present two provable Least-Squares Value Iteration (LSVI) algorithms with runtime complexity sublinear in the number of actions for linear MDPs. We formulate the value function estimation procedure in VI as an approximate maximum inner product search problem and propose a Locality Sensitive Hashing (LSH) type data structure to solve this problem with sublinear time complexity. Our major contribution is combining the guarantees of approximate maximum inner product search with the regret analysis of reinforcement learning. We prove that, with the appropriate choice of approximation factor, there exists a sweet spot. Our proposed Sublinear LSVI algorithms maintain the same regret as the original LSVI algorithms while reducing the runtime complexity to sublinear in the number of actions. To the best of our knowledge, this is the first work that combines LSH with reinforcement learning that results in provable improvements. We hope that our novel way of combining data structures and the iterative algorithm will open the door for further study into the cost reduction in reinforcement learning.

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206.

1 INTRODUCTION

Reinforcement learning (RL) is an essential problem in machine learning that targets maximizing the cumulative reward when an agent is taking actions within an unknown environment (Sutton and Barto, 2018). RL has been a trending topic over the last few years. We have seen a remarkable growth of RL applications in Go (Silver et al., 2016), robotics (Kober et al., 2013), dialogue systems (Li et al., 2016) and recommendation (Zheng et al., 2018). In practical RL, most approaches (Watkins and Dayan, 1992; Silver et al., 2014; Jin et al., 2018) formulate the problem as a Markov Decision Process (MDP). Next, they propose iterative-type MDP solvers that modify the choice of actions at each step based on the agent interaction with the environment. This iterative nature causes the training of RL algorithms to be expensive. For instance, it takes around three weeks to train the agent in AlphaGo (Silver et al., 2016). Moreover, the training is conducted on 50 GPUs, which means the training of RL on democratized computational resources is almost impossible. This expensive training cost is even exaggerated in settings with large action space. For instance, in news recommendation systems, the action space is the billion-scale articles on the web. In dialogue system, the action space is the vocabulary of English or French. With giant number of actions as choices, even a linear scan over the action space would be extremely expensive. Therefore, current MDP solvers cannot provide real-time service in practice.

Given the efficiency bottleneck of RL algorithms, it is natural to ask the following question.

Is there any theoretical computer science (TCS) technique that could improve the running time efficiency of iterative-type MDP solvers with large action space?

The practical success of a typical TCS technique, Locality Sensitive Hashing (LSH), shed lights on answering the question. LSH is a randomized data structure with provable efficiency in approximate nearest neighbor search (ANN) (Indyk and Motwani, 1998; Charikar, 2002; Datar et al., 2004; Shakhnarovich et al., 2005; Andoni and Indyk, 2008; Andoni, 2009; Andoni et al., 2014; Andoni and Razenshteyn,

2015; Andoni et al., 2015, 2017b; Christiani, 2017; Razenshteyn, 2017; Andoni et al., 2018; Wei, 2019; Dong et al., 2020). Meanwhile, LSH could also be extended to maximum inner product search (Max-IP) (Shrivastava and Li, 2014). Moreover, in practical machine learning (ML), LSH has been widely used in many fundamental learning problems to improve the practical running time of iterative-type algorithms such as gradient descents (Chen et al., 2019), back-propagation (Chen et al., 2020; Daghaghi et al., 2021; Chen et al., 2021) and MCMC sampling (Luo and Shrivastava, 2019). However, the current empirical combination of LSH with iterative-type algorithms does not have theoretical support. It is unknown to give a provable guarantee for the impact of LSH over the total number of iterations and per cost iteration of iterative-type algorithms.

Inspired by a large number of successes about using LSH to tackle efficiency bottlenecks in practice, it is natural to ask the following question.

Is there an interesting regime (e.g., some iterative-type MDP solvers) where we can apply LSH *to give provable improvement?*

In this work, we answer both questions by proposing a theoretical framework that combines LSH with MDP solver. We focus on value iteration (VI) (Sutton and Barto, 2018; Bradtke and Barto, 1996), a simple and flexible type of MDP solvers that directly optimizes the maximum expected reward based on the outcome of actions that the agent taken at each step. Specifically, we study the VI algorithms for linear MDP, where the reward is a linear function of the embedding of state-action pair. In this setting, (Jin et al., 2020) have provided theoretical guarantees for VI algorithms. However, the running time efficiency of VI in linear MDP requires improvement in practical scenarios. We identify that the runtime complexity of VI in linear MDP is dominated by the value function estimation procedure. Value function estimation requires a linear scan over all the actions at each step, which is unscalable in real RL tasks with large action space. For instance, in news recommendation systems, the action of an RL agent is recommending an article to the users. The iterative-type VI algorithm scan over all articles at each iteration to find the action that maximizes the expected reward. In practice, the scale of this search space is in billions, so that linear scan is prohibitive. Therefore, reducing the enormous overhead in value function estimation over the large action space becomes a significant research problem in VI.

We focus on applying LSH techniques to reduce this value function estimation overhead in the iterative-type VI algorithm. However, combing LSH with any iterative-type VI algorithm for linear MDP is challenging due to four major reasons: (1) It remains unknown whether the linear scan over all possible actions in VI could be formulated as an ANN or Max-IP problem (2) LSH accelerate this linear scan by introducing an error in estimating value function. This approximation error would accumulate in the value iteration and break the current upper bound for regret. (3) Although LSH has demonstrated success in practical machine learning, its theoretical efficiency guarantee in RL remains unknown. (4) The VI algorithm would query LSH at each step. As the query in each step depends on the previous step, the total failure probability of LSH over this adaptive query sequence could not be union bounded due to correlations.

In this work, we solve these challenges affirmatively by presenting a VI algorithm for linear MDP that uses LSH type approximate Max-IP data structure. We focus on the Least-Squares Value Iteration (LSVI) (Bradtke and Barto, 1996) and its extensions with Upper Confidence Bound (UCB) exploration (LSVI-UCB (Jin et al., 2020)). These are two typical VI algorithms with theoretical foundations as well as practical insights to various RL settings (Gao et al., 2021; Wang et al., 2020a). We connect the theory of Max-IP with reinforcement learning by formulating the value function estimation in LSVI and LSVI-UCB as an approximate Max-IP problem. Then, we propose Sublinear LSVI and Sublinear LSVI-UCB, two algorithms with LSH that have value iteration running time sublinear in the number of actions. For LSVI-UCB, we extend the LSH type Max-IP data structure to approximate maximum matrix norm search so that Sublinear LSVI-UCB could also enjoy the sublinear value iteration complexity over actions. Moreover, we theoretically prove that, with our choice of approximation factor, both Sublinear LSVI and Sublinear LSVI-UCB achieve the same regret with their original versions. Furthermore, we identify the potential risks of LSH type approximate Max-IP data structure in iterative-type algorithm and propose a series of techniques to reduce them.

2 RELATED WORK

Approximate Maximum Inner Product Search *Maximum Inner Product Search* (Max-IP) is a fundamental yet challenging problem in theoretical computer science (Williams, 2005; Abboud et al., 2017; Chen, 2018; Chen and Williams, 2019; Williams, 2018). Given a query $x \in \mathbb{R}^d$ and a dataset $Y \subset \mathbb{R}^d$ with n vectors, the goal of Max-IP is to retrieve a $z \in Y$ so that $x^\top z = \arg\max_{z \in Y} x^\top y$. The brute-force algorithm solves Max-IP in $O(dn)$ time for x by linear scanning over all elements in Y. To improve the Max-IP efficiency in practice, approximation methods are proposed to achieve sublinear query time complexity by returning point with a multiplicative approximation ratio to the Max-IP solution.

Chen (Chen, 2018) show that for bichromatic Max-IP[i] with two sets of n vectors from $\{0,1\}^d$, there is a $n^{2-\Omega(1)}$ time al-

[i]Given two n-point set $A \in \mathbb{R}^d$ and $B \in \mathbb{R}^d$, the goal of bichromatic Max-IP is to find $b \in B$ that maximize inner product for every $a \in A$.

gorithm with $(d/\log n)^{\Omega(1)}$ approximation ratio. Moreover, Chen (Chen, 2018) show that this algorithm is conditionally optimal as such a $(d/\log n)^{o(1)}$ approximation algorithm would refute Strong Exponential Time Hypothesis (SETH) (Impagliazzo and Paturi, 2001)[ii].

Most previous approximate Max-IP approaches reduce the Max-IP to nearest neighbor (NN) search problem and apply approximate nearest neighbor (ANN) data structures such as Locality Sensitive Hashing (LSH) (Shrivastava and Li, 2014, 2015a; Neyshabur and Srebro, 2015; Shrivastava and Li, 2015b; Yan et al., 2018). Given a query $x \in \mathbb{R}^d$ and a dataset $Y \subset \mathbb{R}^d$ with n vectors, the goal of $(\overline{c}, r)$-ANN with $\overline{c} > 1$ is to retrieve a $z \in Y$ so that $\|x - z\|_2 \leq \overline{c} \cdot r$ if there $\min_{y \in Y} \|x - y\|_2 \leq r$. The LSH solves this problem with query time in $O(d \cdot n^{\rho + o(1)})$. Here, $\rho < 1$ and it depends on $\overline{c}$. For randomized LSH that is independent of data, Antoni, Indyk and Razenshteyn (Andoni et al., 2018) show that $\rho \geq 1/\overline{c}^2$. To further reduce ρ, Antoni and Razenshteyn (Andoni and Razenshteyn, 2015) proposes a data-dependent LSH that achieves $\rho = 1/(2\overline{c}^2 - 1)$ with preprocessing time and space in $O(n^{1+\rho} + dn)$. Andoni, Laarhoven, Razenshteyn and Waingarten (Andoni et al., 2017a) propose a improved proposes a data-dependent LSH that solves $(\overline{c}, r)$-ANN with query time $O(d \cdot n^{\rho_q + o(1)})$, space $O(n^{1+\rho_u + o(1)} + dn)$ and preprocessing time $O(dn^{1+\rho_u + o(1)})$. Andoni, Laarhoven, Razenshteyn and Waingarten (Andoni et al., 2017a) also states that for $\overline{c} > 1$, $r > 0$, $\rho_u \geq 0$ and $\rho_q \geq 0$, we have $\overline{c}^2\sqrt{\rho_q} + (\overline{c}^2 - 1)\sqrt{\rho_u} \geq \sqrt{2\overline{c}^2 - 1}$. Moreover, if we achieve $\rho_u = 0$, we could reduce the preprocessing overhead to $O(n^{1+o(1)} + dn)$ while achieving $\rho_q = \frac{2}{\overline{c}^2} - \frac{1}{\overline{c}^4}$. These LSH approaches have concise theoretical guarantees on the trade-off between search quality and query time for Max-IP.

Meanwhile, other non-reduction approximate Max-IP approaches build efficient data structures such as quantization codebooks (Guo et al., 2016, 2020), trees (Yu et al., 2017), alias tables (Ding et al., 2019) and graphs (Morozov and Babenko, 2018; Zhou et al., 2019; Tan et al., 2019). However, there exists few theoretical guarantee on these non-reduction approaches so that their evaluation is totally empirical.

Locality Sensitive Hashing Applications In practice, well-implemented LSH algorithms are developed (Lv et al., 2007; Andoni et al., 2015) and have demonstrated their superiority in tackling efficiency bottlenecks in practical applications. In optimization, Chen et al. (2019) proposes a LSH based approach to estimate gradients in large scale linear models. Xu et al. (2021) proposes a LSH based Frank-Wolfe algorithm that improves the running time over some well-known conditional gradient methods. Moreover, this idea has been extended to neural network training (Chen et al., 2020, 2021). Further more, Besides deep learning, Luo and Shrivastava (2019) also proposes a LSH method for efficient MCMC sampling. Charikar and Siminelakis (2017); Backurs et al. (2018); Siminelakis et al. (2019); Backurs et al. (2019); Charikar et al. (2020) use LSH for efficient kernel density estimation. Zandieh et al. (2020) proposes a LSH based approach for kernel ridge regression. Yang et al. (2021) proposes an LSH algorithm for efficient linear bandits. Li and Li (2021) and Li et al. (2021) demonstrate how to use LSH and improve the efficiency of large scale statistical estimation. Coleman et al. (2022) presents the strengh of LSH in large scale specie classification on genomic sequence streams.

Provable Efficient Reinforcement Learning The theoretical analysis on the efficiency of modern reinforcement learning (RL) approaches has drawn a lot of attention recently (Jin et al., 2018; Bai et al., 2019; Song and Sun, 2019; Jin et al., 2020; Yang and Wang, 2020; Cai et al., 2020; Wang et al., 2020b; Zhang et al., 2020; Wang et al., 2020a; Du et al., 2020; Feng et al., 2020; Du et al., 2021; Xiong et al., 2021). Jin et al. (2018) presents the first Q-learning with UCB exploration algorithm with provable sublinear regret. Jin et al. (2020) proposes a provable RL algorithm namely LSVI-UCB with linear function approximation that achieves both polynomial runtime and polynomial sample complexity. Gao et al. (2021) extends the LSVI-UCB proposed in Jin et al. (2020) with policy switch limitation. Wang et al. (2020a) studies the model-free version of LSVI-UCB. There also exist other works that benefit the community with theoretical analysis on efficient RL (Du et al., 2020; Yang and Wang, 2020; Cai et al., 2020).

Speedup Cost Per Iteration Recently, there have been many works discussing how to improve the cost per iteration for optimization problems (e.g., linear programming, cutting plane method, maximum matching, training neural networks) while maintaining the total number of iterations in achieving the same final error guarantees. However, most of these algorithms are built on sketching (Lee et al., 2019; Jiang et al., 2020, 2021; Song et al., 2021; Song and Yu, 2021; Brand et al., 2021), sampling (Cohen et al., 2019; Brand et al., 2020b; Dong et al., 2021), vector-maintenance (Brand, 2020; Jiang et al., 2021), sparse recovery (Brand et al., 2020b,a) techniques, none of them have used LSH. We hope that our novel combination of data structures and iterative algorithms will open the door for further study into cost reduction in optimization.

3 BACKGROUND

In this section, we describe the background knowledge in both LSH data structures and RL.

[ii] SETH (Strong Exponential Time Hypothesis) states that for every $\epsilon > 0$ there is a k such that k-SAT cannot be solved in $O((2-\epsilon)^n)$ time.

3.1 Locality Sensitive Hashing

We present a well-known data structure called Locality Sensitive Hashing (Indyk and Motwani, 1998) for approximate nearest neighbor search and approximate maximum inner produce search.

Definition 3.1 (Locality Sensitive Hashing)**.** *Let $\overline{c}$ denote a parameter such that $\overline{c} > 1$. Let r denote a parameter. Let p_1, p_2 denote two parameters such that $0 < p_2 < p_1 < 1$. A function family $\mathcal{H}$ is $(r, \overline{c} \cdot r, p_1, p_2)$-sensitive if and only if, for any two vector $x, y \in \mathbb{R}^d$, a function h chosen uniformly from family $\mathcal{H}$ has the following properties:*

- *if $\|x - y\|_2 \leq r$, then $\Pr_{h \sim \mathcal{H}}[h(x) = h(y)] \geq p_1$,*
- *if $\|x - y\|_2 \geq \overline{c} \cdot r$, then $\Pr_{h \sim \mathcal{H}}[h(x) = h(y)] \leq p_2$.*

We want to remark that the original LSH definition supports more general distance function than ℓ_2 distance. In our application, ℓ_2 distance is sufficient, therefore we only define LSH based on ℓ_2 distance. It is well-known that an efficient LSH family implies data structure $(\overline{c}, r)$-ANN.

Definition 3.2 (Approximate Near Neighbor (ANN))**.** *Let $\overline{c} > 1$. Let $r \in (0, 2)$. Given an n-vector dataset $P \subset \mathbb{S}^{d-1}$ on the sphere, $(\overline{c}, r)$-Approximate Near Neighbor Search (ANN) aims at constructing a data structure such that, for a query $q \in \mathbb{S}^{d-1}$ with the promise that there exists a data vector $p \in P$ with $\|p - q\|_2 \leq r$, the data structure reports a data vector $p' \in P$ with distance less than $\overline{c} \cdot r$ from q.*

In the reinforcement learning algorithm, we care about the dual version of the problem (Definition 3.3),

Definition 3.3 (Approximate Max-IP)**.** *Let $c \in (0, 1)$. Let $\tau \in (0, 1)$. Given an n-vector dataset $P \subset \mathbb{S}^{d-1}$ on the sphere, the (c, τ)-Maximum Inner Product Search (Max-IP) aims at building a data structure such that, for a query $q \in \mathbb{S}^{d-1}$ with the promise that there exists a datapoint $p \in P$ with $\langle p, q \rangle \geq \tau$, the data structure reports a datapoint $p' \in P$ with similarity $\langle p', q \rangle \geq c \cdot \tau$.*

We briefly discuss the connection. Let us consider the distance function as Euclidean distance and similarity function as inner product. We also assume all the points are from unit sphere. In this setting, the relationship between two problems are primal vs dual. For any two points x, y with $\|x\|_2 = \|y\|_2 = 1$, we have $\|x - y\|_2^2 = 2 - 2\langle x, y \rangle$. This implies that $r^2 = 2 - 2\tau$. Further, if we have a data structure for $(\overline{c}, r)$-ANN, it automatically becomes a data structure for (c, τ)-Max-IP with parameters $\tau = 1 - 0.5r^2$ and $c = \frac{1 - 0.5\overline{c}^2 r^2}{1 - 0.5r^2}$. This implies that $\overline{c}^2 = \frac{1 - c(1 - 0.5r^2)}{0.5r^2} = \frac{1 - c\tau}{1 - \tau}$.

Our algorithmic result is mainly built on this data structure.

Theorem 3.4 (Andoni and Razenshteyn (Andoni and Razenshteyn, 2015))**.** *Let $\overline{c} > 1$. Let $r \in (0, 2)$. Let $\rho = \frac{1}{2\overline{c}^2 - 1} + o(1)$. The $(\overline{c}, r)$-ANN (see Definition 3.2) on the unit sphere $\mathbb{S}^{d-1}$ can be solved by a data structure using $O(d \cdot n^\rho)$ query time and $O(n^{1+\rho} + dn)$ space.*

Using the standard reduction, we can derive the following.

Corollary 3.5. *Let $c \in (0, 1)$. Let $\tau \in (0, 1)$. The (c, τ)-Max-IP (see Definition 3.3) on a unit sphere $\mathbb{S}^{d-1}$ can be solved in preprocessing time/space $O(n^{1+\rho} + dn)$ and query time $O(d \cdot n^\rho)$, where $\rho = \frac{1 - \tau}{1 - 2c\tau + \tau} + o(1)$.*

Using Andoni et al. (2017a), we can improve the preprocessing time and space to $n^{1+o(1)} + dn$ while having a slightly weaker ρ in query. We provide a detailed and formal version of Corollary 3.5 in Theorem B.2. We present our main result based on that. Moreover, it is reasonable for us to regard $d = n^{o(1)}$ using Johnson-Lindenstrauss Lemma (Johnson and Lindenstrauss, 1984).

Finally, to combine the maximum inner product search with reinforcement learning algorithm to get sublinear time cost per iteration, we still need to deal with many issues, such as the inner product can be negative, τ is arbitrarily close to 0, and τ can arbitrarily close to 1. We will explain how to handle these challenges in later section.

3.2 Reinforcement Learning

In this section, we introduce some backgrounds about reinforcement learning. We start with defining the episodic Markov decision process. Let MDP$(\mathcal{S}, \mathcal{A}, H, \mathbb{P}, r)$ denote the episodic Markov Decision Process, where $\mathcal{S}$ denotes the set of available states, $\mathcal{A}$ denotes the set of available actions, $H \in \mathbb{N}$ denotes the total number of steps in each episode, $\mathbb{P} = \{\mathbb{P}_h\}_{h=1}^H$ with $\mathbb{P}_h[s'|s, a]$ denotes the probability of transition from state $s \in \mathcal{S}$ to state $s' \in \mathcal{S}$ when take actions $a \in \mathcal{A}$ at step h, $r = \{r_h\}_{h=1}^H$ denotes the reward obtained at each step. Here the reward r_h is a function that maps $\mathcal{S} \times \mathcal{A}$ to $[0.55, 1]$[iii] In practice, we build an agent in MDP$(\mathcal{S}, \mathcal{A}, H, \mathbb{P}, r)$ and play K episodes.

In this work, we focus on the linear Markov Decision Process (linear MDP). In this setting, each pair of state and action is represented as an embedding vector $\phi(s, a)$, where $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$. Moreover, the probability $\mathbb{P}_h[s'|s, a]$ for state transition and function r_h are linear in this embedding vector.

In the MDP framework, a policy $\pi = \{\pi_1, \cdots, \pi_H\}$ is defined as sequence such that $\pi_h : \mathcal{S} \to \mathcal{A}$ for each step h. $\pi_h(s) = a$ represents the action taken when we are at step h and state s. Next, we represent the Bellman equation with policy π as

$$Q_h^\pi(s, a) = [r_h + \mathbb{P}_h V_{h+1}^\pi](s, a),$$
$$V_h^\pi(s) = Q_h^\pi(s, \pi_h(s))$$

[iii] Note that in standard reinforcement learning, we assume reward is $[0, 1]$, but it is completely reasonable to do a shift. We will provide more discussion in Section 5.1.

$$V_{H+1}^{\pi}(s) = 0.$$

where $Q^{\pi}(s,a)$ denotes the Q function for policy π when taking action a at state s and $V^{\pi}(s)$ denotes the value function of state s at step h. We use $[\mathbb{P}_h V_{h+1}](s,a)$ to represent the expect value functions when taking action a at state s at step h. For more definitions, please refer to Section A.

4 OUR RESULTS

We present the results in this section. We start with summarizing all of our main results in Table 1. According to Table 1, we reduce the value iteration complexity of LSVI (Bradtke and Barto, 1996) and LSVI with upper confidence bound (LSVI-UCB) (Jin et al., 2020), from linear to sublinear in action space. Meanwhile, the total regret is preserved as same as before. To achieve this, we pay tolerable time to preprocess pairs of state-action into LSH type approximate Max-IP data structure. In the following section, we would elaborate on the details for these main results.

4.1 Sublinear Least-Squares Value Iteration

In LSVI (Bradtke and Barto, 1996) with large action space, the runtime in each value iteration step is dominated by computing the estimated value function as below:

$$\widehat{V}_h(s) = \max_{a \in \mathcal{A}_{\text{core}}} \langle \widehat{w}_h, \phi(s,a) \rangle, \tag{1}$$

where $\widehat{w}_h$ is computed by solving the least-squares problem, $\mathcal{A}_{\text{core}}$ is the core action set and $\phi(s,a)$ is the embedding for state-action pair. Eq. (1) is a standard Max-IP problem and thus, takes $O(Ad)$ to obtain the exact solution. In this work, we relax Eq. (1) into an (c,τ)-Max-IP problem, where $c \in (0,1)$ is the approximation parameter and τ is close to the $\max_{a \in \mathcal{A}_{\text{core}}} \langle \widehat{w}_h, \phi(s,a) \rangle$. Then, we apply LSH type data structure to obtain $\widehat{V}_h(s)$ such that

$$\widehat{V}_h(s) \geq c \cdot \max_{a \in \mathcal{A}_{\text{core}}} \langle \widehat{w}_h, \phi(s,a) \rangle.$$

Note that this operation takes $o(A) \cdot O(d)$ time.

Next, we present our main theorem for Sublinear LSVI in Theorem 4.1, which gives the same $O(LH^2\sqrt{\iota/n})$ regret as LSVI Bradtke and Barto (1996) and reduce the value iteration complexity from $O(HSdA)$ to $O(HSd) \cdot o(A)$.

Theorem 4.1 (Main result, convergence result of Sublinear Least-Squares Value Iteration (Sublinear LSVI), an informal version of Theorem C.2)**.** *Let* MDP$(\mathcal{S}, \mathcal{A}, H, \mathbb{P}, r)$ denote a linear MDP. Let p denote a fixed probability. Let $\iota = \log(Hd/p)$. If we set approximate Max-IP parameter $c = 1 - \Theta(\sqrt{\iota/n})$, then Sublinear LSVI has regret at most $O(H^2\sqrt{\iota/n})$ with probability at least $1-p$. Moreover, with $SA^{1+o(1)} + SdA$ preprocessing time and space, the value iteration complexity of Sublinear LSVI is $O(HSdA^{\rho})$ where $\rho = 1 - \Theta(\iota/n)$.

Note that we could improve the value iteration complexity to with $\rho = 1 - \Theta(\sqrt{\iota/n})$ by increasing the preprocessing time and space to $O(SA^{1+\rho} + SdA)$ using Theorem A.14. We provide a detailed and formal version of Theorem 4.1 in Theorem C.2.

Theorem 4.1 provide the theoretical guidance on the choice of LSH parameters for LSVI. If we set the approximation ratio c, we could anticipate the resulting regret and running time before experiment.

4.2 Sublinear Least-Squares Value Iteration with UCB

We extend the Sublinear LSVI with UCB exploration in this section. In LSVI-UCB (Jin et al., 2020) with large action space, the runtime in each value iteration step is dominated by computing the estimated value function as below:

$$\widehat{V}_h(s_{h+1}^{\tau}) = \max_{a \in \mathcal{A}} \min\{\langle w_h^k, \phi(s_{h+1}^{\tau}, a) \rangle + \beta \cdot \|\phi(s_{h+1}^{\tau}, a)\|_{\Lambda_h^{-1}}, H\}, \tag{2}$$

where w_h^k is computed by solving the least-squares problem, $\phi(s_{h+1}^{\tau}, a)$ is the embedding for state-action pair and

$$\Lambda_h = \sum_{\tau=1}^{k-1} \phi(s_h^{\tau}, a_h^{\tau}) \phi(s_h^{\tau}, a_h^{\tau})^{\top} + \lambda \cdot \mathbf{I}_d.$$

Note that the complexity for Eq. (2) is $O(Ad^2)$.

The key challenge of the proposed Sublinear LSVI-UCB is that Eq. (2) cannot be formulated as a Max-IP problem. First, to deal with this issue, we propose a value function estimation approach as below:

$$\widehat{V}_h(s_{h+1}^{\tau}) = \max_{a \in \mathcal{A}} \min\{\|\phi(s_{h+1}^{\tau}, a)\|_{2\beta^2 \Lambda_h^{-1} + 2 w_h^k w_h^{k\top}}, H\}, \tag{3}$$

where $\|\phi(s_{h+1}^{\tau}, a)\|_{2\beta^2 \Lambda_h^{-1} + 2 w_h^k w_h^{k\top}}$ is the upper bound of $\langle w_h^k, \phi(s_{h+1}^{\tau}, a) \rangle + \beta \cdot \|\phi(s_{h+1}^{\tau}, a)\|_{\Lambda_h^{-1}}$.

Next, we relax this maximum matrix norm search as a (c,τ)-Max-IP problem, where $c \in (0,1)$ is the approximation parameter and τ is the maximum inner product for Eq. (3). Then, we apply LSH type data structure to obtain $\widehat{V}_h(s_{h+1}^{\tau})$ such that

$$\widehat{V}_h(s_{h+1}^{\tau}) \geq c \cdot \max_{a \in \mathcal{A}} \min\{\|\phi(s_{h+1}^{\tau}, a)\|_{2\beta^2 \Lambda_h^{-1} + 2 w_h^k w_h^{k\top}}, H\}.$$

Note that this operation takes $o(A) \cdot O(d^2)$ time complexity.

Using LSH data structure for maximum matrix norm search, we present our main theorem for Sublinear LSVI-UCB in Theorem 4.3, which gives the same $O(\sqrt{d^3 H^4 K \iota^2})$ regret as LSVI-UCB (Jin et al., 2020) and reduce the value iteration complexity from $O(HKd^2A)$ to $O(HKd^2A) \cdot o(A)$. We start with the setting up the parameters for our algorithm.

Table 1: Comparison between our algorithms with previous results such as LSVI, LSVI-UCB, LGSC and MF. We compare our algorithm with: (1) LSVI denotes the Least-Square Value Iteration algorithm (Bradtke and Barto, 1996) (2) LSVI-UCB denotes the Least-Square Value Iteration algorithm with UCB in (Jin et al., 2020). Note that "**V. Iter. C.**" denotes the Value iteration complexity. Let S denote the number of available states. Let A denote the number of available actions. Let d denote the dimension of $\phi(s,a)$. Let H denote the number of steps per episode. Let K denote the total number of episodes. Let n be the quantity of times played for each core pair of state-action. Let $\iota = \log(Hd/p)$ and p is the failure probability. We ignore the big-Oh notation "O" in the table. Let $\rho \in (0,1)$ denote a parameter determined by data structure. In fact, the preprocessing time for Sublinear LSVI-UCB is $O(SA^{1+o(1)} + Sd^2A)$. Since $K > S$, we write the preprocessing time as $O(KA^{1+o(1)} + Kd^2A)$. This table is a union of simplified version of Table 3 (both our algorithm and LSVI have the exact dependence on another L, we omit here and discuss this dependence in Section C.) and Table 4.

	Statement	**Preprocess**	**#Regret**	**V. Iter. C.**
LSVI	Bradtke and Barto (1996)	0	$H^2\sqrt{\iota/n}$	$HSdA$
Ours	Theorem 4.1	$SA^{1+o(1)} + SdA$	$H^2\sqrt{\iota/n}$	$HSdA^\rho$
LSVI-UCB	Jin et al. (2020)	0	$\sqrt{H^4Kd^3\iota^2}$	HKd^2A
Ours	Theorem 4.3	$KA^{1+o(1)} + Kd^2A$	$\sqrt{H^4Kd^3\iota^2}$	HKd^2A^ρ

Definition 4.2 (Sublinear LSVI-UCB Parameteres)**.** *Let* MDP$(\mathcal{S}, \mathcal{A}, H, \mathbb{P}, r)$ denote a linear MDP. For this MDP, we set LSVI-UCB parameter $\lambda = 1$. Let $c = 1 - \frac{1}{\sqrt{K}}$ denote the approximate Max-IP parameter. Let p denote a fixed probability. Let $\iota = \log(2dT/p)$.

Then, we present the Theorem.

Theorem 4.3 (Main result, convergence result of Sublinear Least-Squares Value Iteration with UCB (Sublinear LSVI-UCB), an informal version of Theorem D.12)**.** *With parameters defined in Definition 4.2, Sublinear LSVI-UCB (Algorithm 4) has total regret at most $O(\sqrt{d^3H^4K\iota^2})$ with probability at least $1-p$. Moreover, with $O(KA^{1+o(1)} + Kd^2A)$ preprocessing time and space, the value iteration complexity of Sublinear LSVI-UCB is $O(HKd^2A^\rho)$, where $\rho = 1 - 1/K$.*

Similarly, we could improve the value iteration complexity to with $\rho = 1 - \frac{1}{\sqrt{K}}$ by increasing the preprocessing time and space to $O(KA^{1+\rho} + KdA)$ using Theorem A.14. We provide a detailed and formal version of Theorem 4.3 in Theorem D.12.

Theorem 4.3 provides a LSH solution to one of the most famous extensions of LSVI. Moreover, it provides a solution to solve general similarity search in RL algorithms by transforming it into Max-IP. With this result, we demonstrate that our techniques is compatible with recent VI algorithms.

5 OUR TECHNIQUES

As mentioned in Section 3.1, we need to tackle five major issues to use LSH based approximate Max-IP algorithm for sublinear runtime time LSVI and LSVI-UCB in RL.

- How to prevent the maximum inner product between query and data from being negative or arbitrary close to 0? If the maximum inner product is negative, Max-IP data structures cannot be applied to solve this problem with theoretical guarantee. If the maximum inner product is arbitrary close to 0, the query time of (c,τ)-Max-IP would be close to $O(dn)$.

- How to prevent the maximum inner product between query and data from being close to one? If τ is close to one, the time cost would also be $O(dn)$ so that (c,τ)-Max-IP cannot reduce the time cost from linear to sublinear.

- How to apply (c,τ)-Max-IP for LSVI with UCB exploration? The estimated value function with an additional UCB bonus term could not be written as an inner product, which prevents Max-IP techniques from accelerating the runtime efficiency.

- How to generalize the Max-IP data structure to support maximum matrix norm search? Is Max-IP equivalent to maximum matrix norm search?

- How to improve the running time while preserving the regret? Although approximate Max-IP could accelerate the computation for estimated value function, it brings errors to the value function estimation and thus, affects the total regret. Therefore, a key challenge is quantifying the relationship between regret and the approximation factor c in (c,τ)-Max-IP.

- How to handle the adaptive queries? The weight $\widehat{w}_h$ in Eq. (1) and w_h^k in Eq. (2) are dependent to $h-1$ step. Therefore, the queries for (c,τ)-Max-IP during the Q-learning are adaptive but not arbitrary. Thus, we could not union bound the failure probability of LSH for (c,τ)-Max-IP.

Next, we provide technical details on how we handle these problems.

5.1 Avoid Negative Inner Product or Inner Product Close to 0

In our setting, we assume the reward function r lies in $[0.55, 1]$[iv]. This shift on the reward function would not affect the convergence results of our Sublinear LSVI and Sublinear LSVI-UCB. Moreover, it would benefits the Max-IP by generating acceptable maximum inner product. For Sublinear LSVI, as $r_h(s,a) \in [0.55, 1]$, the optimal value function $V_h^*(s) \geq 0.55$. Then according to Theorem 4.1 the estimated $\widehat{V}_h(s) = \max_{a\in\mathcal{A}} \langle w_h, \psi(s,a)\rangle$ satisfies $|V_h^*(s) - \widehat{V}_h(s)| \leq \epsilon$ if we query each pair of state-action from span matrix for $n = O(\epsilon^{-2}L^2H^4\iota)$ times. In this way, we could assure the maximum inner product is greater than 0.5 if we set $\epsilon \leq 0.05$. For Sublinear LSVI-UCB, the Max-IP is applied on $\widehat{V}_h(s) = \max_{a\in\mathcal{A}} Q_h^k(s,a)$, where $Q_h^k(s,a)$ is a Q function with additional UCB term. From (Jin et al., 2020), we know that for all pair of state-action ,$Q_h^k(s,a) \geq Q_h^*(s,a)$. Therefore, the maximum inner product for Sublinear LSVI-UCB is always greater than 0.5.

5.2 Avoid Inner Product Close to 1

In the optimization problem that could be accelerated by Max-IP, the query and data vectors are usually not unit vectors. To apply results in Section 3.1, we should demonstrate how to transform both query and data vectors into unit vectors. Moreover, we also modify the transformation to avoid the inner product from being too close to 1.

In our work, we introduce a pair of asymmetric transformations as below. Given two vector $x, y \in \mathbb{R}^d$ with $\|y\|_2 \leq 1$ and $\|x\|_2 \leq D_x$, we apply the following transformations

$$\begin{aligned} P(y) = & \begin{bmatrix} y^\top & \sqrt{1 - \|y\|_2^2} & 0 \end{bmatrix}^\top, \\ Q(x) = & \begin{bmatrix} \frac{0.8\cdot x^\top}{D_x} & 0 & \sqrt{1 - \frac{0.64\cdot \|x\|_2^2}{D_x^2}} \end{bmatrix}^\top. \end{aligned} \tag{4}$$

Using this transformations, we transform x, y into unit vectors $P(y)$ and $Q(x)$. Therefore, the Max-IP of $Q(x)$ with respect to $P(Y)$ is equivalent to the ANN problem of $Q(x)$ with respect to $P(Y)$, which could be solved via LSH.

Moreover, we show that

$$Q(x)^\top P(y) = \frac{0.8 x^\top y}{D_x} \leq \frac{0.8 \cdot \|x\|_2 \|y\|_2}{D_x} = 0.8.$$

Further more, it is sufficient to show that

$$\begin{aligned} \arg\max_y Q(x)^\top P(y) = & \arg\max_y \frac{0.8 \cdot x^\top y}{D_x} \\ = & \arg\max_y x^\top y. \end{aligned}$$

[iv]Note that for any reward range $[a, b]$, there exists a shift c and scaling α so that $(a + c)/\alpha = 0.55$ and $(b + c)/\alpha = 1$.

If we perform maximum inner product search on $Q(x)$ and $P(y)$ using the LSH data structures described in Section 3.1, we have

$$\tau = \max_y Q(x)^\top P(y) \leq 0.8.$$

In this way, we could assure τ is not close to 1 so that we could reduce the runtime complexity of value function estimation to be sublinear over actions.

5.3 Approximate Max-IP Data Structure for LSVI-UCB

As shown in Section 4.2, Eq. (2) cannot be formulated as a Max-IP problem. To overcome this barrier, we bound the term $Q_h(s_{h+1}^\tau, a) = \{w_h^\top \phi(s_{h+1}^\tau, a) + \beta \cdot \|\phi(s_{h+1}^\tau, a)\|_{\Lambda_h^{-1}}, H\}$ by matrix norms. Then, we perform the maximum matrix norm search for value function estimation.

We start with the upper bound of $w_h^\top \phi(s_{h+1}^\tau, a) + \beta \cdot \|\phi(s_{h+1}^\tau, a)\|_{\Lambda_h^{-1}}$. As both $\langle w_h^k, \phi(s_{h+1}^\tau, a)\rangle$ and $\|\phi(s_{h+1}^\tau, a)\|_{\Lambda_h^{-1}}$ are non-negative, we have

$$\begin{aligned} & \langle w_h^k, \phi(s_{h+1}^\tau, a)\rangle + \beta \cdot \|\phi(s_{h+1}^\tau, a)\|_{\Lambda_h^{-1}} \\ \leq & \sqrt{2(w_h^\top \phi(s_{h+1}^\tau, a))^2 + 2\beta^2 \cdot \|\phi(s_{h+1}^\tau, a)\|_{\Lambda_h^{-1}}^2} \\ = & \|\phi(s_{h+1}^\tau, a)\|_{2\beta^2\Lambda_h^{-1} + 2w_h^k (w_h^k)^\top} \end{aligned}$$

where the first step follows from $a + b \leq \sqrt{2a^2 + 2b^2}$.

Next, we lower bound the $w_h^\top \phi(s_{h+1}^\tau, a) + \beta \cdot \|\phi(s_{h+1}^\tau, a)\|_{\Lambda_h^{-1}}$ as

$$\begin{aligned} & w_h^\top \phi(s_{h+1}^\tau, a) + \beta \cdot \|\phi(s_{h+1}^\tau, a)\|_{\Lambda_h^{-1}} \\ \geq & \sqrt{(w_h^\top \phi(s_{h+1}^\tau, a))^2 + \beta^2 \cdot \|\phi(s_{h+1}^\tau, a)\|_{\Lambda_h^{-1}}^2} \\ = & \|\phi(s_{h+1}^\tau, a)\|_{\beta^2\Lambda_h^{-1} + w_h^k (w_h^k)^\top}, \end{aligned}$$

where the first step follows from the fact that both $w_h^\top \phi(s_{h+1}^\tau, a)$ and $\|\phi(s_{h+1}^\tau, a)\|_{\Lambda_h^{-1}}$ are non-negative and $a + b \geq \sqrt{a^2 + b^2}$ if $a, b \geq 0$, the second step is an reorganization.

After we lower and upper bound $w_h^\top \phi(s_{h+1}^\tau, a) + \beta \cdot \|\phi(s_{h+1}^\tau, a)\|_{\Lambda_h^{-1}}$, we could also lower bound the term $\{w_h^\top \phi(s_{h+1}^\tau, a) + \beta \cdot \|\phi(s_{h+1}^\tau, a)\|_{\Lambda_h^{-1}}, H\}$ with $\min\{\|\phi(s_{h+1}^\tau, a)\|_{\beta^2\Lambda_h^{-1} + w_h^k (w_h^k)^\top}, H\}$ and upper bound it with $\min\{\|\phi(s_{h+1}^\tau, a)\|_{2\beta^2\Lambda_h^{-1} + 2w_h^k (w_h^k)^\top}, H\}$.

Next we use this lower and upper bound and propose a modified value function estimation shown in Eq. (3). Therefore, our problem becomes designing an approximate maximum matrix norm search data structure. We will discuss this in the following sections.

5.4 Generalize the Approximate Max-IP Data Structure for Max-MatNorm

In this section, we demonstrate how to extend Max-IP to maximum matrix norm search for Sublinear LSVI-UCB in this section. We first define the approximate Maximum Matrix Norm problem. Let $c \in (0,1)$ and $\tau \in (0,1)$. Given an n-point dataset $Y \subset \mathbb{R}^d$, the goal of the (c,τ)-Maximum Matrix Norm (Max-MatNorm) is to construct a data structure that, given a query matrix $x \in \mathbb{R}^{d \times d}$ with the promise that there exists a datapoint $y \in Y$ with $\|y\|_x \geq \tau$, it reports a datapoint $z \in Y$ with $\|z\|_x \geq c \cdot \tau$. We refer the readers to Definition B.4 in the appendix for more details.

We solve the approximate maximum matrix norm by transform it into an approximate Max-IP problem (see Definition 3.3). We start with showing the relationship between Max-MatNorm and Max-IP as

$$\begin{aligned} \text{Max-MatNorm}(X,Y)^2 &= \max_{y \in Y} y^\top x y \\ &= \max_{y \in Y} \langle \mathrm{vec}(x), \mathrm{vec}(yy^\top) \rangle, \end{aligned}$$

where vec vectorizes $d \times d$ matrix x into a d^2 vector.

Next, we show that if we obtain $z \in Y$ by (c^2, τ^2)-Max-IP so that

$$\langle \mathrm{vec}(x), \mathrm{vec}(zz^\top) \rangle \geq c^2 \tau^2,$$

we use z and obtain

$$\|z\|_x = \sqrt{\langle \mathrm{vec}(x), \mathrm{vec}(zz^\top) \rangle} \geq c\tau.$$

In other words, z is a candidate for (c,τ)-Max-MatNorm. In this way, we build a data-structure for (c^2,τ^2)-Max-IP to solve (c,τ)-Max-MatNorm. We summarize our approach for Max-MatNorm as three steps:

- Transform matrix x into $\mathrm{vec}(x)$ and y into vector $\mathrm{vec}(yy^\top)$.
- Transform $\mathrm{vec}(x)$ and $\mathrm{vec}(yy^\top)$ into unit vectors following Eq. (4).
- Use LSH to solve the Max-IP with respect to dataset on the unit sphere.

5.5 Preserving Regret While Reducing the Runtime

In our work, we maintain the same regret with LSVI (Bradtke and Barto, 1996) and LSVI-UCB (Jin et al., 2020) by carefully setting the approximation parameter $c \in (0,1)$ in Max-IP. For Sublinear LSVI, we set $c = 1 - \Theta(\sqrt{\iota/n})$ so that the final regret is as same as LSVI (Bradtke and Barto, 1996). In Sublinear LSVI-UCB, we set $c = 1 - \frac{1}{\sqrt{K}}$ so that the final regret is as same as LSVI-UCB (Jin et al., 2020). Because K, ι and n are global parameter, we could set c in the preprocessing step before value iteration. In this way, we show that our two algorithms are novel demonstration of combining LSH with reinforcement learning without losing on the regret.

We would like to emphasize the significance of these results. In machine learning with LSH, algorithms use LSH data structures as a black box, which introduces extra hyperparameters to the learning procedure. In other words, there would be multiple trials of training to find the best hyperparameters. As a result, the efficiency improvements or potential accuracy deductions remain unknown until all trials are finished. In contrast, our method provides a specification on the LSH parameters before performing an iterative algorithm. Therefore, our results could avoid extensive LSH parameter tuning in practice.

5.6 Handle Adaptive Queries in (c,τ)-Max-IP

In this section, we demonstrate our techniques to handle the adaptive queries to the (c,τ)-Max-IP data structures. We use a quantization method to handle adaptive queries. We denote Q as the convex hull of all queries for (c,τ)-Max-IP. Our method contains two steps: (1) Preprocessing: we quantize Q to a lattice $\widehat{Q}$ with quantization error λ/d. In this way, each coordinate would be quantized into the multiples of λ/d. (2) Query: given a query q in the adaptive sequence $X \subset Q$, we first quantize it to the nearest $\widehat{q} \in \widehat{Q}$ and perform (c,τ)-Max-IP. As each $\widehat{q} \in \widehat{Q}$ is independent, we could union bound the failure probability of adaptive queries. On the other hand, this would generate an λ additive error in the returned inner product. Our analysis indicates that the additive error λ could be handled without breaking the regret.

6 CONCLUSION

In this paper, we study the value iteration algorithms for solving linear Markov Decision Process (MDP) with large action space. We identify that one of the major efficiency bottlenecks for this setting is the value function estimation procedure over all available actions. To tackle this issue, we propose two provable Least-Squares Value Iteration (LSVI) algorithms with runtime complexity sublinear in the number of actions. By formulating the value function estimation procedure in LSVI as an approximate maximum inner product search problem, we bridge the gap between the regret analysis in reinforcement learning and the theory of Locality Sensitive Hashing (LSH) type data structure. The theoretical analysis indicates that with our choice of approximation factor, there exists a LSVI algorithm that has the same order of regret as the original LSVI algorithm while reducing runtime complexity to sublinear in the number of actions. Moreover, we show that our techniques could be extended to one important LSVI variant: LSVI with Upper Confidence Bound (UCB). In this way, our proposal could support more sample-efficient VI algorithms. Although the implementation consumes energy, we hope our novel combination of data structures and the iterative algorithm will inspire further study into cost reduction in optimization.

7 ACKNOWLEDGEMENTS

Zhaozhuo Xu and Anshumali Shrivastava are supported by the National Science Foundation IIS-1652131, BIGDATA-1838177, AFOSR-YIP FA9550-18-1-0152, the ONR DURIP Grant and the ONR BRC grant on Randomized Numerical Linear Algebra.

References

Amir Abboud, Aviad Rubinstein, and Ryan Williams. Distributed pcp theorems for hardness of approximation in p. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 25–36. IEEE, 2017.

Alexandr Andoni. *Nearest neighbor search: the old, the new, and the impossible*. PhD thesis, Massachusetts Institute of Technology, 2009.

Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117, 2008.

Alexandr Andoni and Ilya Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing (STOC)*, pages 793–801, 2015.

Alexandr Andoni, Piotr Indyk, Huy L Nguyen, and Ilya Razenshteyn. Beyond locality-sensitive hashing. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 1018–1028. SIAM, 2014.

Alexandr Andoni, Piotr Indyk, TMM Laarhoven, Ilya Razenshteyn, and Ludwig Schmidt. Practical and optimal lsh for angular distance. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1225–1233. Curran Associates, 2015.

Alexandr Andoni, Thijs Laarhoven, Ilya Razenshteyn, and Erik Waingarten. Optimal hashing-based time-space trade-offs for approximate near neighbors. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 47–66. SIAM, 2017a.

Alexandr Andoni, Ilya Razenshteyn, and Negev Shekel Nosatzki. Lsh forest: Practical algorithms made theoretical. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 67–78. SIAM, 2017b.

Alexandr Andoni, Piotr Indyk, and Ilya Razenshteyn. Approximate nearest neighbor search in high dimensions. In *Proceedings of ICM*, volume 7, 2018.

Arturs Backurs, Moses Charikar, Piotr Indyk, and Paris Siminelakis. Efficient density evaluation for smooth kernels. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 615–626. IEEE, 2018.

Arturs Backurs, Piotr Indyk, and Tal Wagner. Space and time efficient kernel density estimation in high dimensions. *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

Yu Bai, Tengyang Xie, Nan Jiang, and Yu Xiang Wang. Provably efficient q-learning with low switching cost. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

Omri Ben-Eliezer, Rajesh Jayaram, David P Woodruff, and Eylon Yogev. A framework for adversarially robust streaming algorithms. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS)*, pages 63–80, 2020.

Steven J Bradtke and Andrew G Barto. Linear least-squares algorithms for temporal difference learning. *Machine learning*, 22(1):33–57, 1996.

Jan van den Brand. A deterministic linear program solver in current matrix multiplication time. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 259–278. SIAM, 2020.

Jan van den Brand, Yin-Tat Lee, Danupon Nanongkai, Richard Peng, Thatchaphol Saranurak, Aaron Sidford, Zhao Song, and Di Wang. Bipartite matching in nearly-linear time on moderately dense graphs. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 919–930. IEEE, 2020a.

Jan van den Brand, Yin Tat Lee, Aaron Sidford, and Zhao Song. Solving tall dense linear programs in nearly linear time. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 775–788, 2020b.

Jan van den Brand, Binghui Peng, Zhao Song, and Omri Weinstein. Training (overparametrized) neural networks in near-linear time. In *12th Innovations in Theoretical Computer Science Conference (ITCS)*, 2021.

Qi Cai, Zhuoran Yang, Chi Jin, and Zhaoran Wang. Provably efficient exploration in policy optimization. In *International Conference on Machine Learning (ICML)*, pages 1283–1294. PMLR, 2020.

Moses Charikar and Paris Siminelakis. Hashing-based-estimators for kernel density in high dimensions. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1032–1043. IEEE, 2017.

Moses Charikar, Michael Kapralov, Navid Nouri, and Paris Siminelakis. Kernel density estimation through density constrained near neighbor search. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 172–183. IEEE, 2020.

Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing (STOC)*, pages 380–388, 2002.

Beidi Chen, Yingchen Xu, and Anshumali Shrivastava. Lsh-sampling breaks the computation chicken-and-egg loop in adaptive stochastic gradient estimation. *arXiv preprint arXiv:1910.14162*, 2019.

Beidi Chen, Tharun Medini, James Farwell, sameh gobriel, Charlie Tai, and Anshumali Shrivastava. Slide : In defense of smart algorithms over hardware acceleration for large-scale deep learning systems. In *Proceedings of Machine Learning and Systems (MLSys)*, volume 2, pages 291–306, 2020.

Beidi Chen, Zichang Liu, Binghui Peng, Zhaozhuo Xu, Jonathan Lingjie Li, Tri Dao, Zhao Song, Anshumali Shrivastava, and Christopher Re. MONGOOSE: A learnable LSH framework for efficient neural network training. In *International Conference on Learning Representations (ICLR)*, 2021.

Lijie Chen. On the hardness of approximate and exact (bichromatic) maximum inner product. In *33rd Computational Complexity Conference (CCC)*, 2018.

Lijie Chen and Ryan Williams. An equivalence class for orthogonal vectors. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 21–40. SIAM, 2019.

Tobias Christiani. A framework for similarity search with space-time tradeoffs using locality-sensitive filtering. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 31–46. SIAM, 2017.

Michael B Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, 2019.

Benjamin Coleman, Benito Geordie, Li Chou, RA Leo Elworth, Todd Treangen, and Anshumali Shrivastava. One-pass diversified sampling with application to terabyte-scale genomic sequence streams. In *International Conference on Machine Learning*, pages 4202–4218. PMLR, 2022.

Shabnam Daghaghi, Nicholas Meisburger, Mengnan Zhao, and Anshumali Shrivastava. Accelerating slide deep learning on modern cpus: Vectorization, quantizations, memory optimizations, and more. *Proceedings of Machine Learning and Systems*, 3, 2021.

Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry (SoCG)*, pages 253–262, 2004.

Qin Ding, Hsiang-Fu Yu, and Cho-Jui Hsieh. A fast sampling algorithm for maximum inner product search. In *The 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 3004–3012. PMLR, 2019.

Sally Dong, Yin Tat Lee, and Guanghao Ye. A nearly-linear time algorithm for linear programs with small treewidth: A multiscale representation of robust central path. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*. arXiv preprint arXiv:2011.05365, 2021.

Yihe Dong, Piotr Indyk, Ilya Razenshteyn, and Tal Wagner. Learning space partitions for nearest neighbor search. In *International Conference on Learning Representations (ICLR)*. arXiv preprint arXiv:1901.08544, 2020.

Simon S Du, Sham M Kakade, Ruosong Wang, and Lin F Yang. Is a good representation sufficient for sample efficient reinforcement learning? In *International Conference on Learning Representations (ICLR)*, 2020.

Simon S Du, Sham M Kakade, Jason D Lee, Shachar Lovett, Gaurav Mahajan, Wen Sun, and Ruosong Wang. Bilinear classes: A structural framework for provable generalization in rl. In *ICML*, 2021.

Fei Feng, Ruosong Wang, Wotao Yin, Simon S Du, and Lin Yang. Provably efficient exploration for reinforcement learning using unsupervised learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 2020.

Minbo Gao, Tianle Xie, Simon S Du, and Lin F Yang. A provably efficient algorithm for linear markov decision process with low switching cost. *arXiv preprint arXiv:2101.00494*, 2021.

Ruiqi Guo, Sanjiv Kumar, Krzysztof Choromanski, and David Simcha. Quantization based fast inner product search. In *Artificial Intelligence and Statistics (AISTATS)*, pages 482–490. PMLR, 2016.

Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. Accelerating large-scale inference with anisotropic vector quantization. In *International Conference on Machine Learning (ICML)*, pages 3887–3896. PMLR, 2020.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing (STOC)*, pages 604–613, 1998.

Haotian Jiang, Yin Tat Lee, Zhao Song, and Sam Chiu-wai Wong. An improved cutting plane method for convex optimization, convex-concave games and its applications. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, 2020.

Shunhua Jiang, Zhao Song, Omri Weinstein, and Hengjie Zhang. Faster dynamic matrix inverse for faster lps. In

Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC). arXiv preprint arXiv:2004.07470, 2021.

Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I Jordan. Is q-learning provably efficient? *Advances in Neural Information Processing Systems (NeurIPS)*, 2018:4863–4873, 2018.

Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan. Provably efficient reinforcement learning with linear function approximation. In *Conference on Learning Theory (COLT)*, pages 2137–2143. PMLR, 2020.

William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research (IJRR)*, 32(11):1238–1274, 2013.

François Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th international symposium on symbolic and algebraic computation (ISSAC)*, pages 296–303. ACM, 2014.

Yin Tat Lee, Zhao Song, and Qiuyi Zhang. Solving empirical risk minimization in the current matrix multiplication time. In *International Conference on Computational Learning Theory (COLT)*, 2019.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP-IJCNLP)*, pages 1192–1202, 2016.

Ping Li, Xiaoyun Li, Gennady Samorodnitsky, and Weijie Zhao. Consistent sampling through extremal process. In *Proceedings of the Web Conference 2021*, pages 1317–1327, 2021.

Xiaoyun Li and Ping Li. Rejection sampling for weighted jaccard similarity revisited. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

Chen Luo and Anshumali Shrivastava. Scaling-up split-merge mcmc with locality sensitive sampling (lss). In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 33, pages 4464–4471, 2019.

Qin Lv, William Josephson, Zhe Wang, Moses Charikar, and Kai Li. Multi-probe lsh: efficient indexing for high-dimensional similarity search. In *33rd International Conference on Very Large Data Bases (VLDB)*, pages 950–961. Association for Computing Machinery, Inc, 2007.

Francisco S Melo and M Isabel Ribeiro. Q-learning with linear function approximation. In *International Conference on Computational Learning Theory (COLT)*, pages 308–322. Springer, 2007.

Stanislav Morozov and Artem Babenko. Non-metric similarity graphs for maximum inner product search. *Advances in Neural Information Processing Systems (NeurIPS)*, 31: 4721–4730, 2018.

Vasileios Nakos, Zhao Song, and Zhengyu Wang. (nearly) sample-optimal sparse fourier transform in any dimension; ripless and filterless. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1568–1577. IEEE, 2019.

Behnam Neyshabur and Nathan Srebro. On symmetric and asymmetric lshs for inner product search. In *International Conference on Machine Learning (ICML)*, pages 1926–1934. PMLR, 2015.

Ilya Razenshteyn. *High-dimensional similarity search and sketching: algorithms and hardness*. PhD thesis, Massachusetts Institute of Technology, 2017.

Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk. Nearest-neighbor methods in learning and vision. In *Neural Information Processing*, 2005.

Anshumali Shrivastava and Ping Li. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). *Advances in Neural Information Processing Systems (NIPS)*, pages 2321–2329, 2014.

Anshumali Shrivastava and Ping Li. Improved asymmetric locality sensitive hashing (alsh) for maximum inner product search (mips). In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 812–821, 2015a.

Anshumali Shrivastava and Ping Li. Asymmetric minwise hashing for indexing binary inner products and set containment. In *Proceedings of the 24th international conference on world wide web (WWW)*, pages 981–991, 2015b.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning (ICML)*, pages 387–395. PMLR, 2014.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587): 484–489, 2016.

Paris Siminelakis, Kexin Rong, Peter Bailis, Moses Charikar, and Philip Levis. Rehashing kernel evaluation in high dimensions. In *International Conference on Machine Learning (ICML)*, pages 5789–5798. PMLR, 2019.

Zhao Song and Wen Sun. Efficient model-free reinforcement learning in metric spaces. *arXiv preprint arXiv:1905.00475*, 2019.

Zhao Song and Zheng Yu. Oblivious sketching-based central path method for solving linear programming problems. In *38th International Conference on Machine Learning (ICML)*, 2021.

Zhao Song, David Woodruff, Zheng Yu, and Lichen Zhang. Fast sketching of polynomial kernels of polynomial degree. In *International Conference on Machine Learning*, pages 9812–9823. PMLR, 2021.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Shulong Tan, Zhixin Zhou, Zhaozhuo Xu, and Ping Li. On efficient retrieval of top similarity vectors. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5239–5249, 2019.

Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. *arXiv preprint arXiv:1011.3027*, 2010.

Ruosong Wang, Simon S Du, Lin Yang, and Russ R Salakhutdinov. On reward-free reinforcement learning with linear function approximation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 17816–17826. Curran Associates, Inc., 2020a.

Ruosong Wang, Peilin Zhong, Simon S Du, Russ R Salakhutdinov, and Lin F Yang. Planning with general objective functions: Going beyond total rewards. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2020b.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Alexander Wei. Optimal las vegas approximate near neighbors in ℓ_p. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1794–1813. SIAM, 2019.

Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2-3):357–365, 2005.

Ryan Williams. On the difference between closest, furthest, and orthogonal pairs: Nearly-linear vs barely-subquadratic complexity. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1207–1215. SIAM, 2018.

Virginia Vassilevska Williams. Multiplying matrices faster than coppersmith-winograd. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing (STOC)*, pages 887–898. ACM, 2012.

Zhihan Xiong, Ruoqi Shen, and Simon S Du. Randomized exploration is near-optimal for tabular mdp. *arXiv preprint arXiv:2102.09703*, 2021.

Zhaozhuo Xu, Zhao Song, and Anshumali Shrivastava. Breaking the linear iteration cost barrier for some well-known conditional gradient methods using maxip data-structures. *Advances in Neural Information Processing Systems*, 34, 2021.

Xiao Yan, Jinfeng Li, Xinyan Dai, Hongzhi Chen, and James Cheng. Norm-ranging lsh for maximum inner product search. *Advances in Neural Information Processing Systems (NeurIPS)*, 31:2952–2961, 2018.

Lin Yang and Mengdi Wang. Reinforcement learning in feature space: Matrix bandit, kernels, and regret bound. In *International Conference on Machine Learning (ICML)*, pages 10746–10756, 2020.

Shuo Yang, Tongzheng Ren, Sanjay Shakkottai, Eric Price, Inderjit S Dhillon, and Sujay Sanghavi. Linear bandit algorithms with sublinear time complexity. *arXiv preprint arXiv:2103.02729*, 2021.

Hsiang-Fu Yu, Cho-Jui Hsieh, Qi Lei, and Inderjit S Dhillon. A greedy approach for budgeted maximum inner product search. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)*, pages 5459–5468, 2017.

Amir Zandieh, Navid Nouri, Ameya Velingker, Michael Kapralov, and Ilya Razenshteyn. Scaling up kernel ridge regression via locality sensitive hashing. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 4088–4097. PMLR, 2020.

Zihan Zhang, Yuan Zhou, and Xiangyang Ji. Almost optimal model-free reinforcement learningvia reference-advantage decomposition. *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 2020.

Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. Drn: A deep reinforcement learning framework for news recommendation. In *Proceedings of the 2018 World Wide Web Conference (WWW)*, pages 167–176, 2018.

Zhixin Zhou, Shulong Tan, Zhaozhuo Xu, and Ping Li. Möbius transformation for fast inner product search on graph. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

Appendix

Contents

Roadmap. Section A introduces the preliminaries of this work, including notations and definitions, Section B introduces the LSH data structure in detail, Section C presents the results for Sublinear LSVI, Section D presents the results for Sublinear LSVI-UCB, Section E shows how to process adaptive queries in Max-IP.

A PRELIMINARIES

Table 2: Notations related to reinforcement learning.

Notation	Meaning
$\mathcal{S}$	states space
$\mathcal{A}$	action space
$\mathcal{S}_{\mathrm{core}}$	core state set
$\mathcal{A}_{\mathrm{core}}$	core action set
S	# states
A	# actions
H	number of steps per episode
K	number of episodes
s'	next state of state s
$\mathbb{P}$	state transition probability
$\mathbb{P}_h[s'\|s,a]$	transition probability when we take action $a \in \mathcal{A}$ at step $h \in [H]$ from state $s \in \mathcal{S}$.
$r_h(s,a)$	reward at step h given state s and action a
r	$\{r_h\}_{h=1}^H$
$\phi(s,a)$	feature map $\phi(s,a) \in \mathbb{R}^d$
$\mu_h(s)$	unknown measure that $\mathbb{P}_h[s'\|s,a] = \langle \phi(s,a), \mu_h(s') \rangle$
θ_h	unknown measure that $r_h(s,a) = \langle \phi(s,a), \theta_h \rangle$
Φ	$\Phi \in \mathbb{R}^{d \times M}$
n	number of samples played given from each ϕ_j.

This section introduces the preliminaries for our work.

- In Section A.1, we present the basic notations used in our work.
- In Section A.2, we introduce several reinforcement learning.
- In Section A.3, we list the standard properties of linear MDP.
- In Section A.4, we introduces the definitions of Locality Sensitive Hashing data structures and their applications in nearest neighbor search.
- In Section A.5, we list the probabilistic tools used in our work.
- In Section A.6, we list the inequalities to help the proof.

A.1 Basic Notations

We use $\Pr[]$ to denote probability and $\mathbb{E}[]$ to denote expectation if it exists.

For a matrix A, we use $\|A\|_F := (\sum_{i,j} A_{i,j}^2)^{1/2}$ to denote the Frobenius norm of A, we use $\|A\|_1 := \sum_{i,j} |A_{i,j}|$ to denote the entry-wise ℓ_1 norm of A, we use $\|A\|$ to denote the spectral norm of A. We say matrix $A \in \mathbb{R}^{d \times d}$ is a positive semidefinite matrix if for all $x \in \mathbb{R}^d$, $x^\top A x \geq 0$. We say matrix $A \in \mathbb{R}^{d \times d}$ is a positive definite matrix if for all $x \in \mathbb{R}^d$, $x^\top A x > 0$.

For a vector x, we use $\|x\|_2 := (\sum_i x_i^2)^{1/2}$ to denote the ℓ_2 norm of x, we use $\|x\|_1 := \sum_i |x_i|$ to denote the ℓ_1 norm of x, we use $\|x\|_\infty$ to denote the ℓ_∞ norm.

For a vector $x \in \mathbb{R}^d$ and a psd matrix $A \in \mathbb{R}^{d \times d}$, we use $\|x\|_A := (x^\top A x)^{1/2}$ to denote the matrix norm of x over A.

We use $\mathbb{S}^{d-1}$ to denote the unit sphere.

A.2 Notations and Definitions

In this section, we present the notation and definitions for reinforcement learning. We summarize our notations in Table 2.

We start with the definition of the Episodic Markov decision process.

Definition A.1 (Episodic Markov decision process (episodic MDP)). *Let* MDP$(\mathcal{S}, \mathcal{A}, H, \mathbb{P}, r)$ denote the episodic Markov decision process, where $\mathcal{S}$ denotes the set of available states, $\mathcal{A}$ denotes the set of available actions, $H \in \mathbb{N}$ denotes the total number of steps in each episode, $\mathbb{P} = \{\mathbb{P}_h\}_{h=1}^H$ with $\mathbb{P}_h[s'|s,a]$ denotes the probability of transition from state $s \in \mathcal{S}$ to state $s' \in \mathcal{S}$ when take actions $a \in \mathcal{A}$ at step h, $r = \{r_h\}_{h=1}^H$ denotes the reward obtained at each step. Here the reward r_h is a function that maps $\mathcal{S} \times \mathcal{A}$ to $[0.55, 1]$[v]

Note that for any reward range $[a, b]$, there exists a shift c and scaling α so that $(a + c)/\alpha = 0.55$ and $(b + c)/\alpha = 1$ The shift in reward is designed for sublinear runtime in maximum inner product search. We will provide more discussion in Section B.5.

In this work, we focus on linear Markov decision process (linear MDP). In this setting, each pair of state-action is represented as an embedding vector. Moreover, the transition probability $\mathbb{P}_h[s'|s,a]$ and reward function r_h are linear in this embedding vector.

Definition A.2 (Linear MDP (Bradtke and Barto, 1996; Melo and Ribeiro, 2007)). *The* MDP$(\mathcal{S}, \mathcal{A}, H, \mathbb{P}, r)$ becomes a linear MDP if there exists a function $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ and an unknown signed measure set $\mu_h = (\mu_h^{(1)}, \ldots, \mu_h^{(d)})$ over $\mathcal{S}$ such that the transition probability $\mathbb{P}_h[s'|s,a] = \langle \phi(s,a), \mu_h(s') \rangle$ at any step any $h \in [H]$. Here we assume $\max_{(s,a)\in\mathcal{S}\times\mathcal{A}} \|\phi(s,a)\|_2 \leq 1$. Moreover, there exists a hidden vector $\theta_h \in \mathbb{R}^d$ so that $r_h(s,a) = \langle \phi(s,a), \theta_h \rangle$. Here we assume $\max_{h\in[H]}\{\|\mu_h(\mathcal{S})\|_2, \|\theta_h\|_2\} \leq \sqrt{d}$.

In the MDP framework, we define the policy π as a sequence of functions that map state to actions.

Definition A.3 (Policy). *Given a MDP with form* MDP$(\mathcal{S}, \mathcal{A}, H, \mathbb{P}, r)$, a policy $\pi = \{\pi_1, \cdots, \pi_H\}$ is defined as sequence such that $\pi_h : \mathcal{S} \to \mathcal{A}$ for each step h. $\pi_h(s) = a$ represents the action taken when we are at state s and step h.

Moreover, we use $V_h^\pi(s) : \mathcal{S} \to \mathbb{R}$ to define the value of cumulative rewards in expectation if the agent follows received under a given policy π when the start state is s and the start step is h.

Definition A.4 (Value function). *Given a MDP with form* MDP$(\mathcal{S}, \mathcal{A}, H, \mathbb{P}, r)$, we let the value function be:

$$V_h^\pi(s) := \mathbb{E}\left[\sum_{h'=h}^H r_h(s'_h, \pi_h(s'_h)) \,\middle|\, s_h = s\right], \qquad \forall s \in \mathcal{S}, h \in [H].$$

Further more, we define the Q function $Q_h^\pi(s,a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ as the expected cumulative rewards if a agent follows policy π and starts from takeing action a at state s and step h. This representation of $Q_h^\pi(s,a)$ is also associated with the well-known Bellman equation (Sutton and Barto, 2018).

Definition A.5 (Q-Learning). *Let* MDP$(\mathcal{S}, \mathcal{A}, H, \mathbb{P}, r)$ denote an episodic MDP. We use a simplified notation $[\mathbb{P}_h V_{h+1}](s,a) := \mathbb{E}_{s'\sim\mathbb{P}_h[s'|s,a]}[V_{h+1}(s')]$. Then, we represent the Bellman equation with policy π as

$$Q_h^\pi(s,a) = [r_h + \mathbb{P}_h V_{h+1}^\pi](s,a), \qquad V_h^\pi(s) = Q_h^\pi(s, \pi_h(s)), \qquad V_{H+1}^\pi(s) = 0.$$

Similarly, for optimal policy π^*, we have

$$Q_h^*(s,a) = [r_h + \mathbb{P}_h V_{h+1}^*](s,a), \qquad V_h^*(s) = \max_{a\in\mathcal{A}} Q_h^*(s,a), \qquad V_{H+1}^*(s) = 0. \tag{5}$$

Note that as $r_h \in [0,1]$. All Q_h^π and V_h^π are upper bounded by $H + 1 - h$.

After formulate the MDP and its value functions, we start listing conditions on the space of state and action for the convenience of our Sublinear LSVI and Sublinear LSVI-UCB. We first present the definition for the convex hull.

Definition A.6 (Convex hull). *Given a set $\{x_1, x_2, \cdots, x_n\} \subset \mathbb{R}^d$ that denotes as a matrix $A \in \mathbb{R}^{d\times n}$, we define its convex hull $\mathcal{B}(A)$ to be the collection of all finite linear combinations y that satisfies $y = \sum_{i=1}^n a_i \cdot x_i$, where $a_i \in [0,1]$ for all $i \in [n]$ and $\sum_{i\in[n]} a_i = 1$.*

[v] Note that in standard reinforcement learning, we assume reward is $[0, 1]$, but it is completely reasonable to do a shift. We will provide more discussion in Section 5.1.

In this work, we focus on the Sublinear LSVI under continuous state and action space. Given the action space $\mathcal{A}$ and state space $\mathcal{S}$, we formulate $\phi((\mathcal{S} \times \mathcal{A}))$ as the convex hull of $\phi(\mathcal{S}_{\text{core}} \times \mathcal{A}_{\text{core}})$, where $\mathcal{S}_{\text{core}}$ is core state set and $\mathcal{A}_{\text{core}}$ is core action set.

Definition A.7 (Core state and core action sets)**.** *Given a linear MDP with form* MDP$(\mathcal{S}, \mathcal{A}, H, \mathbb{P}, r)$, we define set $\mathcal{S}_{\text{core}} \subset \mathcal{S}$ as the core states set and $\mathcal{A}_{\text{core}} \subset \mathcal{A}$ as the core action set. We denote cardinality of $\mathcal{S}_{\text{core}}$ and $\mathcal{A}_{\text{core}}$ as S and A. Specifically, we have $\mathcal{B}(\phi(\mathcal{S}_{\text{core}} \times \mathcal{A}_{\text{core}})) = \phi(\mathcal{S} \times \mathcal{A})$. Without loss of generality, we let $A \geq d$.

In LSVI (Bradtke and Barto, 1996), the value iteration procedure requires a span matrix that contains state-action embeddings. Moreover, there also exists a series of assumptions on the span matrix. We provide these assumptions as below:

Definition A.8 (Span matrix)**.** *Given a linear MDP with form* MDP$(\mathcal{S}, \mathcal{A}, H, \mathbb{P}, r)$, we define the span matrix $\Phi \in \mathbb{R}^{d \times M}$ as follows: in total $M \leq d$ columns, the jth column is denoted as $\phi_j = \phi(s_j, a_j)$, where $(s_j, a_j) \in \mathcal{S} \times \mathcal{A}$. Moreover, $\{\phi_1, \phi_2, \cdots, \phi_M\}$ is the linear span of $\phi(S \times A)$. Specifically, Φ satisfies:

- $\phi(s, a) = \sum_{j=1}^{M} w_j \phi_j$, $w_j \in \mathbb{R}$ for all $(s, a) \in \mathcal{S} \times \mathcal{A}$,
- $\text{rank}(\Phi) = M$,
- $\max_{(s,a) \in \mathcal{S} \times \mathcal{A}} \|\Phi^{-1} \phi(s, a)\|_1 \leq L$.

Next, we follow Jin et al. (2020) and making assumptions for Sublinear LSVI-UCB. Given a linear MDP with form MDP$(\mathcal{S}, \mathcal{A}, H, \mathbb{P}, r)$, we assume $\mathcal{S}$ is finite with cardinally S and $\mathcal{A}$ is finite with cardinally A.

A.3 Standard Properties of Linear MDP

We list the tools for analyzing linear MDPs properties from Jiang et al. (2021) in this section.

Lemma A.9 (Proposition 2.3 (Jin et al., 2020))**.** *The Q function with form $Q_h^\pi(s, a)$ in linear MDP could be represented it as a inner product $Q_h^\pi(s, a) = \langle \phi(s, a), w_h^\pi \rangle$, where $w_h^\pi \in \mathbb{R}^d$ is a weight vector.*

Next, we show the upper bound of weight w_h^π for any policy π.

Lemma A.10 (Lemma B.2 (Jin et al., 2020))**.** *Given a linear MDP, let w_h^π denote the weight that achieves $Q_h^\pi(s, a) = \langle \phi(s, a), w_h^\pi \rangle$ for all $(s, a) \in \mathcal{S} \times \mathcal{A}$ at step $h \in [H]$. We show that for $\|w_h^\pi\|_2 \leq 2H\sqrt{d}$ for any $h \in [H]$,*

A.4 Locality Sensitive Hashing

We define Locality Sensitive Hashing (LSH). These definitions are very standard, e.g., see Indyk and Motwani (Indyk and Motwani, 1998).

Definition A.11 (Locality Sensitive Hashing)**.** *Let* dist *denote a metric distance. Let $\overline{c}$ denote a parameter such that $\overline{c} > 1$. Let p_1, p_2 denote two parameters such that $0 < p_2 < p_1 < 1$. A family $\mathcal{H}$ is called $(r, \overline{c} \cdot r, p_1, p_2)$-sensitive if and only if, for any two point $x, y \in \mathbb{R}^d$, a function h chosen uniformly from the family $\mathcal{H}$ has the following properties:*

- *if* $\text{dist}(x, y) \leq r$*, then* $\Pr_{h \sim \mathcal{H}}[h(x) = h(y)] \geq p_1$,
- *if* $\text{dist}(x, y) \geq \overline{c} \cdot r$*, then* $\Pr_{h \sim \mathcal{H}}[h(x) = h(y)] \leq p_2$.

We focus on situations where dist is ℓ_2 or cosine distance.

LSH is designed to accelerate the runtime of the Approximate Nearest Neighbor (ANN) problem. We start with define the exact NN problem as:

Definition A.12 (Exact Nearest Neighbor (NN))**.** *Given an n-point dataset $Y \subset \mathbb{S}^{d-1}$ on the sphere, the goal of the Nearest Neighbor (NN) problem is to find a datapoint $y \in Y$ for a query $x \in \mathbb{S}^{d-1}$ such that*

$$\text{NN}(x, Y) := \min_{y \in Y} \|x - y\|_2.$$

Indyk and Motwani (1998) relax the NN problem in Definition A.12 as with approximation and define the Approximate Nearest Neighbor (ANN) problem.

Definition A.13 (Approximate Nearest Neighbor (ANN)). *Let $\overline{c} > 1$. Let $r \in (0, 2)$. Given an n-point dataset $P \subset \mathbb{S}^{d-1}$ on the sphere, the $(\overline{c}, r)$-Approximate Near Neighbor Search (ANN) aims at developing a data structure that, given a query $q \in \mathbb{S}^{d-1}$ with the promise that there exists a datapoint $p \in P$ with $\|p - q\|_2 \leq r$, the data structure reports a datapoint $p' \in P$ with distance less than $\overline{c} \cdot r$ from q.*

Then, the query complexity of ANN is reduced to sublinear by LSH following Theorem A.14 and Theorem A.15. Note that here we write $O(1/\sqrt{\log n})$ as $o(1)$.

Theorem A.14 (Andoni and Razenshteyn (Andoni and Razenshteyn, 2015)). *Let $\overline{c} > 1$ and $r \in (0, 2)$. The $(\overline{c}, r)$-ANN on a unit sphere $\mathbb{S}^{d-1}$ can be solved by a data structure with query time $O(d \cdot n^{\rho})$, space $O(n^{1+\rho} + dn)$ and preprocessing time $O(dn^{1+\rho})$, where $\rho = \frac{1}{2\overline{c}^2 - 1} + o(1)$.*

Theorem A.15 (Andoni, Laarhoven, Razenshteyn and Waingarten (Andoni et al., 2017a)). *Let $\overline{c} > 1$. Let $r \in (0, 2)$. There exists a data structure that solves $(\overline{c}, r)$-ANN on the unit sphere $\mathbb{S}^{d-1}$ with query time $O(d \cdot n^{\rho})$, space $O(n^{1+o(1)} + dn)$ and preprocessing time $O(dn^{1+o(1)})$, where $\rho = \frac{2}{\overline{c}^2} - \frac{1}{\overline{c}^4} + o(1)$.*

In this work, we focus on the Max-IP, which is a well-known problem in the field of computational complexity, we follow the standard notation in this work Chen (2018). We define the exact and approximate Max-IP problem as follows:

Definition A.16 (Exact Max-IP). *Given a data set $Y \subseteq \mathbb{R}^d$, we define Max-IP for a query point $x \in \mathbb{R}^d$ with respect to Y as follows:*

$$\text{Max-IP}(x, Y) := \max_{y \in Y} \langle x, y \rangle.$$

Definition A.17 (Approximate Max-IP). *Let $c \in (0, 1)$ and $\tau \in (0, 1)$. Given an n-point dataset $Y \subset \mathbb{S}^{d-1}$, the (c, τ)-Max-IP aims at building a data structure that, given a query $x \in \mathbb{S}^{d-1}$ with the promise that there exists a datapoint $y \in Y$ with $\langle x, y \rangle \geq \tau$, the data structure reports a datapoint $z \in Y$ with similarity $\langle x, z \rangle$ greater than $c \cdot \text{Max-IP}(x, Y)$.*

To solve (c, τ)-Max-IP, we define a dual version of LSH data structure (Shrivastava and Li (Shrivastava and Li, 2014) call it asymmetric LSH):

Definition A.18 (Locality Sensitive Hashing for similarity). *Let c denote a parameter such that $c \in (0, 1)$. Let τ denote a parameter such that $\tau > 0$. Let p_1, p_2 denote two parameters such that $0 < p_2 < p_1 < 1$. Let $\text{sim}(x, y)$ denote a binary similarity function between $x, y \in \mathbb{R}^d$. A family $\mathcal{H}$ is called $(\tau, c \cdot \tau, p_1, p_2)$-sensitive if and only if, for any query point $x \in \mathbb{R}^d$ and a data point $y \in \mathbb{R}^d$, h chosen uniformly from $\mathcal{H}$ has the following properties:*

- *if $\text{sim}(x, y) \geq \tau$ then $\Pr_{h \sim \mathcal{H}}[h(x) = h(y)] \geq p_1$,*
- *if $\text{sim}(x, y) \leq c \cdot \tau$ then $\Pr_{h \sim \mathcal{H}}[h(x) = h(y)] \leq p_2$.*

It is shown from Shrivastava and Li (2014) that LSH type data structure with asymmetric transformations could achieve sublinear runtime complexity of (c, τ)-Max-IP.

A.5 Probabilistic Tools

Lemma A.19 (Hoeffding bound (Hoeffding, 1963)). *Let $x_1, \cdots, x_n$ be n independent bounded variables in $[a_i, b_i]$. Let , then we show the Hoeffding bound over $x = \sum_{i=1}^n x_i$ as:*

$$\Pr[|x - \mathbb{E}[x]| \geq t] \leq 2 \exp\Big(- \frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2} \Big).$$

A.6 Inequalities

In this sections, we present the supporting inequalities for our work.

Fact A.20 (Lemma D.1 in Jin et al. (2020)). *Given a matrix $\Lambda_t = \lambda \mathbf{I}_d + \sum_{i=1}^t \phi_i \phi_i^\top$ with $\phi_i \in \mathbb{R}^d$ and $\lambda > 0$, we show that:*

$$\sum_{i=1}^t \phi_i^\top (\Lambda_t)^{-1} \phi_i \leq d.$$

Lemma A.21 (Lemma D.4 in Jin et al. (2020)). *Let $\mathcal{V}$ denote a function family that $\max_{V\in\mathcal{V},x\in\mathcal{S}}|V(x)|\leq H$. Let G denote the ϵ-covering number of $\mathcal{V}$. Let $\mathcal{S}$ denote a state space. Let $\{\mathcal{F}_\tau\}_{\tau=0}^\infty$ denote the filtration of $\mathcal{S}$. Let $\{x_\tau\}_{\tau=1}^\infty$ denote a random process defined on $\mathcal{S}$. Let $\{\phi_\tau\}_{\tau=0}^\infty$ denote a real valued random process in $\mathbb{R}^d$. Moreover, $\phi_\tau\in\mathcal{F}_{\tau-1}$ and we have upper bound $\|\phi_\tau\|_2\leq 1$. Given a matrix $\Lambda_k\in\mathbb{R}^{d\times d}$ so that $\Lambda_k=\lambda I_d+\sum_{\tau=1}^k\phi_\tau\phi_\tau^\top$, for any $\delta>0$, for any $k\geq 0$, for any $V\in\mathcal{V}$, we have*

$$\Big\|\sum_{\tau=1}^k\phi_\tau(V(x_\tau)-\mathbb{E}[V(x_\tau)\mid\mathcal{F}_{\tau-1}])\Big\|_{\Lambda_k^{-1}}^2\leq 4H^2\big(d\log(1+k/\lambda)+\log(G_\epsilon/\delta)\big)+8k^2\epsilon^2/\lambda.$$

B DATA STRUCTURES

This section presents the data Structures for our work.

- In Section B.1, we introduce the transformations that build primal-dual connections between approximate Max-IP and ANN.
- In Section B.2, we present our data structure that achieves sublinear query time in approximate Max-IP.
- In Section B.3, we show how to perform approximate Max-MatNorm via approximate Max-IP data structure.
- In Section B.4, we present our efficient transformations for Max-IP in optimization.
- In Section B.5, we formally provide the theoretical results of sublinear approximate Max-IP using one LSH data structure.
- In Section B.6, we provide the theoretical results of sublinear approximate Max-IP using another LSH data structure.

B.1 Existing Transformation from Primal to Dual

In this section, we show a transformation that builds the connection between Max-IP and NN. Under this asymmetric transformation, NN is formulated as a dual problem of Max-IP.

We start with presenting the asymmetric transformation.

Definition B.1 (Asymmetric transformation (Neyshabur and Srebro, 2015)). *Let $Y\in\mathbb{R}^d$ and $\|y\|_2\leq 1$ for all $y\in Y$. Let $x\in\mathbb{R}^d$ and $\|x\|_2\leq D_x$. We define the following asymmetric transform:*

$$\begin{aligned} P(y) &= \begin{bmatrix} y^\top & \sqrt{1-\|y\|_2^2} & 0\end{bmatrix}^\top, \\ Q(x) &= \begin{bmatrix}(xD_x^{-1})^\top & 0 & \sqrt{1-\|xD_x^{-1}\|_2^2}\end{bmatrix}^\top. \end{aligned} \tag{6}$$

Therefore, we have

$$\|Q(x)-P(y)\|_2^2=2-2D_x^{-1}\langle x,y\rangle,\quad \arg\max_{y\in Y}\langle x,y\rangle=\arg\min_{y\in Y}\|Q(x)-P(y)\|^2.$$

In this way, we regard Max-IP as the primal problem and NN as a dual problem.

B.2 Sublinear Max-IP Data Structure

In this section, we show the theorem that provides sublinear query time for Max-IP problem using LSH type data structure.

Theorem B.2 (Formal statement of Corollary 3.5). *Let $c\in(0,1)$ and $\tau\in(0,1)$. Given a set of n-points $Y\subset\mathcal{S}^{d-1}$ on the sphere, one can build a data structure with preprocessing time $\mathcal{T}_{\mathsf{init}}$ and space $\mathcal{S}_{\mathsf{space}}$ so that for any query $x\in\mathcal{S}^{d-1}$, we take $O(d\cdot n^\rho)$ query time:*

- *if* Max-IP$(x,Y)\geq\tau$, *then we output a vector in Y which is a (c,τ)-*Max-IP *with respect to (x,Y) with probability at least 0.9*[vi], *where $\rho:=f(c,\tau)+o(1)$.*

[vi] It is obvious to boost probability from constant to δ by repeating the data structure $\log(1/\delta)$ times.

- *otherwise, we output* fail.

Further,

- *If* $\mathcal{T}_{\mathsf{init}} = O(dn^{1+\rho})$ *and* $\mathcal{S}_{\mathsf{space}} = O(n^{1+\rho} + dn)$*, then* $f(c,\tau) = \frac{1-\tau}{1-2c\tau+\tau}$.
- *If* $\mathcal{T}_{\mathsf{init}} = O(dn^{1+o(1)})$ *and* $\mathcal{S}_{\mathsf{space}} = O(n^{1+o(1)} + dn)$*, then* $f(c,\tau) = \frac{2(1-\tau)^2}{(1-c\tau)^2} - \frac{(1-\tau)^4}{(1-c\tau)^4}$.

Proof. We start with showing that for any two points x, y with $\|x\|_2 = \|y\|_2 = 1$, we have $\|x-y\|_2^2 = 2 - 2\langle x, y\rangle$. This implies that $r^2 = 2 - 2\tau$ for a $(\overline{c}, r)$-ANN and a (c,τ)-Max-IP on x, Y.

Further, if we have a data structure for $(\overline{c}, r)$-ANN, it automatically becomes a data structure for (c,τ)-Max-IP with parameters $\tau = 1 - 0.5r^2$ and $c = \frac{1-0.5\overline{c}^2 r^2}{1-0.5r^2}$. This implies that

$$\overline{c}^2 = \frac{1 - c(1-0.5r^2)}{0.5r^2} = \frac{1-c\tau}{1-\tau}.$$

Next, we show how to solve (c,τ)-Max-IP by solving $(\overline{c}, r)$-ANN using two different data structures.

Part 1. If we initialize the data-structure following Theorem A.14, we show that the (c,τ)-Max-IP on a unit sphere $\mathcal{S}^{d-1}$ can be solved by solving $(\overline{c}, r)$-ANN with query time $O(d \cdot n^{\rho})$, space $O(n^{1+\rho} + dn)$ and preprocessing time $O(dn^{1+\rho})$, where

$$\rho = \frac{1}{2\overline{c}^2 - 1} + o(1) = \frac{1}{2\frac{1-c\tau}{1-\tau} - 1} + o(1) = \frac{1-\tau}{1-2c\tau+\tau} + o(1).$$

Thus, $f(c,\tau) = \frac{1-\tau}{1-2c\tau+\tau}$.

Part 2. If we initialize the data-structure following Theorem A.15, we show that the (c,τ)-Max-IP on a unit sphere $\mathcal{S}^{d-1}$ can be solved by solving $(\overline{c}, r)$-ANN with query time $O(d \cdot n^{\rho})$, space $O(n^{1+o(1)} + dn)$ and preprocessing time $O(dn^{1+o(1)})$, where

$$\rho = \frac{2}{\overline{c}^2} - \frac{1}{\overline{c}^4} + o(1) = \frac{2(1-\tau)^2}{(1-c\tau)^2} - \frac{(1-\tau)^4}{(1-c\tau)^4} + o(1).$$

Thus, $f(c,\tau) = \frac{2(1-\tau)^2}{(1-c\tau)^2} - \frac{(1-\tau)^4}{(1-c\tau)^4}$.

□

In practice, we tune parameter τ close to Max-IP(x, Y) to achieve higher c. Moreover, Theorem B.2 could be applied to general Max-IP problem. To do this, we first apply asymmetric transformation in Definition B.1 and transfer it to a (c,τ)-Max-IP problem over $Q(x)$ and $Q(Y)$. Then, we solve this (c,τ)-Max-IP problem by solving its dual problem, which is $(\overline{c}, r)$-ANN. Finally, the solution to the $(\overline{c}, r)$-ANN would be the approximate solution to the original Max-IP(x, Y). Meanwhile, it is reasonable for us to regard $d = n^{o(1)}$ using Johnson-Lindenstrauss Lemma Johnson and Lindenstrauss (1984).

B.3 Sublinear Max-IP Data Structure for Maximum Matrix Norm Search

In this section, we extend LSH type Max-IP data structure for maximum matrix norm search.

Definition B.3 (Exact Maximum Matrix Norm (Max-MatNorm))**.** *Given a data set* $Y \subseteq \mathbb{R}^d$ *and a query matrix* $x \in \mathbb{R}^{d \times d}$*, we define Maximum Matrix Norm as follows:*

$$\text{Max-MatNorm}(x, Y) := \max_{y \in Y} \|y\|_x.$$

Next, we define the approximate version of the Maximum Matrix Norm.

Definition B.4 (Approximate Max-MatNorm)**.** *Let $c \in (0,1)$ and $\tau \in (0,1)$. Let vec denote the vectorization of $d \times d$ matrix into a d^2 vector. Given an n-point dataset $Y \subset \mathbb{R}^d$ and $yy^\top \in \mathbb{S}^{d^2-1}$ for all $y \in Y$, the goal of the (c,τ)-*Max-MatNorm *is to cosntruct a data structure that, given a query matrix $x \in \mathbb{R}^{d \times d}$ and $\mathrm{vec}(x) \in \mathbb{S}^{d^2-1}$ with the promise that there exists a datapoint $y \in Y$ with $\|y\|_x \geq \tau$, it reports a datapoint $z \in Y$ with $\|z\|_x \geq c \cdot$ Max-MatNorm(x, Y).*

Next, we show the relationship between Max-MatNorm and Max-IP

Lemma B.5 (Relation between Max-MatNorm and Max-IP)**.** *We show that*

$$\textsf{Max-MatNorm}(X,Y)^2 = \max_{y \in Y} \langle \mathrm{vec}(x), \mathrm{vec}(yy^\top) \rangle$$

where vec *vectorizes $d \times d$ matrix x into a d^2 vector.*

Proof. We show that

$$\begin{aligned} \textsf{Max-MatNorm}(x,Y)^2 = & \max_{y \in Y} \|y\|_x^2 \\ = & \max_{y \in Y} y^\top x y \\ = & \max_{y \in Y} \langle \mathrm{vec}(x), \mathrm{vec}(yy^\top) \rangle \end{aligned}$$

where the first step follows the definition of Max-MatNorm, the second step follows from the definition of $\|y\|_x^2$, the third step decomposes the quadratic form into a inner product.

□

Next, we present our main theorem for Max-MatNorm(x, Y).

Theorem B.6. *Let c denote a parameter such that $c \in (0,1)$. Let τ denote a parameter such that $\tau \in (0,1)$. Let vec denote the vectorization of $d \times d$ matrix into a d^2 vector. Given a n-points set $Y \subseteq \mathbb{R}^d$ and $yy^\top \in \mathbb{S}^{d^2-1}$ for all $y \in Y$, one can construct a data structure with $\mathcal{T}_{\mathsf{init}}$ preprocessing time and $\mathcal{S}_{\mathsf{space}}$ so that for any query matrix $x \in \mathbb{R}^{d \times d}$ with $\mathrm{vec}(x) \in \mathbb{S}^{d^2-1}$, we take query time complexity $O(d^2 n^\rho \cdot \log(1/\delta))$:*

- *if* Max-MatNorm$(x, Y) \geq \tau$*, then we output a vector in Y which is a (c,τ)-*Max-MatNorm *with respect to (x, Y) with probability at least $1 - \delta$, where $\rho := f(c,\tau) + o(1)$.*
- *otherwise, we output* fail*.*

Further,

- *If $\mathcal{T}_{\mathsf{init}} = O(d^2 n^{1+\rho} \cdot \log(1/\delta))$ and $\mathcal{S}_{\mathsf{space}} = O((n^{1+\rho} + d^2 n) \cdot \log(1/\delta))$, then $f(c,\tau) = \frac{1-\tau^2}{1-c^2\tau^2+\tau^2}$.*
- *If $\mathcal{T}_{\mathsf{init}} = O(d^2 n^{1+o(1)} \cdot \log(1/\delta))$ and $\mathcal{S}_{\mathsf{space}} = O((n^{1+o(1)} + d^2 n) \cdot \log(1/\delta))$, then $f(c,\tau) = \frac{2(1-\tau^2)^2}{(1-c^2\tau^2)^2} - \frac{(1-\tau^2)^4}{(1-c^2\tau^2)^4}$.*

Proof. We start with showing that if we have a (c^2, τ^2)-Max-IP data structure over $\mathrm{vec}(x)$ and every $\mathrm{vec}(yy^\top)$, $y \in Y$, we would obtain a $z \in Y$ such that

$$\langle \mathrm{vec}(x), \mathrm{vec}(zz^\top) \rangle \geq c^2 \max_{y \in Y} \langle \mathrm{vec}(x), \mathrm{vec}(yy^\top) \rangle, \tag{7}$$

we could use it and derive the following propriety for z:

$$\begin{aligned} \|z\|_x = & \sqrt{\langle \mathrm{vec}(x), \mathrm{vec}(zz^\top) \rangle} \\ \geq & \sqrt{c^2 \max_{y \in Y} \langle \mathrm{vec}(x), \mathrm{vec}(yy^\top) \rangle} \end{aligned}$$

$$= c \max_{y \in Y} \sqrt{\langle \mathrm{vec}(x), \mathrm{vec}(yy^\top) \rangle}$$
$$= c \max_{y \in Y} \|y\|_x$$

where the second step follows from Eq. (7).

Therefore, z is the solution for (c, τ)-Max-MatNorm(x, Y).

Next, we show how to retrieve z via two data structures used for (c, τ)-Max-IP(x, Y) in Theorem B.2.

Part 1. If we initialize the data structure following Theorem A.14, we can construct a data structure with $O((n^{1+\rho} + d^2 n) \cdot \log(1/\delta))$ preprocessing time and $O((n^{1+\rho} + d^2 n) \cdot \log(1/\delta))$ space so that for any query matrix $x \in \mathbb{R}^{d \times d}$ with $\mathrm{vec}(x) \in \mathbb{S}^{d^2-1}$, we take query time complexity $O(d^2 n^\rho \cdot \log(1/\delta))$ to retrieve z. Here $\rho = \frac{1-\tau^2}{1-c^2\tau^2+\tau^2} + o(1)$ and we are able to improve the failure probability to δ by repeating the LSH for $\log(1/\delta)$ times.

Part 2. If we initialize the data structure following Theorem A.15, we can construct a data structure with $O((n^{1+o(1)} + d^2 n) \cdot \log(1/\delta))$ preprocessing time and $O((n^{1+o(1)} + dn) \cdot \log(1/\delta))$ space so that for any query matrix $x \in \mathbb{R}^{d \times d}$ with $\mathrm{vec}(x) \in \mathbb{S}^{d^2-1}$, we take query time complexity $O(d^2 n^\rho \cdot \log(1/\delta))$ to retrieve z. Here $\rho = \frac{2(1-\tau^2)^2}{(1-c^2\tau^2)^2} - \frac{(1-\tau^2)^4}{(1-c^2\tau^2)^4} + o(1)$ and we also improve the failure probability to δ by repeating the LSH for $\log(1/\delta)$ times.

□

Moreover, Theorem B.6 could be applied to general Max-MatNorm problem. To do this, we first apply transform (c, τ)-Max-MatNorm problem into a (c^2, τ^2)-Max-IP problem using Lemma B.5. Next, we apply transformations in Definition B.1 and transfer the (c^2, τ^2)-Max-IP problem to a (c^2, τ^2)-Max-IP problem over $Q(x)$ and $Q(Y)$. Then, we solve this (c^2, τ^2)-Max-IP problem by solving its dual problem, which is $(\overline{c}, r)$-ANN. Finally, the solution to the $(\overline{c}, r)$-ANN would be the approximate solution to the original Max-MatNorm(x, Y).

B.4 Transformation for Efficient Query

In the optimization problem that could be accelerated by (c, τ)-Max-IP, the query and data vectors are usually not unit vectors so that we apply transformations in Definition B.1 to map both query and data vectors into unit vectors. However, if the mapped inner product is too close to 1. The formulation of ρ would break and the time complexity would be linear. To avoid this, we propose a new set of asymmetric transformations:

Definition B.7 (Efficient asymmetric transformation)**.** *Let $Y \in \mathbb{R}^d$ and $\|y\|_2 \leq 1$ for all $y \in Y$. Let $x \in \mathbb{R}^d$ and $\|x\|_2 \leq D_x$. We define the following asymmetric transform:*

$$P(y) = \begin{bmatrix} y^\top & \sqrt{1 - \|y\|_2^2} & 0 \end{bmatrix}^\top, \quad Q(x) = \begin{bmatrix} \frac{0.8 \cdot x^\top}{D_x} & 0 & \sqrt{1 - \frac{0.64 \cdot \|x\|_2^2}{D_x^2}} \end{bmatrix}^\top.$$

Next, we use Lemma B.8 to show how to enforce τ to be away from 1 via our efficient asymmetric transformation.

Lemma B.8. *Given the transformation P and Q defined in Definition B.7, we show that both* Max-IP$(Q(x), P(Y))$ *and* NN$(Q(x), P(Y))$ *are equivalent to* Max-IP(x, Y)*. Moreover,*

$$\text{Max-IP}(Q(x), P(Y)) \leq 0.8.$$

Proof. Using transformations in Definition B.7, for all $y \in Y$, we have

$$Q(x)^\top P(y) = \frac{0.8 \cdot x^\top y}{D_x} \leq 0.8 \cdot \frac{\|x\|_2 \|y\|_2}{D_x} \leq 0.8$$

where the third step follows from $\|x\|_2 \leq D_x$ and $\|y\|_2 \leq 1$.

Next, we show that Max-IP$(Q(x), P(Y))$ is equivalent to Max-IP(x, Y).

$$\arg\max_{y \in Y} Q(x)^\top P(y) = \arg\max_{y \in Y} \frac{0.8 \cdot \langle x, y \rangle}{D_x} = \arg\max_{y \in Y} \langle x, y \rangle.$$

Further more, $\mathsf{NN}(Q(x), P(Y))$ (see Definition A.12) is equivalent to $\mathsf{Max\text{-}IP}(x, Y)$.

$$\|Q(x) - P(y)\|_2^2 = 2 - 1.6 D_x^{-1} \langle x, y \rangle, \quad \arg\min_{y \in Y} \|Q(x) - P(y)\|^2 = \arg\max_{y \in Y} \langle x, y \rangle.$$

□

B.5 Sublinear Query Time: Part 1

In this section, we show that ρ is strictly less than 1 using LSH in Andoni and Razenshteyn (2015).

Lemma B.9. *If* LSH *data structure's parameters* c *and* τ *satisfy that* $c \in [0.5, 1)$ *and* $\tau \in [0.5, 1)$ *then, we could upper bound* ρ *as:*

$$\rho < 1 - \frac{\gamma}{2} + O(1/\sqrt{\log n})$$

where $\gamma = 1 - c$.

Proof. We can upper bound ρ as follows:

$$\begin{aligned}
\rho &= \frac{1-\tau}{1-2c\tau+\tau} + O(1/\sqrt{\log n}) \\
&= 1 - \frac{2\tau - 2c\tau}{1-2c\tau+\tau} + O(1/\sqrt{\log n}) \\
&= 1 - (1-c) \cdot \frac{2\tau}{1-2c\tau+\tau} + O(1/\sqrt{\log n}) \\
&\leq 1 - (1-c) \cdot \frac{1}{1-2c\tau+\tau} + O(1/\sqrt{\log n}) && \text{by } \tau \geq 0.5 \\
&< 1 - (1-c) \cdot \frac{1}{2} + O(1/\sqrt{\log n}) && \text{by } \tau < 1 \\
&= 1 - \frac{\gamma}{2} + O(1/\sqrt{\log n})
\end{aligned}$$

where the second and third steps are reorganizations, the forth step follows from $\tau \geq 0.5$, the fifth step follows from $\tau < 1$ and $c \geq 0.5$, the last step is a reorganization.

Therefore, we complete the proof. □

For Sublinear LSVI, we set $c = 1 - C_0 L \sqrt{\iota/n}$ and and $\tau \geq 0.5$ by shifting the reward function. In this way, we have

$$\rho < 1 - \frac{C_0 L \sqrt{\iota/n}}{2} + O(\frac{1}{\sqrt{\log A}}) < 1 - \frac{1}{4} C_0 L \sqrt{\iota/n} \tag{8}$$

where the first step follows from $\gamma = 1 - c = C_0 L \sqrt{\iota/n}$, the second step follows from $\frac{1}{4} C_0 L \sqrt{\iota/n} > \Omega(\frac{1}{\sqrt{\log A}})$.

For Sublinear LSVI-UCB, we set $c = 1 - \frac{1}{\sqrt{K}}$ and $\tau \geq 0.5$ by shifting the reward function. In this way, we have

$$\rho < 1 - \frac{1}{2\sqrt{K}} + O(\frac{1}{\sqrt{\log A}}) < 1 - \frac{1}{4\sqrt{K}} \tag{9}$$

where the first step follows from $\gamma = 1 - c = \frac{1}{\sqrt{K}}$, the second step follows from $\frac{1}{4\sqrt{K}} > \Omega(\frac{1}{\sqrt{\log A}})$.

Therefore, we show that sublinear value iteration can be achieved while preserving the same regret.

B.6 Sublinear Query Time: Part 2

In this section, we show that ρ is strictly less than 1 using LSH in Andoni et al. (2017a).

Using Andoni et al. (2017a), the ρ for LSH based $\mathsf{Max\text{-}IP}$ data structure with parameters c and τ becomes

$$\rho = \frac{2(1-\tau)^2}{(1-c\tau)^2} - \frac{(1-\tau)^4}{(1-c\tau)^4} + o(1)$$

where is a function over c and τ.

To upper bound the ρ, we start with showing that it is decreasing as τ increase when $c \in [0.5, 1)$ and $\tau \in [0.5, 1)$.

Lemma B.10. *Let $c \in [0.5, 1)$ and $\tau \in [0.5, 1)$. We show that function*

$$f(c, \tau) := \frac{2(1-\tau)^2}{(1-c\tau)^2} - \frac{(1-\tau)^4}{(1-c\tau)^4},$$

is decreasing as τ increase.

Proof. We take the derivative of $f(c, \tau)$ in τ and get

$$\frac{\partial}{\partial \tau} f(c, \tau) = -\frac{4(c-1)^2(\tau-1)\tau(c\tau+\tau-2)}{(1-c\tau)^5} < 0,$$

where the second step follows from $c \in [0.5, 1)$ and $\tau \in [0.5, 1)$.

Thus, $f(c, \tau)$ is decreasing as τ increase when $c \in [0.5, 1)$ and $\tau \in [0.5, 1)$.

□

Next, we have our results in upper bounding ρ.

Lemma B.11. *If* LSH *data structure's parameters c and τ satisfy that $c \in [0.5, 1)$ and $\tau \in [0.5, 1)$ then, we could upper bound ρ as:*

$$\rho < 1 - \frac{\gamma^2}{4} + O(1/\sqrt{\log n}),$$

where $\gamma = 1 - c$.

Proof. Let $\gamma = 1 - c$, we have

$$\begin{aligned}
\rho &= \frac{2}{\overline{c}^2} - \frac{1}{\overline{c}^4} + O(1/\sqrt{\log n}) \\
&= \frac{2(1-\tau)^2}{(1-c\tau)^2} - \frac{(1-\tau)^4}{(1-c\tau)^4} + O(1/\sqrt{\log n}) \\
&\leq \frac{0.5}{(1-0.5c)^2} - \frac{0.0625}{(1-0.5c)^4} + O(1/\sqrt{\log n}) \\
&= \frac{0.5}{(0.5+0.5\gamma)^2} - \frac{0.0625}{(0.5+0.5\gamma)^4} + O(1/\sqrt{\log n}) \\
&= \frac{2}{(1+\gamma)^2} - \frac{1}{(1+\gamma)^4} + O(1/\sqrt{\log n}) \\
&= \frac{2+4\gamma+2\gamma^2-1}{(1+\gamma)^4} + O(1/\sqrt{\log n}) \\
&= \frac{1+4\gamma+2\gamma^2}{(1+\gamma)^4} + O(1/\sqrt{\log n}) \\
&= 1 - \frac{4\gamma^2+4\gamma^3+\gamma^4}{(1+\gamma)^4} + O(1/\sqrt{\log n}) \\
&< 1 - \frac{4\gamma^2}{(1+\gamma)^4} + O(1/\sqrt{\log n}) && \text{by } \gamma > 0 \\
&< 1 - \frac{\gamma^2}{4} + O(1/\sqrt{\log n}) && \text{by } \gamma < 1,
\end{aligned}$$

where the second step follows from $\overline{c}^2 = \frac{1-c\tau}{1-\tau}$, the third step follows from that ρ is monotonic decrease as τ increase and $\tau \geq 0.5$, the forth to eighth steps are reorganizations, the ninth step follows from $\gamma = 1 - c > 0$, the tenth step follows from $\gamma = 1 - c < 1$.

□

For Sublinear LSVI, we set $c = 1 - C_0 L\sqrt{\iota/n}$ and and $\tau \geq 0.5$ by shifting the reward function. In this way, we have

$$\rho < 1 - \frac{C_0^2 L^2 \iota}{4n} + O(\frac{1}{\sqrt{\log A}}) < 1 - \frac{1}{8} C_0^2 L^2 \iota / n, \tag{10}$$

where the first step follows from $\gamma = 1 - c = C_0 L\sqrt{\iota/n}$, the second step follows from $\frac{1}{8} C_0^2 L^2 \iota / n > \Omega(\frac{1}{\sqrt{\log A}})$.

For Sublinear LSVI-UCB, we set $c = 1 - \frac{1}{\sqrt{K}}$ and $\tau \geq 0.5$ by shifting the reward function. In this way, we have

$$\rho < 1 - \frac{1}{4K} + O(\frac{1}{\sqrt{\log A}}) < 1 - \frac{1}{8K}, \tag{11}$$

where the first step follows from $\gamma = 1 - c = \frac{1}{\sqrt{K}}$, the second step follows from $\frac{1}{8K} > \Omega(\frac{1}{\sqrt{\log A}})$.

Therefore, we show that sublinear value iteration can be achieved while preserving the same regret.

C SUBLINEAR LEAST-SQUARES VALUE ITERATION

This section presents the Sublinear Least-Squares Value Iteration (Sublinear LSVI)

- In Section C.1, we introduce the Sublinear LSVI algorithm.
- In Section C.2, we provide the upper bound of the difference between the optimal value function and the estimated value function.
- In Section C.3, we present the regret analysis of Sublinear LSVI.
- In Section C.4, we perform a runtime analysis on the building blocks of Sublinear LSVI to analyze its efficiency.
- In Section C.5, we compare Sublinear LSVI with LSVI (Bradtke and Barto, 1996) in regret and value iteration complexity.

C.1 Algorithm

We present our Sublinear LSVI algorithm in Algorithm 1. We summarize our algorithm as several steps: (1) sample collection: we query a pair of state and action in the span matrix for n times at each step and observe its reward and next state, (2) data structure construction, we preprocess the embeddings for state and action pairs and build a nearest neighbor data structure, (3) we perform least-squares solver to estimate the weight in the linear MDP model, (4) we use LSH for value function estimation, (5) we construct policy based on the estimated value function.

C.2 Value Difference

In this section, we provide the tools for regret analysis. The goal of this section is to prove Lemma C.1.

Lemma C.1. *Let* MDP$(\mathcal{S}, \mathcal{A}, H, \mathbb{P}, r)$ denote a linear MDP. Let $V_1^*(s)$ be the optimal value function defined in Definition A.5. Let $\widehat{V}_1(s)$ be the estimated value function defined in Definition A.5. We show that via Algorithm 1, the difference $V_1^*(s) - \widehat{V}_1(s)$ is upper bounded by:

$$V_1^*(s) - \widehat{V}_1(s) \leq \mathop{\mathbb{E}}_{\pi^*} \Big[\sum_{h=1}^{H} [(\mathbb{P}_h - \widehat{\mathbb{P}}_h)\widehat{V}_{h+1}](s_h, a_h) | s_1 = s \Big] + \frac{1-c}{2} \cdot H(H+1), \tag{12}$$

where c is the parameter for Max-IP.

Proof. We start with lower bounding $\widehat{V}_h(s)$ as

$$\begin{aligned} \widehat{V}_h(s) &\geq c \cdot \max_{a \in \mathcal{A}_{core}} \langle \widehat{w}_h, \phi(s,a) \rangle \\ &= c \max_{a \in \mathcal{A}} \widehat{Q}_h(s,a), \end{aligned} \tag{13}$$

where the first step follows from Theorem B.2, the second step follows from the definition of $\widehat{Q}_h(s,a)$ in Definition A.5 and the definition of convex hull.

Next, we upper bound $V_h^*(s) - \widehat{V}_h(s)$ as

$$\begin{aligned} V_h^*(s) - \widehat{V}_h(s) &= \max_{a \in \mathcal{A}} Q_h^*(s,a) - \widehat{V}_h(s) \\ &\leq \max_{a \in \mathcal{A}} Q_h^*(s,a) - c \max_{a \in \mathcal{A}} \widehat{Q}_h(s,a) \\ &\leq Q_h^*(s, \pi^*(s)) - c \max_{a \in \mathcal{A}} \widehat{Q}_h(s,a) \\ &\leq Q_h^*(s, \pi^*(s)) - c \widehat{Q}_h(s, \pi^*(s)) \\ &= c \Big(Q_h^*(s, \pi^*(s)) - \widehat{Q}_h(s, \pi^*(s)) \Big) + (1-c) Q_h^*(s, \pi^*(s)) \end{aligned}$$

Algorithm 1 Sublinear LSVI

data structure LSH ▷ Theorem B.2
  INIT($S \subset \mathbb{R}^d$, $n \in \mathbb{N}$, $d \in \mathbb{N}$, $c \in (0,1)$, $\tau \in (0,1)$)
    ▷ $|S| = n$, c, τ is the approximate Max-IP parameter and d is the dimension of data
  QUERY($x \in \mathbb{R}^d$)
end data structure

procedure SUBLINEARLSVI($\mathcal{S}_{\text{core}}$, $\mathcal{A}_{\text{core}}$, $N \in \mathbb{N}$, $H \in \mathbb{N}$, $c_{\mathsf{LSH}} \in (0,1)$, $\tau_{\mathsf{LSH}} \in (0,1)$)
    ▷ $\mathcal{S}_{\text{core}}$ and $\mathcal{A}_{\text{core}}$ are in Definition A.7
  /*Collect Samples*/
  for step $h \in [H]$ **do**
    $\mathcal{D}_h \leftarrow \emptyset$
    for $j = 1, \cdots, M$ **do** ▷ For each column in the span matrix defined in Definition A.8
      for $l = 1, \cdots, n$ **do** ▷ Play n times
        Query (s_j, a_j) at step h, observe the next state s'_{jl}.
        ▷ s_j, a_j defined in Definition A.8
        $\mathcal{D}_h \leftarrow \mathcal{D}_h \cup \{(s_j, a_j, s'_{jl})\}$ ▷ $|\mathcal{D}_h| = Mn$
      end for
    end for
  end for
  /*Preprocess data and build a nearest neighbor data structure*/
    ▷ This step takes $O(S \cdot (A^{1+\rho} + dA))$
  for $s \in \mathcal{S}_{\text{core}}$ **do**
    $\Phi_s \leftarrow \{\phi(s,a) | \forall a \in \mathcal{A}_{\text{core}}\}$
    static LSH LSH_s
    LSH_s.INIT($\Phi_s, A, d, c_{\mathsf{LSH}}, \tau_{\mathsf{LSH}}$)
  end for
  /*Precompute Λ matrix*/ ▷ This step takes $O(Md^2 + d^\omega)$
  $\Lambda \leftarrow n \sum_{j=1}^M \phi(s_j, a_j) \phi(s_j, a_j)^\top$
  Compute Λ_h^{-1}
  /*Update value function*/ ▷ This step takes $O(H(d^2 + Md + Mn + SdA^\rho))$
  for step $h = H, \ldots, 1$ **do**
    $\widehat{w}_h \leftarrow \Lambda^{-1} \sum_{(\dot{s}, \dot{a}, \dot{s}_l') \in \mathcal{D}_h} \phi(\dot{s}, \dot{a}) \left(r_h(\dot{s}, \dot{a}) + \widehat{V}_{h+1}(\dot{s}_l') \right)$
    for all $s \in \mathcal{S}_{\text{core}}$ **do**
      $a \leftarrow \text{LSH}_s$.QUERY($\widehat{w}_h$)
      $\widehat{V}_h(s) \leftarrow \langle \widehat{w}_h, \phi(s,a) \rangle$
    end for
  end for
  /*Construct policy*/ ▷ This step takes $O(HSdA)$
  policy $\widehat{\pi} \leftarrow \emptyset$
  for step $h = 1, \ldots, H$ **do**
    $\widehat{\pi}_h(s) \leftarrow \arg\max_{a \in \mathcal{A}_{\text{core}}} \langle \widehat{w}_h, \phi(s,a) \rangle$ for all $s \in \mathcal{S}_{\text{core}}$
  end for
  return $\widehat{\pi}$
end procedure

$$
\begin{aligned}
&\leq c\Big(Q_h^*(s, \pi^*(s)) - \widehat{Q}_h(s, \pi^*(s))\Big) + (1-c)(H+1-h) \\
&\leq \Big(Q_h^*(s, \pi^*(s)) - \widehat{Q}_h(s, \pi^*(s))\Big) + (1-c)(H+1-h), \qquad (14)
\end{aligned}
$$

where the first step follows from $V_h^*(s) = \max_{a \in \mathcal{A}} Q_h^*(s,a)$, the second step follows from Eq. (13), the third step follows from $\max_{a \in \mathcal{A}} Q_h^*(s,a) = Q_h^*(s, \pi^*(s))$ and the forth step follows from $\max_{a \in \mathcal{A}} \widehat{Q}_h(s,a) \geq \widehat{Q}_h(s, \pi^*(s))$, the fifth step is an reorganization, the sixth step follows the upper bound for Q_h^* in Definition A.5, the seventh step follows from $c \in (0,1)$ and c is close to 1.

Next, we can write the difference $Q_h^*(s,a) - \widehat{Q}_h(s,a)$ as,

$$\begin{aligned} Q_h^*(s,a) - \widehat{Q}_h(s,a) &= [r_h + \mathbb{P}_h V_{h+1}^*](s,a) - [r_h + \widehat{\mathbb{P}}_h \widehat{V}_{h+1}](s,a) \\ &= [\mathbb{P}_h V_{h+1}^*](s,a) - [\widehat{\mathbb{P}}_h \widehat{V}_{h+1}](s,a) \\ &= [\mathbb{P}_h V_{h+1}^*](s,a) - [\mathbb{P}_h \widehat{V}_{h+1}](s,a) + [\mathbb{P}_h \widehat{V}_{h+1}](s,a) - [\widehat{\mathbb{P}}_h \widehat{V}_{h+1}](s,a) \\ &= [\mathbb{P}_h (V_{h+1}^* - \widehat{V}_{h+1})](s,a) + [(\mathbb{P}_h - \widehat{\mathbb{P}}_h)\widehat{V}_{h+1}](s,a), \end{aligned} \tag{15}$$

where the first step follows from the definition of $Q_h(s,a)$ in Definition A.5, the second step follows from eliminating the common term $r_h(s,a)$, the third step follows from inserting an additional term $[\mathbb{P}_h \widehat{V}_{h+1}](s,a)$, and the last step is a reorganization.

Combining Eq. (14) and Eq. (15), we have

$$\begin{aligned} & V_h^*(s) - \widehat{V}_h(s) \\ \leq & \left(Q_h^*(s, \pi^*(s)) - \widehat{Q}_h(s, \pi^*(s)) \right) + (1-c)(H+1-h) \\ = & [\mathbb{P}_h (V_{h+1}^* - \widehat{V}_{h+1})](s, \pi^*(s)) + [(\mathbb{P}_h - \widehat{\mathbb{P}}_h)\widehat{V}_{h+1}](s, \pi^*(s)) + (1-c)(H+1-h) \\ = & \mathop{\mathbb{E}}_{\pi^*} \left[(V_{h+1}^* - \widehat{V}_{h+1})(s_{h+1}) \,\middle|\, s_h = s \right] + \mathop{\mathbb{E}}_{\pi^*} \left[[(\mathbb{P}_h - \widehat{\mathbb{P}}_h)\widehat{V}_{h+1}](s_h, a_h) \,\middle|\, s_h = s \right] \\ & + (1-c)(H+1-h) \\ = & (V_{h+1}^* - \widehat{V}_{h+1}) + \mathop{\mathbb{E}}_{\pi^*} \left[[(\mathbb{P}_h - \widehat{\mathbb{P}}_h)\widehat{V}_{h+1}](s_h, a_h) \,\middle|\, s_h = s \right] \\ & + (1-c)(H+1-h), \end{aligned}$$

where the first step follows the Eq. (14), the second step follows the Eq. (15), the third step rewrites both terms into an expectation over π^*, and the last step follows the definition of V_{h+1}^* and $\widehat{V}_{h+1}$.

Using induction from 1 to H, we have

$$\begin{aligned} V_1^*(s) - \widehat{V}_1(s) \leq & \mathop{\mathbb{E}}_{\pi^*} \left[\sum_{h=1}^H [(\mathbb{P}_h - \widehat{\mathbb{P}}_h)\widehat{V}_{h+1}](s_h, a_h) \,\middle|\, s_1 = s \right] + (1-c) \sum_{h=1}^H (H+1-h) \\ = & \mathop{\mathbb{E}}_{\pi^*} \left[\sum_{h=1}^H [(\mathbb{P}_h - \widehat{\mathbb{P}}_h)\widehat{V}_{h+1}](s_h, a_h) \,\middle|\, s_1 = s \right] + \frac{1-c}{2} \cdot H(H+1), \end{aligned}$$

where the second step is a reorganization. □

C.3 Regret Analysis

The goal of this section is to prove Theorem C.2.

Theorem C.2 (Convergence Result of Sublinear Least-Squares Value Iteration (Sublinear LSVI), a formal version of Theorem 4.1)**.** *Given a linear MDP with form* MDP$(\mathcal{S}, \mathcal{A}, H, \mathbb{P}, r)$ with core sets $\mathcal{S}_{\text{core}}$, $\mathcal{A}_{\text{core}}$ defined in Definition A.7, if we chose $n = O(C_0^2 \cdot \epsilon^{-2} L^2 H^4 \iota)$, where $\iota = \log(Hd/p)$ and C_0 is a constant, the Sublinear LSVI (Algorithm 1) with approximate Max-IP parameter $c = 1 - \Theta(L \cdot \sqrt{\iota/n})$ has regret at most $O(LH^2\sqrt{\iota/n})$ with probability at least $1-p$.

Proof. We have two definitions for $\widehat{Q}_h(s,a)$. The first definition is given by Definition A.1, it says

$$\widehat{Q}_h(s,a) = r_h(s,a) + [\widehat{\mathbb{P}}_h \cdot \widehat{V}_{h+1}](s,a). \tag{16}$$

The second definition is given by Definition A.2, it says

$$\widehat{Q}_h(s,a) = \phi(s,a)^\top \widehat{w}_h. \tag{17}$$

Given the second definition, our goal is to derive $\widehat{\mathbb{P}}_h$.

To do this, we write $\widehat{Q}_h(s,a)$ as

$$\begin{aligned}
&\widehat{Q}_h(s,a) \\
=& \phi(s,a)^\top \widehat{w}_h \\
=& \phi(s,a)^\top \Lambda^{-1} \sum_{(\dot{s},\dot{a},\dot{s_l}')\in\mathcal{D}_h} \phi(\dot{s},\dot{a}) \Big(r_h(\dot{s},\dot{a}) + \widehat{V}_{h+1}(\dot{s_l}') \Big) \\
=& \phi(s,a)^\top \Lambda^{-1} \sum_{(\dot{s},\dot{a},\dot{s_l}')\in\mathcal{D}_h} \phi(\dot{s},\dot{a}) \Big(\phi(\dot{s},\dot{a})^\top \theta_h + \widehat{V}_{h+1}(\dot{s_l}') \Big) \\
=& \phi(s,a)^\top \Lambda^{-1} \sum_{(\dot{s},\dot{a},\dot{s_l}')\in\mathcal{D}_h} \phi(\dot{s},\dot{a})\phi(\dot{s},\dot{a})^\top \theta_h + \phi(s,a)^\top \Lambda^{-1} \sum_{(\dot{s},\dot{a},\dot{s_l}')\in\mathcal{D}_h} \phi(\dot{s},\dot{a}) \widehat{V}_{h+1}(\dot{s_l}') \\
=& \phi(s,a)^\top \theta_h + \phi(s,a)^\top \Lambda^{-1} \sum_{(\dot{s},\dot{a},\dot{s_l}')\in\mathcal{D}_h} \phi(\dot{s},\dot{a}) \widehat{V}_{h+1}(\dot{s_l}') \\
=& \phi(s,a)^\top \theta_h + \int \Big(\phi(s,a)^\top \Lambda^{-1} \sum_{(\dot{s},\dot{a},\dot{s_l}')\in\mathcal{D}_h} \phi(\dot{s},\dot{a}) \delta(s',\dot{s_l}') \Big) \widehat{V}_{h+1}(s') \mathrm{d}s' \\
=& r_h(s,a) + \int \Big(\phi(s,a)^\top \Lambda^{-1} \sum_{(\dot{s},\dot{a},\dot{s_l}')\in\mathcal{D}_h} \phi(\dot{s},\dot{a}) \delta(s',\dot{s_l}') \Big) \widehat{V}_{h+1}(s') \mathrm{d}s', \qquad (18)
\end{aligned}$$

where the first step follows the definition of $\widehat{Q}_h(s,a)$ in Definition A.5, the second step follows the definition of $\widehat{w}_h$ in Algorithm 4, the third step follows the definition of reward r_h in Definition A.2, the forth step is an reorganization, the fifth step follows from $\Lambda^{-1} \sum_{(\dot{s},\dot{a},\dot{s_l}')\in\mathcal{D}_h} \phi(\dot{s},\dot{a})\phi(\dot{s},\dot{a})^\top = \mathbf{I}_d$, the sixth step rewrites the second term in a integral format, where $\delta(x,y)$ is a Dirichlet function, the last step follows the definition of reward r_h in Definition A.2.

By comparing Eq. (18) with Eq. (16), we should define $\widehat{\mathbb{P}}_h(s'|s,a)$ as

$$\widehat{\mathbb{P}}_h(s'|s,a) = \phi(s,a)\Lambda^{-1} \sum_{(\dot{s},\dot{a},\dot{s_l}')\in\mathcal{D}_h} \phi(\dot{s},\dot{a})\delta(s',\dot{s_l}'). \qquad (19)$$

Combining Eq. (19) with the definition of $[\widehat{\mathbb{P}}_h \widehat{V}_{h+1}](s,a)$ in Definition A.5.

$$[\widehat{\mathbb{P}}_h \widehat{V}_{h+1}](s,a) = \phi(s,a)^\top \Lambda^{-1} \sum_{(\dot{s},\dot{a},\dot{s_l}')\in\mathcal{D}_h} \phi(\dot{s},\dot{a}) \widehat{V}_{h+1}(\dot{s_l}'). \qquad (20)$$

In the next a few paragraphs, we will explain how to rewrite $[(\mathbb{P}_h - \widehat{\mathbb{P}}_h)\widehat{V}_{h+1}](s,a)$.

$$\begin{aligned}
&[(\mathbb{P}_h - \widehat{\mathbb{P}}_h)\widehat{V}_{h+1}](s,a) \\
=& \phi(s,a)^\top \int \widehat{V}_{h+1}(s') \mathrm{d}\mu(s') - [\widehat{\mathbb{P}}_h \widehat{V}_{h+1}](s,a) \\
=& \phi(s,a)^\top \int \widehat{V}_{h+1}(s') \mathrm{d}\mu(s') - \phi(s,a)^\top \Lambda^{-1} \sum_{(\dot{s},\dot{a},\dot{s_l}')\in\mathcal{D}_h} \phi(\dot{s},\dot{a}) \widehat{V}_{h+1}(\dot{s_l}') \\
=& \phi(s,a)^\top \Lambda^{-1} \sum_{(\dot{s},\dot{a},\dot{s_l}')\in\mathcal{D}_h} \phi(\dot{s},\dot{a})\phi(\dot{s},\dot{a})^\top \int \widehat{V}_{h+1}(s') \mathrm{d}\mu(s') \\
& - \phi(s,a)^\top \Lambda^{-1} \sum_{(\dot{s},\dot{a},\dot{s_l}')\in\mathcal{D}_h} \phi(\dot{s},\dot{a}) \widehat{V}_{h+1}(\dot{s_l}') \\
=& \phi(s,a)^\top \Lambda^{-1} \sum_{(\dot{s},\dot{a},\dot{s_l}')\in\mathcal{D}_h} \phi(\dot{s},\dot{a}) \Big(\phi(\dot{s},\dot{a})^\top \int \widehat{V}_{h+1}(s') \mathrm{d}\mu(s') - \widehat{V}_{h+1}(\dot{s_l}') \Big) \\
=& \phi(s,a)^\top \Lambda^{-1} \sum_{(\dot{s},\dot{a},\dot{s_l}')\in\mathcal{D}_h} \phi(\dot{s},\dot{a}) \Big(\int \widehat{V}_{h+1}(s') \phi(\dot{s},\dot{a})^\top \mathrm{d}\mu(s') - \widehat{V}_{h+1}(\dot{s_l}') \Big) \\
=& \phi(s,a)^\top \Lambda^{-1} \sum_{(\dot{s},\dot{a},\dot{s_l}')\in\mathcal{D}_h} \phi(\dot{s},\dot{a}) \Big(\int \widehat{V}_{h+1}(s') \mathbb{P}_h[s'|\dot{s},\dot{a}] \mathrm{d}s' - \widehat{V}_{h+1}(\dot{s_l}') \Big)
\end{aligned}$$

$$= \phi(s,a)^\top \Lambda^{-1} \sum_{(\dot{s},\dot{a},\dot{s_l}')\in\mathcal{D}_h} \phi(\dot{s},\dot{a})\Big(\mathbb{E}[\widehat{V}_{h+1}(s')|\dot{s},\dot{a}] - \widehat{V}_{h+1}(s_i')\Big), \tag{21}$$

where the first step follows the definition of $\mathbb{P}_h[s'|s_i,a_i] = \phi(s_i,a_i)\mu_h(s')$, the second step follows Eq. (20), the third step adds the $\Lambda^{-1}\sum_{(\dot{s},\dot{a},\dot{s_l}')\in\mathcal{D}_h}\phi(\dot{s},\dot{a})\phi(\dot{s},\dot{a})^\top = \mathbf{I}_d$ to the left term, the forth and fifth steps are reorganizations, the sixth step follows the definition of $\mathbb{P}_h$ in Definition A.2, the last step follows the definition of expectation.

Next, we rewrite $\sum_{(\dot{s},\dot{a},\dot{s_l}')\in\mathcal{D}_h}\phi(\dot{s},\dot{a})$ as

$$\sum_{(\dot{s},\dot{a},\dot{s_l}')\in\mathcal{D}_h} \phi(\dot{s},\dot{a}) = n\sum_{j=1}^M \phi_(s_j,a_j) = n\sum_{j=1}^M \phi_j, \tag{22}$$

where the first steps follows from Algorithm 1 that for each $\phi(s_j,a_j)$, we query it n times and put all $\{(s_j,a_j,s_{j1}'),\cdots,(s_j,a_j,s_{jn}')\}$ in $\mathcal{D}_h$, the second step follows by $\phi_j = \phi(s_j,a_j)$ in Definition A.7.

Next, we rewrite Λ as

$$\begin{aligned}\Lambda &= \sum_{(\dot{s},\dot{a},\dot{s_l}')\in\mathcal{D}_h}\phi(\dot{s},\dot{a})\phi(\dot{s},\dot{a})^\top \\ &= n\sum_{j=1}^M \phi_(s_j,a_j)\phi_(s_j,a_j)^\top \\ &= n\Phi\Phi^\top,\end{aligned} \tag{23}$$

where the first steps follows by the definition of Λ in Algorithm 1,the second steps follows from Algorithm 1 that for each $\phi(s_j,a_j)$, we query it n times and put all $\{(s_j,a_j,s_{j1}'),\cdots,(s_j,a_j,s_{jn}')\}$ in $\mathcal{D}_h$, the third step follows from the definition of Φ in Definition A.7.

Combining Eq. (23) with Eq. (21), we get

$$[(\mathbb{P}_h - \widehat{\mathbb{P}}_h)\widehat{V}_{h+1}](s,a) = \phi(s,a)^\top(n\Phi\Phi^\top)^{-1}\sum_{(\dot{s},\dot{a},\dot{s_l}')\in\mathcal{D}_h}\phi(\dot{s},\dot{a})\Big(\mathbb{E}[\widehat{V}_{h+1}(s')|\dot{s},\dot{a}] - \widehat{V}_{h+1}(s_i')\Big). \tag{24}$$

Next, we further bound $[(\mathbb{P}_h - \widehat{\mathbb{P}}_h)\widehat{V}_{h+1}](s,a)$ as:

$$\begin{aligned}&[(\mathbb{P}_h - \widehat{\mathbb{P}}_h)\widehat{V}_{h+1}](s,a) \\ &= \phi(s,a)^\top(n\Phi\Phi^\top)^{-1}\sum_{(\dot{s},\dot{a},\dot{s_l}')\in\mathcal{D}_h}\phi(\dot{s},\dot{a})\Big(\mathbb{E}[\widehat{V}_{h+1}(s')|\dot{s},\dot{a}] - \widehat{V}_{h+1}(s_i')\Big) \\ &= \phi(s,a)^\top(n\Phi\Phi^\top)^{-1}n\sum_{j=1}^M\phi(s_j,a_j)\sum_{l=1}^n\Big(\mathbb{E}[\widehat{V}_{h+1}(s')|s_j,a_j] - \widehat{V}_{h+1}(s_{jl}')\Big) \\ &= \phi(s,a)^\top(\Phi\Phi^\top)^{-1}\sum_{j=1}^M\phi(s_j,a_j)\Big(\mathbb{E}[\widehat{V}_{h+1}(s')|s_j,a_j] - \frac{1}{n}\sum_{l=1}^n\widehat{V}_{h+1}(s_{jl}')\Big) \\ &= \phi(s,a)^\top(\Phi\Phi^\top)^{-1}\sum_{j=1}^M\phi_j\Big(\mathbb{E}[\widehat{V}_{h+1}(s')|s_j,a_j] - \frac{1}{n}\sum_{l=1}^n\widehat{V}_{h+1}(s_{jl}')\Big),\end{aligned}$$

where the first step follows from Eq. (24), the second steps follows from Algorithm 1 that for each $\phi(s_j,a_j)$, we query it n times and put all $\{(s_j,a_j,s_{j1}'),\cdots,(s_j,a_j,s_{jn}')\}$ in $\mathcal{D}_h$, the third step is an reorganization, the last step follows the definition of ϕ_j in Definition A.7.

For each $j\in[M]$, we define random variable

$$z_j := \mathbb{E}[\widehat{V}_{h+1}(s')|\phi_j] - \frac{1}{n}\sum_{l=1}^n\widehat{V}_{h+1}(s_{jl}').$$

By Hoefding Inequality in Lemma A.19, we can show

$$|z_j| \leq C_0 \cdot H \cdot \sqrt{\iota/n}$$

For convenient, we define vector $z \in \mathbb{R}^M$ to be $z := [z_1, \cdots, z_M]$.

Now, we can upper bound $[(\mathbb{P}_h - \widehat{\mathbb{P}}_h)\widehat{V}_{h+1}](s,a)$ as follows:

$$\begin{aligned}
[(\mathbb{P}_h - \widehat{\mathbb{P}}_h)\widehat{V}_{h+1}](s,a) &= \phi(s,a)^\top (\Phi\Phi^\top)^{-1}\Phi z \\
&= \phi(s,a)^\top (\Phi^\dagger)^\top z \\
&= \Big(\Phi^\dagger \phi(s,a)\Big)^\top z \\
&\leq \|\Phi^\dagger \phi(s,a)\|_1 \cdot \|z\|_\infty \\
&\leq L \cdot C_0 \cdot H \cdot \sqrt{\iota/n}, \qquad (25)
\end{aligned}$$

where the first step follows the $\sum_{j=1}^M \phi_j z_j = \Phi z$, the second step is an reorganization, the third step follows the holders inequality, the last step uses the bound for $\|\Phi^{-1}\phi(s,a)\|_1$ in Definition A.7, Definition A.8 and $\|z_j\|_2$.

Combining Eq. (25) with Lemma C.1, we could upper bound $V_1^*(s) - \widehat{V}_1(s)$

$$\begin{aligned}
V_1^*(s) - \widehat{V}_1(s) \leq & \mathop{\mathbb{E}}_{\pi^*}\Big[\sum_{h=1}^H [(\mathbb{P}_h - \widehat{\mathbb{P}}_h)\widehat{V}_{h+1}](s_h,a_h)|s_1 = s\Big] + \frac{1-c}{2}\cdot H(H+1) \\
&\leq H \cdot L \cdot C_0 \cdot H \cdot \sqrt{\iota/n} + \frac{1-c}{2}\cdot H(H+1) \\
&= L \cdot C_0 \cdot H^2 \cdot \sqrt{\iota/n} + \frac{1-c}{2}\cdot H(H+1) \\
&\leq L \cdot C_0 \cdot H^2 \cdot \sqrt{\iota/n} + (1-c)H^2 \\
&\leq 2C_0 L H^2 \sqrt{\iota/n} \\
&\leq \epsilon,
\end{aligned}$$

where the first step follows from Lemma C.1, the second step follows the upper bound of $[(\mathbb{P}_h - \widehat{\mathbb{P}}_h)\widehat{V}_{h+1}](s,a)$ in Eq. 25, the third step is an reorganization, the forth step follows from $H \geq 1$ so that $H^2 \geq H$, the fifth step follows from $1 - c = C_0 L\sqrt{\iota/n}$, the sixth step follows from $n = O(C_0^2 \cdot \epsilon^{-2} L^2 H^4 \iota)$. Here we choose c that maintain the level of regret.

□

C.4 Running Time Analysis

Lemma C.3. *The running time of pre-computing Λ^{-1} takes*

$$O(Md^2 + d^\omega).$$

Proof. It takes $O(Md^2)$ to sum up every $\phi(s_j,a_j)\phi(s_j,a_j)^\top$. It takes $O(d)$ constant to multiply the sum results by n. Computing the inverse matrix of Λ takes $O(d^\omega)$. Combining the complexity together, we obtain the pre-computing complexity $O(Md^2 + d^\omega)$. □

Lemma C.4. *The running time of updating value takes*

$$O(H \cdot (d^2 + Md + Mn + SdA^\rho)).$$

Further more,

- *If initialize the* LSH *data-structure using Theorem A.14, $\rho = 1 - \frac{1}{4}C_0 L\sqrt{\iota/n}$.*

Algorithm 2 LSVI (Bradtke and Barto, 1996)

procedure LSVI($\mathcal{S}$, $\mathcal{A}$, $N \in \mathbb{N}$, $H \in \mathbb{N}$) ▷ $\mathcal{S}$ and $\mathcal{A}$ are in Definition A.2
  /*Collect Samples*/
  for $h \in [H]$ **do**
    $\mathcal{D}_h \leftarrow \emptyset$
    for $j = 1, \cdots, M$ **do** ▷ For each element in the span set defined in Definition A.8
      for $l = 1, \cdots, n$ **do** ▷ Play n times
        Query (s_j, a_j) at step h, observe the next state s'_{jl}.
        ▷ s_j, a_j defined in Definition A.8
        $\mathcal{D}_h \leftarrow \mathcal{D}_h \cup \{(s_j, a_j, s'_{jl})\}$ ▷ $|\mathcal{D}_h| = Mn$
      end for
    end for
  end for
  /*Precompute Λ matrix*/ ▷ This step takes $O(Md^2 + d^\omega)$
  $\Lambda \leftarrow n \sum_{j=1}^M \phi(s_j, a_j)\phi(s_j, a_j)^\top$ ▷ $\Lambda \in \mathbb{R}^{d \times d}$
  Compute Λ^{-1}
  /*Update value function*/ ▷ This step takes $O(H(d^2 + Md + Mn + SAd))$
  for $h = H, \ldots, 1$ **do**
    $\widehat{w}_h \leftarrow \Lambda^{-1} \sum_{(\dot{s}, \dot{a}, \dot{s}_l{}') \in \mathcal{D}_h} \phi(\dot{s}, \dot{a}) \left(r_h(\dot{s}, \dot{a}) + \widehat{V}_{h+1}(\dot{s}_l{}') \right)$
    for all $s \in \mathcal{S}_{\mathrm{core}}$ **do**
      $\widehat{V}_h(s) \leftarrow \max_{a \in \mathcal{A}_{\mathrm{core}}} \langle \widehat{w}_h, \phi(s, a) \rangle$
    end for
  end for
  /*Construct policy*/ ▷ This step takes $O(HSAd)$
  policy $\widehat{\pi} \leftarrow \emptyset$
  for $h = 1, \ldots, H$ **do**
    $\widehat{\pi}_h(s) \leftarrow \arg\max_{a \in \mathcal{A}_{\mathrm{core}}} \langle \widehat{w}_h, \phi(s, a) \rangle$ for all $s \in \mathcal{S}$
  end for
  return $\widehat{\pi}$
end procedure

- *If initialize the* LSH *data-structure using Theorem A.15,* $\rho = 1 - \frac{1}{8} C_0^2 L^2 \iota / n$.

Proof. We can rewrite $\widehat{w}_h$ as follows:

$$\begin{aligned} \widehat{w}_h &= \Lambda^{-1} \sum_{(\dot{s}, \dot{a}, \dot{s}_l{}') \in \mathcal{D}_h} \phi(\dot{s}, \dot{a}) \left(r_h(\dot{s}, \dot{a}) + \widehat{V}_{h+1}(\dot{s}_l{}') \right) \\ &= \Lambda^{-1} n \sum_{j=1}^M \phi(s_j, a_j)(r_h(s_j, a_j) + \frac{1}{n} \sum_{l=1}^n \widehat{V}_{h+1}(s'_{jl})), \end{aligned}$$

where the second step follows the definition of $\mathcal{D}_h$.

For each of the H step,

- It takes $O(SdA^\rho)$ to compute $\widehat{V}_h(s'_{jl})$ for each state $s_j \in \mathcal{S}_{\mathrm{core}}$. If we initialize the LSH data-structure using Theorem A.14, we determine $\rho = 1 - \frac{1}{4} C_0 L \sqrt{\iota / n}$ using Lemma B.9. If we initialize the LSH data-structure using Theorem A.15, we determine $\rho = 1 - \frac{1}{8} C_0^2 L^2 \iota / n$ using Lemma B.10.
- It takes $O(Mn)$ to compute $r_h(s_j, a_j) + \frac{1}{n} \sum_{l=1}^n \widehat{V}_{h+1}(s'_{jl})$ for the total n number of s'_{jl} observed by (s_j, a_j).
- It takes $O(Md)$ to sum up the M dimensional vector $\phi(s_j, a_j)(r_h(s_j, a_j) + \frac{1}{n} \sum_{l=1}^n \widehat{V}_{h+1}(s'_{jl}))$.
- It takes $O(d^2)$ to multiply Λ with the sum of vectors.

- All other operations take $O(d)$.

Combining the complexity together and multiply by H steps, we finish the proof. □

Lemma C.5. *The running time of constructing policy takes*

$$O(HSdA)$$

Proof. For each step, it takes $O(SdA)$ to find the optimal action. Thus, it takes $O(HSdA)$ for inference. □

C.5 Comparison

In this section, we show the comparison between our Sublinear LSVI with LSVI Bradtke and Barto (1996).

We start with presenting the LSVI algorithm in Algorithm 2.

Next, we show the comparison results in Table 3.

Table 3: Comparison between Our Sublinear LSVI with LSVI. Let S and A denote the cardinality of $\mathcal{S}_{core}$ and $\mathcal{A}_{core}$. Let d denote the dimension of $\phi(s,a)$. Let H be the number of steps played in each episode. Let n denote the quantity of times played for each pair of core state-action. Let L denote the constant in Definition A.8. Let $\iota = \log(Hd/p)$ and p is the failure probability. Let $\rho_1 = 1 - \frac{1}{4}C_0 L\sqrt{\iota/n}$ be the parameter of data structures in Theorem A.14 and $\rho_2 = 1 - \frac{1}{8}C_0^2 L^2 \iota/n$ be the parameter of data structure Theorem A.15. This table is a detailed version of corresponding part of Table 1.

Algorithm	**Preprocess**	**#Value Iteration**	**Regret**
Ours	$O(SdA^{1+\rho_1})$	$O(HSdA^{\rho_1})$	$O(C_0 LH^2\sqrt{\iota/n})$
Ours	$O(SdA^{1+o(1)})$	$O(HSdA^{\rho_2})$	$O(C_0 LH^2\sqrt{\iota/n})$
LSVI	0	$O(HSdA)$	$O(C_0 LH^2\sqrt{\iota/n})$

D SUBLINEAR LEAST-SQUARES VALUE ITERATION WITH UCB

This section extend the Sublinear LSVI with UCB exploration.

- In Section D.1, we present the Sublinear LSVI-UCB algorithm.
- In Section D.2, we define several simplified notations for the convenience of proof.
- In Section D.3, we provide the upper bound of weight estimated by Sublinear LSVI-UCB.
- In Section D.4, we introduce a modified version of net argument for Sublinear LSVI-UCB.
- In Section D.5, we upper bound the fluctuation on the value function when performing Sublinear LSVI-UCB Algorithm.
- In Section D.6, we provide the upper bound on the difference between the estimated Q function and the actual Q function.
- In Section D.7, we given the upper bound on the difference between the estimated Q function and the actual Q function at the first step using induction.
- In Section D.8, we introduce the recursion formula for the regret analysis.
- In Section D.9, we formally provide the regret analysis of LSVI-UCB.
- In Section D.10, we analyze the runtime Sublinear LSVI-UCB by calculating the time complexity for each block.
- In Section D.11, we compare Sublinear LSVI-UCB with LSVI-UCB (Jin et al., 2020) in terms of regret and value iteration complexity.

In the following sections we show how to tackle the problem and provide our Sublinear LSVI-UCB. Moreover, we provide the regret analysis of our Sublinear LSVI-UCB.

D.1 Algorithm

In LSVI-UCB (Jin et al., 2020) with large action space, the runtime in each value iteration step is dominated by by computing the estimated value function as below:

$$\widehat{V}_h(s_{h+1}^\tau) = \max_{a \in \mathcal{A}} \min\{\langle w_h^k, \phi(s_{h+1}^\tau, a)\rangle + \beta \cdot \|\phi(s_{h+1}^\tau, a)\|_{\Lambda_h^{-1}}, H\} \tag{26}$$

where w_h^k is computed by solving the least-squares problem and $\phi(s_{h+1}^\tau, a)$ is the embedding for a pair of state-action. The complexity for Eq. (26) is $O(d^2 A)$

The key challenge of Sublinear LSVI-UCB here is that Eq. (2) cannot be formulated as a Max-IP problem.

To handle this, we demonstrate how to develop Sublinear LSVI-UCB algorithm. We start with bounding the Q function in Jin et al. (2020) as

Lemma D.1. *We show that*

$$\min\{\|\phi(s_{h+1}^\tau, a)\|_{\beta^2 \Lambda_h^{-1} + w_h^k (w_h^k)^\top}, H\} \leq Q_h(s_{h+1}^\tau, a) \leq \min\{\|\phi(s_{h+1}^\tau, a)\|_{2\beta^2 \Lambda_h^{-1} + 2 w_h^k (w_h^k)^\top}, H\}.$$

Proof. We start with rewriting $Q_h(s_{h+1}^\tau, a)$,

$$Q_h(s_{h+1}^\tau, a) = \min\{w_h^\top \phi(s_{h+1}^\tau, a) + \beta \cdot \|\phi(s_{h+1}^\tau, a)\|_{\Lambda_h^{-1}}, H\}.$$

Next, we show that

$$w_h^\top \phi(s_{h+1}^\tau, a) + \beta \cdot \|\phi(s_{h+1}^\tau, a)\|_{\Lambda_h^{-1}} \leq \sqrt{2(w_h^\top \phi(s_{h+1}^\tau, a))^2 + 2\beta^2 \cdot \|\phi(s_{h+1}^\tau, a)\|_{\Lambda_h^{-1}}^2}$$

$$= \|\phi(s_{h+1}^\tau, a)\|_{2\beta^2\Lambda_h^{-1}+2w_h^k(w_h^k)^\top},$$

where the first step follows from Cauchy-Schwartz inequality, the second step is an reorganization.

Next, we show that

$$\begin{aligned} w_h^\top \phi(s_{h+1}^\tau, a) + \beta \cdot \|\phi(s_{h+1}^\tau, a)\|_{\Lambda_h^{-1}} &\geq \sqrt{(w_h^\top \phi(s_{h+1}^\tau, a))^2 + \beta^2 \cdot \|\phi(s_{h+1}^\tau, a)\|_{\Lambda_h^{-1}}^2} \\ &= \|\phi(s_{h+1}^\tau, a)\|_{\beta^2\Lambda_h^{-1}+w_h^k(w_h^k)^\top}, \end{aligned}$$

where the first step follows from the fact that both $w_h^\top \phi(s_{h+1}^\tau, a)$ and $\|\phi(s_{h+1}^\tau, a)\|_{\Lambda_h^{-1}}$ are non-negative, the second step is an reorganization.

Finally, consider the propriety of $\min$ function, we finish the proof of the lemma.

□

Algorithm 3 Modified LSVI-UCB

1: **for** $k = 1, \ldots, K$ **do**
2: Initialize the state to s_1^k.
3: **for** $h = H, \ldots, 1$ **do**
4: /*Compute Λ_h^{-1}*/ ▷ This step takes $O(Kd^2 + d^\omega)$
5: $\Lambda_h \leftarrow \sum_{\tau=1}^{k-1} \phi(s_h^\tau, a_h^\tau)\phi(s_h^\tau, a_h^\tau)^\top + \lambda \cdot \mathbf{I}_d$.
6: Compute Λ_h^{-1}
7: /* Value Iteration*/ ▷ This takes $O(AKd^2)$
8: $w_h^k \leftarrow \Lambda_h^{-1} \sum_{\tau=1}^{k-1} \phi(s_h^\tau, a_h^\tau) \cdot (r_h(s_h^\tau, a_h^\tau) + \widehat{V}_{h+1}(s_{h+1}^\tau))$
9: **for** $\tau = 1, \cdots, k-1$ **do**
10: **for** $a \in \mathcal{A}$ **do**
11: $Q_h(s_{h+1}^\tau, a) \leftarrow \min\{\|\phi(s_{h+1}^\tau, a)\|_{2\beta^2\Lambda_h^{-1}+2w_h^k w_h^{k\top}}, H\}$.
12: **end for**
13: $\widehat{V}_h(s_h^\tau) \leftarrow \max_{a\in\mathcal{A}} Q_h(s_h^\tau, a)$
14: $a_h^\tau \leftarrow \arg\max_{a\in\mathcal{A}} Q_h(s, a)$ ▷ a_h^τ is the maximum value action taken at state s_h^τ.
15: **end for**
16: **end for**
17: /* Construct Policy*/
18: **for** $h = 1, \ldots, H$ **do**
19: Given state s_h^k, take action a_h^k, and observe s_{h+1}^k.
20: **end for**
21: **end for**

Next, we present a modified version of LSVI-UCB in Algorithm 3. The major difference between our modified version of LSVI-UCB and Jin et al. (2020) lies in in Line 11 of Algorithm 3. Here we choose $Q_h(s_{h+1}^\tau, a) \leftarrow \min\{\|\phi(s_{h+1}^\tau, a)\|_{2\beta^2\Lambda_h^{-1}+2w_h^k(w_h^k)^\top}, H\}$, which is the upper bound of $\min\{w_h^\top \phi(s_{h+1}^\tau, a) + \beta \cdot \|\phi(s_{h+1}^\tau, a)\|_{\Lambda_h^{-1}}, H\}$ according to Lemma D.1.

Based on Algorithm 3, we propose our Sublinear LSVI-UCB in Algorithm 4, which reduce the value iteration complexity to sublinear in actions. Note that to let ρ strict less than 1, we set $c^2 \in [0.5, 1)$ and $\tau^2 \in [0.5, 0.8]$) following Lemma B.9.

D.2 Notations for Proof of Convergence

Next, we start the regret analysis of our Sublinear LSVI-UCB. We first define a series of notations. At episode k, we first estimate the weight w_h^k and matrix Λ_h^k. Next, we use them to estimate Q function Q_h^k. Then, using our LSH data structures, we obtain the value function $V_h^k(s)$ following line 25 of Algorithm 4. We also obtain the corresponding action associated with the value function and form the polity π_k following Line 30 of Algorithm 4. We also simplify $\phi(s_h^k, a_h^k)$ as ϕ_h^k.

Algorithm 4 Sublinear LSVI-UCB

data structure MATRIXLSH ▷ Theorem B.6
  INIT($S \subset \mathbb{R}^d$, $n \in \mathbb{N}$, $d \in \mathbb{N}$, $c \in (0,1)$, $\tau \in (0,1)$)
    ▷ $|S| = n$, c, τ is the approximate Max-MatNorm parameter and d is the dimension of data
  QUERY($x \in \mathbb{R}^d$)
end data structure

procedure SUBLINEARLSVI-UCB($\mathcal{S}$, $\mathcal{A}$, $N \in \mathbb{N}$, $H \in \mathbb{N}$, $c_{\mathsf{MatLSH}} \in (0,1)$, $\tau_{\mathsf{MatLSH}} \in (0,1)$)
  /*Preprocess $\phi(s,a)$ and build a LSH data structure*/ ▷ This step takes $O(S \cdot (A^{1+\rho} + d^2 A))$
  for $s \in \mathcal{S}$ **do**
    $\Phi_s \leftarrow \{\phi(s,a) | \forall a \in \mathcal{A}\}$
    static MATRIXLSH MATLSH$_s$
    MATLSH$_s$.INIT($\Phi_s, A, d, c_{\mathsf{MatLSH}}, \tau_{\mathsf{MatLSH}}$)
  end for

  for $k = 1, \ldots, K$ **do**
    Initialize state to s_1^k.
    for $h = H, \ldots, 1$ **do**
      /*Compute Λ_h^{-1}*/ ▷ This step takes $O(Kd^2 + d^\omega)$
      $\Lambda_h \leftarrow \sum_{\tau=1}^{k-1} \phi(s_h^\tau, a_h^\tau)\phi(s_h^\tau, a_h^\tau)^\top + \lambda \cdot \mathbf{I}_d$.
      Compute Λ_h^{-1}
      /* Value Iteration*/ ▷ This takes $O(Kd^2 A^\rho)$
      $w_h^k \leftarrow \Lambda_h^{-1} \sum_{\tau=1}^{k-1} \phi(s_h^\tau, a_h^\tau) \cdot (r_h(s_h^\tau, a_h^\tau) + \widehat{V}_{h+1}(s_{h+1}^\tau))$
      for $\tau = 1, \cdots, k-1$ **do**
        $a_h^\tau \leftarrow$ MATLSH$_s$.QUERY($2\beta^2 \Lambda_h^{-1} + 2 w_h^k w_h^{k\top}$)
        $\widehat{V}_h(s_h^\tau) \leftarrow \min\{\|\phi(s_{h+1}^\tau, a_h^\tau)\|_{2\beta^2 \Lambda_h^{-1} + 2 w_h^k w_h^{k\top}}, H\}$
      end for
    end for
    /* Construct Policy*/
    for step $h = 1, \ldots, H$ **do**
      Take action a_h^k at s_h^k, and observe s_{h+1}^k.
    end for
  end for
end procedure

D.3 Upper Bound on Weights in Sublinear LSVI-UCB

In this section, we show how to bound the weights w_h^k in Algorithm 4 using Lemma D.2. The weight we would like to bound is different from Jin et al. (2020). But the bound inequalities is very standard and similar to the proof in Jin et al. (2020).

Lemma D.2. *The weight w_h^k in Algorithm 4 at episode $k \in [K]$ and step $h \in [H]$ satisfies:*

$$\|w_h^k\|_2 \leq 2H\sqrt{dk/\lambda}.$$

Proof. If we perform $v^\top w_h^k$ where $v \in \mathbb{R}^d$ could be any vector in $\mathbb{R}^d$, we could bound $|v^\top w_h^k|$ as

$$\begin{aligned}
|v^\top w_h^k| &= \Big| v^\top (\Lambda_h^k)^{-1} \sum_{\tau=1}^{k-1} \phi_h^\tau \Big(r(s_h^\tau, a_h^\tau) + \widehat{V}_{h+1}(s_{h+1}^\tau) \Big) \Big| \\
&\leq \Big| v^\top (\Lambda_h^k)^{-1} \sum_{\tau=1}^{k-1} \phi_h^\tau \Big(r(s_h^\tau, a_h^\tau) + \max_{a \in \mathcal{A}} Q_{h+1}(s_{h+1}^\tau, a) \Big) \Big| \\
&\leq 2H \cdot \Big| v^\top (\Lambda_h^k)^{-1} \sum_{\tau=1}^{k-1} \phi_h^\tau \Big|
\end{aligned}$$

$$
\begin{aligned}
&= 2H \cdot \sum_{\tau=1}^{k-1} \left| v^\top (\Lambda_h^k)^{-1} \phi_h^\tau \right| \\
&\leq 2H \cdot \Big((\sum_{\tau=1}^{k-1} v^\top (\Lambda_h^k)^{-1} v) \cdot (\sum_{\tau=1}^{k-1} (\phi_h^\tau)^\top (\Lambda_h^k)^{-1} \phi_h^\tau) \Big)^{1/2} \\
&\leq 2H \|v\|_2 \sqrt{dk/\lambda},
\end{aligned}
$$

where the first step follows from the definition of w_h^k in Algorithm 4, the second step follows from the definition of $\widehat{V}_{h+1}$ in Algorithm 4, the third step follows from Definition A.2 that $r(s,a) + \widehat{V}_{h+1}(s) \leq 2H$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$, the forth step is a reorganization, the fifth step follows Cauchy–Schwarz inequality, the last step follows from Lemma A.20.

Next, we rewrite $\|w_h^k\|_2 = \max_{v:\|v\|_2=1} |v^\top w_h^k|$, in this way,

$$
\|w_h^k\|_2 = \max_{v:\|v\|_2=1} |v^\top w_h^k| \leq 2H\sqrt{dk/\lambda},
$$

where the last step follows from $|v^\top w_h^k| \leq 2H\sqrt{dk/\lambda}$.

□

D.4 Our Net Argument

We present our net argument to support the proof in the this section. We start with defining the covering number of euclidean ball.

Lemma D.3. *Let $\mathcal{B}$ denote a Euclidean ball in $\mathbb{R}^d$. $\mathcal{B}$ has radius greater than 0. For any $\epsilon > 0$, we upper bound the ϵ-covering number of $\mathcal{B}$ by $(1 + 2R/\epsilon)^d$.*

This is a standard statement. We reder readers to Vershynin (2010) for more details.

Next, we upper bound the covering number of a function $\mathcal{V}(s) = \min\Big\{ \|\phi(s,a)\|_{\beta^2\Lambda^{-1} + ww^\top}, H \Big\}$. The $\mathcal{V}$ we would like to bound is different from Jin et al. (2020). But the net argument is very standard and similar to proof in Jin et al. (2020).

Lemma D.4 (Our Net Argument)**.** *Let $\Lambda \in \mathbb{R}^{d\times d}$ denote a invertible matrix whose minimum eigenvalue is greater than a constant λ. Let w denote a vector such that $\|w\|_2 \leq L$. Let $\beta \in [0, B]$. Let $\max_{(s,a)\in\mathcal{S}\times\mathcal{A}} \|\phi(s,a)\|_2 \leq 1$. Let $\mathcal{V}$ denote a famility of functions such that $V : \mathcal{S} \to \mathbb{R}$ for any $V \in \mathcal{V}$ Let $\mathcal{N}_\epsilon$ denote the ϵ-covering number of $\mathcal{V}$. The ϵ-covering number is defined on distance $\mathrm{dist}(V, V') = \max_{s\in\mathcal{S}} |V(s) - V'(s)|$. If for any $V \in \mathcal{V}$, we have the form*

$$
V(s) = \min\Big\{ \|\phi(s,a)\|_{\beta^2\Lambda^{-1} + ww^\top}, H \Big\}. \tag{27}
$$

Then we have

$$
\log \mathcal{N}_\epsilon \leq d \log(1 + 4L/\epsilon) + d^2 \log\Big(1 + 8d^{1/2}B^2/(\lambda\epsilon^2)\Big).
$$

Proof. For given two arbitrary functions $V_1, V_2 \in \mathcal{V}$, we have

$$
\begin{aligned}
\mathrm{dist}(V_1, V_2) &\leq \sup_{s,a} \Big(\|\phi(s,a)\|_{\beta_1^2\Lambda_1^{-1} w_1 w_1^\top} - \|\phi(s,a)\|_{\beta_2^2\Lambda_2^{-1} + w_2 w_2^\top} \Big) \\
&\leq \sup_{\phi:\|\phi\|_2\leq 1} \Big(\|\phi\|_{\beta_1^2\Lambda_1^{-1} + w_1 w_1^\top} - \|\phi\|_{\beta_2^2\Lambda_2^{-1} + w_2 w_2^\top} \Big) \\
&\leq \sup_{\phi:\|\phi\|_2\leq 1} \sqrt{|\phi^\top(\beta_1^2\Lambda_1^{-1} + w_1 w_1^\top - \beta_2^2\Lambda_2^{-1} - w_2 w_2)\phi|} \\
&\leq \sup_{\phi:\|\phi\|_2\leq 1} \Big((w_1 - w_2)^\top \phi \Big) + \sup_{\phi:\|\phi\|_2\leq 1} \sqrt{|\phi^\top(\beta_1^2\Lambda_1^{-1} - \beta_2^2\Lambda_2^{-1})\phi|} \\
&= \|w_1 - w_2\| + \sqrt{\|\beta_1^2\Lambda_1^{-1} - \beta_2^2\Lambda_2^{-1}\|_2}
\end{aligned}
$$

$$\leq \|w_1 - w_2\| + \sqrt{\|\beta_1^2 \Lambda_1^{-1} - \beta_2^2 \Lambda_2^{-1}\|_F}, \tag{28}$$

where the first step follows the definition of V_1 and V_2 in Eq. (27), the second step follows from the fact that $\|\phi(s,a)\|_2 \leq 1$ in Definition A.2, the third step step follows from the fact that for any $x, y \geq 0$, we have $|\sqrt{x} - \sqrt{y}| \leq \sqrt{|x-y|}$, the forth step follows from $\sqrt{x+y} \leq \sqrt{x} + \sqrt{y}$ for any $x, y \geq 0$, the fifth step follows from the fact that the Frobenius norm of matrix is greater than the ℓ_2 norm.

Next, we denote $\mathcal{C}_w$ as the $(\epsilon/2)$-cover of a ball $\{w \in \mathbb{R}^d \mid \|w\|_2 \leq L\}$. Using Lemma D.3, we show that it can be upper bound as: $|\mathcal{C}_w| \leq (1 + 4L/\epsilon)^d$.

Similarly, we denote $\mathcal{C}_\Lambda$ as the $(\epsilon^2/4)$-cover of a ball $\{\beta^2 \Lambda^{-1} \in \mathbb{R}^{d \times d} \mid \|\beta^2 \Lambda^{-1}\|_F \leq d^{1/2} B^2 \lambda^{-1}\}$. Here we define the ball in $\|\cdot\|_F$. Using Lemma D.3, we show that it can be upper bound as: $|\mathcal{C}_\Lambda| \leq (1 + 8d^{1/2}B^2/(\lambda\epsilon^2))^{d^2}$.

Using, Eq. (28), we know that given any $V_1 \in \mathcal{V}$, we could find a $V_2 \in \mathcal{V}$ with form $V_2(s) = \min\left\{\|\phi(s,a)\|_{\beta_2^2 \Lambda_2^{-1} + w_2 w_2^\top}, H\right\}$ where $w_2 \in \mathcal{C}_w$ and $\beta_2^2 \Lambda_2^{-1} \in \mathcal{C}_\Lambda$, such that $\mathrm{dist}(V_1, V_2) \leq \epsilon$. Therefore, $\mathcal{N}_\epsilon \leq |\mathcal{C}_w| \cdot |\mathcal{C}_\mathcal{A}|$. Using this inequality, we have

$$\begin{aligned}
\log \mathcal{N}_\epsilon \leq & \log |\mathcal{C}_\mathcal{A}| + \log |\mathcal{C}_w| \\
\leq & d \log(1 + 4L/\epsilon) + d^2 \log(1 + 8d^{1/2}B^2/(\lambda\epsilon^2)).
\end{aligned}$$

Thus, we conclude the proof. □

D.5 Upper Bound on Fluctuations

We present a concentration lemma so that the fluctuations in LSVI-UCB is upper bounded in this section. The analysis is very standard and similar to proof in Jin et al. (2020). However, we improve the proof of Jin et al. (2020) with more detailed constant dependence.

Lemma D.5. *Let $C_\beta > 1$ denote a fixed constant. Let $\beta = C_\beta \cdot dH\sqrt{\iota}$. Let $\iota = \log(2dT/p)$. We show that for any probability $p \in [0,1]$ that is fixed, if we have an ξ event satisfying that for all $k \in [K]$ and $h \in [H]$:*

$$\Big\| \sum_{\tau=1}^{k-1} \phi_h^\tau \big(V_{h+1}^k(s_{h+1}^\tau) - [\mathbb{P}_h V_{h+1}^k](s_h^\tau, a_h^\tau)\big) \Big\|_{(\Lambda_h^k)^{-1}} \leq 30 \cdot dH\sqrt{\iota + \log(5C_\beta)},$$

Then, we have

$$\Pr[\xi] \geq 1 - p/2.$$

Proof. We show that any fixed $\epsilon > 0$, we have

$$\begin{aligned}
& \Big\| \sum_{\tau=1}^{k-1} \phi_h^\tau \big(V_{h+1}^k(s_{h+1}^\tau) - [\mathbb{P}_h V_{h+1}^k](s_h^\tau, a_h^\tau)\big) \Big\|_{(\Lambda_h^k)^{-1}}^2 \\
\leq & 4H^2 \left(d \log(1 + k/\lambda) + d \log\left(1 + \frac{8H\sqrt{dk}}{\epsilon\sqrt{\lambda}}\right) + d^2 \log\left(1 + \frac{8d^{1/2}\beta^2}{\epsilon^2 \lambda}\right) + \log(2/p) \right) + \frac{8k^2\epsilon^2}{\lambda} \\
\leq & 4H^2 (d \log(1+k) + d \log(1 + 8\sqrt{k^3/d}) + d^2 \log(1 + 8C_\beta^2 d^{0.5} K^2 \iota) + \log(2/p)) + 8d^2H^2 \\
\leq & 30 \cdot d^2 H^2 \log(10 C_\beta dT/p) \\
= & 30 \cdot d^2 H^2 (\iota + \log(5C_\beta)),
\end{aligned} \tag{29}$$

where the first step follows from combining Lemmas A.21 and D.4,the second step follows from $\lambda = 1$, $\epsilon = dH/K$, and $\beta = C_\beta \cdot dH\sqrt{\iota}$, the third step follows from $C_\beta \geq 1$ and $\iota = \log(2dT/p)$, the last step follows from $\iota = \log(2dT/p)$.

Thus, we complete the proof. □

D.6 Upper Bound of Difference of Q Function

In this section, we bound like to bound the difference between the Q function Q_h^k (see Section A.1) selected by Algorithm 4 and the value function Q_h^π (see Definition A.4) of any policy π. We bound the their difference by bounding $\langle \phi(s,a), w_h^k \rangle - Q_h^\pi(s,a)$. The analysis is very standard and similar to proof in Jin et al. (2020). However, we improve the proof of Jin et al. (2020) with more detailed constant dependence.

Lemma D.6. *Let $\lambda = 1$ in Algorithm 4. Let $\iota = \log(2dT/p)$. We show that for any policy π that is fixed, for all $s \in \mathcal{S}$, $a \in \mathcal{A}$, $h \in [H]$ and $k \in [K]$, on the event ξ defined in Lemma D.5, we show that exists an absolute constant $C_\beta \geq 100$ such that*

$$\langle \phi(s,a), w_h^k \rangle - Q_h^\pi(s,a) - [\mathbb{P}_h(V_{h+1}^k - V_{h+1}^\pi)](s,a) \leq C_\beta dH\sqrt{\iota} \cdot \|\phi(s,a)\|_{(\Lambda_h^k)^{-1}}$$

Proof. We start with rewriting $Q_h^\pi(s,a)$ as

$$Q_h^\pi(s,a) := \langle \phi(s,a), w_h^\pi \rangle = r_h(s,a) + [\mathbb{P}_h V_{h+1}^\pi](s,a).$$

where the first step follows from Proposition A.9, and the second step follows from Eq. (5).

Next, we show that

$$\begin{aligned} w_h^k - w_h^\pi &= (\Lambda_h^k)^{-1} \sum_{\tau=1}^{k-1} \phi_h^\tau (r_h^\tau + V_{h+1}^k(s_{h+1}^\tau)) - w_h^\pi \\ &= (\Lambda_h^k)^{-1} \Big(- \lambda w_h^\pi + \sum_{\tau=1}^{k-1} \phi_h^\tau \big(V_{h+1}^k(s_{h+1}^\tau) - [\mathbb{P}_h V_\pi^{h+1}](s_h^\tau, a_h^\tau) \big) \Big) \\ &= p_1 + p_2 + p_3. \end{aligned}$$

where the first step follows from the definition of w_h^k, the second step follows from the definition of w_h^π. the last step follows from

$$\begin{aligned} p_1 &:= -\lambda (\Lambda_h^k)^{-1} w_h^\pi \\ p_2 &:= (\Lambda_h^k)^{-1} \sum_{\tau=1}^{k-1} \phi_h^\tau \big(V_{h+1}^k(s_{h+1}^\tau) - [\mathbb{P}_h V_{h+1}^k](s_h^\tau, a_h^\tau) \big) \\ p_3 &:= (\Lambda_h^k)^{-1} \sum_{\tau=1}^{k-1} \phi_h^\tau [\mathbb{P}_h (V_{h+1}^k - V_{h+1}^\pi)](s_h^\tau, a_h^\tau) \end{aligned}$$

Next, we upper bound p_1, p_2 and p_3 separately.

We upper bound p_1 as,

$$\begin{aligned} |\langle \phi(s,a), q_1 \rangle| &= \lambda \cdot |\langle \phi(s,a), (\Lambda_h^k)^{-1} w_h^\pi \rangle| \\ &\leq \lambda \cdot \|w_h^\pi\|_2 \cdot \|\phi(s,a)\|_{(\Lambda_h^k)^{-1}} \\ &\leq 2H\sqrt{d\lambda} \cdot \|\phi(s,a)\|_{(\Lambda_h^k)^{-1}} \\ &\leq 2H\sqrt{d} \cdot \|\phi(s,a)\|_{(\Lambda_h^k)^{-1}}, \end{aligned} \tag{30}$$

where the second step follows from $|\langle a, b \rangle| \leq \|a\|_2 \cdot \|b\|_2$, and the third step follows from $\|w_h^\pi\|_2 \leq 2H\sqrt{d/\lambda}$ (see Lemma D.2), and the last step follows from $\lambda = 1$.

We upper bound p_2 as,

$$|\langle \phi(s,a), q_2 \rangle| \leq 30 \cdot dH\sqrt{\iota + \log(5C_\beta)} \|\phi(s,a)\|_{(\Lambda_h^k)^{-1}}, \tag{31}$$

where the first step follows from Lemma D.5 on the event ξ.

We upper bound q_3 as,

$$\begin{aligned}\langle \phi(s,a), p_3 \rangle &= \left\langle \phi(s,a), (\Lambda_h^k)^{-1} \sum_{\tau=1}^{k-1} \phi_h^\tau [\mathbb{P}_h(V_{h+1}^k - V_{h+1}^\pi)](s_h^\tau, a_h^\tau) \right\rangle \\ &= \left\langle \phi(s,a), (\Lambda_h^k)^{-1} \sum_{\tau=1}^{k-1} \phi_h^\tau (\phi_h^\tau)^\top \int (V_{h+1}^k - V_{h+1}^\pi)(s') \mathrm{d}\mu_h(s') \right\rangle \\ &= q_1 + q_2,\end{aligned}$$

where the first step is a reorganization, the second step decomposes the right hand side as:

$$\begin{aligned}q_1 &:= \left\langle \phi(s,a), \int (V_{h+1}^k - V_{h+1}^\pi)(s') \mathrm{d}\mu_h(s') \right\rangle, \\ q_2 &:= -\lambda \left\langle \phi(s,a), (\Lambda_h^k)^{-1} \int (V_{h+1}^k - V_{h+1}^\pi)(s') \mathrm{d}\mu_h(s') \right\rangle.\end{aligned}$$

Then, we rewrite $q_1 = \mathbb{P}_h(V_{h+1}^k - V_{h+1}^\pi)(s,a)$ following Definition A.2.

Next, we upper bound q_2 as

$$|q_2| \leq 2H\sqrt{d} \|\phi(s,a)\|_{(\Lambda_h^k)^{-1}}, \tag{32}$$

where the first step follows from Lemma D.2.

Finally, because $\langle \phi(s,a), w_h^k \rangle - Q_h^\pi(s,a) = \langle \phi(s,a), p_1 + p_2 + p_3 \rangle$, we have

$$\begin{aligned}&|\langle \phi(s,a), w_h^k \rangle - Q_h^\pi(s,a) - \mathbb{P}_h(V_{h+1}^k - V_{h+1}^\pi)(s,a)| \\ &= \langle \phi(s,a), p_1 + p_2 + q_2 \rangle \\ &\leq (2H\sqrt{d} + 30 \cdot dH\sqrt{\iota + \log(5C_\beta)} + 2H\sqrt{d}) \cdot \|\phi(s,a)\|_{(\Lambda_h^k)^{-1}} \\ &\leq dH(30\sqrt{\iota + \log(5C_\beta)} + 4) \cdot \|\phi(s,a)\|_{(\Lambda_h^k)^{-1}},\end{aligned}$$

where the second step follows from combining Eq. (30), Eq. (31) and Eq. (32), the third step follows from $d \geq 1, H \geq 1$.

Finally, we choose an absolute constant C_β that satisfies:

$$30(\sqrt{\iota + \log(5C_\beta)} + 4) \leq C_\beta \sqrt{\iota}. \tag{33}$$

Note that $\iota = \log(2dT/p) \geq 4$, as long as $C_\beta \geq 100$ the above inequality holds

Finally, with this choice of C_β, we finish the proof. □

D.7 Q Function Difference by Induction

In this section, we build a connection between $Q_1^k(s,a)$ selected by Algorithm 4 and $Q_1^*(s,a)$. We show in Lemma D.7 that $Q_1^*(s,a)$ is upper bounded by $Q_1^k(s,a)$ plus an error term related to the parameter c for approximate Max-MatNorm in Algorithm 4.

Lemma D.7. *Let $Q_1^k(s,a)$ denote the estimated Q function for state s when taking action a at the first step. Let $Q_1^*(s,a)$ denote the optimal Q function for state s when taking action a at the first step. Let H denote the total steps. Let c is the parameter for approximate* Max-MatNorm. *We show that using Sublinear LSVI-UCB (see Algorithm 4), we have*

$$Q_1^*(s,a) - Q_1^k(s,a) \leq H - c\frac{1-c^H}{1-c}.$$

Proof. We start with bounding on the relationship between $Q_h^k(s,a)$ and $Q_h^*(s,a)$.

$$\langle \phi(s,a), w_h^k \rangle + \beta \|\phi(s,a)\|_{(\Lambda_h^k)^{-1}} \geq Q_h^*(s,a) + [\mathbb{P}_h(V_{h+1}^k - V_{h+1}^*)](s,a), \tag{34}$$

where the first step follows Lemma D.5.

Next, when $h = H$, as the value functions are all zero in $H + 1$ step, we have

$$\begin{aligned} Q_H^k(s,a) &\geq \langle \phi(s,a), w_H^k \rangle + \beta \|\phi(s,a)\|_{(\Lambda_H^k)^{-1}} \\ &\geq Q_H^*(s,a), \end{aligned} \tag{35}$$

where the first step follows from Lemma D.1, the second step follows from Lemma D.6.

Next, we have

$$\begin{aligned} \max_{a \in \mathcal{A}} Q_H^k(s,a) &\geq \max_{a \in \mathcal{A}} Q_H^*(s,a) \\ &\geq V_H^*(s), \end{aligned} \tag{36}$$

where the first step follows from Eq. (35), the second step follows from the definition of $V_H^*(s)$ in Definition A.4.

Next, when $h = H - 1$, we bound $[\mathbb{P}_h(V_H^k - V_H^*)](s,a)$ as

$$\begin{aligned} [\mathbb{P}_h(V_H^k - V_H^*)](s,a) &\geq [\mathbb{P}_h(c \max_{a \in \mathcal{A}} Q_H^k(s,a) - V_H^*)](s,a) \\ &\geq c[\mathbb{P}_h(\max_{a \in \mathcal{A}} Q_H^k(s,a) - V_H^*)](s,a) - (1-c)[\mathbb{P}_h V_H^*](s,a) \\ &\geq c[\mathbb{P}_h(\max_{a \in \mathcal{A}} Q_H^k(s,a) - V_H^*)](s,a) - (1-c) \cdot 1 \\ &\geq -(1-c) \cdot 1, \end{aligned} \tag{37}$$

where the first step comes from the property of data structure MATRIXLSH in Algorithm 4, the second step is an reorganization, the third step follows the definition of $V_H^*(s)$ in Definition A.4, the last step follows the Eq. (36).

Next, we have

$$\begin{aligned} Q_{H-1}^k(s,a) &\geq \langle \phi(s,a), w_{H-1}^k \rangle + \beta \|\phi(s,a)\|_{(\Lambda_{H-1}^k)^{-1}} \\ &\geq Q_{H-1}^*(s,a) + [\mathbb{P}_h(V_H^k - V_H^*)](s,a) \\ &\geq Q_{H-1}^*(s,a) - (1-c) \cdot 1, \end{aligned} \tag{38}$$

where the first step follows from the Lemma D.1, the second step follows from Eq. (34) ,and the third step follows Eq. (37).

Next, we have

$$\begin{aligned} \max_{a \in \mathcal{A}} Q_{H-1}^k(s,a) &\geq \max_{a \in \mathcal{A}} Q_{H-1}^*(s,a) - (1-c) \cdot 1 \\ &\geq V_{H-1}^*(s) - (1-c) \cdot 1, \end{aligned} \tag{39}$$

where the first step follows from Eq. (38), and the second step follows the definition of $Q_{H-1}^*(s,a)$ in section A.1.

Next, when $h = H - 2$, we lower bound $[\mathbb{P}_h(\widehat{V_{H-1}^k} - V_{H-1}^*)](s,a)$ as

$$\begin{aligned} [\mathbb{P}_h(V_{H-1}^k - V_{H-1}^*)](s,a) &\geq [\mathbb{P}_h(c \max_{a \in \mathcal{A}} Q_{H-1}^k(s,a) - V_{H-1}^*)](s,a) \\ &\geq c[\mathbb{P}_h(\max_{a \in \mathcal{A}} Q_{H-1}^k(s,a) - V_{H-1}^*)](s,a) - (1-c) \cdot [\mathbb{P}_h V_{H-1}^*)](s,a) \\ &\geq c[\mathbb{P}_h(\max_{a \in \mathcal{A}} Q_{H-1}^k(s,a) - V_{H-1}^*)](s,a) - (1-c) \cdot 2 \\ &\geq -c(1-c) \cdot 1 - (1-c) \cdot 2, \end{aligned} \tag{40}$$

where the first step comes from the MATRIXLSH in Algoritm 4, the second step is an reorganization, the third step follows the definition of $V_H^*(s)$ in section A.1, the last step follows the Eq. (39).

Next, we have

$$\begin{aligned} Q_{H-2}^k(s,a) &\geq \langle \phi(s,a), w_{H-2}^k \rangle + \beta \|\phi(s,a)\|_{(\Lambda_{H-2}^k)^{-1}} \\ &\geq Q_{H-2}^*(s,a) + [\mathbb{P}_h(V_{H-1}^k - V_{H-1}^*)](s,a) \\ &\geq Q_{H-2}^*(s,a) - c(1-c)\cdot 1 - (1-c)\cdot 2, \end{aligned} \tag{41}$$

where the first step follows from the Lemma D.1, the second step follows from Eq. (34) ,and the third step follows Eq. (40).

using induction from H to 1, we have

$$\begin{aligned} Q_1^k(s,a) &\geq Q_1^*(s,a) - (1-c)\sum_{h=1}^{H} c^{h-1}(H+1-h) \\ &= Q_1^*(s,a) - (1-c)\frac{H - cH - c + c^{H+1}}{(1-c)^2} \\ &= Q_1^*(s,a) - \frac{H - cH - c + c^{H+1}}{1-c} \\ &= Q_1^*(s,a) - \frac{H - cH}{1-c} + \frac{-c + c^{H+1}}{1-c} \\ &= Q_1^*(s,a) - (H - \frac{c - c^{H+1}}{1-c}) \\ &= Q_1^*(s,a) - (H - c\frac{1-c^H}{1-c}), \end{aligned} \tag{42}$$

where the first step follows the induction rule, the remain steps are reorganizations.

□

We notice from Lemma D.7 that there exists a term $H - c\frac{1-c^H}{1-c}$. Here we use Fact D.8 to bound this term.

Fact D.8. *Let $H \in \mathbb{N}$. Let $c = 1-\gamma$, for any $\gamma \in (0, 1/(10H))$, then we have*

$$H - c\frac{1-c^H}{1-c} \leq 2\gamma H^2.$$

Proof. First, by definition $\gamma = 1 - c \in (0,1)$, then we can rewrite LHS as

$$\begin{aligned} H - c\frac{1-c^H}{1-c} &= H - (1-\gamma)(1-(1-\gamma)^H)/\gamma \\ &\leq H - (1-\gamma)(1-e^{-H\gamma})/\gamma \\ &\leq H - (1-\gamma)(H - 0.5(H^2\gamma)) \\ &= H(1-(1-\gamma)(1-0.5(H\gamma))) \\ &\leq H\cdot(2H\gamma) = 2\gamma H^2, \end{aligned}$$

where the second step follows from $(1-\gamma)^{1/\gamma} \leq e^{-1}$, the third step follows from $1 - e^{-x} \geq x - 0.5x^2, \forall x \in [0, 1/10]$.

□

D.8 Recursive Formula

In this section, we bound the difference between $Q_h^k(s_h^k, a)$ and $Q_h^{\pi_k}(s_h^k, a)$ in a recursive formula.

Lemma D.9 (Recursion)**.** *Let δ_h^k denote the difference $Q_h^k(s_h^k,a) - Q_h^{\pi_k}(s_h^k,a)$. Let $\zeta_{h+1}^k = \mathbb{E}[\delta_{h+1}^k | s_h^k, a] - \delta_{h+1}^k$ denote the error between expectation and observed difference. Let $\beta = C_\beta dH\sqrt{\iota}$. Given the event ξ defined in Lemma D.5, we bound $\delta_h^k - \delta_{h+1}^k$ for any $k \in [K]$ and $h \in [H]$ as*

$$\delta_h^k - \delta_{h+1}^k \leq \zeta_{h+1}^k + 2\beta\|\phi(s_h^k, a)\|_{(\Lambda_h^k)^{-1}}.$$

Proof. We bound the δ_h^k as

$$\begin{aligned}\delta_h^k &= Q_h^k(s,a) - Q_h^{\pi_k}(s,a) \\ &\leq [\mathbb{P}_h(V_{h+1}^k - V_{h+1}^{\pi_k})](s,a) + 2C_\beta dH\sqrt{\iota}\|\phi(s,a)\|_{(\Lambda_h^k)^{-1}}. \\ &= \zeta_{h+1}^k + \delta_{h+1}^k + 2\beta\|\phi(s,a)\|_{(\Lambda_h^k)^{-1}},\end{aligned}$$

where the second step follows from Lemma D.6, the third step follows from $[\mathbb{P}_h(V_{h+1}^k - V_{h+1}^{\pi_k})](s,a) = \zeta_{h+1}^k + \delta_{h+1}^k$.

Thus, we finish the proof.

□

As our algorithm have the same upper bound on recursion with Jin et al. (2020), the upper bound on ζ_{h+1}^k in Jin et al. (2020) could also be used in our analysis. We state the bound as

Lemma D.10 (Jin et al. (2020))**.** *Let $\zeta_{h+1}^k = \mathbb{E}[\delta_{h+1}^k | s_h^k, a] - \delta_{h+1}^k$. With probability at least $1 - p/2$, we show that*

$$\sum_{k=1}^K \sum_{h=1}^H \zeta_h^k \leq 2H\sqrt{T\iota}.$$

We could also upper bound $\sum_{k=1}^K \sum_{h=1}^H \|\phi(s_1^k, a_1^{k*})\|_{(\Lambda_h^k)^{-1}}$ following Jin et al. (2020).

Lemma D.11 (Jin et al. (2020))**.** *Let $a_1^{k*} \in \mathcal{A}$ denote the optimal action at state $s_1^k \in \mathcal{S}$. Given, Λ_h^k estimated in each step, we have*

$$\sum_{k=1}^K \sum_{h=1}^H \|\phi(s_1^k, a_1^{k*})\|_{(\Lambda_h^k)^{-1}} \leq H \cdot \sqrt{2dK\iota}.$$

D.9 Regret Analysis

In this section, we prove main theorem in Theorem D.12.

Theorem D.12 (Convergence Result of Sublinear Least-Squares Value Iteration with UCB (Sublinear LSVI-UCB), a formal version of Theorem 4.3)**.** *In a linear MDP in Definition A.2, we set $\lambda = 1$. Let $C_\beta \geq 100$ denote a fixed constant and $\iota = \log(2dT/p)$. If we set approximate* Max-MatNorm *parameter $c = 1 - \frac{\iota}{\sqrt{K}}$, then for any $p \in (0,1)$ that is fixed, with probability $1 - p$, Sublinear LSVI-UCB (Algorithm 4) has the cumulative regret at most $O(C_\beta \cdot \sqrt{d^3H^3T\iota^2})$.*

Proof. We start with upper bounding the regret as:

$$\begin{aligned}\text{Regret}(K) &= \sum_{k=1}^K \big(V_1^*(s_1^k) - V_1^{\pi_k}(s_1^k)\big) \\ &= \sum_{k=1}^K \left(\max_{a\in\mathcal{A}} Q_1^*(s_1^k, a) - \max_{a\in\mathcal{A}} Q_1^{\pi_k}(s_1^k, a)\right) \\ &\leq \sum_{k=1}^K \left(\max_{a\in\mathcal{A}} Q_1^*(s_1^k, a_1^{k*}) - Q_1^{\pi_k}(s_1^k, a_1^{k*})\right) \\ &\leq \sum_{k=1}^K (Q_1^k(s_1^k, a_1^{k*}) - Q_1^{\pi_k}(s_1^k, a_1^{k*}) + 2\gamma H^2) \\ &= 2\gamma KH^2 + \sum_{k=1}^K \delta_1^k \\ &\leq 2\gamma KH^2 + \sum_{k=1}^K \sum_{h=1}^H \zeta_h^k + 2\beta \sum_{k=1}^K \sum_{h=1}^H \|\phi(s_1^k, a_1^{k*})\|_{(\Lambda_h^k)^{-1}}, \end{aligned} \tag{43}$$

where the first step follows the definition of regret, the second steps follows from the definition of value function in Definition A.4, the third step follows from that $\max_{a\in\mathcal{A}} Q_1^{\pi_k}(s_1^k, a) \geq Q_1^{\pi_k}(s_1^k, a_1^{k*})$, where a_1^{k*} is the optimal action chosen at state s_1^k, the forth step follows from Lemma D.7, the fifth step is a follows the definition of δ_h^k and ζ_h^k as in Lemma D.9, the sixth step follows from Lemma D.9.

Next, with probability $1-p$, we show that

$$\begin{aligned}
\text{Regret}(K) &\leq 2\gamma KH^2 + \sum_{k=1}^{K}\sum_{h=1}^{H} \zeta_h^k + 2\beta \sum_{k=1}^{K}\sum_{h=1}^{H} \|\phi(s_1^k, a_1^{k*})\|_{(\Lambda_h^k)^{-1}} \\
&\leq 2\gamma KH^2 + 2H\sqrt{T\iota} + 2\beta \sum_{k=1}^{K}\sum_{h=1}^{H} \|\phi(s_1^k, a_1^{k*})\|_{(\Lambda_h^k)^{-1}} \\
&\leq 2K\gamma H^2 + 2H\sqrt{T\iota} + \beta H\sqrt{2dK\iota} \\
&= 2K\gamma H^2 + 2H\sqrt{T\iota} + C_\beta \cdot \sqrt{2d^3H^4K\iota^2} \\
&\leq 2\sqrt{H^4K\iota^2} + 2\sqrt{H^3K\iota} + C_\beta \cdot \sqrt{2d^3H^4K\iota^2} \\
&\leq 2C_\beta\sqrt{d^3H^4K\iota^2}, \qquad (44)
\end{aligned}$$

where the second step follows from Lemma D.10, the third step follows from Lemma D.11, the forth step from $\beta = C_\beta \cdot dH\sqrt{\iota}$, the fifth step follows from $\gamma = \frac{1}{\sqrt{K}}$, the sixth step is a reorganizationm the seventh step follows from $C_\beta \geq 100$.

Thus, we finish our proof.

□

D.10 Running Time Analysis

We present the running time analysis of our Sublinear LSVI-UCB. We first introduce the running time of each procedure of LSVI-UCB in Section D.10.1. Next, we introduce the running time of Sublinear LSVI-UCB in Section D.10.2. Therefore, we could compare their efficiency in the next section.

D.10.1 LSVI-UCB

First, we show the LSVI-UCB algorithm in Algorithm 5

Lemma D.13. *The running time of pre-computing Λ^{-1} in Algorithm 5 takes time*

$$O(Kd^2 + d^\omega),$$

where $\omega \approx 2.373$ is the exponent of matrix multiplication Williams (2012); Le Gall (2014).

Proof. It takes $O(Kd^2)$ to compute and sum up every $\phi(s_h^\tau, a_h^\tau)\phi(s_h^\tau, a_h^\tau)^\top$. Computing the inverse matrix of Λ takes $O(d^\omega)$. All other operations take $O(d)$. Combining the complexity together, we obtain the pre-computing complexity $O(Kd^2 + d^\omega)$. □

Lemma D.14. *The running time of value iteration in Algorithm 5 takes*

$$O(HKd^2A).$$

Proof. For each of the H step,

- It takes $O(Kd^2A)$ to compute $\widehat{V}_{h+1}(s_{h+1}^\tau)$ for each state s_{h+1}^τ.
- It takes $O(Kd)$ to sum up $\phi(s_h^\tau, a_h^\tau) \cdot (r_h(s_h^\tau, a_h^\tau) + \widehat{V}_{h+1}(s_{h+1}^\tau))$.
- It takes $O(d^2)$ to multiply Λ with the sum of vectors.
- All other operations take $O(d)$.

Combining them together, we have $O(HKd^2A)$. □

Algorithm 5 LSVI-UCB (Jin et al., 2020)

for $k = 1, \ldots, K$ **do**
  Initialize the state s_1^k.
  for $h = H, \ldots, 1$ **do**
    /*Compute Λ_h^{-1}*/ ▷ This step takes $O(Kd^2 + d^\omega)$
    $\Lambda_h \leftarrow \sum_{\tau=1}^{k-1} \phi(s_h^\tau, a_h^\tau)\phi(s_h^\tau, a_h^\tau)^\top + \lambda \cdot \mathbf{I}_d$.
    Compute Λ_h^{-1}
    /* Value Iteration*/ ▷ This step takes $O(Kd^2A)$
    $w_h^k \leftarrow \Lambda_h^{-1} \sum_{\tau=1}^{k-1} \phi(s_h^\tau, a_h^\tau) \cdot (r_h(s_h^\tau, a_h^\tau) + \widehat{V}_{h+1}(s_{h+1}^\tau))$
    for $\tau = 1, \cdots, k-1$ **do**
      for $a \in \mathcal{A}$ **do**
        $Q_h(s_{h+1}^\tau, a) \leftarrow \min\{\langle w_h^k, \phi(s_{h+1}^\tau, a)\rangle + \beta \cdot \|\phi(s_{h+1}^\tau, a)\|_{\Lambda_h^{k-1}}, H\}$.
      end for
      $\widehat{V}_h(s_h^\tau) \leftarrow \max_a Q_h(s_h^\tau, a)$
      $a_h^\tau \leftarrow \arg\max_a Q_h(s, a)$ ▷ a_h^τ is the maximum value action taken at state s_h^τ.
    end for
  end for
  /* Construct Policy*/
  for $h = 1, \ldots, H$ **do**
    Take action a_h^k, and observe s_{h+1}^k.
  end for
end for

D.10.2 Sublinear LSVI-UCB

In this section, we show the runtime analysis of our Sublinear LSVI-UCB in Algorithm 4.

Lemma D.15. *The running time of pre-computing Λ^{-1} in Algorithm 4 takes*

$$O(Kd^2 + d^\omega),$$

where $\omega \approx 2.373$ is the exponent of matrix multiplication (Williams, 2012; Le Gall, 2014).

Proof. It takes $O(Kd^2)$ to compute and sum up every $\phi(s_h^\tau, a_h^\tau)\phi(s_h^\tau, a_h^\tau)^\top$. Computing the inverse matrix of Λ takes $O(d^\omega)$. All other operations take $O(d)$. Combining the complexity together, we obtain the pre-computing complexity $O(Kd^2 + d^\omega)$. □

Lemma D.16. *The running time of value iteration in Algorithm 4 takes*

$$O(HKd^2A^\rho),$$

Further more,

- *If initialize the* LSH *data-structure using Theorem A.14, $\rho = 1 - \frac{1}{4\sqrt{K}}$.*
- *If initialize the* LSH *data-structure using Theorem A.15, $\rho = 1 - \frac{1}{8K}$.*

Proof. For each of the H step,

- It takes $O(Kd^2A^\rho)$ to compute $\widehat{V}_{h+1}(s_{h+1}^\tau)$ for each state s_{h+1}^τ. If we initialize the LSH data-structure using Theorem A.14, we determine $\rho = 1 - \frac{1}{4\sqrt{K}}$ using Lemma B.9. If we initialize the LSH data-structure using Theorem A.15, we determine $\rho = 1 - \frac{1}{8K}$ using Lemma B.10.
- It takes $O(Kd)$ to sum up $\phi(s_h^\tau, a_h^\tau) \cdot (r_h(s_h^\tau, a_h^\tau) + \widehat{V}_{h+1}(s_{h+1}^\tau))$.
- It takes $O(d^2)$ to multiply Λ with the sum of vectors.

- All other operations take $O(d)$.

□

D.11 Comparison

In this section, we show the comparison between our Sublinear LSVI-UCB with LSVI-UCB (Jin et al., 2020). We show the comparison results in Table 4.

Table 4: Comparison between Our Sublinear LSVI-UCB with LSVI-UCB (Jin et al., 2020). Let S denote the quantity of available states and A denote the quantity of available actions. Let d denote the dimension of $\phi(s,a)$. Let H denote the number of steps per episode. Let K denote the total number of episodes. Let $\iota = \log(2Hd/p)$ and p is the failure probability. Let $\rho_1 = 1 - \frac{1}{4\sqrt{K}}$ be the parameter determined by data structure in Theorem A.14 and $\rho_2 = 1 - \frac{1}{8K}$ be the parameter determined by data structure Theorem A.15. Since $K > S$, we write the preprocessing time as $O(Kd^2A^{1+o(1)})$. This table is a detailed version of corresponding part of Table 1.

Algorithm	**Preprocess**	**#Value Iteration**	**Regret**
Ours	$O(Kd^2A^{1+\rho_1})$	$O(HKd^2A^{\rho_1})$	$O(C_\beta\sqrt{d^3H^4K\iota^2})$
Ours	$O(Kd^2A^{1+o(1)})$	$O(HKd^2A^{\rho_2})$	$O(C_\beta\sqrt{d^3H^4K\iota^2})$
LSVI	0	$O(HKd^2A)$	$O(C_\beta\sqrt{d^3H^4K\iota^2})$

E MORE DATA STRUCTURES: ADAPTIVE Max-IP QUERIES

In this section, we show how to tackle the adaptive Max-IP queries in RL. In both Sublinear LSVI and Sublinear LSVI-UCB, the queries for (c,τ)-Max-IP during the value iteration are adaptive but not arbitrary. Thus, we could not union bound the failure probability of LSH for (c,τ)-Max-IP. In this work, we present a quantization method to union bound the failure probability of adaptive Max-IP queries. This section is organized as:

- In Section E.1, we introduce the LSH data structure for adaptive Max-IP queries and theoretical guarantee of Sublinear LSVI with this data structure.
- In Section E.2, we present the LSH data structure for adaptive Max-MatNorm queries and theoretical guarantee of Sublinear LSVI-UCB with this data structure.

E.1 Sublinear LSVI with Adaptive Max-IP Queries

In this section, we show how to tackle adaptive Max-IP queries in Sublinear LSVI. We start with defining the quantized approximate Max-IP.

Definition E.1 (Quantized approximate Max-IP)**.** *Let $c \in (0,1)$ and $\tau \in (0,1)$. Let $\lambda \geq 0$. Given an n-point dataset $Y \subset \mathbb{S}^{d-1}$, the goal of the (c,τ,λ)-Max-IP is to build a data structure that, given a query $x \in \mathbb{S}^{d-1}$ with the promise that there exists a datapoint $y \in Y$ with $\langle x,y\rangle \geq \tau$, it reports a datapoint $z \in Y$ with similarity $\langle x,z\rangle \geq c \cdot \text{Max-IP}(x,Y) - \lambda$.*

Next, we show a standard way of performing approximate Max-IP via LSH. We denote Q as the convex hull of all queries for (c,τ)-Max-IP and denote its maximum diameter in ℓ_2 distance as D_X. Our quantization method[vii] contains two steps: (1) Preprocessing: we quantize Q to a lattice $\widehat{Q}$ with quantization error λ/d. In this way, each coordinate would be quantized into the multiples of λ/d. (2) Query: given a query $x \in Q$, we first quantize it to the nearest $\widehat{q} \in \widehat{Q}$ and perform (c,τ)-Max-IP. As each $\widehat{q} \in \widehat{Q}$ is independent, we could union bound the failure probability of adaptive queries. On the other hand, this would generate an λ additive error in the returned inner product.

Next, we show our theorem for (c,τ,λ)-Max-IP over adaptive queries in Theorem E.2.

Theorem E.2 (A modified version of Theorem B.2)**.** *Let $c \in (0,1)$, $\tau \in (0,1)$ and $\lambda \in (0,1)$. Given a set of n-points $Y \subset \mathbb{S}^{d-1}$ on the sphere, one can construct a data structure with preprocessing time $\mathcal{T}_{\text{init}} \cdot \kappa$ and space $\mathcal{S}_{\text{space}} \cdot \kappa$ so that for every $x \in \mathbb{S}^{d-1}$ in an adaptive query sequence $X = \{x_1, x_2, \cdots, x_T\}$, we take $O(dn^\rho \cdot \kappa)$ query time:*

[vii] This is a standard trick in the field of sketching and streaming (Nakos et al., 2019; Ben-Eliezer et al., 2020).

- *if* Max-IP$(x, Y) \geq \tau$*, then we output a vector in* Y *which is a* (c, τ, λ)-Max-IP *with respect to* (x, Y) *with probability at least* $1 - \delta$*, where* $\rho = f(c, \tau) + o(1)$.
- *otherwise, we output* fail.

where $\kappa := d \log(ndD_X/(\lambda\delta))$ *and* $\rho \in (0, 1)$*. We use* D_X *to represent maximum diameter in* ℓ_2 *distance of all queries in* X.

Further more,

- *If* $\mathcal{T}_{\mathsf{init}} = O(dn^{1+\rho})$ *and* $\mathcal{S}_{\mathsf{space}} = O(n^{1+\rho} + dn)$*, then* $f(c, \tau) = \frac{1-\tau}{1-2c\tau+\tau}$.
- *If* $\mathcal{T}_{\mathsf{init}} = O(dn^{1+o(1)})$ *and* $\mathcal{S}_{\mathsf{space}} = O(n^{1+o(1)} + dn)$*, then* $f(c, \tau) = \frac{2(1-\tau)^2}{(1-c\tau)^2} - \frac{(1-\tau)^4}{(1-c\tau)^4}$.

Proof. The failure probability for an adaptive sequence X is equivalent to the probability that at least one query $\widehat{q} \in \widehat{Q}$ fail in solving all κ number of (c, τ)-Max-IP. We bound this failure probability as

$$\Pr[\exists \widehat{q} \in \widehat{Q} \quad \text{s.t all } (c, \tau)\text{-Max-IP fail}] = n \cdot (\frac{dD_X}{\lambda})^d \cdot (\frac{1}{10})^\kappa \leq \delta,$$

where the last step follows from $\kappa := d \log(\frac{ndD_X}{\lambda\delta}/)$.

For the success queries, it introduces a λ error in the inner product. Thus, the results is (c, τ, λ)-Max-IP. Then, following Theorem B.2, we finish the proof. □

Next, we show a modified Version of Theorem C.2 with (c, τ, λ)-Max-IP.

Theorem E.3 (Modified Version of Theorem C.2)**.** *Let* MDP$(\mathcal{S}, \mathcal{A}, H, \mathbb{P}, r)$ denote a linear MDP with core sets $\mathcal{S}_{\text{core}}$, $\mathcal{A}_{\text{core}}$ (see Definition A.7) and span matrix Φ (see Definition A.8). If we query each $\phi(s_j, a_j)$ in the jth row of Φ for $n = O(\epsilon^{-2}L^2H^4\iota)$ times, where $\iota = \log(Hd/p)$, the output policy of Sublinear LSVI with (c, τ, λ)-Max-IP parameter $c = 1 - C_0L \cdot \sqrt{\iota/n}$ and $\lambda = C_0LH \cdot \sqrt{\iota/n}$ would be ϵ-optimal with probability at least $1 - p$. In other words, the regret of Sublinear LSVI is at most $O(C_0LH^2\sqrt{\iota/n})$. Moreover, with $\mathcal{T}_{\mathsf{init}} \cdot \kappa$ preprocessing time and $\mathcal{S}_{\mathsf{space}} \cdot \kappa$ space, the value iteration complexity of Sublinear LSVI is $O(HSdA^\rho \cdot \kappa)$, where $\kappa := d \log(ndD_X/(\lambda\delta))$, D_X is the maximum diameter of weight.

Further more,

- If $\mathcal{T}_{\mathsf{init}} = O(SdA^{1+\rho})$ and $\mathcal{S}_{\mathsf{space}} = O(SA^{1+\rho} + SdA)$, then $\rho = 1 - \frac{C_0L\sqrt{\iota/n}}{4}$.
- If $\mathcal{T}_{\mathsf{init}} = O(SdA^{1+o(1)})$ and $\mathcal{S}_{\mathsf{space}} = O(SA^{1+o(1)} + SdA)$, then $\rho = 1 - \frac{C_0^2L^2\iota}{8n}$.

Proof. We start with showing the modified version of value difference. Because the quantization transforms (c, τ)-Max-IP into a (c, τ, λ)-Max-IP with a λ additive error, we rewrite the value difference as:

$$\begin{aligned} V_1^*(s) - \widehat{V}_1(s) \leq & \mathop{\mathbb{E}}_{\pi^*}\Big[\sum_{h=1}^H [(\mathbb{P}_h - \widehat{\mathbb{P}}_h)\widehat{V}_{h+1}](s_h, a_h) \Big| s_1 = s\Big] + (1-c)\sum_{h=1}^H (H + 1 - h) + \lambda \cdot H \\ = & \mathop{\mathbb{E}}_{\pi^*}\Big[\sum_{h=1}^H [(\mathbb{P}_h - \widehat{\mathbb{P}}_h)\widehat{V}_{h+1}](s_h, a_h) \Big| s_1 = s\Big] + \frac{1-c}{2} \cdot H(H+1) + \lambda \cdot H \end{aligned} \tag{45}$$

where the first step adds λ error over each step based on Lemma C.1, and the second step is a reorganization.

Next, we bound the $V_1^*(s) - \widehat{V}_1(s)$ as:

$$V_1^*(s) - \widehat{V}_1(s) \leq \mathop{\mathbb{E}}_{\pi^*}\Big[\sum_{h=1}^H [(\mathbb{P}_h - \widehat{\mathbb{P}}_h)\widehat{V}_{h+1}](s_h, a_h) | s_1 = s\Big] + \frac{1-c}{2} \cdot H(H+1) + \lambda \cdot H$$

$$
\begin{aligned}
&\leq H \cdot L \cdot C_0 \cdot H \cdot \sqrt{\iota/n} + \frac{1-c}{2} \cdot H(H+1) + \lambda \cdot H \\
&= L \cdot C_0 \cdot H^2 \cdot \sqrt{\iota/n} + \frac{1-c}{2} \cdot H(H+1) + \lambda \cdot H \\
&\leq L \cdot C_0 \cdot H^2 \cdot \sqrt{\iota/n} + (1-c)H^2 + \lambda \cdot H \\
&\leq 2C_0 L H^2 \sqrt{\iota/n} + \lambda \cdot H \\
&\leq 3C_0 L H^2 \sqrt{\iota/n} \\
&\leq \epsilon,
\end{aligned}
$$

where the first step follows from Eq. (45), the second step follows the upper bound of $[(\mathbb{P}_h - \widehat{\mathbb{P}}_h)\widehat{V}_{h+1}](s,a)$ in Eq. (25), the third step is an reorganization, the forth step follows from $H \geq 1$ so that $H^2 \geq H$, the fifth step follows from $1 - c = C_0 L\sqrt{\iota/n}$, the sixth step follows from $\lambda = C_0 LH \cdot \sqrt{\iota/n}$, the seventh step follows from $n = O(C_0^2 \cdot \epsilon^{-2} L^2 H^4 \iota)$.

Using Theorem E.2, we derive the preprocessing time, space and query time for value iteration in Sublinear LSVI. Because the value iteration complexity dominates Sublinear LSVI, the final runtime complexity is $O(HSdA^\rho \cdot \kappa)$ with ρ strictly smaller than 1. □

E.2 Sublinear LSVI-UCB with Adaptive Max-MatNorm Queries

In this section, we show how to tackle adaptive Max-MatNorm queries in sublinear LSVI-UCB.

We start with defining the quantized approximate Max-MatNorm.

Definition E.4 (Quantized Approximate Max-MatNorm)**.** *Let $c \in (0,1)$ and $\tau \in (0,1)$. Let $\lambda \geq 0$. Given an n-point dataset $Y \subset \mathbb{S}^{d-1}$, the goal of the (c,τ,λ)-Max-MatNorm is to build a data structure that, given a query $x \in \mathbb{S}^{d-1}$ with the promise that there exists a datapoint $y \in Y$ with $\langle x, y\rangle \geq \tau$, it reports a datapoint $z \in Y$ with similarity $\langle x, z\rangle \geq c \cdot \textsf{Max-MatNorm}(x, Y) - \lambda$.*

Next, we present how to extend quantized approximate Max-IP to approximate Max-MatNorm.

Theorem E.5 (A modified version of Theorem B.6)**.** *Let $c \in (0,1)$, $\tau \in (0,1)$ and $\lambda \in (0,1)$. Let vec denote the vectorization of $d \times d$ matrix into a d^2 vector. Given a set of n-points Y and $yy^\top \in \mathbb{S}^{d^2-1}$ for all $y \in Y$, one can construct a data structure with with $\mathcal{T}_{\textsf{init}} \cdot \kappa$ preprocessing time and $\mathcal{S}_{\textsf{space}} \cdot \kappa$ space so that for every query $x \in \mathbb{R}^{d\times d}$ with $\mathrm{vec}(x) \in \mathbb{S}^{d^2-1}$ in an adaptive sequence $X = \{x_1, x_2, \cdots, x_T\}$, we take query time $O(d^2 n^\rho \cdot \kappa)$:*

- *if $\textsf{Max-MatNorm}(x, Y) \geq \tau$, then we output a vector in Y which is a (c,τ,λ)-Max-MatNorm with respect to (x, Y) with probability at least δ, where $\rho := f(c,\tau) + o(1)$.*
- *otherwise, we output* fail.

where $\kappa := d\log(ndD_X/(\lambda\delta))$ and $\rho \in (0,1)$. We use D_X to represent maximum diameter in ℓ_2 distance of all queries in X after vectorization.

Further more,

- *If $\mathcal{T}_{\textsf{init}} = O(d^2 n^{1+\rho})$ and $\mathcal{S}_{\textsf{space}} = O(n^{1+\rho} + d^2 n)$, then $f(c,\tau) = \frac{1-\tau^2}{1-c^2\tau^2+\tau^2}$.*
- *If $\mathcal{T}_{\textsf{init}} = O(d^2 n^{1+o(1)})$ and $\mathcal{S}_{\textsf{space}} = O(n^{1+o(1)} + d^2 n)$, then $f(c,\tau) = \frac{2(1-\tau^2)^2}{(1-c^2\tau^2)^2} - \frac{(1-\tau^2)^4}{(1-c^2\tau^2)^4}$.*

Proof. We start with applying (c^2, τ^2, λ)-Max-IP data structure over $\mathrm{vec}(x)$ and $\mathrm{vec}(YY^\top)$. Then, we would obtain a $z \in Y$ that

$$
\langle \mathrm{vec}(x), \mathrm{vec}(zz^\top)\rangle \geq c^2 \max_{y\in Y} \langle \mathrm{vec}(x), \mathrm{vec}(yy^\top) - \lambda\rangle \tag{46}
$$

we could use it and derive the following propriety for z:

$$\begin{aligned}
\|z\|_x &= \sqrt{\langle \mathrm{vec}(x), \mathrm{vec}(zz^\top)\rangle} \\
&\geq \sqrt{c^2 \max_{y\in Y} \langle \mathrm{vec}(x), \mathrm{vec}(yy^\top)\rangle - \lambda} \\
&\geq \sqrt{c^2 \max_{y\in Y} \langle \mathrm{vec}(x), \mathrm{vec}(yy^\top)\rangle} - \sqrt{\lambda} \\
&\geq c \max_{y\in Y} \sqrt{\langle \mathrm{vec}(x), \mathrm{vec}(yy^\top)\rangle} - \lambda \\
&= c \max_{y\in Y} \|y\|_x - \lambda,
\end{aligned}$$

where the second step follows from Eq. (46), the third step follows from Cauchy-Schwartz inequality, the forth follows from $\lambda \in (0,1)$, the last step is a reorganization.

Thus, z is the solution for (c,τ,λ)-Max-MatNorm(x,Y). Next, applying Theorem E.2 , we finish the proof.

□

Theorem E.6 (Modified Version of Theorem D.12)**.** *Let* MDP$(\mathcal{S},\mathcal{A},H,\mathbb{P},r)$ denote a linear MDP. For any probability $p \in (0,1)$ that is fixed, if we set approximate Max-MatNorm parameter $c = 1 - \frac{\iota}{\sqrt{K}}$, quantization error $\lambda \leq \sqrt{H^2 K}$ and Sublinear LSVI-UCB parameter $\beta = \Theta(dH\sqrt{\iota})$ with $\iota = \log(2dT/p)$, then the Sublinear LSVI-UCB (Algorithm 4) has regret at most $O(C_\beta \cdot \sqrt{d^3H^4K\iota^2})$ with probability $1-p$. Moreover, with $\mathcal{T}_{\mathsf{init}} \cdot \kappa$ preprocessing time and $\mathcal{S}_{\mathsf{space}} \cdot \kappa$ space, the value iteration complexity of Sublinear LSVI-UCB is $O(HKd^2A^\rho \cdot \kappa)$, where $\kappa := d\log(ndD_X/(\lambda\delta))$, D_X is the maximum diameter of weight.

Further more

- If $\mathcal{T}_{\mathsf{init}} = O(Kd^2A^{1+\rho})$ and $\mathcal{S}_{\mathsf{space}} = O(KA^{1+\rho} + Kd^2A)$, then $\rho = 1 - \frac{1}{4\sqrt{K}}$.
- If $\mathcal{T}_{\mathsf{init}} = O(Kd^2A^{1+o(1)})$ and $\mathcal{S}_{\mathsf{space}} = O(KA^{1+o(1)} + Kd^2A)$, then $\rho = 1 - \frac{1}{8K}$.

Proof. We start with showing the modified version of Q-function difference $Q_1^*(s,a) - Q_1^k(s,a)$. Because the quantization transforms (c,τ)-Max-IP into a (c,τ,λ)-Max-IP with a λ additive error, we rewrite the $Q_1^*(s,a) - Q_1^k(s,a)$ as:

$$Q_1^*(s,a) - Q_1^k(s,a) \leq (H - c\frac{1-c^H}{1-c}) + H\lambda.$$

Next, we could upper bound the regret with probability $1-p$ as:

$$\begin{aligned}
\mathrm{Regret}(K) &\leq 2K\gamma H^2 + 2H\sqrt{T\iota} + \beta H\sqrt{2dK\iota} + H\lambda \\
&= 2K\gamma H^2 + 2H\sqrt{T\iota} + C_\beta \cdot \sqrt{2d^3H^4K\iota^2} + H\lambda \\
&= 2\sqrt{H^4K\iota^2} + 2\sqrt{H^3K\iota} + C_\beta \cdot \sqrt{2d^3H^4K\iota^2} + H\lambda \\
&= 3\sqrt{H^4K} + 2\sqrt{H^3K\iota} + C_\beta \cdot \sqrt{2d^3H^4K\iota^2} \\
&\leq 2C_\beta\sqrt{d^3H^4K\iota^2},
\end{aligned}$$

where the first step follows from Eq. (44), the second step follows from $\beta = C_\beta \cdot dH\sqrt{\iota}$, the third step follows from $\gamma = \frac{1}{\sqrt{K}}$, the forth step is a reorganization follows from $\lambda \leq \sqrt{H^2K}$, the last step follows from $C_\beta \geq 100$.

Using Theorem E.5, we derive the preprocessing time, space and query time for value iteration in Sublinear LSVI-UCB. Because the value iteration complexity dominates LSVI-UCB, the final runtime complexity is $O(HKd^2A^\rho \cdot \kappa)$ with ρ strictly smaller than 1. We alternate the S in preprocessing and space by K since $K > S$. Note that to let ρ strict less than 1. We set $c^2 \in [0.5, 1)$ and $\tau^2 \in [0.5, 1)$. □