A Survey on the Computational Hardness of linear-structured Markov Decision Processes

by

Chuhuan Huang

A Thesis Presented to the
FACULTY OF THE DANA AND DAVID DORNSIFE COLLEGE OF LETTERS,
ARTS AND SCIENCE
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
MASTER OF ARTS
(APPLIED MATHEMATICS)

May 2023

Acknowledgements

This survey paper and related research are conducted under the supervision of Professor Steven Heilman, and funded by Graduate Teaching Assistantship, Department of Mathematics, University of Southern California, 2022-2023 academic year.

The author would like to thank Professor Steven Heilman, Professor Jianfeng Zhang, and Professor Remigijus Mikulevicius for their patient mentoring and advising in his early career in Mathematics and in the application for his Ph.D. program, as the thesis committee.

The author would like to thank his family: father Xuegong Huang and mother Qian Cheng, for their caring and support since November 1997. Though they sometime may be hard to communicate, indeed their love exists in a strong sense.

The author would like to thank his friends, both here in California and back in the People's Republic of China, for their mental support and inspiration.

The author would like to thank a philosopher from Philadelphia, who was known for his mentality and resilience. He inspired the author so much in every stage of life and every aspect of the author's life philosophy: something greater will.

In loving memory of his mother's mother and the philosopher.

Table of Contents

Acknowledgements ii

Abstract iv

Chapter 1: Introduction 1

Chapter 2: Preliminaries 7

2.1 Reinforcement learning and Markov decision processes 7

2.2 Complexity theory terminologies and computationally hard problems 12

Chapter 3: The Computational-Statistical gaps of Linear Q^*/V^* Class 14

3.1 The Sharp Result 16

3.2 The Proof of the Reduction Proposition 17

3.3 High rank remark 22

Chapter 4: Linear MDPs are both computational and statistical efficient 23

4.1 The Algorithm and its Time Complexity 25

4.2 Correctness and Sample Complexity 28

4.3 Remarks and Discussions 31

Chapter 5: In showing the analogous result for Linear Mixture MDPs 32

Bibliography 35

Appendix: Technical Details 37

Abstract

The curse of dimensionality in reinforcement learning leads to the function approximation technique to approximate the (action) value function. However, unfortunately, this approximation technique makes the provability much challenging in general. In [5], they proved there exists a reinforcement learning algorithm that is both statistically and computationally efficient by assuming the linearity of the transition probability and reward function. Therefore, it is natural to ask,*what is the minimal requirement for the environments/ dynamics to ensure the existence of a RL algorithm that is provably both statistically and computationally efficient?*

Driven by this question, this thesis will survey the results of [6] and [5], along with some other related results. We will discuss some analogous work that could be done for the Linear Mixture MDPs, introduced in [8], [1], [17], which is a main focus of our future work.

Chapter 1

Introduction

Suppose in a virtual world, a warrior wants to strengthen a piece of his equipment X, to make it more powerful and triple its market price. To strengthen this equipment successfully, he has to elevate the numerical attribute of X from 50 to 110 in precisely 7 steps. In each step, he could choose 3 different types of tools to elevate the numeric, but different tools have different random effects on X and have different price. How should he choose the tool in each step to maximize his gain, even facing uncertainties and randomness? This is a typical question considered in the context of reinforcement learning.

Since the 1956 workshop in Dartmouth, Machine Learning (ML), as a major field in Artificial Intelligence (AI), has been growing rapidly. This field was initially aiming in training a machine to predict a general pattern (the model) from limited observed data. The machine is said to be "learning" throughout a "training process." Some exmaple of recent progress include: advancements in the machines playing games such as Go, namely AlphaGo's superhuman performance versus Lee Sedol in 2016, together with land-changing literature [11], the invention of the Transformer and attention machanism [16] in 2017, and, the game-changing chatbot storm from GPT-4

[9]. Artificial intelligence and Machine Learning and their applications have been a worldwide influential phenomenon in every aspect of social and economic life.

Throughout their history, different machine learning paradigms emerged[10], including supervised learning, unsupervised learning, and reinforcement learning.

Supervised learning is a type of machine learning paradigm where the model is trained by manually-labeled data (the input data set is called the training set). Upon finishing a successful training process, the machine should be able to predict the label for unlabeled data outside of the training set. Two typical jobs of supervised learning are classification and regression, corresponding to discrete labels and continuous labels, respectively.

Unsupervised learning is a type of machine learning paradigm where the model is trained by an unlabeled training set, and upon finishing the training process, the machine should be able to partition the feeding datasets properly based on their intrinsic features. A typical job of unsupervised learning is clustering, where, upon successfully trained, the model should partitions the input data set into several subsets based on their intrinsic attribute.

In between the previous two paradigms, semi-supervised learning is a machine learning paradigm where the model is trained by a dataset that is largely unlabeled, except for a small labeled portion. In practice, large-scale data labeling is generally expensive, so semi-supervised learning alleviates this issue and combines the advantages of accuracy in supervised learning and the cost-efficiency in unsupervised learning.

Supervised learning, semi-supervised learning, and unsupervised learning are listed in decreasing order in the dependence on human guidance. However, human is not, or never was, the only guidance for the learning process; the environment is also a natural guidance, which leads to our main interest, reinforcement learning.

Reinforcement learning (RL)[10][14] is a machine learning paradigm where the model (typically we use the word agent) that is trained through a sequence of interactions with the surrounding environment: in each step, the agent observes the current environment, makes actions, gets numerical feedback, called a reward, directly from the environment. In contrast to the previous paradigms, the agent receives a numerical reward instead of a manual label, and the environment changes reacting to the agent's action. The typical goal of reinforcement learning is to train the model, such that the agent would be able to choose a proper action in each round of interactions to maximize the total collected rewards from the environment.

Consider the example in the opening paragraph. The warrior could realize that there is only finite many possible numeric (called state) he could go, and he has only finite many choices in actions. If he knows the distribution of tool's effect on sending a given numeric to another (called transition probability), he could apply Dynamic Programming (DP) to solve to maximize his expected revenue. Unfortunately, this distribution is typically unknown to the warrior. In this case, he could apply Monte Carlo method[10] to get an empirical distribution and then apply DP. However, the sampling process in Monte Carlo method is usually expensive in terms of time and monetary cost required, but in order to get an empirical distribution close enough to the true distribution, the warrior has to sample as much as he could. To overcome this dilemma between the accuracy and the cost, it is sensible for the warrior to apply SARSA [14].

In SARSA[14], instead of estimating transition probability for all state-action pair, the model estimates the collected reward $Q(s, a)$, given initial action a and initial state s directly, and moreover it updates the estimation by the Bellman equation, current state s and chosen action a, the reward r collected in this step, and next state s' after the action a and next action a'; The action choosing policy chooses the action such that it maximizes the estimation. Under this implementation, the the total amount of computation (time complexity) decreases, but it is still exponential in the size of all states ($|\mathcal{S}|$) and all actions ($|\mathcal{A}|$), which makes the larger scale implementation impractical.

The curse of dimensionality in reinforcement learning [5] refers to the fact that the computational and sample complexity (numbers of samples) of learning algorithms tends to grow exponentially with the number of dimensions or features in the state space, and hence it becomes much more difficult to compute the exact value function V (expected reward to be collected, a function of the initial state) and learn a good policy based on the actual V.

A typical way to overcome this curse is to apply (value) function approximation, in theory, and in practice, where the (action) value function is parameterized and estimated during training, typically via deep neural networks (e.g. Deep Q Network, DQN[7], is a modified Q-Learning where the action value function Q is parameterized by a deep neural network, and this network can be trained by Stochastic Gradient Descent where the loss function is defined as L^2 distance between the current estimate and scaled next estimate plus an immediate reward[7]).

Even though the function approximation technique is applied in practice to accommodate to large-scale $|\mathcal{S}|$ and $|\mathcal{A}|$, the amount of time involved training's computation is still intimidating. For example, the world-famous AlphaZero [12] went through a 13-day-training, even with 5000 tensor processing units (TPUs); DOTA2 bots trained by OpenAI Five took 10 months in real-time, with 128000 GPUs [2].

Function approximation unleashes the power of RL by enlarging the effective states and action spaces, but its provability is technically challenging: the neighborhood of most of the states remains unvisited during training episodes, impairing the reliability of the estimates of the value functions [14]. To deal with this challenge, a typical approach is to assume some linearity in the structure of the dynamics, hoping to facilitate the theoretical proofs. In [5], by assuming the linearity of the transition probability and reward function with respect to some known feature maps, they proved there exists a reinforcement learning algorithm that is both statistically and computationally efficient (the amount of samples and the amount of computation required are polynomial in some intrinsic structural property), in the function approximation setting.

Thus, as of our main interest, a natural question is, *what is the minimal requirement for the environments/ dynamics to ensure the existence of a RL algorithm that is provably both statistically and computationally efficient?*

Driven by the natural question above, discussion in statistical efficiency, or say sample efficiency, becomes as a main body in the reinforcement learning theory community. In [3], by assuming the linearity of the optimal action value function and optimal value function with respect to some known feature, sample efficiency is possible. However, recently, in [6], by following the same assumption: the linear optimal value function and action value function with respect to some known feature, they first proposed a gap between the statistical efficiency and computational efficiency. That is, under their assumed linearity, the sample efficiency is possible [3], but there is no randomized algorithm to solve the proposed RL problem in polynomial time, where it is polynomial with respect to a known feature dimension d.

This thesis will survey the results of [6] and [5] along with some other related results. We will review some necessary background preliminaries in section 2, then we will review the computational hardness for Linear Q^*/V^* MDPs in [6] in section 3; in section 4, we will go over the main idea and the structure in [5] for Linear MDPs, showing the existence of mathematically provable RL algorithm that is both computational and statistically efficient. Further, in section 5, we will discuss some analogous work that could be done for the Linear Mixture MDPs, introduced in [8], [1], [17], which is a main focus of our future work. We also attach an appendix section for some technical details.

Chapter 2

Preliminaries

In this chapter, we review some preliminaries that are used throughout this thesis.

2.1 Reinforcement learning and Markov decision processes

In reinforcement learning (RL), the agent interacts with the surroundings, gets feedback, and learns to adjust its strategy to survive and thrive. A typically choice of the underlying framework is the Markov decision process[14].

Given a filtered probability space $(\Omega, \mathscr{F}, \mathbb{F}, \mathbb{P})$, where Ω is the sample space, $\mathscr{F}$ is the σ-algebra on Ω, $\mathbb{F}$ is the filtration, and $\mathbb{P}$ is the probability measure on Ω.

Definition 2.1. *A (discrete time, episodic, uniformly-bounded reward, time-homogeneous)* ***Markov Decision Process (MDP)*** *M is a stochastic process defined by a tuple $(\mathcal{S}, \mathcal{A}, p(\cdot|\cdot,\cdot), R, H)$, where*

- *$H \in \mathbb{Z}^+$ is the time horizon, or say the length of each episode.*

- *$\mathcal{S}$ is the non-empty set of all possible states, called the **State Space**, and there is a distinguished terminal state in $\mathcal{S}$, denoted as $s_\perp$. $\mathcal{S}$ is equipped with its own σ-algebra $\mathscr{F}_\mathcal{S}$ that contains all singleton $\{s\}, \forall s \in \mathcal{S}$.*

- *$\mathcal{A}$, the **Action Space**, is the set of all possible actions, also equipped with its own σ-algebra $\mathscr{F}_\mathcal{A}$ that contains all singleton $\{a\}, \forall a \in \mathcal{A}$.*

- *$p : \mathscr{F}_\mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0,1]$, defined as, for every $(B, s', a) \in \mathscr{F}_\mathcal{S} \times \mathcal{A} \times \mathcal{S}$, $p(B|s,a) := \mathbb{P}(S_{t+1} \in B | S_t = s, A_t = a)$, is the **transition kernel**, where*

  - *$S : \{0, 1, 2, ..., H\} \times \Omega \to \mathcal{S}$ is the state process;*
  - *$A : \{0, 1, 2, ..., H\} \times \Omega \to \mathcal{A}$ is the action process;*
  - *the dynamics satisfy the Markov property, i.e. for any bounded measurable function f,*

$$\mathbb{E}[f(S_{t+1})|\sigma(S_t, A_t, S_{t-1}, ..., S_0, A_0)] = \mathbb{E}[f(S_{t+1})|\sigma(S_t, A_t)]. \tag{2.1}$$

  Note here we assume p does not depend on t, or say the transition is assumed to be time-homogenous, and we sometimes abuse the notation by writing $p(s'|s,a) := p(\{s'\}|s,a)$.

- *$R : \mathcal{S} \times \mathcal{A} \to \triangle([0,1])$, defined by*

$$R(s,a) := R_{s,a},$$

is the ***(immediate, stochastic) reward****, where* $\triangle([0,1])$ *is the space of all probability measures over* $[0,1]$*. For all* $(s,a) \in \mathcal{S} \times \mathcal{A}$*, we could also define the* ***expected reward*** $r : \mathcal{S} \times \mathcal{A} \to [0,1]$ *by*

$$r(s,a) := \int_{[0,1]} x R_{s,a}(dx).$$

In an episode, we illustrate the agent-environment interactions as the following: an agent starts to observe some initial state $s_0 \in \mathcal{S}$ of the environment. At each step t, the agent observe that the environment is on state s_t; he takes action a_t, and collects a reward sampling from the distribution corresponding to R_{s_t,a_t}; the environment reacts to his action and transitions from the current state s_t to a next state, following the transition $p(\cdot|s_t,a_t)$. The agent continues to interact with the environment until he observes $s_\perp$ or $t = H$. We shall denote

$$\tau := \inf\{t \in \mathbb{Z}^+ : S_t = s_\perp\} \wedge H,$$

as the first time that the interaction stops. We say this sequence of states from s_0 to s_τ, a **trajectory**.

A **reinforcement learning problem** (**RL problem**) is, given an MDP M, to find an action-choosing function (called **policy**) for the agent upon observing the current state of the environment that maximize his expected rewards collected through the trajectory. Sometimes, we may abuse the terminology that we interchangeably use the phrases "solving an RL problem" and "solving an MDP".

More rigorously, a (deterministic) **policy** is a function $\pi : \mathcal{S} \to \mathcal{A}$ (write as $\pi \in \mathcal{A}^{\mathcal{S}}$, where $\mathcal{A}^{\mathcal{S}} := \{f : \mathcal{S} \to \mathcal{A}\}$) and we define the **state value function** $V^{\pi} : \mathcal{S} \to \mathbb{R}$, the expected reward to be collected along the path given a starting state $s \in \mathcal{S}$ under policy π:

$$V^{\pi}(s) := \mathbb{E}\Big[\sum_{t=0}^{\tau-1} r(S_t, \pi(S_t))\Big| S_0 = s\Big]$$

where $S_1, A_1, ..., S_{\tau-1}, A_{\tau-1}$ are obtained via executing π in M, i.e. $A_t = \pi(S_t), \forall t \in [0, \tau-1] \cap \mathbb{Z}$.

We also define the **state-action value function** $Q^{\pi} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, the expected reward to be collected along the path given a starting state s and initial action a, under policy π:

$$Q^{\pi}(s,a) := r(s,a) + \mathbb{E}\Big[\sum_{t=1}^{\tau-1} r(S_t, \pi(S_t))\Big| S_0 = s\Big].$$

We remarked that, given a fixed policy $\pi \in \mathcal{A}^{\mathcal{S}}$, the sequence $(S_t)_{t=0}^{\tau}$ is a Markov process. Therefore, by applying the Markov property, we obtain the **Bellman equation**[14] for action-value function Q and the value function V.

Lemma 2.1 (Bellman equation). *Given an MDP $M = (\mathcal{S}, \mathcal{A}, p, R, H)$ and a policy π, for any $s \in \mathcal{S}$, $a \in \mathcal{A}$, V^{π} and Q^{π} satisfy the following equation:*

$$Q^{\pi}(s,a) = r(s,a) + \mathbb{E}_{s' \sim p(\cdot|s,a)} V^{\pi}(s'),$$

where $\mathbb{E}_{s' \sim p(\cdot|s,a)} f(s') := \int_{\mathcal{S}} f(s') p(ds'|s,a)$.

We conclude this section by our natural interest in

$$V^*(s) := \sup_{\pi \in \mathcal{A}^{\mathcal{S}}} V^\pi(s)$$

and

$$Q^*(s,a) := \sup_{\pi \in \mathcal{A}^{\mathcal{S}}} Q^\pi(s,a),$$

the **optimal state value function** and **optimal state-action value function**, respectively. Similarly to the Bellman equation, we have the **Bellman optimality equation**[14]:

Lemma 2.2 (Bellman optimality equation)**.** *Given an MDP $M = (\mathcal{S}, \mathcal{A}, p, R, H)$, for any $s \in \mathcal{S}$, $a \in \mathcal{A}$, V^* and Q^* satisfy the following equations:*

$$V^*(s) = \sup_{a \in \mathcal{A}} Q^*(s,a), \quad Q^*(s,a) = r(s,a) + \mathbb{E}_{s' \sim p(\cdot|s,a)} V^*(s')$$

The **optimal policy** $\pi^* : \mathcal{S} \to \mathcal{A}$ is a policy s.t. $V^{\pi^*} = V^*$ and $Q^{\pi^*} = Q^*$.

2.2 Complexity theory terminologies and computationally hard problems

We first recall some complexity concepts from theoretical computer science [13].

A decision problem is a computational problem in that for any inputs, there are only two possible outputs (*yes* or *no*). We say a decision problem is in **RP** class if it can be solved by a randomized algorithm in polynomial time; a decision problem is in **NP** class if its solution can be verified by an algorithm in polynomial time. We further say it is **NP-complete** if every problem in NP is polynomial-time reducible to it.

It is well-known that the RP class is a subset of the NP class, i.e. every decision problem in the RP class is in the NP class, while whether the reverse inclusion is true remains a major open problem in the community of theoretical computer science.

Now we move to a more concrete problem: the **UNIQUE-3-SAT** problem. We begin with some pre-requisite terminologies.

A **Boolean expression**, or **propositional logic formula**, is a sequence of Boolean variables, operators **AND** (conjunction, denoted as $\wedge$)[*], **OR** (disjunction, denoted as $\vee$), **NOT** (negation, denoted as $\neg$), and parentheses. For example, $(x_1 \vee x_2 \vee x_3 \vee \neg x_4) \wedge x_5$ is a Boolean expression. A formula is said to be **satisfiable** if it can be made **TRUE** by assigning Boolean values (i.e. **TRUE**, **FALSE**) to its variables. The **Boolean Satisfiability Problem**, or **SAT**, is to check whether a given input formula is satisfiable.

[*]Please distinguish the notation $\wedge$ for a minimum of two and for AND based on the context, i.e. it means AND when it comes to Boolean variables and it means minimum otherwise.

A **literal** is either a variable or the negation of a variable. A **clause** is a parenthesized disjunction of literals (or a single literal). A formula is said to be **3-conjunctive normal form**, or **3-CNF**, if it is a conjunction of clauses (or single clause), where clauses are of length 3, i.e. a clause that is connected by two $\vee$. An example of 3-CNF formula is $(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$. **UNIQUE-3-SAT** problem is an SAT problem where the input 3-CNF formula is ensured to have either 0 or 1 satisfying assignment.

Also, we may frequently make statements[6] like that a randomized algorithm A solves a problem in time t with error probability p, by which we means that after running $\mathcal{O}(t)$ total amount of steps, and, with at least $1-p$ probability, A solves the problem. Here $\mathcal{O}$ is the big O notation. More specifically, for an SAT problem, it returns **YES** on a satisfiable formula with at least $1-p$ probability and it with probability 1 returns **NO** on an unsatisfiable formula. For a reinforcement learning problem, with probability at least $1-p$, it returns a ε-optimal policy π, such that $\sup_{s\in\mathcal{S}} |V^*(s) - V^\pi(s)| < \varepsilon$. We typically use $poly(n)$ as the abbreviation of "polynomial in n".

We finish this section by pointing out a main result related to the Unique-3-SAT problem. It is well-known that the Unique-3-SAT problem is NP-complete by Cook-Levin Theorem[13]; further, [15] states a stronger result:

Theorem 2.3 (Valiant-Vazirani, 1985[15][6]). *Unless $NP = RP$, no randomized algorithm can, with probability at least $7/8$, solve UNIQUE-3-SAT with v variables in $poly(v)$ time.*

Chapter 3

The Computational-Statistical gaps of Linear Q^*/V^* Class

In this chapter, we review a recent article [6] in the reinforcement learning theory community that first reveals a sharp computational-statistical gap in reinforcement learning.

Before we specify this gap, we first introduce a classification for MDPs:

Definition 3.1 ([6]). *We say an MDP M satisfies **linear** Q^*/V^* **condition** if there exists a $d \in \mathbb{N}$, known feature maps $\psi : \mathcal{S} \to \mathbb{R}^d$, $\tilde{\psi} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$, and an unknown*[*] *fixed $\theta, \omega \in \mathbb{R}^d$ s.t. for all $s \in \mathcal{S}, a \in \mathcal{A}$,*

$$V^*(s) = \langle \psi(s), \omega \rangle, \quad Q^*(s,a) = \langle \tilde{\psi}(s,a), \theta \rangle.$$

*We shall call this class of MDPs the **linear** Q^*/V^* **class**, and refer d as the **feature dimension**.*

A basic assumption is that, if we try to use an algorithm to solve the RL problem associated with a linear Q^*/V^* MDP, transition p, the features ψ, $\tilde{\psi}$, and the immediate reward R are assumed accessible by this algorithm. The call of each function above is assumed to take constant runtime; input-parsing/output-computing for these functions takes $poly(d)$ time . In this sense, the transition p and the immediate reward R sampling from the probability measure are known.

[*]By known, we mean it is at most polynomial-time computable, while constant time is possible but not required. By unknown, we don't know if there is a randomized algorithm to compute it in polynomial time.

Now, with the notion above in mind, we may specify the computational-statistical gap in reinforcement learning: MDPs in linear Q^*/V^* class, as long as $|\mathcal{A}|$ is a fixed and finite constant, are statistically efficient to solve as presented in [3], meaning it merely requires $poly(d)$ samples; yet they are computationally hard to solve, i.e. unless $NP = RP$, there is no randomized algorithm can be used to solve them in $poly(d)$ time[6].

For the rest of the section, we first review the computational hardness for this class of MDPs. We first specify the RL problem of major interest: LINEAR-3-RL[6].

A **LINEAR-3-RL** problem is a RL problem where the MDP M satisfies the following properties:

- $|\mathcal{S}| < \infty$.
- it has deterministic transition P, i.e. given $(s, a) \in \mathcal{S} \times \mathcal{A}$, $\exists s' \in \mathcal{S}$ uniquely s.t. $p(s'|s, a) = 1$, and $p(\cdot|s, a) = 0$ elsewhere. We accordingly define $P : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ by $P(s, a) = s'$ if $p(s'|s, a) = 1$.
- $|\mathcal{A}| = 3$, i.e. it has 3 actions.
- it satisfies the linear Q^*/V^* condition.
- it has horizon $H = \mathcal{O}(d)$, where d is the feature dimension in the linear Q^*/V^* condition.

A LINEAR-3-RL is viewed as a tuple (Oracle M, goal G), where the M is an MDP satisfying the properties stated above and the goal G is to find a policy π s.t. $\sup_{s\in\mathcal{S}} |V^* - V^\pi| < \frac{1}{4}$, or say this LINEAR-3-RL is solved by finding such policy.

3.1 The Sharp Result

We now present the major computational hardness result of reinforcement learning in [6].

Theorem 3.1 (D. Kane et al., 2022[6]). *Unless $NP = RP$, no randomized algorithm can solve, with error probability at least $9/10$, LINEAR-3-RL with feature dimension d in $poly(d)$ time.*

The punchline of the proof of this theorem is the following reduction that connects the computational hardness of solving the given MDP to the well-known computational hardness of solving the UNIQUE-3-SAT problem.

Proposition 3.2 ([6])**.** *Suppose $q \geq 1$. If LINEAR-3-RL with feature dimension d can be solved in time d^q with probability at least 9/10, then UNIQUE-3-SAT with v variables can be solved in time $poly(v)$ with probability at least $7/8$.*

Together with the well-established computational hardness result for UNIQUE-3-SAT, the theorem 2.3, this reduction implies the computational hardness of the LINEAR-3-RL, the theorem 3.1.

3.2 The Proof of the Reduction Proposition

In this sub-subsection, we sketch the proof of the reduction and leave the technical details in the appendix for reference.

We divide the proof into several steps:

1. We construct an MDP M_φ, based on the given 3-CNF formula φ, which happens to satisfy the linear Q^*/V^* condition.

2. We construct an algorithm $\mathcal{A}_{SAT}$ to solve UNIQUE-3-SAT using an algorithm $\mathcal{A}_{RL}$ solving the M_φ, such that by solving the M_φ, i.e. the RL algorithm returns a good policy, $\mathcal{A}_{SAT}$ solves the UNIQUE-3-SAT problem by finding the satisfying assignment.

More specifically, given a 3-CNF formula φ that has v variables in total and $\mathcal{O}(v)$ clauses in total, and $w^* \in \{-1,1\}^v$ the unique satisfying assignment, we construct the M_φ as the following

- each state $s = (l, w)$, where $l \in \mathbb{N}$ is the depth and $w \in \{-1,1\}^v$ is the associated assignment. M_φ starts at $s_0 := (0, w_0)$, where $w_0 := (-1, -1, ..., -1) \in \{-1,1\}^v$.

- the deterministic dynamics are defined as:

  – for a non-satisfying state s with assignment $w \neq w^*$ and $l < H$, consider the first unsatisfied clause with variables $x_{i_1}, x_{i_2}, x_{i_3}$. Under action $j \in \{1,2,3\}$, state s transition to s' with assignment $(w'_1, ..., w'_v)$ where $w'_k = \neg w_k$ if $k = i_j$ and $w'_k = w_k$ elsewhere, i.e. the i_j-th bit in the assignment is flipped, and the depth level $l' = l + 1$.

  – for a non-satisfying state s with $w \neq w^*$ and $l = H$, or a satisfying state s with $w = w^*$, every $j \in \{1,2,3\}$ takes s to $s_\perp$.

- the reward is everywhere 0 except on states with satisfying assignments or states on the last layer H. In both case, $s = (l, w)$, for any $a \in \{1, 2, 3\}$, $R(s, a) \sim Ber(g(l, w))$, where $Ber(p)$ is the Bernoulli distribution with parameter p,

$$g(l, w) := \Big(1 - \frac{\text{dist}(w, w^*) + l}{H + v}\Big)^r.$$

  r is a parameter[†] that we will specify it in the later context, and $\text{dist}(\cdot, \cdot)$ is the hamming distance.

Now we claim M_φ satisfies the linear Q^*/V^* condition, and hence the associated RL problem is a LINEAR-3-RL problem.

Claim 3.1. *Given an action a and a state $s = (l, w)$, i.e. it is in level l and it has an assignment w,*

i) We have $V^*(s) = g(l, w)$;

ii) By assuming v is large enough, we could construct features maps $\psi : \mathcal{S} \to \mathbb{R}^d$, $\tilde{\psi} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ with $d \leq 2v^r$ involve nothing beyond information from s and a, and $\theta \in \mathbb{R}^d$ involves nothing beyond information from w^* such that V^* and Q^* are in linear form with respect to the features ψ and $\tilde{\psi}$ i.e.

$$Q^*(s, a) = \langle \theta, \tilde{\psi}(s, a) \rangle, V^*(s) = \langle \theta, \psi(s) \rangle.$$

Now supposedly we are given a randomized algorithm $\mathcal{A}_{RL}$ for the LINEAR-3-RL problem, we could further construct a randomized algorithm $\mathcal{A}_{SAT}$ for the UNIQUE-3-SAT problem based on $\mathcal{A}_{RL}$. Note that the runtime of the $\mathcal{A}_{RL}$ accumulates each call through accessing the transition kernel, and the features, and may assume they require constant time to finish computing.

[†]the notation r represents either this parameter or the expected reward, but it should not be confusing as it is easily distinguishable from the context.

However, in collecting the reward, it is necessary to know the w^* in advance, which cannot be computed efficiently. Therefore, we instead replace the MDP M_φ with a simulator $\bar{M}_\varphi$.

$\bar{M}_\varphi$ has the same transition kernel and features, but it always assigns zero reward at the last layer[‡] i.e. the simulator $\bar{M}_\varphi$ assigns zero reward except on the satisfying state. Now since by our previous assumption on the polynomial runtime of the reward, transition, and features, now whenever $\bar{M}_\varphi$ is called, it takes $poly(d)$-time to execute.

We construct $\mathcal{A}_{SAT}$ for UNIQUE-3-SAT as the following:

On input 3-CNF formula φ,

- $\mathcal{A}_{SAT}$ calls $\mathcal{A}_{RL}$, but each call to M_φ is replaced with the call to $\bar{M}_\varphi$.
- Suppose $\mathcal{A}_{RL}$ returns a policy that the interactions between the agent to the environment end on a state $s = (l, w)$. Then, if w satisfies the formula, $\mathcal{A}_{SAT}$ returns **YES**. Otherwise, it returns **NO**.

To see its correctness, we claim the following:

Claim 3.2. *Suppose $r > 1$ and $H = v^r$. If $\mathcal{A}_{RL}$ returns a policy π such that*

$$\sup_{s \in \mathcal{S}} |V^*(s) - V^\pi(s)| < \frac{1}{4},$$

*then, if φ is satisfiable, $\mathcal{A}_{SAT}$ returns **YES**; otherwise, it returns **NO**.*

[‡]By last layer, we mean state $s = (l, w)$ with $l = H$ and $w \neq w^*$.

Claim 3.3. *Suppose* $r \geq 2$ *and* $H = v^r$. *Suppose* $\mathcal{A}_{RL}$ *with* M_φ *access runs* $v^{\frac{r^2}{4}}$ *number of time steps, and, with probability* $\frac{9}{10}$, *it returns a policy* π *such that*

$$\sup_{s \in \mathcal{S}} |V^*(s) - V^\pi(s)| < \frac{1}{4}.$$

Then, still running $v^{\frac{r^2}{4}}$ *number of time steps,* $\mathcal{A}_{RL}$ *with* $\bar{M}_\varphi$ *access will return a policy* π *such that* $\sup_{s \in \mathcal{S}} |V^*(s) - V^\pi(s)| < \frac{1}{4}$ *with probability at least* $\frac{7}{8}$.

We remark that the π showed up in Claim 3.3 twice and they are not necessary to be the same one.

Together by claim 3.2 and claim 3.3, if there is an algorithm $\mathcal{A}_{RL}$ such that, with probability at least $9/10$, it solves the LINEAR-3-RL with feature dimension $d = 2v^r$ in $v^{r^2/4}$ number of time steps, then the constructed algorithm $\mathcal{A}_{SAT}$ can solve, with probability at least 7/8, the UNIQUE-3-SAT problem in time $v^{r^2/4} \cdot poly(d)$ (it takes $poly(d)$ time for each call to the $\bar{M}_\varphi$, so we multiply this factor).

At this moment, we choose

$$r = 8q,$$

and $q \geq 1$ implies $r \geq 2$, and further we observe,

$$d^q \leq v^{\frac{r^2}{4}} \tag{3.1}$$

and

$$poly(d) \cdot v^{\frac{r^2}{4}} = poly(v), \tag{3.2}$$

where the inequality (1) is given by

$$v^{\frac{r^2}{4}} = (v^r)^{\frac{r}{4}} \geq d^{\frac{r}{8}} = d^q$$

when v is large enough.

Conclusively, if there is an algorithm $\mathcal{A}_{RL}$ that, with probability at least $9/10$, solves the LINEAR-3-RL with feature dimension d in time d^q, by choosing v s.t. $d = 2v^r$, $\mathcal{A}_{RL}$, with probability at least $9/10$, solves the LINEAR-3-RL with feature dimension d in time $v^{r^2/4}$, and hence we know by claim 3.2 and claim 3.3, there is a randomized algorithm $\mathcal{A}_{SAT}$ that, with probability at least $\frac{7}{8}$, solves the v-variable UNIQUE-3-SAT problem in time $v^{\frac{r^2}{4}} \cdot poly(d) = poly(v)$, i.e. the reduction proposition (proposition 3.2)[6] is proved.

3.3 High rank remark

We finish this section by remarking that, by fixing a deterministic policy $\pi \in \mathcal{A}^{\mathcal{S}}$ in the MDP associated with the LINEAR-3-RL problem, the transition matrix in the corresponding Markov chain is a $2^v \times 2^v$ matrix and it is high-rank, i.e. the rank of the transition matrix is of order higher than polynomial in d.

To see this, suppose $N < 2^v/v$ is the number of next states reachable from 2^v initial states, then by pigeonhole principle there exists a next state that can be reached from more than v states, which is impossible as for each state, we may only allow to flip at most v bits, resulting at most v next states. Then, since there are at least $2^v/v$ the next states, it is equivalent to say there are at least $2^v/v$ distinct standard unit vectors, and hence the rank is at least $2^v/v$. Now, as we set $v = (d/2)^{1/r}$, the rank is at least $2^{d/2^{1/r}}/(d/2)^{1/r}$, which is not in polynomial order.

Chapter 4

Linear MDPs are both computational and statistical efficient

In the previous chapter, we reviewed the established computational-statistical gap for the linear Q^*/V^* class. To find the boundary, or say the equivalent conditions for computational- statistical gap, in this section, we shift to another class of MDPs with a different notion of linear structure, presented in [5], that it turns out this condition ensures that both computational and statistical efficiency, i.e. there is no computational-statistical gap in the following class of MDPs.

Definition 4.1 ([5])**.** *We say an MDP M is a* ***linear MDP*** *if there exists a $d \in \mathbb{N}$, a known feature map $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$, an unknown vector-valued measure $\mu : \mathscr{F}_{\mathcal{S}} \to \mathbb{R}^d$, and an unknown fixed $\theta \in \mathbb{R}^d$, s.t. for all $(s, a) \in \mathcal{S} \times \mathcal{A}$,*

$$p(\cdot|s,a) = \langle \phi(s,a), \mu(\cdot) \rangle, \quad r(s,a) = \langle \phi(s,a), \theta \rangle.$$

By scaling, we may, without loss of generality, assume $\|\phi(s,a)\| \leq 1$ for $\forall (s,a) \in \mathcal{S} \times \mathcal{A}$, and $\max\{\|\mu(S)\|, \|\theta\|\} \leq \sqrt{d}$.

We remark that, if we try to use an algorithm to solve an RL problem associated with a linear MDP, we assume, except for the feature ϕ, the algorithm has **NO DIRECT** access to the associated reward function R, and transition P; the reward actually received in each step is still accessible, and takes constant time to receive. That is, we assume for $\forall (s,a) \in \mathcal{S} \times \mathcal{A}$, $\phi(s,a)$ is computable in polynomial time, with respect to the feature dimension d, while the transition and reward are parameterized by known feature map ϕ and unknown parameters μ and θ.

Remark 4.1. *We also remark that linear MDPs are in the linear Q^* Class.*

That is, for any $\pi \in \mathcal{A}^{\mathcal{S}}$, the action value function Q^π and the optimal value function Q^* are also linear in the feature map ϕ.

To see this, we apply the Bellman optimality equation (lemma 2.2), so we have for any $(s,a) \in \mathcal{S} \times \mathcal{A}$,

$$\begin{aligned} Q^*(s,a) &= r(s,a) + \mathbb{E}_{s' \sim p(\cdot|s,a)} V^*(s') \\ &= r(s,a) + \int_S V^*(s') dp(s'|s,a) \\ &= \langle \phi(s,a), \theta \rangle + \int_S V^*(s') \langle \phi(s,a), \mu(ds') \rangle \\ &= \langle \phi(s,a), \theta + \int_S V^*(s') \mu(ds') \rangle. \end{aligned} \tag{4.1}$$

By define $\tilde{\theta}^* := \theta + \int_S V^*(s')\mu(ds')$, we know Q^* is linear in the feature map ϕ. The proof for Q^π is similar.

4.1 The Algorithm and its Time Complexity

From now on, we assume $|\mathcal{A}| < \infty$, and reward R is deterministic so r coincides[*] with R.

In the rest of the section, we review the literature [5], and present both computational and statistical efficiency results for linear MDPs, by first introducing the algorithm and the main result in [5][†] that directly implies both polynomial runtime and polynomial sample complexity.

Before we introduce the algorithm, we first add a significant notion, the **total (expected) regret**.

Definition 4.2. *Suppose we repeat the interactions of the agent and the environment for K episodes and for each episode $k \in \{1, 2, ..., K\}$, the agent applies the policy π^k. We define the the **total (expected) regret** as*

$$\textit{Regret}(K) := \sum_{k=1}^{K} \left[V^*(s_0^k) - V^{\pi_k}(s_0^k) \right],$$

where s_0^k is the starting state in the k-th episode.

Now we present the algorithm **LSVI-UCB**, the **Least-Square Value Iteration with Upper-Confidence Bounds**.

In Algorithm 1, each episode consists of a backward and a forward loop over all steps. In the first loop, (w_h, Λ_h) are updated to build the Q functoin. Note here we start the loop from $h = H$, as we need to build the Q function backward as in the dynamic programming. In the second loop, the agent greedily chooses the action,

$$a_h^k \leftarrow \arg\sup_{a \in \mathcal{A}} Q(s_h^k, a),$$

[*]In [Jin et al., 2019], it is remarked that the following results are readily generalized to stochastic rewards.

[†]In our review, we still assume the time-homogeneity of the MDP for notation simplicity, and the result for a general time-inhomogeneous setting is proved in [5]

Algorithm 1 Least-Square Value Iteration with Upper-Confidence Bounds (LSVI-UCB), [5]

```
1: for episodes $k = 1, ..., K$ do
2:     Initiate the starting state $s_1^k$ by sampling from a given distribution on $\mathcal{S}$.
3:     for step $h = H, ..., 1$ do
4:         $\Lambda_h \leftarrow \lambda \cdot \mathbf{I} + \sum_{i=1}^{k-1} \phi(s_h^i, a_h^i)\phi(s_h^i, a_h^i)^\intercal$.
5:         $w_h \leftarrow \Lambda_h^{-1} \sum_{i=1}^{k-1} \phi(s_h^i, a_h^i)\Big[\sup_{a\in\mathcal{A}} Q(s_{h+1}^i, a) + r(s_h^i, a_h^i)\Big]$.
6:         $Q(\cdot,\cdot) \leftarrow \min\{H, \beta\sqrt{\phi(\cdot,\cdot)^\intercal \Lambda_h^{-1}\phi(\cdot,\cdot)} + w_h^\intercal \phi(\cdot,\cdot)\}$.
7:     end for
8:     for step $h = 1, ..., H$ do
9:         $a_h^k \leftarrow \arg\sup_{a\in\mathcal{A}} Q(s_h^k, a)$.
10:        the agent takes action $a_h^k$, transitions to $s_{h+1}^k$ based on transition $p(\cdot|s_h^k, a_h^k)$.
11:    end for
12: end for
```

to approach the maximum of the Q function built in the first loop. We remark a boundary case that in line 4-5 the summation sums from 1 to 0 when $k = 1$, so we resolve this by assigning w_h with 0 and Λ_h with $\lambda\mathbf{I}$. In this case, there is no actual update happening in line 6.

This Least-Square Value Iteration is inspired by the classical Value Iteration (VI) algorithm. In the classical VI, the Q function is updated following the Bellman optimality equation (Lemma 2.2),

$$Q(s, a) \leftarrow r(s, a) + \mathbb{E}_{s'\sim p(\cdot|s,a)} \sup_{a\in\mathcal{A}} Q(s', a), \quad \forall (s, a) \in \mathcal{S} \times \mathcal{A}.$$

In practice, this update may be hard to implement because it is impossible to iterate all $(s, a) \in \mathcal{S} \times \mathcal{A}$ when $|\mathcal{S}| = \infty$. However, recall in Remark 4.1 we could parameterize $Q^*(s, a) = w^\intercal \phi(s, a)$ for a parameter $w \in \mathbb{R}^d$, together with the known feature ϕ, so it is natural to think the following L^2-regularized least-squares problem (L^2-LS), at the k-th episode and the h-th step:

$$\arg\min_{\tilde{w}\in\mathbb{R}^d} \sum_{i=1}^{k-1} \Big[r(s_h^i, a_h^i) + \sup_{a\in\mathcal{A}} Q(s_{h+1}^i, a) - \tilde{w}^\intercal \phi(s_h^i, a_h^i)\Big]^2 + \lambda \|\tilde{w}\|_2^2,$$

and hence we naturally apply the solution of this L^2-LS in updating the parameter w_h, namely, the line 4-5, where Λ_h is exactly the normal matrix appearing in the solution.

Moreover, to encourage exploration, they added an additional UCB bonus term

$$\beta\sqrt{\phi(\cdot,\cdot)^\intercal\Lambda_h^{-1}\phi(\cdot,\cdot)},$$

where

$$\frac{1}{\phi(\cdot,\cdot)^\intercal\Lambda_h^{-1}\phi(\cdot,\cdot)}$$

is naively the effective numbers of samples, along the ϕ direction, that our agent has observed till step h, and the uncertainty along the ϕ direction is naively represented by the bouns term.

Now we first analyze the time complexity for Algorithm 1, and postpone the correctness of the algorithm till we finish the presentation of the main result, which implies both the correctness and the sample efficiency at the same time.

In the Algorithm 1, Sherman-Morrison formula allows us to compute Λ_h^{-1} in $\mathcal{O}(d^2)$ time, so the time complexity of Algorithm 1 is largely depending on the time complexity in computing $\sup_{a\in\mathcal{A}} Q(s_{h+1}^i, a)$ for all $i \in [k]$. For each step, it takes $\mathcal{O}(d^2|\mathcal{A}|K)$ time, where the K term comes from the summation $k \leq K$, the $|\mathcal{A}|$ term comes from the comparing each action to find $\max Q$, and d^2 term comes from computing $Q(x_{h+1}^i, a)$ as it involves all previous computations, which are of d^2 order, e.g. computing Λ_h and Λ_h^{-1}. That is, in total, the computational complexity is $\mathcal{O}(d^2|\mathcal{A}|KT)$.

4.2 Correctness and Sample Complexity

Now we could present the main result: the $\sqrt{T}$-regret bound of the LSVI-UCB, where $T := KH$ is the count of steps in total.

Theorem 4.2 (Jin et al., 2019[5]). *Given a linear MDP M, there exists an absolute constant $c > 0$ s.t. for any fixed $p \in (0, 1)$, if we set $\lambda := 1$ and $\beta := c \cdot dH\sqrt{\iota}$ in Algorithm 1 with $\iota := \log(2dT/p)$, then with probability $1 - p$, the total regret of LSVI-UCB is at most $\mathcal{O}(\sqrt{d^3H^3T\iota^2})$, where $\mathcal{O}(\cdot)$ hides only absolute constants.*

To see the correctness of Algorithm 1 implied by theorem 1, we follow a similar discussion in section 3.1 [4], that, given

$$\sum_{k=1}^{K}[V^*(s_1) - V^{\pi_k}(s_1)] \le CT^{\frac{1}{2}},$$

with probability $1 - p$, where $C := C'\sqrt{d^3H^3\iota^2}$ and C' is an absolute constant, then, under the condition that the previous inequality holds, by uniformly choosing $\pi = \pi_k$ for $k = 1, 2, ..., K$, we have

$$V^*(s_1) - V^{\pi}(s_1) \le 8CHT^{-\frac{1}{2}} = 8CH^{\frac{1}{2}}\frac{1}{\sqrt{K}},$$

with (conditional) probability at least $7/8$. This is true as, if $V^*(s_1) - V^{\pi}(s_1) > 8CHT^{-\frac{1}{2}}$ for probability greater than $1/8$, then

$$\sum_{k=1}^{K}[V^*(s_1) - V^{\pi_k}(s_1)] > \frac{K}{8}8CHT^{-\frac{1}{2}} = CT^{\frac{1}{2}},$$

and we derive a contradiction. Therefore, given $\varepsilon > 0$, we want to choose K large enough s.t.

$$8CH^{\frac{1}{2}}\frac{1}{\sqrt{K}} < \varepsilon.$$

By plugging in $C = C'\sqrt{d^3H^3\iota^2}$, our K should satisfy

$$8C'\sqrt{d^3H^3}\Big[\log(\frac{2dH}{p}) + \log(K)\Big]\frac{1}{\sqrt{K}} < \varepsilon.$$

Notice $\lim_{K\to\infty}\frac{\log(K)}{\sqrt{K}} = 0$, we may choose K large enough to further satisfy

$$8C'\sqrt{d^3H^3}\frac{\log(K)}{\sqrt{K}} < \frac{\varepsilon}{2},$$

and

$$8C'\sqrt{d^3H^3}\log(\frac{2dH}{p})\frac{1}{\sqrt{K}} < \frac{\varepsilon}{2}. \tag{4.2}$$

The inequality (11) finally allows us to choose $K = \tilde{\mathcal{O}}(d^3H^3/\varepsilon^2)$, where $\tilde{\mathcal{O}}(\cdot)$ hides absolute constants and log polynomial terms, and hence

$$V^*(s_1) - V^\pi(s_1) < \varepsilon,$$

i.e. π is an ε-optimal policy.

That is, by running the Algorithm 1 for $K = \tilde{\mathcal{O}}(d^3H^3/\varepsilon^2)$ episodes and drawing $HK = \tilde{\mathcal{O}}(d^3H^4/\varepsilon^2)$ samples, with probability $\frac{7}{8}(1-p)$, the Algorithm 1 learns an ε-optimal policy satisfies $V^*(s_1) - V^\pi(s_1) < \varepsilon$, and the policy π was chosen uniformly from $\{\pi_1, ..., \pi_K\}$, and each π_k was generated based on the function Q function at the corresponding episode.

Together, it is established that there is no computational-statistical gap for linear MDPs.

4.3 Remarks and Discussions

Remark 4.3. *we first remark that, given a linear MDP M with $|\mathcal{S}| < \infty$, if we fix a deterministic policy π, transition matrix P in the corresponding Markov chain $\{S_t\}$ is low-rank factorizable, i.e. it can be written as a product of matrices with rank $\leq d$.*

To see this, denote $\tilde{\phi}(s) := \phi(s, \pi(s))$.

$$\Phi := \begin{bmatrix} \tilde{\phi}(s_1) & \tilde{\phi}(s_2) & \cdots & \tilde{\phi}(s_{|\mathcal{S}|}) \end{bmatrix}$$

is a $d \times |\mathcal{S}|$ matrix has rank $\leq d$ as $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$.

$$\mathcal{M} := \begin{bmatrix} \mu(s_1) & \mu(s_2) & \cdots & \mu(s_{|\mathcal{S}|}) \end{bmatrix}$$

is also a $d \times |\mathcal{S}|$ matrix has rank $\leq d$ as $\mu : \mathcal{S} \to \mathbb{R}^d$. Then,

$$P = \mathcal{M}^\intercal \Psi.$$

We finish this section by pointing out that the relationship between the computational efficiency of an MDP and the rank of the transition matrix in the corresponding Markov chain is still *unknown*, to the best of our knowledge. We presented examples: Linear MDPs are computationally efficient and low-rank, while the Linear Q^*/V^* MDPs are computationally hard and high-rank.

Chapter 5

In showing the analogous result for Linear Mixture MDPs

In the previous chapters, we presented two classes of MDPs that are both statistically efficient, but they differ in their computational efficiency: Linear Q^*/V^* MDPs[6] are computationally hard while the Linear MDPs[5] are computationally efficient. We remarked on their intrinsic difference in their transition matrices: linear MDPs have a low-rank transition matrix, while linear Q^*/V^* MDPs have a high-rank transition matrix. Therefore, it is natural to consider the relationship between the computational efficiency of an MDP and the rank of the transition matrix in the corresponding Markov chain.

In our way of investigating this relationship, we present another class of MDPs introduced in [8], [1], [17] in the following.

Definition 5.1. *We say an MDP M is a* ***linear mixture MDP*** *if there exists a $d \in \mathbb{N}$, known feature maps $\phi : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$, $\psi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$, and an unknown fixed $\theta \in \mathbb{R}^d$, s.t. for all $(s, a) \in \mathcal{S} \times \mathcal{A}, s' \in \mathcal{S}$,*

$$p(s'|s, a) = \langle \phi(s', s, a), \theta \rangle, \quad r(s, a) = \langle \psi(s, a), \theta \rangle.$$

From [3], linear Mixture MDPs are statistically efficient, and by applying the similar argument in remark 4.3, the corresponding transition matrices in the linear Mixture MDPs also enjoy low-rank factorization. In fact, given fixed $\pi \in \mathcal{A}^{\mathcal{S}}$, by denoting $\Theta \in \mathbb{R}^{d\times|\mathcal{S}|^2}$ and $\Theta := \begin{bmatrix} \theta & \theta & \dots & \theta \end{bmatrix}$ and $\Phi \in \mathbb{R}^{d\times|\mathcal{S}|^2}$ with $\Phi_{ss'} := \phi(s', s, \pi(s))$, we have

$$P = \Theta^{\intercal}\Phi,$$

where $\text{rank}(\Theta) = \text{rank}(\Theta^{\intercal}) = 1$ and $\text{rank}(\Phi) \leq d$. However, it is not clear if the Q^* of a Linear Mixture MDP is linear in a *known* feature map, where by known we still mean there is an algorithm to compute in polynomial time. To see this, observing from the Bellman optimality equation, we have

$$\begin{aligned} Q^*(s,a) &= r(s,a) + \mathbb{E}_{s\sim p(\cdot|s,a)}V^*(s') \\ &= \langle \psi(s,a), \theta\rangle + \int_{\mathcal{S}} V^*(s')p(ds'|s,a) \\ &= \langle \psi(s,a) + \int_{\mathcal{S}} V^*(s')\phi(ds', s, a), \theta\rangle. \end{aligned} \tag{5.1}$$

This linear form seems attractive. However, we typically have no a priori information on the computational time on V^* or Q^* themselves and it is not that natural to assume they are polynomial time computable in the first place. The fact that Q^* is unclear to be a linear function of a known feature map suggests that a fundamentally different approach is required to either prove or disprove the computational efficiency of linear mixture MDPs, as we recall from the previous section that the linear parameterization of Q^* in linear MDPs[5] played a vital role in updating the weights in the Least-Square Value Iteration algorithm.

Our future work continues to investigate whether linear mixture MDPs are computationally efficient, or whether there is a reduction from a linear mixture MDP to a well-known computationally hard problem, which should imply in general linear mixture MDPs are computationally hard. To construct such a reduction, if possible, we will try to consider the computational problem $NASH$, which is to find a Nash equilibrium in a d-player game. This consideration follows the following “logic": the existence of a mixed Nash equilibrium is implied by the Brouwer fixed point theorem, while the action value function is a fixed point acted by the Bellman operator; if there is a polynomial time reduction from the updating step in the Bellman operator to the updating step in the Brouwer fixed point theorem, we are “good".

Bibliography

[1] Alex Ayoub, Zeyu Jia, Csaba Szepesvari, Mengdi Wang, and Lin F Yang. "Model-based reinforcement learning with value-targeted regression". In: *arXiv preprint arXiv:2006.01107* (2020).

[2] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Christopher Hesse, Rafal Jozefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Ponde de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. "Dota 2 with large scale deep reinforcement learning". In: *CoRR* (2019).

[3] 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: *arXiv preprint arXiv:2103.10897* (2021).

[4] Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I Jordan. "Is Q-Learning Provably Efficient?" In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/d3b1fb02964aa64e257f9f26a31f72cf-Paper.pdf.

[5] Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I. Jordan. "Provably Efficient Reinforcement Learning with Linear Function Approximation". In: *arXiv preprint arXiv:1907.05388* (2019).

[6] Daniel Kane, Sihan Liu, Shachar Lovett, and Gaurav Mahajan. "Computational-Statistical Gaps in Reinforcement Learning". In: *arXiv preprint arXiv:2202.05444* (2022).

[7] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, and et al. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (2015), pp. 529–533.

[8] Aditya Modi, Nan Jiang, Ambuj Tewari, and Satinder Singh. “Sample complexity of reinforcement learning using linearly combined model ensembles”. In: *Conference on Artificial Intelligence and Statistics*. 2020.

[9] OpenAI. “GPT-4 Technical Report”. In: *arXiv preprint arXiv:2303.08774* (2023).

[10] Xipeng Qiu. *Neural Networks and Deep Learning*. Publishing House of Electronics Industry, 2019. ISBN: 978-7-111-64968-7.

[11] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. “Mastering the game of Go with deep neural networks and tree search”. In: *Nature* 529.7587 (2016), pp. 484–489. DOI: 10.1038/nature16961.

[12] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. “Mastering the game of Go without human knowledge”. In: *Nature* 550.7676 (2017), pp. 354–359. DOI: 10.1038/nature24270.

[13] Michael Sipser. *Introduction to the Theory of Computation*. 3rd. Cengage Learning, 2012. ISBN: 978-1-133-18779-0.

[14] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 2018. ISBN: 978-0-262-19398-6.

[15] L.G. Valiant and V.V. Vazirani. “NP is as easy as detecting unique solutions”. In: *Theoretical Computer Science* 47 (1986), pp. 85–93. ISSN: 0304-3975. DOI: https://doi.org/10.1016/0304-3975(86)90135-0.

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. “Attention Is All You Need”. In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[17] Dongruo Zhou, Jiafan He, and Quanquan Gu. “Provably efficient reinforcement learning for discounted MDPs with feature mapping”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 12793–12802.

Appendix: Technical Details

Proof of Lemma 2.1. Note

$$\begin{aligned} Q^{\pi}(s,a) &= r(s,a) + \mathbb{E}\Big[\sum_{t=1}^{\tau-1} r(S_t,\pi(S_t))|S_0=s\Big] \\ &= r(s,a) + \mathbb{E}_{s'\sim p(\cdot|s,a)}\mathbb{E}\Big[\sum_{t=1}^{\tau-1} r(S_t,\pi(S_t))|S_1=s',S_0=s\Big] \\ &= r(s,a) + \mathbb{E}_{s'\sim p(\cdot|s,a)}\mathbb{E}\Big[\sum_{t=1}^{\tau-1} r(S_t,\pi(S_t))|S_1=s'\Big] \text{ by Markov property (1)} \\ &= r(s,a) + \mathbb{E}_{s'\sim p(\cdot|s,a)}V^{\pi}(s') \end{aligned} \tag{2}$$

□

Proof of Lemma 2.2.

Note whenever the optimal policy π^* exists, the second equation holds automatically by plugging the π^* into the lemma 2.1.

More generally, when the existence of π^* is not assumed. Fix $(s,a) \in \mathcal{S} \times \mathcal{A}$, observe

$$\begin{aligned} Q^*(s,a) &:= \sup_\pi Q^\pi(s,a) \\ &= \sup_\pi \Big[r(s,a) + \mathbb{E}_{s' \sim p(\cdot|s,a)} V^\pi(s') \Big] \\ &= r(s,a) + \sup_\pi \mathbb{E}_{s' \sim p(\cdot|s,a)} V^\pi(s') \\ &\le r(s,a) + \mathbb{E}_{s' \sim p(\cdot|s,a)} \sup_\pi V^\pi(s') \\ &= r(s,a) + \mathbb{E}_{s' \sim p(\cdot|s,a)} V^*(s'). \end{aligned} \tag{3}$$

To see the other direction, recall that we set a finite time horizon, so we know the $V^*(s') = \sup_\pi V^\pi(s') \le H, \forall s' \in \mathcal{S}$ and then by dominated convergence theorem, for every $\varepsilon > 0$, $\exists \pi \in \mathcal{A}^{\mathcal{S}}$, s.t.

$$\mathbb{E}_{s' \sim p(\cdot|s,a)} \Big[V^*(s') - V^\pi(s') \Big] = \mathbb{E}_{s' \sim p(\cdot|s,a)} \Big| V^*(s') - V^\pi(s') \Big| < \varepsilon$$

Hence, we have

$$\begin{aligned} Q^*(s,a) &\ge Q^\pi(s,a) \\ &= r(s,a) + \mathbb{E}_{s' \sim p(\cdot|s,a)} V^\pi(s') \\ &> r(s,a) + \mathbb{E}_{s' \sim p(\cdot|s,a)} V^*(s') - \varepsilon, \end{aligned} \tag{4}$$

that is, we established the second equation.

To show $V^*(s) = \sup_{a \in \mathcal{A}} Q^*(s,a)$, note for any deterministic policy $\pi : \mathcal{S} \to \mathcal{A}$,

$$V^\pi(s) = Q^\pi(s, \pi(s)),$$

and by $V^*(s) := \sup_\pi V^\pi(s)$, we left to show

$$\sup_\pi Q^\pi(s, \pi(s)) = \sup_{a \in \mathcal{A}} Q^*(s,a).$$

To show left inequality, notice

$$Q^\pi(s, \pi(s)) \leq Q^*(s, \pi(s)) \leq \sup_{a \in \mathcal{A}} Q^*(s,a), \tag{5}$$

and hence it is given by taking sup over π in both sides of the inequality.

To show the other way around, if $V^*(s) < \sup_{a \in \mathcal{A}} Q^*(s,a)$ for some $s \in \mathcal{S}$, then there $\exists a' \in \mathcal{A}$ s.t.

$$Q^*(s, a') > V^*(s),$$

and moreover there $\exists \pi \in \mathcal{A}^{\mathcal{S}}$ s.t.

$$Q^\pi(s, a') > V^*(s).$$

Therefore, if we choose a modified policy $\bar{\pi} : \mathcal{S} \to \mathcal{A}$ s.t. for all $x \in \mathcal{S}$,

- $\bar{\pi}(s) = a'$;
- $Q^\pi(x, \bar{\pi}(x)) \geq Q^\pi(x, \pi(x)) = V^\pi(x)$.

We left to show $V^{\bar{\pi}}(s) \geq Q^{\pi}(s, a')$. But this is given by the policy improvement theorem [14], together with the bellman equation (lemma 2.1). More precisely, we have

$$\begin{aligned} V^{\bar{\pi}}(s) - Q^{\pi}(s, a') &= Q^{\bar{\pi}}(s, \bar{\pi}(s)) - Q^{\pi}(s, a') \\ &= \mathbb{E}_{s' \sim p(\cdot|s,a')}[V^{\bar{\pi}}(s') - V^{\pi}(s')] \\ &\geq 0, \end{aligned} \tag{6}$$

as $V^{\bar{\pi}}(s') \geq V^{\pi}(s'), \forall s' \in \mathcal{S}$ by the policy improvement theorem. □

Proof of Claim 3.1 i).

This claim is showed in two steps.

1. We construct a policy π such that $V^{\pi}(s) = g(l, w)$;

2. Given any other policy $\pi' \in \mathcal{A}^{\mathcal{S}}$, we show $V^{\pi'}(s) \leq g(l, w)$.

In step 1), we construct the policy π as the following: for every non-satisfying state s, i.e. $w \neq w^*$, $\pi(s) := a$ is such an action that the hamming distance dist(w, w^*) from the current assignment w to the satisfying assignment w^* is decreased by 1. Note that we can always find such action, because by definition, all clauses are satisfied by the satisfying assignment.

Therefore, from a state $s = (l, w)$, together with transition P and the policy π, $s' := P(s, \pi(s)) = (l + 1, w_1)$ satisfying either

1. $w_1 = w^*$; or

2. w_1 is on the optimal path (in a level later than w) i.e. dist$(w, w^*) =$ dist$(w, w_1)+$dist(w_1, w^*).

In both cases,

$$V^{\pi}(s) = \Big(1 - \frac{\text{dist}(w, w_1) + l + \text{dist}(w_1, w^*)}{H + v}\Big)^r = g(l, w).$$

In step 2). For any other policy π' that leads s to end on state $s' = (l', w')$ (that is, either it is on the last layer $l' = H$, or it reaches the satisfying assignment $w' = w^*$, we have

$$\begin{aligned} V^{\pi'}(s) &= \Big(1 - \frac{\text{dist}(w', w^*) + l'}{H + v}\Big)^r \\ &\leq \Big(1 - \frac{\text{dist}(w, w') + l + \text{dist}(w', w^*)}{H + v}\Big)^r \\ &\leq g(l, w), \end{aligned} \tag{7}$$

where $l' - l \geq \text{dist}(w, w')$ contributes to the first $\leq$ and the triangle inequality

$$\text{dist}(w, w^*) \leq \text{dist}(w, w') + \text{dist}(w', w^*)$$

contributes to the second $\leq$. $\square$

Proof of Claim 3.1 ii).

This claim is showed in two steps.

1. We can write $V^*(s)$ as a polynomial in w and w^*, with degree at most r.

2. We construct the ψ and $\tilde{\psi}$ directly from the polynomial in the previous step.

In the step 1, we observe that

$$\text{dist}(w, w^*) = \frac{v - \langle w, w^* \rangle}{2},$$

and $g(l, w)$ can automatically be written as a polynomial in $\text{dist}(w, w^*)$, of degree r.

For the second, we construct vector θ as the following:

- Let $\mathscr{S}$ be the collection of all multiset $S \subset [v]$ and $|S| \leq r$;

- for each $S \in \mathscr{S}$, let $\theta_S := \prod_{i \in S} w_i^*$;

- Let $\theta := (\theta_S)_{S \in \mathscr{S}}$.

That is, each coordinate θ_S is a monomial in the coordinate of w^*, e.g. $w_i^* w_j^* w_k^*$ where w_i^*, w_j^*, w_k^* is i-th, j-th, and k-th coordinate of w^*, respectively.

Then, note that

$$g(l, w) = \Big(1 - \frac{\text{dist}(w, w^*) + l}{H + v}\Big)^r$$

$$= \Big(1 - \frac{\frac{v - \langle w, w^* \rangle}{2} + l}{H + v}\Big)^r$$

and $\langle w, w^* \rangle = \sum_1^v w_i w_i^*$. We set $\psi(s)$ to be the corresponding coefficient vector (each coordinate corresponds to a θ_S) and hence $V^*(s) = \langle \theta, \psi(s) \rangle$ follows. Note there are at most $\sum_{i=0}^r v^i \leq 2v^r$

(when v large enough) many coefficients (for multiset $S \subset [v]$ with $|S| = i$, there at most v^i possible choice), we may set the feature dimension as $d = 2v^r$.

Lastly, $\tilde{\psi}(s, a) := \psi(P(s, a))$ as P is deterministic and the linear representation of Q^* follows by $Q^*(s, a) = V^*(P(s, a))$. $\square$

Proof of Claim 3.2.

$\mathcal{A}_{SAT}$ will always return **NO** on unsatisfiable formula is clear. We are left to consider those satisfiable formulas.

Let φ be a satisfiable formula. In the MDP oracle M_φ, we remark the following observation: assuming large enough v and r, the rewards collected in the last layer and the optimal value function V^* are separated.

More specifically,

- the reward is comparatively small in the last layer:

$$\mathbb{E}[R] = \Big(1 - \frac{\text{dist}(w, w^*) + H}{H + v}\Big)^r \leq (\frac{v}{H+v})^r \leq v^{r-r^2} < \frac{1}{4},$$

- V^* is comparatively large:

$$V^* = \Big(1 - \frac{l}{H+v}\Big)^r \geq \Big(1 - \frac{v}{H+v}\Big)^r = \Big(1 + \frac{v}{v^r}\Big)^{-r} \geq 1 + (-r)\frac{v}{v^r} \geq \frac{1}{2},$$

  where $l \leq v$ contributes to the first inequality(action choosing follows from the proof of claim 3.1 and this $l \leq v$ follows), the second from Bernoulli's inequality and the third is some elementary calculus bounding.

Therefore, if $\mathcal{A}_{RL}$ returns a policy π s.t. it produces large value function V^π, i.e.

$$\sup_{s \in \mathcal{S}} |V^* - V^\pi| < \frac{1}{4},$$

then the policy π has to lead to a satisfying state and $\mathcal{A}_{SAT}$ will return **YES** by its construction. $\square$

Proof of Claim 3.3. Let

- - $\mathbb{P}_{M_\varphi}$ be the joint probability measure on the generated policies and observed rewards when $\mathcal{A}_{RL}$ has access to M_φ.
  - $\mathbb{P}_{\overline{M}_\varphi}$ be the joint probability measure when $\mathcal{A}_{RL}$ has access access to $\overline{M}_\varphi$ the simulator.
- R_i be the reward collected along i-th trajectory. recall reward is only granted on the last layer or the satisfying state.
- After finishing running $\mathcal{A}_{RL}$ with access to M_φ, we denote T as the count of all trajectories within the episode H.

As in our assumption, $\mathcal{A}_{RL}$ runs $v^{r^2/4}$ amount of steps and hence

$$T \leq v^{r^2/4}.$$

As in the proof of Claim 3.2, assuming large enough r and v, the expected reward in the MDP M_φ is comparatively small on the last layer

$$\mathbb{E}[R] \leq v^{-\frac{r^2}{2}},$$

and states on the last layer are visited by $\mathcal{A}_{RL}$ at most $v^{r^2/4}$ times, it is with **high probability** that the rewards at the last level are all zeros.

More precisely, when large enough r, v is still assumed,

$$\begin{aligned}\mathbb{P}_{M_\varphi}(\{R_i = 0, \forall i \in [T]\}) &= 1 - \mathbb{P}_{M_\varphi}(\{R_i \neq 0, \exists i \in [T]\}) \\ &\geq 1 - v^{-\frac{r^2}{4}} \\ &\geq \frac{4}{5},\end{aligned} \tag{8}$$

where the first inequality in (7) follows from

$$\begin{aligned}\mathbb{P}_{M_\varphi}(\{R_i \neq 0, \exists i \in [T]\}) &= \mathbb{P}_{M_\varphi}(\cup_{i\in[T]}\{R_i \neq 0\}) \\ &\leq \sum_{i\in[T]} \mathbb{P}_{M_\varphi}(\{R_i \neq 0\}) \\ &\leq T \cdot v^{-\frac{r^2}{2}} \\ &\leq v^{\frac{r^2}{4}} \cdot v^{-\frac{r^2}{2}} \\ &= v^{-\frac{r^2}{4}}\end{aligned} \tag{9}$$

and

$$\mathbb{P}_{M_\varphi}(\{R_i \neq 0\}) = \mathbb{E}_{M_\varphi}[R_i] \leq v^{-\frac{r^2}{2}}$$

by proof of Claim 2 and the nature of the Bernoulli distribution, and the second inequality in (7) is a general bounding assuming v is large enough.

Let

- $\mathbb{S}_{\mathcal{A}_{RL}, M_\varphi}$ (or $\mathbb{S}_{\mathcal{A}_{RL}, \overline{M}_\varphi}$) be the event that $\mathcal{A}_{RL}$, with access to M_φ (or $\overline{M}_\varphi$), after running at most $v^{r^2/4}$ steps, returns a policy π that satisfies $\sup_{s\in\mathcal{S}} |V^* - V^\pi| < \frac{1}{4}$. ($\mathbb{S}$ is short for succeeds)

- $\mathbb{V}_{[T]}$ be the event that the reward collected in the last layer is zero for all trajectories in $[T]$ ($\mathbb{V}$ is short for void).

It is assumed that

$$\mathbb{P}_{M_\varphi}(\mathbb{S}_{\mathcal{A}_{RL},M_\varphi}) = \frac{9}{10},$$

and from previous reasoning we know

$$\mathbb{P}_{M_\varphi}(\mathbb{V}_{[T]}) \geq \frac{4}{5}.$$

Therefore, together we have

$$\begin{aligned}
&\mathbb{P}_{M_\varphi}(\mathbb{S}_{\mathcal{A}_{RL},M_\varphi}|\mathbb{V}_{[T]}) \\
=&\frac{\mathbb{P}_{M_\varphi}(\mathbb{S}_{\mathcal{A}_{RL},M_\varphi} \cap \mathbb{V}_{[T]})}{\mathbb{P}_{M_\varphi}(\mathbb{V}_{[T]})} \\
\geq&\frac{\frac{9}{10}-\frac{1}{5}}{\frac{4}{5}} = \frac{7}{8}
\end{aligned} \tag{10}$$

where the inequality by

$$\mathbb{P}(A \cap B) = 1 - \mathbb{P}(A^c \cup B^c) \geq 1 - [\mathbb{P}(A^c) + \mathbb{P}(B^c)].$$

We remark that since the only difference between M_φ and simulator $\overline{M}_\varphi$ is on the last layer, and hence the marginal distribution $\mathbb{P}_{M_\varphi}$ and $\mathbb{P}_{\overline{M}_\varphi}$ on policy π, conditioned on $\mathbb{V}_{[T]}$, coincide.

That is, we have

$$\mathbb{P}_{\overline{M}_\varphi}(\mathbb{S}_{\mathcal{A}_{RL},\overline{M}_\varphi}|\mathbb{V}_{[T]}) = \mathbb{P}_{M_\varphi}(\mathbb{S}_{\mathcal{A}_{RL},M_\varphi}|\mathbb{V}_{[T]})$$

Since, $\mathbb{P}_{\overline{M}_\varphi}(\mathbb{V}_{[T]}) = 1$ by construction of the simulator, we conclude that

$$\mathbb{P}_{\overline{M}_\varphi}(\mathbb{S}_{\mathcal{A}_{RL},\overline{M}_\varphi}) \geq \frac{7}{8},$$

i.e. with probability at least $7/8$, $\mathcal{A}_{RL}$, with access to $\overline{M}_\varphi$ the simulator, after running at most $v^{r^2/4}$ steps, returns a policy π that satisfies $\sup_{s\in\mathcal{S}} |V^* - V^\pi| < \frac{1}{4}$. $\square$