A SURVEY ON THE COMPUTATIONAL HARDNESS OF LINEAR-STRUCTURED MARKOV DECISION PROCESSES

CHUHUAN HUANG

Contents

Date: March, 2023.

1. Introduction

A problem occurred to me when I was playing a role-playing computer game in my teen age. To strengthen a piece of equipment for my role successfully, which will typically triple its original market price, I have to transition from a starting numeric 50 to end numeric 110 in exactly 7 steps. In each steps, I could choose 3 different types of tool to elevate the numeric: tool "6" could randomly increment the numeric by 6, 7, or 8; tool "7" could randomly increment the numeric by 7, 8, or 9; tool "8" could randomly increment the numeric by 8, 9, or 10. Moreover, it was set as the game rules that for a successful strengthening, the last step should be a transition exactly from numeric 100 to numeric 110, and if one reaches a numeric more than 100 but other than 110, no more tools can be employed and this round of strengthening is a fail, requiring a re-do if one still insists to do this strengthening. Further, tools in the same type behaves in the same way independently, namely following the same unknown distribution, and the current effect of the chosen tool is independent of the effects of the previous chosen tools. However, tools in different type does not necessary to follow the same distribution, and different types of tools have different costs: tool "6" takes 1 buck, tool "7" takes 5 bucks, tool "8" takes 10 bucks.

How should I choose the tool in each step to maximize my "profit"? I had no elegant idea in approaching nor patience in plain calculations until later I encountered the machine learning.

Since the 1956 workshop in Dartmouth, Machine Learning (ML), as a major field in Artificial Intelligence (AI), initially aiming in training the machine to predict a general pattern (the model) from limited observed data (machine is "learning" throughout the training process), has been growing rapidly: the astonishing game of Go series: AlphaGo versus Lee Sedol in 2016, together with land-changing literature [D. Silver et al., 2016], the invention of the Transformer [Vaswani, et al., 2017], and, the game-changing storm by GPT-4 [openAI, 2023], Artificial Inteligience and Machine Learning and their applications have been a world-wide influential phenomenon in every aspect of social and economic life.

Throughout the history, different genres of machine learning paradigms emerged. Less restrictively speaking, they are: 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 together as a set is called the training set). Upon finishing a successful training process, the machine should be able to predict the label for an 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.

Comparatively, unsupervised learning is a type of machine learning paradigm where the model is trained by a 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.

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

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

Reinforcement learning (RL) is a type of machine learning paradigm that the model (typically we use the word agent) is trained through a sequence of interactions with the surrounding environment: in each step, the agent observe the current environment, make actions, gets numerical feedback, called reward, directly from the environment, instead of manual label, and the environment changes reacting to agent's action. The typical goal of the 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 collected rewards from the environment after he finishes interactions. The problem occured to me in my teen age is a natural example approachable by reinforcement learning. The environment is the game, the agent is me, and my available actions correspond to choosing different tools.

Notice if the distribution of action's effect from a given numeric (called transition probability) is known, a typical approach to this problem is the dynamic programming, or Value Iteration, which takes $\mathcal{O}(|\mathcal{S}|^2|\mathcal{A}|)$ time for each iteration, where $\mathcal{S}$ is set of all possible numeric (called state space), $\mathcal{A}$ is set of 3 actions {use tool"6", use tool "7", use tool "8"}. Unfortunately, this transition probability is unknown. One possible way is to apply Monte Carlo method, and when the number of trials is large enough, by law of large numbers, the estimated transition probability and its true value only differ by a small error, and then we could apply the dynamic programming way by using the estimated transition probability.

However, not only this transition probability is unknown, but we know for sure each trial of sampling is expensive (large negative reward in each choice unless the numeric 110 is reached). To overcome this dillemma between the accuracy and the cost, a sensible approach is SARSA [Rummery et al., 1994].

In SARSA, instead of enumerating transition probability for all numeric-action pair, the model estimates the collected reward $Q(s, a)$, given initial action a and initial numeric s directly, and moreover it update the estimation by the Bellman equation, current numeric s and choosed action a, the reward r collected in this step, and next numeric s' after the action a and next action a'; The action choosing policy chooses the action such that it maximize the estimation. Under this implementation, the time complexity decreases, but it is still exponential in $|\mathcal{S}|$ and $|\mathcal{A}|$, which makes the larger scale implementation impractical.

The curse of dimensionality in the reinforcement learning refers to the fact that the computational and sample complexity 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 (collected reward) and learn an good policy.

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 the during training, typically via deep neural networks (e.g. Deep Q Network, DQN, is modified Q-Learning where action value function Q is parameterized by a deep neural network; the 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 a immediate reward)[Mnih et al.,2015].

Even though the function approximation technique is applied, so that the state and action space could be increased to large-scale, the computational requirements are still intimidating. For example, the world-famous AlphaZero [Silver et al., 2018] 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 [Berner et al., 2019].

While function approximation greatly elevates the potential of RL by enlarging the effective states and action spaces, it raises theoretical challenges: the neighborhood of most of the states remains unvisited during training episodes, impairing the reliability of the estimates of the value functions [Sutton, Barto, 2018]. To deal with this credibility challenge, a typical approach is to assume some linearity in the structure of the dynamics, hoping to facilitate the theoretical proofs. In [Yang and Wang, 2019], by assuming the linearity of the transition probability and reward function with respect to some known feature maps, they proved there exists a provably reinforcement learning algorithm that is both statistical and computational efficicent, 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 provably RL algorithm that is both statistical and computational efficient?*

Driven by the natural question above, statistical efficiency, or say sample efficiency, has been growing as a main body in the reinforcement learning theory community. In [Du et al., 2021], by assuming the linearity of the optimal value function and action value function with respect to some known feature, the sample efficiency is possible. However, recently, in [D.Kane et al., 2022], by following the same assumption: the linear optimal value function and action value function with respect to some known feature, they firstly proposed a gap between the statistical efficiency and computational efficiency. That is, under their assumed linearity, the sample efficiency is possible [Du et al., 2021], 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.

In our survey, we will review some necessary background preliminaries in section 2, then we will review the computational hardness for Linear V^*/Q^* MDPs in [D.Kane et al., 2022] in section 3; in section 4, we will go over the main idea and the structure in [Yang and Wang, 2019] for Linear MDPs, showing the existence of mathematically provable RL algorithm that is both computational and statistical efficient. Further, in section 5, we will discuss some analogous work could be done for the Linear Mixture MDPs, introduced in [Modi et al., 2020, Ayoub et al., 2020, Zhou et al., 2021], which is of our main focus in the future work.

2. Preliminaries

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

2.1. **Reinforcement learning and Markov decision processes.**

Reinforcement Learning (RL) is a type of learning where an agent interacts with the surroundings, gets feedback and learns to adjust its strategy to survive and thrive in the environment. The framework of the reinforcement learning is the Markov decision process.

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.*
- *$\mathcal{S}$, the* ***State Space****, is the non-empty set of all possible states, 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 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 $(s', s, s_{t-1}, ..., s_0) \in \mathcal{S}^{t+2}$ and $(a, a_{t-1}, ..., a_0) \in \mathcal{A}^{t+1}$*

$$\begin{aligned}&\mathbb{P}(S_{t+1} = s' | S_t = s, A_t = a, S_{t-1} = s_{t-1}, A_{t-1} = a_{t-1}, ..., S_0 = s_0, A_0 = a_0)\\&= \mathbb{P}(S_{t+1} = s' | S_t = s, A_t = a).\end{aligned} \tag{1}$$

  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, the interaction bewteen the agent and the environment can be illustrated 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 state s_t, takes action a_t, and receives 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 the next state s_{t+1}, following the transition probability $p(s_{t+1}|s_t,a_t)$. 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 interactions stop. 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 a 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 interchangably use the phrases "solving a 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 $\tau := \inf\{t \in \mathbb{Z}^+ : S_t = s_\perp\} \wedge H$ is the first time that the process M stops, a stopping time, and $S_1, A_1, ..., S_{\tau-1}, A_{\tau-1}$ 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** for the value function V and action-value function Q.

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

Proof of Lemma 2.1. Note

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

$\square$

We conclude this subsection 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. Simlilarly to the Bellman equation, we have the **Bellman optimality equation**:

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 $Q^{\pi^*} = Q^*$.

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

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

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 $V^*(s) \coloneqq \sup_\pi V^\pi(s)$, 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 side of the inequlity.

To show the other way around, if $V^*(s) < \sup_{a\in\mathcal{A}} Q^*(s,a)$, 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 define $\tilde{\pi} : \mathcal{S} \to \mathcal{A}$ s.t.

$$\tilde{\pi}(s') \coloneqq \begin{cases} a' & \text{if } s' = s \\ \pi(s') & \text{otherwise} \end{cases},$$

we know

$$V^{\tilde{\pi}}(s) = Q^{\tilde{\pi}}(s,\tilde{\pi}(s)) = Q^{\tilde{\pi}}(s,a') \geq Q^\pi(s,a') > V^*(s),$$

where the first inequality follows from that whenever s is reached, action a' will increase the expected rewards. Thus, we derive our desired contradiction. $\square$

2.2. **Complexity theory terminologies and computationally hard problems.**

Before we introduce a problem that is well-known to be computionally hard, we first recall some complexity concepts from theoretical computer science.

A decision problem is a computational problem 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 an 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 RP class is a subset of NP class, i.e. every decision problem in RP class is in 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-requiste terminologies.

A **Boolean expression**, or **propositional logic formula**, is a sequence of Boolean variables, operators **AND** (conjunction, denoted as $\wedge$)[1], **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.

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 a SAT problem where the input 3-CNF formula is ensured to have either 0 or 1 satisfying assignment.

Also, we may frequently make statements like that a randomized algorithm A solves a problem in time t with error probability p, by which we means that A runs in time $\mathcal{O}(t)$[2] and solve the problem with probability at least $1-p$. More specifically, for a SAT problem, it returns **YES** on a satisfiable formula with probability at least $1-p$ and it returns **NO** on an unsatisfiable formula with probability 1. For a reinforcement learning problem, it returns a ε optimal policy π with probability at least $1-p$, 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 Unique-3-SAT problem is NP-complete by Cook-Levin Theorem; further, [Valiant-Vazirani, 1985] states a stronger result:

Theorem 2.3 (Valiant-Vazirani, 1985). *Unless $NP = RP$, no randomized algorithm can solve UNIQUE-3-SAT with v variables in $poly(v)$ time with error probability $\frac{1}{8}$.*

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

In this section we review a recent literature in the reinforcement learning theory community that firstly reveals a sharp computational-statistical gap in reinforcement learning.

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

[1]Please distinguish the notation $\wedge$ for 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.

[2]Let f, g be real-valued function defined on an unbounded subset of the positive real numbers, then we say $f(x) = \mathcal{O}(g(x))$ if there exists $M \in R^+, x_0 \in R^+$, s.t. $|f(x)| \leq Mg(x)$ whenever $x \geq x_0$.

Definition 3.1. *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*[3] *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****.*

We remark that, if we try to use an algorithm to solve the the RL problem associated to a linear Q^*/V^* MDP, we assume the algorithm has access to the associated reward function R, transition p, and the features ψ and $\tilde{\psi}$. We may assume the call of each function above accrues constant runtime and input/output for these functions takes polynomial time in the feature dimension d. In this sense, transition p and reward R are known.

Now, with the notion above in mind, we may specify the computational-statistical gap in reinforcement learning: MDPs in linear Q^*/V^* class, regardless of the number of actions, are statistically efficient to solve [Du et al. 2021], meaning it merely requires $poly(d)$ samples; yet they are computionally hard to solve, i.e. unless $NP = RP$, there is no randomized algorithm can be used to solve them in time polynomial in d.

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.

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 a MDP satifying the properties stated above and the goal G is to find a policy π s.t. $V^\pi > V^* - \frac{1}{4}$, or say this LINEAR-3-RL is solved by finding a such policy.

3.1. **The Sharp Result.**

We now present the major computational hardness result of reinforcement learning in [D.Kane et al., 2022].

[3]By known, we means 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.

Theorem 3.1 (D.Kane et al., 2022). *Unless $NP = RP$, no randomized algorithm can solve LINEAR-3-RL with feature dimension d in polynomial time in d with error probability $\frac{1}{10}$.*

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. Suppose $q \geq 1$. If LINEAR-3-RL with feature dimension d can be solved in time d^q with error probability $\frac{1}{10}$, then UNIQUE-3-SAT with v variables can be solved in time $poly(v)$ with error probability $\frac{1}{8}$.

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 subsubsection, we sketch the proof of the reduction and leave the technical details in the appendix for reference.

We divide the proof into several steps. In the first step, we construct a MDP M_φ, based on the given 3-CNF formula φ, which happens to satisfy the linear Q^*/V^* condition. In the second step, we associate an algorithm solving the M_φ with an algorithm solving the UNIQUE-3-SAT, such that by solving the M_φ, i.e. the RL algorithm returns a good policy, implies solving the UNIQUE-3-SAT problem, finding the satisfying assignment.

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

- each state $s = (l, w)$, where $w \in \{-1, 1\}^v$ is the associated assignment and $l \in \mathbb{N}$ is the depth of the state. 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 state s with a non-satisfying assignment 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 filpped, and the depth level $l' = l + 1$.
  - for state s with a non-satisfying assignment w and $l = H$ or s with the satisfying assignment w^*, every $j \in \{1, 2, 3\}$ takes s to $s_\perp$.
- the reward is everywhere 0 except on states with satisfying assignment or states on the last level 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,

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

and r is a parameter specified later.

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. For any state s in level l with assignment w and action a,

i) the optimal value function $V^*(s) = g(l, w)$.

ii) for large enough v, there exists features $\psi(s)$, $\tilde{\psi}(s,a) \in \mathbb{R}^d$ with feature dimension $d \leq 2v^r$ depending only on state s and action a; and $\theta \in \mathbb{R}^d$ depending only on w^* such that V^* and Q^* can be written as a linear function of features ψ and $\tilde{\psi}$ i.e. $V^*(s) = \langle \theta, \psi(s) \rangle$, $Q^*(s,a) = \langle \theta, \tilde{\psi}(s,a) \rangle$.

Proof of Claim 3.1 i). We prove i) of this claim by 1) showing there exists a policy π that achieves this value function and 2) by showing other policies are inferior in terms of the value function.

For 1), let π be a policy s.t. for every state s with non-satisfying assignment $w \neq w^*$, π chooses the action that decreases the hamming distance $\operatorname{dist}(w, w^*)$ by 1. Note that, at each state, such action always exists, as in our construction, all clauses are satisfied by the satisfying assignment. Therefore, from a state s at level l with assignment w, we can reach a state with assignment w_1 s.t.

i) $w_1 = w^*$

or

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

In both cases,

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

We now show 2). For any other policy π' that ends on state s' at level l' with assignment w' (i.e. either $l' = H$ or $w' = w^*$), we have

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

where the first inequality by $l' - l \geq \operatorname{dist}(w, w')$ and the second by simply the triangle inequality $\operatorname{dist}(w, w') + \operatorname{dist}(w', w^*) \geq \operatorname{dist}(w, w^*)$. ☐

Proof of Claim 3.1 ii). For ii), we firstly show $V^*(s)$ can be written as a polynomial of degree at most r in w and w^* and then show it is sufficient for V^* and Q^* to be linear function of features ψ and $\tilde{\psi}$.

The first part follows from the observation that

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

and $g(l, w)$ is a polynomial of degree r in $\mathrm{dist}(w, w^*)$.

For the second, we set vector θ to be all monomials in w^* of degree at most r (more precisely in w_i^*,w_j^* coordinates of w^*), where each coordinate of θ corresponds to a multiset $S \subseteq [v]$ of size $|S| \leq r$ and $\theta_S = \prod_{i \in S} w_i^*$, i.e. $\theta = (\theta_S)_{S \in \mathscr{S}}$, where $\mathscr{S}$ is the collection (allowing repetition) of all such multiset S. Then, note that

$$\begin{aligned} g(l, w) &= \Big(1 - \frac{l + \mathrm{dist}(w, w^*)}{H + v}\Big)^r \\ &= \Big(1 - \frac{l + \frac{v - \langle w, w^* \rangle}{2}}{H + v}\Big)^r \end{aligned}$$

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 $S \subseteq [v]$ 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 $Q^*(s, a) = V^*(P(s, a))$. □

Now we construct a randomized algorithm $\mathcal{A}_{SAT}$ for UNIQUE-3-SAT problem using a randomized algorithm $\mathcal{A}_{RL}$ for the RL problem. Note that runtime of the $\mathcal{A}_{RL}$ accumulates each call through accessing the transition kernel, the features, and may assume they require constant time to finish computing. However, in collecting the reward, it 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**[4] 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, we can execute each call to the simulator $\bar{M}_\varphi$ in time *poly*(d).

We construct the algorithm for UNIQUE-3-SAT as the following:

On input 3-CNF formula φ, $\mathcal{A}_{SAT}$ runs the algorithm $\mathcal{A}_{RL}$ replacing each call to MDP oracle M_φ with the corresponding call to simulator $\bar{M}_\varphi$. If the sequence of actions, i.e. the policy, by $\mathcal{A}_{RL}$ ends on a state with assignment w, $\mathcal{A}_{SAT}$ outputs **YES** if w is the satisfying assignment and returns **NO** otherwise.

To see its correctness, we claim the following:

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

Claim 3.2. Suppose $r > 1$ and horizon $H = v^r$. If $\mathcal{A}_{RL}$ outputs a policy π such that $V^\pi > V^* - \frac{1}{4}$, then $\mathcal{A}_{SAT}$ on 3-CNF formula φ outputs **YES** if φ is satisfiable and **NO** otherwise.

Proof of Claim 3.2. $\mathcal{A}_{SAT}$ will always return **NO** on unsatisfiable formula is clear. We left to consider those satisfiable formulas.

Let φ be a satisfiable formula. In the MDP oracle M_φ, notice the following two observation:

i) the reward are "small" everywhere except on reaching the satisfying assignment: the expected reward at the last layer is upper bounded (assuming large enough v and r)

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

ii) the optimal V^* is large (considering large enough v and r):

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

where the first inequality follows from $l \leq v$ (action choosing follows from 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 the value function V^π is large, i.e. $V^\pi > V^* - \frac{1}{4}$, then the policy π has to end on a state with the satisfying assignment w^* and $\mathcal{A}_{SAT}$ will return **YES** by its construction. □

Claim 3.3. Suppose $r \geq 2$ and horizon $H = v^r$. Suppose $\mathcal{A}_{RL}$ with access to MDP oracle M_φ runs in time $v^{\frac{r^2}{4}}$ (total number of time steps) and outputs a policy π such that $V^\pi > V^* - \frac{1}{4}$ with error probability $\frac{1}{10}$. Then, $\mathcal{A}_{RL}$ with access to simulator $\bar{M}_\varphi$, still runs in time $v^{\frac{r^2}{4}}$, outputs a policy π such that $V^\pi > V^* - \frac{1}{4}$ with error probability $\frac{1}{8}$.

Proof of Claim 3.3. Let

- $\mathbb{P}_{M_\varphi}$ and $\mathbb{P}_{\overline{M}_\varphi}$ denote the distribution on the *observed* rewards and output policies induced by the $\mathcal{A}_{RL}$ when running on access to the oracle M_φ and the simulator $\overline{M}_\varphi$ respectively.
- R_i denote the reward received on the last layer (and last layer only) at the end of the i-th trajectory.
- T be the total number of trajectories sampled by algorithm $\mathcal{A}_{RL}$ when running on access to MDP oracle M_φ.

By the assumption, $\mathcal{A}_{RL}$ runs in time $v^{r^2/4}$ (total number of steps) and therefore $T \leq v^{r^2/4}$. Since the expected reward at the **last layer** in the MDP M_φ is upper bounded by (as in the proof of Claim 3.2, assumming large enough v and r)

$$\Big(1-\frac{H}{H+v}\Big)^r \le v^{-r^2+r} \le v^{-\frac{r^2}{2}}$$

and the algorithm only visits states at most $v^{r^2/4}$ times on the last layer, we get by the union bound that with **high probability** all the rewards at the last level are all zeros. More precisely (assuming v is large enough),

$$\begin{aligned}\mathbb{P}_{M_\varphi}(\{R_i=0,\forall i\in[T]\}) &= 1-\mathbb{P}_{M_\varphi}(\{R_i\neq 0,\exists i\in[T]\}) \\ &\ge 1-v^{-\frac{r^2}{4}} \\ &\ge \frac{4}{5},\end{aligned} \tag{6}$$

where the first inequality in (7) follows from

$$\begin{aligned}\mathbb{P}_{M_\varphi}(\{R_i\neq 0,\exists i\in[T]\}) &= \mathbb{P}_{M_\varphi}(\cup_{i\in[T]}\{R_i\neq 0\}) \\ &\le \sum_{i\in[T]}\mathbb{P}_{M_\varphi}(\{R_i\neq 0\}) \\ &\le T\cdot v^{-\frac{r^2}{2}} \\ &\le v^{\frac{r^2}{4}}\cdot v^{-\frac{r^2}{2}} \\ &= v^{-\frac{r^2}{4}}\end{aligned} \tag{7}$$

and

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

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.

We say $\mathcal{A}_{RL}$ succeeds with access to M_φ (or $\overline{M}_\varphi$) if the output policy π after running for time at most $v^{r^2/4}$ satisfies $V^\pi > V^* - 1/4$. The above reasoning and the assumption that $\mathcal{A}_{RL}$ succeeds with access to MDP oracle M_φ with probability $\frac{9}{10}$ implies

$$\begin{aligned}&\mathbb{P}_{M_\varphi}(\mathcal{A}_{RL}\text{ succeeds with access to }M_\varphi|R_i=0,\forall i\in[T]) \\ =&\frac{\mathbb{P}_{M_\varphi}(\{\mathcal{A}_{RL}\text{ succeeds with access to }M_\varphi)\}\cap\{R_i=0,\forall i\in[T]\})}{\mathbb{P}(\{R_i=0,\forall i\in[T]\})} \\ \ge&\frac{\frac{9}{10}-\frac{1}{5}}{\frac{4}{5}}=\frac{7}{8}\end{aligned} \tag{8}$$

where the inequality by

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

Note that the conditional distribution $\mathbb{P}_{M_\varphi}$ and $\mathbb{P}_{\overline{M}_\varphi}$ on output policy π given $R_i=0\forall i\in[T]$ are exactly the same because MDP oracle M_φ and simulator $\overline{M}_\varphi$ only differ on the last layer. Thus, we have

$$\mathbb{P}_{\overline{M}_\varphi}(\mathcal{A}_{RL}\text{ succeeds with access to }\overline{M}_\varphi|R_i=0,\forall i\in[T])$$

$$= \mathbb{P}_{M_\varphi}(\mathcal{A}_{RL} \text{ succeeds with access to } M_\varphi | R_i = 0, \forall i \in [T])$$

Since, $\mathbb{P}_{\overline{M}_\varphi}(\{R_i = 0, \forall i \in [T]\}) = 1$ by construction of the simulator, we conclude that

$$\mathbb{P}_{\overline{M}_\varphi}(\mathcal{A}_{RL} \text{ succeeds with access to } \overline{M}_\varphi) \geq \frac{7}{8}.$$

□

Together by claim 3.2 and claim 3.3, if there is an algorithm $\mathcal{A}_{RL}$ that solves the LINEAR-3-RL with feature dimension $d = 2v^r$ in time $v^{r^2/4}$ with error probability 1/10, then the constructed algorithm $\mathcal{A}_{SAT}$ can solve the UNIQUE-3-SAT problem in time $poly(d) \cdot v^{r^2/4}$ with error probability 1/8 (the $poly(d)$ factor is because each call to the $\bar{M}_\varphi$ takes $poly(d)$ time).

At this moment, we choose

$$r = 8q,$$

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

(9) $$d^q \leq v^{\frac{r^2}{4}}$$

and

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

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.

Together, if there is an algorithm $\mathcal{A}_{RL}$ that solves the LINEAR-3-RL with feature dimension d in time d^q with error probability 1/10, by choosing v s.t. $d = 2v^r$, $\mathcal{A}_{RL}$ solves the LINEAR-3-RL with feature dimension d in time $v^{r^2/4}$ with error probability 1/10, and hence we know by claim 3.2 and claim 3.3, there is an randomized algorithm $\mathcal{A}_{SAT}$ that solves the UNIQUE-3-SAT problem with v variables in time $poly(d) \cdot v^{\frac{r^2}{4}} = poly(v)$ with error probability $\frac{1}{8}$, i.e. the reduction proposition (proposition 3.2) 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 to 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.

4. Linear MDPs are both computational and statistical efficient

In the previous section, 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 [Yang and Wang, 2019], that it turns out this condition ensures the both computational and statistical efficiency, i.e. there is no computational-statistical gap in the following class of MDPs.

Definition 4.1. *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 the 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.

Example 4.1 (Tabular MDPs)**.**

Now we propose that the class of linear MDPs is included in the linear V^*/Q^* class.

From now on, we assume $|\mathcal{A}| < \infty$, and reward $R.$ is deterministic so r coincides[5] with R

In the rest of the section, we review the literature [Yang, Wang et. al. 2019], and present the both computational and statistical efficiency results for linear MDPs, by first introducing the algorithm and the main result in [Yang and Wang, 2019][6] that directly implies the both polynomial runtime and polynomial sample conplexity.

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

[5]In [Yang and Wang, 2019], it is remarked that the following results are readily generalized to stochastic rewards.

[6]In our review, we still assume the time-homogeneity of the MDP for simplicity, for more general setting of time-inhomogeneous setting, please see [Yang and Wang, 2019]

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 (Algorithm 1), **Least-Square Value Iteration with Upper-Confidence Bounds**, or **LSVI-UCB**. In the Algorithm 1, each episode consists

Algorithm 1 Least-Square Value Iteration with Upper-Confidence Bounds (LSVI-UCB), [Yang, Wang et. al. 2019]

```
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:         $\Lambda_h \leftarrow \sum_{i=1}^{k-1} \phi(s_h^i, a_h^i)\phi(s_h^i, a_h^i)^\intercal + \lambda \cdot \mathbf{I}$.
5:         $w_h \leftarrow \Lambda_h^{-1} \sum_{i=1}^{k-1} \phi(s_h^i, a_h^i)\Big[r(s_h^i, a_h^i) + \sup_{a\in\mathcal{A}} Q(s_{h+1}^i, a)\Big]$.
6:         $Q(\cdot,\cdot) \leftarrow \min\{w_h^\intercal \phi(\cdot,\cdot) + \beta[\phi(\cdot,\cdot)^\intercal \Lambda_h^{-1} \phi(\cdot,\cdot)]^{1/2}, H\}$.
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
```

of two loops over all steps. In the first loop, parameters (w_h, Λ_h) are updated to build the action-value function Q. Note here we start the loop from $h = H$, as we need to build the action-value function Q backwards as in the dynamic programming. In the second loop, the agent greedily choose the action,

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

to approach the maximum of the action-value function Q built in the first loop. We remark a boundary case that when $k = 1$, the summation in line 4-5 are form 1 to 0, so we simply assign $\Lambda_h \leftarrow \lambda\mathbf{I}$ and $w_h \leftarrow 0$. In this case, there is no actual update happening in line 6.

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

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

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, if 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, it is natural to think the following L^2-regularized least-squares problem, at the k-th episode and the h-th step:

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

and hence we naturally apply the solution of this regularized least-squares problem in updating the parameter w_h, namely, the line 4-5, where Λ_h is exactly the Gram matrix appearing in the solution of this regularized least-squares problem.

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

$$\beta[\phi(\cdot,\cdot)^\intercal \Lambda_h^{-1} \phi(\cdot,\cdot)]^{1/2},$$

where, intuitively, $m := [\phi(\cdot,\cdot)^\intercal \Lambda_h^{-1} \phi(\cdot,\cdot)]^{1/2}$ represents the effective numbers of samples, along the ϕ direction, that our agent has observed till step h, and $\beta/\sqrt{m}$ represents the uncertainty along the ϕ direction.

Now we first analyze the time complexity for the 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, If we compute Λ_h^{-1} using the Sherman-Morrison formula, which takes $\mathcal{O}(d^2)$ time, the time complexity of the Algorithm 1 is dominated by the time complexity in computing $\sup_{a\in\mathcal{A}} Q(s_{h+1}^i, a)$ for all $i \in [k]$, which takes $\mathcal{O}(d^2|\mathcal{A}|K)$ time per step, 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 $\mathcal{O}(d^2|\mathcal{A}|KT)$.

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

Theorem 4.1 (**Yang, Wang et al., 2019**). *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 the Algorithm 1 implied by the theorem 1, we follow a similar discussion in section 3.1 [Chi et al, 2018], that, given

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

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

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

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 choose K large enough s.t.

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

that is, by plugging in $C = \sqrt{d^3H^3\iota^2}$, our K satisfies

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

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

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

and

$$8\sqrt{d^3H^3}\log(\frac{2dH}{p})\frac{1}{\sqrt{K}} < \frac{\varepsilon}{2}. \tag{11}$$

The inequality (3) further allows us to choose $K = \mathcal{O}(d^3H^3/\varepsilon^2)$, where $\mathcal{O}(\cdot)$ hides only absolute constants, and hence

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

i.e. π is an ε-optimal policy.

That is, by runing the Algorithm 1 for $K = \mathcal{O}(d^3H^3/\varepsilon^2)$ episodes and drawing $HK = \mathcal{O}(d^3H^4/\varepsilon^2)$ 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 action-value function Q at the k-th episode.

Together, it is established that there is no computational-statistical gap for linear MDPs. We remark the low rank property of linear MDPs, i.e. given a linear MDP M, if we fix a deterministic policy π, the corresponding Markov chain has a transition matrix P, with $\text{rank}(P) = \mathcal{O}(d)$.

Remark 4.2. Given a linear MDP M, and a fixed deterministic policy $\pi \in \mathcal{A}^{\mathcal{S}}$, if we define $P \coloneqq [P_{ij}]$ where $P_{ij} \coloneqq p(s_j|s_i, \pi(s_i))$, then $\text{rank}(P) \coloneqq \dim(\text{col}P) = \mathcal{O}(d)$.

We finish this seciton by pointing out that relationship between the computational efficiency of a 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 computational efficicent and low-rank, while the Linear Q^*/V^* MDPs are computationally hard and high-rank.

5. In showing the analogous result for Linear Mixture MDPs

To investigate relationship between the computational efficiency of a MDP and the rank of the transition matrix in the corresponding Markov chain, we present another class of MDPs.

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 $(s,a) \in \mathcal{S} \times \mathcal{A}, s' \in \mathcal{S}$,*

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

6. Acknowledgements

This survey paper and related research are conducted under the supervision of Professor Steven Heilman, and funded by Graduate Teaching Assistantship, Department of Mathematic, 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 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 supporting 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 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 the life and every aspect of the author's life philosophy: something greater will.

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

7. Reference

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Silver, D., Schrittwieser, J., Simonyan, K. et al. Mastering the game of Go without human knowledge. *Nature*, 550: 354–359, 2017.

Daniel Kane, Sihan Liu, Shachar Lovett, and Gaurav Mahajan. Computational-statistical gaps in reinforcement learning. arXiv preprint *arXiv:2202.05444*, 2022.

Simon S. Du, Sham M. Kakade, Jason D. Lee, Shachar Lovett, Gaurav Mahajan, Wen Sun, Ruosong Wang. Bilinear Classes: A Structural Framework for Provable Generalization in RL. arXiv preprint *arXiv:2103.10897*, 2021.

Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan. Provably efficient reinforcement learning with linear function approximation. In *Conference on Learning Theory*, 2020

M. L. Puterman. Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley and Sons, 2014.

L G Valiant and V V Vazirani. Np is as easy as detecting unique solutions. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, page 458–463, 1985.

Michael Sipser. *Introduction to the Theory of Computation*, 3rd edition, ISBN-13: 978-1-133-18779-0.

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

Xipeng Qiu, *Neural Networks and Deep Learning*, ISBN-13: 978-7-111-64968-7.

Richard S.Sutton and Andrew G.Barto, *Reinforcement Learning: An Introduction*, 2nd edition, ISBN-13: 978-0-262-19398-6.

Rummery G A and Niranjan M. On-line q-learning using connectionist systems[R]. Department of Engineering, University of Cambridge, 1994.

Mnih V, Kavukcuoglu K, Silver D, et al., Human-level control through deep reinforcement learning[J]. *Nature*, 518(7540):529-533, 2015.

Ashish Vaswani, Noam Shazeer, et al., Attention is all you need. arXiv preprint *arXiv:1706.03762*, 2017.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Christopher Hesse, Rafal J ´ozefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pond´e 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. *CoRR*, 2019.

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.

Alex Ayoub, Zeyu Jia, Csaba Szepesvari, Mengdi Wang, and Lin F Yang. Model-based reinforcement learning with value-targeted regression. *arXiv:2006.01107*, 2020.

Dongruo Zhou, Jiafan He, and Quanquan Gu. Provably efficient reinforcement learning for discounted mdps with feature mapping. In *International Conference on Machine Learning*, pages 12793–12802. PMLR, 2021.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. arXiv preprint *arXiv: 1312.5602*, 2013.

8. Appendix

8.1. **Proofs of lemmas in Section 4.**

Proof of Proposition 4.1.

we first show linear MDP satisfies the linear V^*/Q^* condition, and then we show linear mixture MDP, under a transformation of the feature maps, satisfies the linear V^*/Q^* condition.

For a linear MDP, we apply the Bellman optimality equation, for any $(s,a) \in \mathcal{S} \times \mathcal{A}$,

$$\begin{aligned} Q^*(s,a) &= r(s,a) + \mathbb{E}_{s' \sim p(\cdot|s,a)} V^*(s') \\ &= r(s,a) + \int_S V^*(s') dp(s'|s,a) \\ &= \langle \phi(s,a), \theta \rangle + \int_S V^*(s') \langle \phi(s,a), d\mu(s') \rangle \\ &= \langle \phi(s,a), \theta + \int_S V^*(s') d\mu(s') \rangle. \end{aligned} \tag{12}$$

By define $\tilde{\theta} := \theta + \int_S V^*(s') d\mu(s')$, we know Q^* is linear in the feature map ϕ, and we left to show $V^*(s) = \langle \tilde{\phi}(s), \tilde{\theta} \rangle$ for some $\tilde{\phi} : \mathcal{S} \to \mathbb{R}^d$. In fact, the existence of such $\tilde{\phi}$ is abundant, as if we consider $l_{\tilde{\theta}}(x) := \langle x, \tilde{\theta} \rangle$, $Image(l_{\tilde{\theta}}) = \mathbb{R}$, so whenever $V^*(s) \in \mathbb{R}$, there is a $x_s \in \mathbb{R}^d$ s.t. $l_{\tilde{\theta}}(x_s) = V^*(s)$, and we could define $\tilde{\phi}(s) := x_s$. □