Parameterized Indexed Value Function for Efficient Exploration in Reinforcement Learning

Tian Tan,1*† Zhihan Xiong,2* Vikranth R. Dwaracherla3
1Department of Civil and Environmental Engineering, Stanford University
2Department of Statistics, Stanford University
3Department of Electrical Engineering, Stanford University
{ttant, zxiong9, vikranth}@stanford.edu

Abstract

It is well known that quantifying uncertainty in the action-value estimates is crucial for efficient exploration in reinforcement learning. Ensemble sampling offers a relatively computationally tractable way of doing this using randomized value functions. However, it still requires a huge amount of computational resources for complex problems. In this paper, we present an alternative, computationally efficient way to induce exploration using index sampling. We use an indexed value function to represent uncertainty in our action-value estimates. We first present an algorithm to learn parameterized indexed value function through a distributional version of temporal difference in a tabular setting and prove its regret bound. Then, in a computational point of view, we propose a dual-network architecture, Parameterized Indexed Networks (PINs), comprising one mean network and one uncertainty network to learn the indexed value function. Finally, we show the efficacy of PINs through computational experiments.

1 Introduction

Efficient exploration is a long-established problem and an active research area in reinforcement learning (RL). It is well known in the RL community that maintaining an uncertainty estimate in the value functions is crucial for efficient exploration (Russo et al. 2018), (Ghavamzadeh et al. 2015), (Osband, Aslanides, and Cassirer 2018). Conventionally, dithering methods, such as ε-greedy and Boltzmann ε-greedy, induce exploration by randomly selecting and experimenting with actions. This is one of the common exploration schemes used in many applications (Mnih et al. 2015), (He et al. 2015), (Tan et al. 2019), for its simplicity. However, these schemes do not take uncertainty into account in their value estimates. As a result, they require a huge amount of data to learn a desirable policy through dithering in an environment with sparse and delayed reward signals.

Although there has been a growing interest in quantifying uncertainty, such as dropout (Srivastava et al. 2014), (Gal and Ghahramani 2016) and variational inference (Fortunato et al. 2017), (Touati et al. 2018), these methods are typically not suitable for sequential decision making (Osband, Aslanides, and Cassirer 2018). Instead, ensemble sampling (Osband et al. 2016), (Lu and Van Roy 2017), (Osband, Aslanides, and Cassirer 2018), which represents uncertainty by learning an approximate posterior distribution over value functions via bootstrapping, has achieved remarkable success in RL. Each member in the ensemble, typically a neural network, learns a mapping from the input state or feature space to action values from a perturbed version of the observed data. However, ensemble sampling methods demand learning and maintaining many neural networks in parallel, which can be both computation and memory intensive.

Index sampling (Dwaracherla, Van Roy, and Ibrahimi 2019) offers a possibility to distill the ensemble process into a “single” network which learns a mapping from the state or feature space and a random index space to the action values. The random index, can be thought as an independent random variable drawn from a fixed distribution \( p_z \). After learning this mapping, the resulting model can be approximately sampled from the posterior over networks conditioned on the observed data by passing different indices. Therefore, one effectively samples a value function from its posterior by sampling a random index \( z \).

In this paper, we consider index sampling via a class of Gaussian indexed value functions as we explicitly parameterize the action-value function as \( Q_z(x,a) = \nu(x,a) + m(x,a)z \), where \((x,a)\) represents a state-action pair, \( z \sim \mathcal{N}(0, 1) \), \( \nu(x,a) \) and \( m(x,a) \) are real valued functions representing mean and standard deviation of the action-values or commonly known as \( Q \)-values. We present an algorithm to learn such a parameterized indexed value function via a distributional variation of temporal difference (TD) learning, which we refer to as the distributional TD or Wasserstein TD when Wasserstein distance is used as the distance metric. We prove that this algorithm enjoys a Bayesian regret bound of

\[
\tilde{O}\left( H^2 \sqrt{|X||A|L} \right)
\]

in finite-horizon episodic MDPs, where \( H \) is the episode length, \( L \) is the number of episodes, and \(|X|\) and \(|A|\) are the cardinality of the state and action spaces. Then, we propose a dual-network architecture Parameterized Indexed Networks

*Equal contribution.
†Corresponding author.
Copyright © 2020, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.
(PINs) to generalize with complex models in deep reinforce-
ment learning. PINs consist of a mean network for learning \( \nu(x, a) \), and an uncertainty network for \( m(x, a) \). We demonstrate the efficacy of PINs on two benchmark problems, Deep-sea and Cartpole Swing-up, that require deep exploration from Deepmind bsuite (Osband et al. 2019). Our open-source implementation of PINs can be found at https://github.com/tiantan522/PINs.

2 Related Work

In the model-based setting, the optimal exploration strategy can be obtained through dynamic programming in Bayesian belief space given a prior distribution over MDPs (Guez, Silver, and Dayan 2012). However, the exact solution is intractable. An alternative way for efficient exploration is via posterior sampling. Motivated by Thompson Sampling, the posterior sampling reinforcement learning (PSRL) was first introduced in (Strens 2000) mainly as a heuristic method. The theoretical aspects of PSRL were poorly understood until very recently. Osband et al. (Osband and Van Roy 2017) established an \( \hat{O}(H^{1.5}|X|\sqrt{|A|L}) \) Bayesian regret bound for PSRL in finite-horizon episodic MDPs. Note that this result improves upon the best previous Bayesian regret bound of \( \hat{O}(H^{3.5}|X|\sqrt{|A|L}) \) for any reinforcement learning algorithm. Although PSRL enjoys state-of-the-art theoretical guarantees, it is intractable for large practical systems. Approximations are called for when exact solution is intractable.

The RLSVI algorithm in (Osband et al. 2017) approximates the posterior sampling for exploration by using randomized value functions sampled from a posterior distribution. However, its performance is highly dependent on choosing a reasonable linear representation of the value function for a given problem. To concurrently perform generalization and efficient exploration with a flexible nonlinear function approximator, (Osband et al. 2016) proposed a bootstrapped Deep Q-Network (BootDQN) which learns and maintains an ensemble of neural networks in parallel, each trained by a perturbed version of the observed data. The issue with bootstrapping is that it possesses no mechanism for estimating uncertainty that does not come from the observed dataset. This restraints BootDQN from performing well in environments with delayed rewards. A simple remedy was proposed recently in (Osband, Aslanides, and Cassirer 2018) where each of the ensemble members is trained and regularized to a prior network/function that is fixed after random initialization. Equivalently, this amounts to adding a randomized untrainable prior function to each ensemble member in linear representation. This additive prior mechanism can also be viewed as some form of self-motivation or curiosity to direct learning and exploration if the agent has never observed a reward.

Index sampling presented in (Dwaracherla, Van Roy, and Ibrahimi 2019) in a bandit setting aims to learn a posterior distribution over value functions with an extra random index as input. In this paper, we extend this to RL setting using distributional TD approach. We first present an algo-

3 Analysis of Parameterized Indexed Value Function

We first consider learning a parameterized indexed value function in a tabular setting. The environment is modelled as an episodic Markov Decision Process (MDP).

Markov Decision Process Setup

Our MDP formulation is adopted from (Osband et al. 2017), which is stated as the following assumption.

Assumption 1. The MDP \( \mathcal{M} = (S, A, H, R, P, \rho) \) is finite-horizon time-inhomogeneous such that the state space can be factorized as \( S = S_0 \cup \cdots \cup S_{H-1} \) and each \( s_h \in S_h \) can be written as a pair \( s_h = (h, x) \), \( x \in \mathcal{X} \) for each time step \( h \in \{0, 1, \ldots, H-1\} \) with \( |\mathcal{X}| < \infty \). Further, we have \( |A| < \infty \) and \( P(s_{h+1} \in S_{h+1} | s_h, a_h) = 1 \) for any \( a_h \in A \), \( h < H-1 \) and the MDP will terminate with probability 1 after taking action \( a_{H-1} \). Finally, the reward is always binary, which means that \( R(s, a) \in \{0, 1\} \) for any \( s \in S, a \in A \).

Then, for each state-action pair \( (h, x, a) \), the transition leads to an observation of \( o = (r, x') \) pair consisting of the next \( x' \in \mathcal{X} \) and a reward \( r \in \{0, 1\} \) after taking action \( a \). We denote this pair \( o \) as an outcome of the transition. Let \( \mathcal{P}_{h,x,a} \) be a categorical distribution after taking an action \( a \) in state \( (h, x) \) over the \( 2|\mathcal{X}| \) possible outcomes, which are finite because of the binary reward. We make the following assumption in our analysis:

Assumption 2. With the above setup, for each \( (h, x, a) \in \{0, 1, \ldots, H-2\} \times \mathcal{X} \times A \), the outcome distribution is drawn from a Dirichlet prior \( \mathcal{P}_{h,x,a} \sim \text{Dirichlet} (\alpha_{h,x,a}^0) \) for \( \alpha_{h,x,a}^0 \in \mathbb{R}_+^{2|\mathcal{X}|} \), and each \( \mathcal{P}_{h,x,a} \) is drawn independently. Further, assume that there exists some \( \beta \geq 3 \) such that \( 1^T \alpha_{h,x,a}^0 = \beta \) for all \( (h, x, a) \).

The Dirichlet distribution is chosen because it is a conjugate prior of categorical distribution. This helps in simplifying our analysis on Bayesian regret bound. Nevertheless, one may extend our analysis to the reward which has a bounded support in \([0, 1]\) by using techniques from (Agrawal and Goyal 2012).

We define a policy to be a mapping from \( S \) to a probability distribution over \( A \) and denote the set of all policies by \( \Pi \). For any MDP \( \mathcal{M} \) and policy \( \pi \in \Pi \), we can define a value function \( V^\pi_{\mathcal{M}, h}(x) = \mathbb{E}_{\mathcal{M}, \pi} \left[ \sum_{t=h}^{H-1} r_{t+1} | x_h = x \right] \) as the expected sum of rewards starting from \( x \) at time step \( h \) and following policy \( \pi \). Then, the optimal value function can be defined as \( V^*_{\mathcal{M}, h}(x) = \max_a V^*_{\mathcal{M}, h}(x, a) \), and the optimal state-action value function at time \( h \) can be defined accordingly:

\[
Q^*_{\mathcal{M}, h}(x, a) = \mathbb{E}_{\mathcal{M}} \left[ r_{h+1} + V^*_{\mathcal{M}, h+1}(x_{h+1}) \mid x_h = x, a_h = a \right]
\] (1)
Algorithm for Tabular RL

In this section, we present our algorithm for learning a parameterized indexed value function in the above tabular setting, which is summarized as Algorithm 1.

Here we explicitly parameterize the state-action value function in episode $l$ as $Q_{Z,h}^l(x,a) = \nu^l(h,x,a) + m^l(h,x,a)Z_{h,x,a}$, where $Z_{h,x,a} \sim \mathcal{N}(0,1)$ is assumed to be independent for each state-action pair $(h,x,a)$ for the ease of analysis. To simplify notation, when $Z$ appears as a subscript of $Q$ function, it implicitly depends on the state-action input of $Q$. With a slight abuse of notation, we sometimes use a lower case subscript $Q_{z,h}^l(x,a) = \nu^l(h,x,a) + m^l(h,x,a)z_{h,x,a}$ as a sampled value function, where $z_{h,x,a}$ is a sample from a standard Gaussian. Further, let $Q \sim \mathcal{N}(\theta, \sigma^2)$ be a prior of the $Q$ function for all $(h,x,a)$.

Denote $D_{h,x,a}^{-1} = \{(r_{h+1}^j, x_{h+1}^j) \mid x_h^j = x, a_h^j = a, j < l\}$, which includes all observed transition outcomes for $(h,x,a)$ up to the start of episode $l$, and $n^l(h,x,a) = |D_{h,x,a}^{-1}|$.

In the spirit of temporal difference learning, $\nu(r,x') \in D_{h,x,a}^{-1}$ and $n^l(h,x,a) \neq 0$, we define a learning target for our parameterized indexed value function $Q_{Z,h}^l(x,a)$ as:

$$y^l(h,x,a,r,x') = r + \frac{\sigma Z}{\sqrt{n^l(h,x,a)}} + \max_{a' \in A} Q_{z,h+1}^{l+1}(x',a')$$

where $Z_{h+1,x',a'} \sim \mathcal{N}(0,1)$ is independently sampled for each $(h+1,x',a')$, $\sigma$ is a positive constant added to perturb the observed reward, and $Z \sim \mathcal{N}(0,1)$ is a standard normal random variable. This target is similar to the one used in RLSVI (Osband et al. 2017) except that $Z$ is a random variable and the added noise $\sigma$ is divided by $\sqrt{n^l(h,x,a)}$. This decay of noise $\sigma$ is needed to show concentration in the distribution of value function as we gather more and more data.

The learning target resembles traditional TD in the sense that it can be viewed as one-step look-ahead. However, a random variable $Z$ is needed in the target for indexing. Specifically, any realization of $Z$ reduces it to a TD error between a current estimate at the sampled index and a perturbed scalar target. To take all values of $Z$ into account, we propose to match their distributions directly, resulting in a distributional version of TD learning. We refer it as distributional temporal difference, or more specifically Wasserstein temporal difference when $p$-Wasserstein distance is used as a metric. In the tabular setting, this leads to the following loss function:

$$L_p(\nu^l(h,x,a), m^l(h,x,a)) = \sum_{(r,x') \in D_{h,x,a}^{-1}} W_p(Q_{Z,h}^l(x,a), y^l(h,x,a,r,x'))^2,$$

where $Q_{Z,h}^l(x,a) = \nu^l(h,x,a) + m^l(h,x,a)Z_{h,x,a}$. Here, $W_p(\cdot, \cdot)$ is the $p$-Wasserstein distance between two distributions. Similarly, we can define a regularization to the prior as $\psi_p(\nu,m) = \beta W_p(\nu + mZ, Q)^2$, where $\beta = \sigma^2/\sigma_0^2$. Therefore, when updating the parameters $({\nu, m})$, one needs to solve $\min_{(\nu, m)} \{L_p + \psi_p\}$.

For two Gaussian random variables $X \sim \mathcal{N}(\mu_X, \sigma_X^2)$ and $Y \sim \mathcal{N}(\mu_Y, \sigma_Y^2)$ the 2-Wasserstein distance between them is simply $W_2(X,Y)^2 = (\mu_X - \mu_Y)^2 + (\sigma_X - \sigma_Y)^2$. Therefore, we can minimize $L_2 + \psi_2$ exactly by

$$L^l(\nu, m) = \frac{\sum_{(r,x') \in D_{h,x,a}^{-1}} (r + \max_{a' \in A} Q_{z,h+1}^{l+1}(x',a')) + \beta \nu}{n^l(h,x,a) + \beta}$$

$$m^l(h,x,a) = \frac{\sqrt{n^l(h,x,a) \sigma + \beta \sigma_0}}{n^l(h,x,a) + \beta}$$

By combining these steps, we get Algorithm 1 for tabular RL. As a sanity check, we see that the standard deviation term $n^l$ will shrink and decrease to zero as $n^l \to \infty$, which is achieved by decaying $\sigma$ by $\sqrt{n^l}$ in the learning target.

Algorithm 1: Tabular Wasserstein TD (WTD)

1. For all $l \in \{0,1,\ldots,L\}$, $h \in \{0,\ldots,H-1\}$, $x \in X$, and $a \in A$, initialize $D_{h,x,a}^0 = \emptyset$ and

   $$\nu^l(H,x,a) = m^l(H,x,a) = 0$$

2. For $l = 1, \ldots, L$ do
   3. Sample $z_{h,x,a} \sim \mathcal{N}(0,1)$ for all $x \in X$, $a \in A$, and $h \in \{0,\ldots,H-1\}$
   4. For $h = H-1, \ldots, 0$ do
      5. For $x \in X$, $a \in A$ do
         6. Update $\nu^l(h,x,a)$ and $m^l(h,x,a)$ using equations (3) and (4)
      7. end
   8. end
   9. Observe $x_0$
   10. $D_{h,x,a}^l \leftarrow D_{h,x,a}^{l-1}$ for all $x \in X$, $a \in A$ and $h \in \{0,\ldots,H-1\}$
   11. For $h = 0, \ldots, H-1$ do
      12. Take action $a_h = \arg \max_{a \in A} Q_{z,h}^l(x_h,a)$ and observe $(r_{h+1}, x_{h+1})$
      13. $D_{h,x,a}^h \leftarrow D_{h,x,a}^h \cup \{(r_{h+1}, x_{h+1})\}$
   14. end
   15. end

Bayesian Regret Bound

The following theorem is the main result of our analysis which establishes a Bayesian regret bound for Algorithm 1.

Theorem 1. Consider an agent WTD with infinite buffer, greedy actions and an MDP stated in Assumption 1 with planning horizon $H$. Under Assumption 2 with $\beta \geq 3$, if Algorithm 1 is applied with $\sigma^2 = 3H^2$, $\theta = H$ and $\beta = \beta$, for any number of episodes $L \in \mathbb{N}$, we have

$$\text{BayesRegret (WTD, L)} \leq 5H^2 \sqrt{\beta |X||A| L \log_+ (2|X||A| H^2)} \log_+ \left(1 + \frac{L}{|X||A|} \right)$$

$$= \bar{O} \left( H^2 \sqrt{|X||A| L} \right)$$
where \( \bar{O}(\cdot) \) ignores all poly-logarithmic terms and \( \log_+ (x) = \max \{1, \log (x)\} \).

**Proof.** The complete proof is included in the supplemental material C. Here, we provide a sketch of the proof, whose framework takes the analysis of RL-SVI in (Osband et al. 2017) as reference. We first define a stochastic Bellman operator as follows.

**Stochastic Bellman Operator** Based on the updating rules (3) and (4), for a general function \( Q \in \mathbb{R}^{[X][A]} \), Algorithm 1 defines a functional operator \( F_{l,h} \):

\[
F_{l,h} Q(x,a) = \frac{\sum_{(r,x') \in D_{l,h,a}^{-1}} (r + \max_{a' \in A} Q(x',a')) + \beta \bar{O}}{n^l(h,x,a) + \beta}
+ \frac{\sqrt{n^l(h,x,a) \sigma + \beta \sigma_0 Z_{h,x,a}}}{n^l(h,x,a) + \beta}, Z_{h,x,a}
\]

where \( Z_{h,x,a} \sim \mathcal{N}(0,1) \) is independent from \( Q \). Specifically, with this operator, we have \( Q_{l,h}^* = F_{l,h} Q_{l',h+1}^* \).

Recall the definition of \( Q_{M,h}^* \) in equation (1). We denote \( F_{M,h} \) as the true Bellman operator that satisfies \( Q_{M,h}^* = F_{M,h} Q_{M,h}^* \).

To prove Theorem 1, we resort to the following key lemma proved in (Osband et al. 2017).

**Lemma 1.** Let \( Q_{Z,0}^*, \ldots, Q_{Z,H}^* \) be the sequence of state-action value function learned by Algorithm 1, where \( Q_{Z,H}^* = 0 \), and \( \pi_l \) be the greedy policy based on these \( Q \)-values. For any episode \( l \in \mathbb{N} \), if we have

\[
\mathbb{E} \left[ \max_{a' \in A} Q_{Z,0}^* (x'_0,a'_0) \right] \geq \mathbb{E} \left[ \max_{a' \in A} Q_{M,0}^* (x'_0,a'_0) \right]
\]

then, by defining \( \Delta_l = V_{M,0}^* (x'_0) - V_{M,0}^* (x'_0) \), we can have

\[
\mathbb{E} [\Delta_l] \leq \mathbb{E} \left[ \sum_{h=0}^{H-1} \left( (F_{l,h} - F_{M,h}) Q_{Z,h+1}^* (x'_h,a'_h) \right) \right]
\]

In order to use the bound (6), it is necessary to verify that the condition (5) is satisfied by our Algorithm 1. To achieve this, we resort to the concept of stochastic optimism, which is defined as the following:

**Stochastic Optimism** A random variable \( X \) is *stochastically optimistic* with respect to another random variable \( Y \), denoted as \( X \geq_{SO} Y \), if \( \mathbb{E}[u(X)] \geq \mathbb{E}[u(Y)] \) holds for all convex increasing function \( u : \mathbb{R} \mapsto \mathbb{R} \).

If Assumption 2 holds, by additionally assuming some technical conditions on parameters \( \sigma^2, \theta \) and \( \beta \), it is possible to show that

\[
Q_{Z,0}^* (x,a) \geq_{SO} Q_{M,0}^* (x,a) \quad \text{for any history } H_{l-1} = \bigcup_{k=1}^{l-1} \{(h,k,x_k',a_k',r_{h+k-1}) \mid h < H \}
\]

and \( (x,a) \in A \times A \).

By using Lemma 2 in (Osband et al. 2017) on preservation of optimism, we can then further show that

\[
\mathbb{E}_{H_{l-1}} \left[ \max_{a' \in A} Q_{Z,0}^* (x'_0,a'_0) \right] \geq \mathbb{E}_{H_{l-1}} \left[ \max_{a' \in A} Q_{M,0}^* (x'_0,a'_0) \right]
\]

Therefore, condition (5) can be obtained by simply taking expectation over all possible \( H_{l-1} \).

**Towards Regret Bound** After verifying condition (5), we can then apply bound (6) in Lemma 1 to our Algorithm 1 to get

\[
\text{BayesRegret} (\text{WTD}, L) = \mathbb{E} \left[ \sum_{l=1}^{L} \Delta_l \right]
\]

Finally, after some algebraic manipulations, it is possible to bound this term by \( \bar{O} (HL^2 \sqrt{E[X][A]} \), which completes the proof.

### 4 Parameterized Indexed Networks

Previously we have proved the efficiency of Wasserstein TD for learning a parameterized indexed value function in the tabular setting. We next discuss the application of this methodology to deep RL. We note that some of the conditions to derive Theorem 1 are violated in most cases of deep RL, which puts us in the territory of heuristics. More precisely, technical conditions on parameters \( \sigma^2, \theta \) may not hold, the number of visitations \( n_l (h,x,a) \) cannot be counted in general, and as we will see soon that the independence among sampled \( z \)'s (in Algorithm 1) required for analysis can be violated owing to algorithmic considerations. Nevertheless, insights from the previous algorithm helped us for our design of a deep learning algorithm that can perform well in practice.

We first recall the essence of index sampling, which aims to effectively sample one value function from posterior by sampling an index \( z \sim \mathcal{N}(0,1) \). To induce temporally consistent exploration or deep exploration (Osband et al. 2016), the agent needs to follow a greedy policy according to the sampled value function in the next episode of interaction. This insight bears a resemblance to the idea of *Thompson sampling* (Thompson 1933) for balancing between exploration and exploitation, and it amounts to sampling one single \( z \) per episode in index sampling. A general algorithm describing the interaction between an agent with index sampling (IS) and its environment is summarized in Algorithm 2 *live-IS*. Note that we use *live-IS* as an *off-policy* algorithm since the agent samples from a replay buffer containing previously observed transitions to obtain its current policy (in *line 3*).

We realize that for any realization of \( z \)'s that is fixed within an episode, applying the updating rules for action-value function (3) and (4) in the tabular case then gives

\[
Q_{l,h} (x,a) = \frac{\sum_{(r,x') \in D_{h,a}^{-1}} (r + Q_{l+1}^* (x',a')) + \beta \bar{O}}{n^l(h,x,a) + \beta}
+ \frac{\sqrt{n^l(h,x,a) \sigma + \beta \sigma_0 z}}{n^l(h,x,a) + \beta}
\]

\[
= \frac{\sum_{(r,x') \in D_{h,a}^{-1}} (r + \bar{O} (h+1,x',a')) + \beta \bar{O}}{n^l(h,x,a) + \beta}
+ \frac{\sqrt{n^l(h,x,a) \sigma + \beta \sigma_0 + \sum_{(r,x') \in D_{h,a}^{-1}} m^l (h+1,x',a')} z}{n^l(h,x,a) + \beta}
\]
Algorithm 2: live.IS

Input: agent, environment
1 for l in (1, 2, …) do
2 Sample $z^l \sim \mathcal{N}(0, 1)$
3 agent.learn_from_buffer()
4 history ← environment.reset()
5 while history is not terminal do
6    action ← agent.act(history) ($z^l$)
7    history ← environment.step(action)
8 end
9 agent.update_buffer(history)
10 end

where $\tilde{a} = \arg\max_{a' \in A} Q_{z^l, h+1}^l (x', a')$ is the best action with respect to the sampled value function at the next time step. Notice that this update for $Q$ function is equivalent to compute the minimization of the distributional loss (2) with a modified target:

$$\left(\nu^l (h, x, a), m^l (h, x, a)\right) = \min_{\nu, m} \sum_{(r, x') \in D_{h, r, a}} W_2 \left(\nu + m Z, \tilde{y}^l\right)^2 + \psi_2 (\nu, m),$$

where

$$\tilde{y}^l (h, x, a, r, x') = r + \frac{\sigma Z}{\sqrt{\nu^l (h, x, a)}} + Q_{z, h+1}^l (x', \tilde{a}) + \sqrt{\sigma Z} \left(\frac{\sigma Z}{\sqrt{\nu^l (h, x, a)}} + m^l (h+1, x', \tilde{a})\right) \text{ mean target}$$

Note that in target $\tilde{y}^l$ the subscript $Z$ is a random variable instead of a realization. We see that the uncertainty measure at the next step $h + 1$ is propagated into the uncertainty at the current step $h$. Although derived from a completely different perspective, this is consistent with previous findings in (O’Donoghue et al. 2017).

Network Design

According to the above derivation from index sampling, Wasserstein TD for learning a parameterized indexed value function is essentially trying to fit the point estimate $\nu (s, a)$ and the uncertainty estimate $m (s, a)$ to their one-step look-ahead mean target and uncertainty target respectively as illustrated in $\tilde{y}^l$. When $W_2 (\cdot, \cdot)$ is used as metric, this reduces to minimizing two separate squared errors. Hence, we propose to use a dual-network architecture named Parameterized Indexed Networks (PINs) for deep RL consisting of a mean network for learning $\nu (s, a)$ and an uncertainty network for learning $m (s, a)$. The two networks are joined by a Gaussian with index $Z \sim \mathcal{N}(0, 1)$.

Let $\theta = (\phi, \omega)$ be parameters of the trainable/online PINs, where $\phi, \omega$ are parameters of the mean and the uncertainty network respectively, and let $\tilde{\theta} = (\tilde{\phi}, \tilde{\omega})$ be the parameters of the target nets (Mnih et al. 2015). Then, in learn_from_buffer() of Algorithm 2, the agent is essentially trying to minimize the following distributional loss:

$$\mathcal{L}_{\text{dist}} \left(\theta, \tilde{\theta}, D\right) = \beta W_2 (Q_{\theta, Z}, \tilde{Q})^2 + \sum_{(s, a, r, x') \in D} W_2 \left(Q_{\theta, Z} (s, a), r + \sigma(l) Z + \gamma Q_{\tilde{\theta}, Z} (s', \tilde{a})\right)^2$$

where $\tilde{Q}$ is a prior distribution, $\beta$ is a prior scale which is treated as a hyperparameter, $Z \sim \mathcal{N}(0, 1)$, $\gamma$ is a discount factor, $D$ represents sampled minibatch of data consisting of past transitions from a replay buffer, and $\sigma (l)$ is a perturbation added to observed reward $r$. Further, being consistent with our previous analysis, we can consider slowly decaying the added noise variance $\sigma (l)^2$ after each episode of learning to improve concentration.

For action selection in the learning target, we find empirically that $\tilde{a}$, obtained by taking $\arg\max$ on the sampled value function, can sometimes affect learning stability. Therefore, we instead use $\bar{a} \in \arg\max_{a' \in A} \mathbb{E} \left[Q_{\theta, Z}^2 (s', a')\right] = \arg\max_{a' \in A} \nu_{\phi} (s', a')$ for action selection, and the mean network reduces to a Deep Q-Network. We include a performance comparison between $\tilde{a}$ and $\bar{a}$ for action selection in supplemental material B.

Figure 1: Parameterized Indexed Networks

The overall architecture of PINs is depicted in Figure 1. We incorporate a prior mechanism by following (Osband, Aslanides, and Cassirer 2018), where each trainable network is paired with an additive prior network. Instead of explicitly regularizing to a prior, i.e. the $\beta W_2 (Q_{\theta, Z}, \tilde{Q})^2$ term in $\mathcal{L}_{\text{dist}}$, we add a randomized prior function to the Q function (Osband, Aslanides, and Cassirer 2018). The prior networks share the exact same architecture as their paired trainable networks and are fixed after random initialization. The structure of the uncertainty network is different from that of the mean network in the following manner:

- Softplus (Glorot, Bordes, and Bengio 2011) output layer to ensure non-negativity in output values;
- Multiple bootstrapped output heads to encourage diversity in uncertainty estimates: the last hidden layer of the
uncertainty net spins out $U$ different output heads. Head $u \in \{1, \ldots, U\}$ maps the input state $s$ to uncertainty estimates $m^u_w(s, a)$ for all $a \in A$. Despite that all but the output layer are shared among different heads, we promote diversity by (1) applying a bootstrap mask $\text{Ber}(0.5)$ to each output head so that data in buffer is not completely shared among heads (Osband et al. 2016), and (2) each head is paired and trained together with a slightly different prior $\bar{m}^u_w(s, a)$.

For each episode $l$, we first sample an index $z^l \sim \mathcal{N}(0, 1)$ and an output head $u \sim \text{Unif}(\{1, \ldots, U\})$. The agent then acts greedily with respect to the sampled value function with additive priors $Q^u_{z^l} = \nu + m^u z^l + \beta_1 \bar{v} + \beta_2 \bar{m}^u z^l$ for consistent exploration in episode $l$, where $\beta_1, \beta_2$ are scaling hyperparameters for the mean prior and the uncertainty prior, respectively. Here the additive prior distribution can be seen as $Q^u_{z^l} = \beta_1 \bar{v} + \beta_2 \bar{m}^u z^l$. A detailed training algorithm for PINs is included in supplemental material A.

5 Experimental Results

We evaluate the performance of PINs on two benchmark problems, Deep-sea and Cartpole Swing-up, that highlight the need for deep exploration from Deepmind bsuite (Osband et al. 2019), and compare it with the state-of-the-art ensemble sampling methods: the bootstrapped DQN with additive prior networks (BSP$_K$) (Osband, Aslanides, and Cassirer 2018) and the bootstrapped DQN without prior mechanism (BS$_K$) (Osband et al. 2016), where $K$ denotes the number of networks in the ensemble.

Deep-sea

Deep-sea is a family of grid-like deterministic environments (Osband et al. 2017), which are indexed by a problem size $N \in \mathbb{N}$, with $N \times N$ cells as states, and sampled action mask $M_{ij} \sim \text{Ber}(0.5), i, j \in \{1, \ldots, N\}$. Action set $A = \{0, 1\}$, and at cell $(i, j)$, $M_{ij}$ represents action “left” and $1 - M_{ij}$ represents action “right”. The agent always starts in the upper-left-most cell at the beginning of each episode. At each cell, action “left” (“right”) takes the agent to the
The usual dithering methods will need $\Omega(2^N)$ episodes to learn the optimal policy, which grows exponentially with the problem size $N$.

Figure 2 shows the cumulative reward of our PINs and various ensemble models on Deep-sea with four different sizes $N = 15, 20, 25, 30$ for $6K$ episodes of learning. Each approach is evaluated over 5 different random seeds. We consider that the agent has learned the optimal policy when there is a linear increase in cumulative reward by the end of training. For example, our agent with PINs successfully learned the optimal policy in $3/5$ seeds when $N = 30$, in $4/5$ seeds when $N = 25$ within $6K$ episodes. All networks are MLP with 1 hidden layer. For PINs, the mean network has 300 units and the uncertainty network has 512 units with $U = 10$ output heads in the output layer. We set $\sigma = 2$ for the added noise without any decay for experiments on Deep-sea, and $\beta_1 = \beta_2 = 2$. For ensemble models (Osband, Aslanides, and Cassirer 2018), each single network in the ensemble contains 50 hidden units, and we set prior scale $\beta = 10$ for BSP as recommended in (Osband, Aslanides, and Cassirer 2018). For BS, we simply let $\beta = 0$ to exclude the prior mechanism. For all bootstrapping, we use Bernoulli mask with $p = 0.5$. We see that the performance of PINs is comparable to that of ensemble methods BSP$^5$ and BSP$^7$ with additive priors. Also, note that the PINs are relatively more efficient in computation as PINs only require 2 back-propagations per update while BSP$^K$ need $K$ backward passes per update. In addition, even equipped with 10 separate networks, BS$^{10}$ struggles to learn the optimal policy as $N$ increases, which highlights the significance of a prior for efficient exploration.

Conceptually, the main advantage of our PINs is that it distributes the tasks of learning an action function and measuring uncertainty in estimates into two separate networks. Therefore, it is possible to further enhance exploration by using a more carefully-crafted and more complex design of the uncertainty network and its prior without concerning about the stability of learning in the mean network. As an example, a more delicate design that induces diverse uncertainty estimates for unseen state-action pairs can potentially drive exploration to the next-level. We consider experimenting with different architectures of the uncertainty network as our future work.

Cartpole Swing-up

We next present our results on a classic benchmark problem: Cartpole Swing-up (Sutton and Barto 2018), (Osband et al. 2019), which requires learning a more complex mapping from continuous states$^1$ to action values. Unlike the original cartpole, the problem is modified so that the pole is hanging down initially and the agent only receives a reward of 1 when the pole is nearly upright, balanced, and centered$^2$. Further, a cost of 0.05 is added to move the cart and the environment is simulated under timescale of 0.01s with a maximum of 1000 time-steps per episode.

Figure 3 shows a comparison on smoothed episodic reward in 3000 episodes of learning over 5 different random seeds. We failed to train a BS$^{10}$ agent that can learn a performant policy on this environment; thus, the results are omitted. To demonstrate computational savings of PINs, all networks used here have 3 hidden layers with 50 units in each layer. Besides, the uncertainty network spins out only $U = 2$ output heads. For added noise in PINs, we use $\sigma = 2$ and linearly decay it to 1 over the course of training to promote concentration in the approximate posterior distribution. As for prior scale, we use $\beta_1 = \beta_2 = 2$ for our PINs, and $\beta = 30$ for BSP as in (Osband, Aslanides, and Cassirer 2018). We see that PINs achieved similar performance to that of the ensemble models but with only two separate neural networks. Additionally, although PINs seem to progress slowly compared to BSP$^K$, they exhibit smaller variance in performance by the end of learning. This experiment demonstrates the computational efficiency that can be brought by PINs and by index sampling for representing uncertainty in action-value estimates.

---

$^1$A 8 dimensional vector in bsuite where we set the threshold of position $x$ to be 5.

$^2$Reward 1 only when $\cos(\theta) > 0.95, |x| < 1, |\dot{x}| < 1$ and $|\dot{\theta}| < 1$. 
6 Conclusion

In this paper, we present a parameterized indexed value function which can be learned by a distributional version of TD. After proving its efficiency in the tabular setting, we introduce a computationally lightweight dual-network architecture, Parameterized Indexed Networks (PINs), for deep RL and show its efficacy through numerical experiments. To the best of our knowledge, we lead the first study of index sampling to achieve efficient exploration in the field of RL.

However, several open questions are still left unanswered. It would be worthwhile to explore other designs for uncertainty and prior networks, and experiment with other distributional metrics to see if one can obtain stronger theoretical guarantees and/or better empirical performance. It would be very interesting to combine the ideas of PINs with more advanced neural architectures such as convolutional networks, and evaluate its performance on Atari games with sparse rewards. We leave this and many possible extensions of PINs as our future work, and we hope that this work can serve as a starting point for future studies on the potential of index sampling for efficient RL.

7 Acknowledgement

We thank Benjamin Van Roy, Chengshu Li for the insightful discussions, and Rui Du for comments on the earlier drafts.

References


A Training Algorithm for PINs

Algorithm 3 describes the subroutine agent.learn_from_buffer() called in Algorithm 2 liveJ5, where the agent runs several SGD steps to update the mean and the uncertainty network. For each observed transition, we sample a $U$-dimensional binary mask $M$ with $M[j] \sim \text{Ber}(0.5)$ defining data sharing among different output heads in the uncertainty network. In acting during inference, we first randomly sample an index and sample an output head in the uncertainty network to obtain $Q$-values. The agent then acts greedily with respect to the sampled value function during the next episode of interaction.

We used a batch size of 64, and number of batches $N = 10$ for experiments on Deep-sea, $N = 100$ on Cartpole Swing-up. The target networks are updated every 10 episodes of learning. Adam optimizer with learning rate $10^{-3}$ is used to train both networks.

Algorithm 3: learn_from_buffer

**buffer**: replay buffer of past transitions

**$N$**: number of minibatches

**$b$**: batch size

**$\nu_\phi, m_\omega$**: trainable networks

**$\nu_\phi, m_\omega$**: target networks, parameters are copied from the trainable nets periodically

**$\bar{\nu}_\phi, \bar{m}_\omega$**: untrainable prior networks

**$\beta_1, \beta_2$**: prior scale

**$\gamma$**: discount factor

**$\alpha_\nu, \alpha_m$**: learning rates

**$\sigma$**: noise/perturbation scale

1. for $n$ in $(1, 2, \ldots, N)$ do
2.   $D \leftarrow$ buffer.sample_minibatch ($b$)
3.   Calculate $\bar{a}_i \in \arg\max_{a' \in A} \left( \nu_{\bar{\phi}} + \beta_1 \bar{\nu}_{\bar{\phi}} \right) (s'_i, a')$ for each $(s_i, a_i, r_i, s'_i, M_i) \in D$
4.   Compute gradient for updating the mean network
5.      $\nabla_\phi L_\nu(\phi, \bar{\phi}, \bar{\phi}, D) = \frac{1}{|D|} \sum_{(s_i, a_i, r_i, s'_i, M_i) \in D} \nabla_\phi \left( \left( \nu_{\bar{\phi}} + \beta_1 \bar{\nu}_{\bar{\phi}} \right) (s_i, a_i) - (r_i + \gamma (\nu_{\phi} + \beta_1 \bar{\nu}_{\bar{\phi}})(s'_i, \bar{a}_i)) \right)^2$
6.   Randomly select a head: $u_i \sim \text{Unif}(\mathcal{I}_i)$, where $\mathcal{I}_i = \{j \in \{1, \ldots, U\} \mid M_i[j] = 1\}$ for each $(s_i, a_i, r_i, s'_i, M_i) \in D$
7.   Compute gradient for updating the uncertainty network
8.      $\nabla_\omega L_m(\omega, \bar{\omega}, \bar{\omega}, D) = \frac{1}{|D|} \sum_{(s_i, a_i, r_i, s'_i, M_i) \in D} \nabla_\omega \left( \left( m_{u_i}^{w_i} + \beta_2 \bar{m}_{u_i}^{w_i} \right) (s_i, a_i) - \left( \sigma + \gamma (m_{u_i}^{w_i} + \beta_2 \bar{m}_{u_i}^{w_i})(s'_i, \bar{a}_i) \right)^2 \right)$
9.   Update
10.     $\phi \leftarrow \phi - \alpha_\nu \nabla_\phi L_\nu(\phi, \bar{\phi}, \bar{\phi}, D)$
11.     $\omega \leftarrow \omega - \alpha_m \nabla_\omega L_m(\omega, \bar{\omega}, \bar{\omega}, D)$
12. Adjust learning rate by Adam optimizer

B Comparison on Different Action Selection Schemes

We compare two different schemes in selecting actions for training our PINs:

$$\bar{a} \in \arg\max_{a' \in A} \mathbb{E} \left[ Q_{\delta, \bar{Z}'}(s', a') + Q_{\delta, \bar{Z}'}(s', a') \right] = \arg\max_{a' \in A} \left\{ \nu_{\bar{\phi}}(s', a') + \beta_1 \bar{\nu}_{\bar{\phi}}(s', a') \right\}$$

as in line 3 of our Algorithm 3, where we denote the additive prior distribution as $\bar{Q}_{\delta, \bar{Z}'}(s, a) = \beta_1 \bar{\nu}_{\bar{\phi}}(s, a) + \beta_2 \bar{\nu}_{\bar{\phi}}(s, a) Z'$, and $Z'$ is a standard normal random variable. Note that $\bar{a}$ does not depend on which head is picked in the uncertainty network; thus, we omit the superscript for identifying heads on $m(s, a)$ here.
Alternatively, we can use a sampled value function for action selection:

\[
\hat{a} \in \arg\max_{a' \in A} \left\{ Q_{\theta, z}^u (s', a') + \bar{Q}_{\bar{\theta}, z}^u (s', a') \right\} = \arg\max_{a' \in A} \left\{ \nu_{\phi} (s', a') + m_{\bar{\omega}}^u (s', a') z + \beta_1 \bar{\nu}_{\phi} (s', a') + \beta_2 \bar{m}_{\bar{\omega}}^u (s', a') z \right\},
\]

where \( z \) is sampled from \( N(0, 1) \), and \( u \) denotes a sampled head in the uncertainty network for a given transition \((s, a, r, s')\) as in line 5 of Algorithm 3. We compare \( \bar{a} \) and \( \hat{a} \) for training PINs on Deep-sea with \( N = 20, 25 \). As we see in Figure 4, \( \bar{a} \) gives a more robust performance across different random seeds.

We also point out that many other options are available for action selection besides \( \bar{a}, \hat{a} \). As an example, instead of sampling one single \( z \), we may sample multiple \( K > 1 \) \( z \)'s independently so that there are \( K \) sampled values for each action \( a' \in A \). Then, given the \( K \) sampled values, we can consider picking an optimistic estimate for each action before taking the argmax.

At the same time, we also need to take the learning stability into account, especially for training the mean network. We believe that this could be an interesting direction to explore in the future work of PINs and of index sampling in general.

![Figure 4: Performance comparison between \( \bar{a} \) and \( \hat{a} \) on Deep-sea over 5 seeds](image)

**C Proof of Theorem 1**

For completeness, we first restate the necessary assumptions and theorem here.

**Assumption 1.** The MDP \( \mathcal{M} = (S, A, H, R, P, \rho) \) is finite-horizon time-inhomogeneous such that the state space can be factorized as \( S = S_0 \cup S_1 \cup \cdots \cup S_{H-1} \) and each \( s_h \in S_h \) can be written as a pair \( s_h = (h, x) \), \( x \in X \) for each \( h \in \{0, 1, \ldots, H - 1\} \) with \( |X| < \infty \). Further, we have \( |A| < \infty \) and \( P(s_{h+1} \in S_{h+1} | s_h \in S_h, a_h) = 1 \) for any \( a_h \in A \), \( h < H - 1 \) and the MDP will terminate with probability 1 after taking action \( a_{H-1} \). Finally, the reward is always binary, which means that \( R(s) \in \{0, 1\} \) for any \( s \in S \).
Assumption 2. With the above setup, for each \((h, x, a) \in \{0, 1, \ldots, H - 2\} \times \mathcal{X} \times \mathcal{A}\), the outcome distribution is drawn from a Dirichlet prior \(\mathcal{P}_{h,x,a} \sim \text{Dirichlet} \left( \alpha_{h,x,a}^{0} \right)\) for \(\alpha_{h,x,a}^{0} \in \mathbb{R}^{\mathcal{X}}_{+}\), and each \(\mathcal{P}_{h,x,a}\) is drawn independently. Further, assume that there exists some \(\beta \geq 3\) such that \(1^{\prime} \alpha_{h,x,a}^{0} = \beta\) for all \((h, x, a)\).

Theorem 1. Consider an agent WTD with infinite buffer, greedy actions and an MDP stated in Assumption 1 with planning horizon \(H\). Under Assumption 2 with \(\beta \geq 3\), if Algorithm 1 is applied with \(\sigma^{2} = 3H^{2}\), \(\bar{\theta} = H\) and \(\frac{\sigma^{2}}{\sigma_{0}^{2}} = \beta\), for any number of episodes \(L \in \mathbb{N}\), we have

\[
\text{BayesRegret}(\text{WTD}, L) \leq 5H^{2}\sqrt{\beta |\mathcal{X}| |\mathcal{A}| L} \log(2 |\mathcal{X}| |\mathcal{A}| H L) \log_{+} \left( 1 + \frac{L}{|\mathcal{X}| |\mathcal{A}|} \right)
\]

= \(\tilde{O} \left( H^{2} \sqrt{|\mathcal{X}| |\mathcal{A}| L} \right)\),

where \(\tilde{O} (\cdot)\) ignores all poly-logarithmic terms and \(\log_{+} (x) = \max \{1, \log (x)\}\).

**Stochastic Bellman Operator**

Based on the updating rules for \(n^{l} (h, x, a)\) and \(m^{l} (h, x, a)\) in the main paper, for a general function \(Q \in \mathbb{R}^{|\mathcal{X}| |\mathcal{A}|}\), Algorithm 1 defines a functional operator \(F_{l,h}\):

\[
F_{l,h} Q (x, a) = \sum_{(r, x') \in \mathcal{D}^{l-1}_{h,x,a}} \left( r + \max_{a' \in \mathcal{A}} Q (x', a') \right) + \beta \bar{\theta} + \sqrt{n^{l} (h, x, a) \sigma + \beta \sigma_{0}} \cdot Z_{h,x,a}
\]

where \(Z_{h,x,a} \sim \mathcal{N} (0, 1)\) is independent from \(Q\). Specifically, with this operator, we have \(Q^{l}_{Z,h} = F_{l,h} Q^{l}_{Z',h+1}\), where \(Z\) and \(Z'\) are independent.

Recall the definition of \(Q^{*}_{M,h}\) in equation (1) of the main paper. We denote \(F_{M,h}\) as the true Bellman operator that satisfies \(Q^{*}_{M,h} = F_{M,h} Q^{*}_{M,h+1}\).

To prove Theorem 1, we resort to the following key lemma proved in (Osband et al. 2017).

**Lemma 1.** Let \((Q^{l}_{Z,0}, \ldots, Q^{l}_{Z,H})\) be the sequence of state-action value function learned by Algorithm 1, where \(Q^{l}_{Z,H} = 0\), and \(\pi^{l}\) be the greedy policy based on them. For any episode \(l \in \mathbb{N}\), if we have

\[
E \left[ \max_{a' \in \mathcal{A}} Q^{l}_{Z,0} (x^{l}_{0}, a') \right] \geq E \left[ \max_{a' \in \mathcal{A}} Q^{*}_{M,0} (x^{l}_{0}, a') \right]
\]

then,

\[
E \left[ V^{*}_{M,0} (x^{l}_{0}) - V^{l}_{M,0} (x^{l}_{0}) \right] \leq E \left[ \sum_{h=0}^{H-1} (F_{l,h} Q^{l}_{Z,h+1} - F_{M,h} Q^{l}_{Z,h+1}) (x^{l}_{h}, a^{l}_{h}) \right]
\]

In order to use bound (9), it is necessary to verify that our Algorithm 1 satisfies condition (8). To achieve this, we can resort to the concept of stochastic optimism, which is defined as the following:

**Stochastic Optimism**

**Definition 1.** A random variable \(X\) is stochastically optimistic with respect to another random variable \(Y\), denoted as \(X \geq_{SO} Y\), if \(E [u (X)] \geq E [u (Y)]\) holds for all convex increasing function \(u : \mathbb{R} \rightarrow \mathbb{R}\).

The following two lemmas, which have been proved in (Osband et al. 2017), will be useful in showing that the Algorithm 1 satisfies condition (8).

**Lemma 2.** Suppose \((X_{1}, \ldots, X_{n})\) and \((Y_{1}, \ldots, Y_{n})\) are two collections of independent random variables with \(X_{i} \geq_{SO} Y_{i}\) for each \(i \in \{1, \ldots, n\}\). Then, for any convex increasing function \(f : \mathbb{R}^{n} \rightarrow \mathbb{R}\), we have

\[
f (X_{1}, \ldots, X_{n}) \geq_{SO} f (Y_{1}, \ldots, Y_{n})
\]

**Lemma 3.** Let \(Y = P^{T} V\) for fixed \(V \in \mathbb{R}^{n}\) and \(P \sim \text{Dirichlet} (\alpha)\) with \(\alpha \in \mathbb{R}^{\mathcal{X}}_{+}\) and \(\alpha^{T} 1 \geq 3\). Let \(X \sim \mathcal{N} (\mu, \sigma^{2})\) with \(\mu \geq \frac{\sigma^{2} 1^{T}}{\sigma^{2}}\) and \(\sigma^{2} \geq \frac{3 \text{Span} (V^{T})}{\alpha^{T} 1}\), then \(X \geq_{SO} Y\), where \(\text{Span} (V) = \max_{i} V_{i} - \min_{i} V_{i}\).

Let \(H_{l-1} = \bigcup_{k=1}^{l-1} \{ (h, x^{k}_{h}, a^{k}_{h}, r^{k}_{h+1}) | h < H \}\) be the history up to the end of episode \(l - 1\). Then, by directly combining Lemma 2 and Equation 7, we can have the following result:
Lemma 5. For two random functions $Q_1, Q_2 \in \mathbb{R}^{|\mathcal{X}||A|}$, suppose that condition on history $\mathcal{H}_{t-1}$, the entries of $Q_1(x, a)$ are drawn independently across $x, a$ and independently of $z$ in Equation 7. Then, if

$$Q_1(x, a) | \mathcal{H}_{t-1} \geq_{SO} Q_2(x, a) | \mathcal{H}_{t-1}, \quad \forall (x, a) \in \mathcal{X} \times A,$$

we can have

$$F_{t, h} Q_1(x, a) | \mathcal{H}_{t-1} \geq_{SO} F_{t, h} Q_2(x, a) | \mathcal{H}_{t-1}, \quad \forall (x, a) \in \mathcal{X} \times A, t \in \{0, \ldots, H-1\}, l \in \mathbb{N}$$

For convenience, we will first introduce several new notations.

Definition 2. For any arbitrary state-action value function $Q$, the induced value function $V_Q \in \mathbb{R}^{2|\mathcal{X}|}$ is defined as $V_Q(r, x') := r + \max_{a' \in A} Q(x', a')$ for all $x' \in \mathcal{X}$ and $r \in \{0, 1\}$

With this definition, we notice that

$$V_{Q}(r, x') \geq V_{Q}(r', x') \quad \forall (r, x'), (r', x') \in \mathcal{D}_{h, x, a}.$$
To show the later one, we notice that

\[
\left( \frac{\sqrt{n^l(h, x, a)} \sigma + \beta \sigma_0}{n^l(h, x, a) + \beta} \right)^2 = \frac{n^l(h, x, a) \sigma^2 + \beta^2 \sigma_0^2 + 2 \sqrt{n^l(h, x, a)} \beta \sigma_0}{(n^l(h, x, a) + \beta)^2}
\]

\[
= \frac{n^l(h, x, a) \sigma^2 + \beta \sigma^2 + 2 \sqrt{n^l(h, x, a)} \beta \sigma_0}{(n^l(h, x, a) + \beta)^2}
\]

\[
= \frac{\sigma^2}{n^l(h, x, a) + \beta} + \frac{2 \sqrt{n^l(h, x, a)} \beta \sigma_0}{(n^l(h, x, a) + \beta)^2}
\]

\[
\geq \frac{\sigma^2}{n^l(h, x, a) + \beta} \geq \frac{3 \cdot \text{Span}(V_Q)^2}{n^l(h, x, a) + \beta}
\]

(Since \(\sigma^2 \geq 3 \cdot \text{Span}(V_Q)^2\))

Therefore, the proof is complete.

Notice that in our MDP, by Assumption 1, the planning horizon is \(H\) and reward can only take 0 or 1. Thus, we must have \(\text{Span}(V_Q) \leq H\) and \(\|V_Q\|_{\infty} \leq H\). Then, by combining Lemma 4 and Lemma 5, we can have the following result, which has been proved in (Osband et al. 2017).

**Corollary 1.** If Assumption 2 holds and Algorithm 1 is applied with parameters \(\sigma^2 = 3H^2, \bar{\theta} = H\) and \(\beta = \frac{\sigma^2}{\sigma_0}\), we can then have

\[
Q^l_{Z,0}(x, a) | \mathcal{H}_{t-1} \geq_{SO} Q^*_{M,0}(x, a) | \mathcal{H}_{t-1}
\]

for any history \(\mathcal{H}_{t-1}\) and pair \((x, a)\) \(\in \mathcal{X} \times \mathcal{A}\).

Then, by using Lemma 2, which is applicable since both \(Z_{h, x, a}\) and \(P_{h, x, a}\) are distributed independently across different states and actions, we can have

\[
E \left[ \max_{a' \in \mathcal{A}} Q^l_{Z,0}(x_0^l, a') | \mathcal{H}_{t-1} \right] \geq E \left[ \max_{a' \in \mathcal{A}} Q^*_{M,0}(x_0^l, a') | \mathcal{H}_{t-1} \right]
\]

Therefore, condition (8) can be obtained by simply taking expectation over all possible \(\mathcal{H}_{t-1}\).

**Towards Regret Bound**

After verifying condition (8), we can then apply bound (9) in Lemma 1 to our Algorithm 1. For simplicity, let

\[
\sigma^l(h, x, a) = \frac{\sqrt{n^l(h, x, a)} \sigma + \beta \sigma_0}{n^l(h, x, a) + \beta}, \quad \text{and} \quad w^l(h, x, a) = \sigma^l(h, x, a) \cdot Z_{h, x, a} \text{ for } Z_{h, x, a} \sim \mathcal{N}(0, 1)
\]

(12)

The following three lemmas will be useful in our proof.

**Lemma 6.** If \(\beta \geq 2\), then with probability 1, we can have the following results

\[
\sum_{l \leq L, h < H} \sum_{a_h} \frac{1}{n^l(h, x_h, a_h^l) + \beta} \leq H |\mathcal{X}| |\mathcal{A}| \log \left(1 + \frac{L}{|\mathcal{X}| |\mathcal{A}|} \right)
\]

\[
\sum_{l \leq L, h < H} \sum_{a_h} \sqrt{\frac{1}{n^l(h, x_h, a_h^l) + \beta}} \leq 2 \sqrt{H^2 L |\mathcal{X}| |\mathcal{A}|}
\]

**Lemma 7.** For each \(h < H\) and \(l \leq L\), we can have

\[
E \left[ w^l(h, x_h^l, a_h^l) \right] \leq \sqrt{2 \log (|\mathcal{A}| |\mathcal{X}|)} E \left[ \sigma^l(h, x_h^l, a_h^l)^2 \right]
\]

**Lemma 8.** If Algorithm 1 is applied with parameters \(\sigma^2 = 3H^2, \bar{\theta} = H\) and \(\beta = \frac{\sigma^2}{\sigma_0} \geq 3\), we can have

\[
E \left[ \max_{l \leq L, h < H} \|V^l_{Z, h+1}\|_{\infty} \right] \leq 2H + 2H^2 \sqrt{\log (2 |\mathcal{X}| |\mathcal{A}| H L)}
\]
Among them, Lemma 6 and 7 have been proved in (Osband et al. 2017) and (Russo and Zou 2015), respectively. Lemma 8 will be proved later. Now, we start to prove the regret bound.

**Proof.** By defining $\Delta_l = V_{M,0}^*(x_0^l) - V_{M,0}^*(x_0^l)$, we have

$$\text{BayesRegret (WTD, } L) = \mathbb{E} \left[ \sum_{l=1}^{L} \Delta_l \right] \leq \mathbb{E} \left[ \sum_{l=1}^{L} \sum_{h=0}^{H-1} (F_{l,h} Q_{Z,h+1}^l - F_{M,h} Q_{Z,h+1}^l) (x_h^l, a_h^l) \right]$$

Then, we can see that

$$\mathbb{E} [\Delta_l] \leq \mathbb{E} \left[ \sum_{h=0}^{H-1} (F_{l,h} Q_{Z,h+1}^l - F_{M,h} Q_{Z,h+1}^l) (x_h^l, a_h^l) \right] = \mathbb{E} \left[ \sum_{h=0}^{H-1} \left( (F_{l,h} Q_{Z,h+1}^l) (x_h^l, a_h^l) - \mathbb{E} [F_{M,h} Q_{Z,h+1}^l (x_h^l, a_h^l) \mid H_{l-1}, x_0^l, a_0^l, \ldots, x_h^l, a_h^l] \right) \right]$$

To further simplify, recall that $P_{h,x,a} | H_{l-1} \sim \text{Dirichlet} \left( \alpha_{h,x,a} \right)$, where

$$\alpha_{h,x,a} = \alpha_{h,x,a}^0 + n^l (h, x, a) \hat{P}_{h,x,a}^l$$

Notice that the value of $n^l (h, x, a)$ and $\hat{P}_{h,x,a}^l$ will not change until we observe the transition outcome of $x_h^l$ and $a_h^l$. Therefore, we have $P_{h,x,a} | H_{l-1}, x_0^l, a_0^l, \ldots, x_h^l, a_h^l \sim \text{Dirichlet} \left( \alpha_{h,x,a}^l \right)$. Thus, for a fixed state-action value function $Q \in \mathbb{R}^{[X] \times |A|}$, we can have

$$\mathbb{E} \left[ F_{M,h} Q (x_h^l, a_h^l) \mid H_{l-1}, x_0^l, a_0^l, \ldots, x_h^l, a_h^l \right] = V_Q^T \mathbb{E} \left[ P_{h,x,a} | H_{l-1}, x_0^l, a_0^l, \ldots, x_h^l, a_h^l \right]$$

(Basicaly equation (10))

$$\mathbb{E} \left[ F_{M,h} Q (x_h^l, a_h^l) - \mathbb{E} [F_{M,h} Q (x_h^l, a_h^l) \mid H_{l-1}, x_0^l, a_0^l, \ldots, x_h^l, a_h^l] \right] = \beta \hat{\theta} + V_Q^T \alpha_{h,x,a}^0 + \frac{n^l (h, x_h^l, a_h^l) + \beta}{n^l (h, x_h^l, a_h^l) + \beta} + w^l (h, x_h^l, a_h^l)$$

(Since $1^T \alpha_{h,x,a}^l = \beta$ by Assumption 2)

Notice that condition on $H_{l-1}$, $F_{M,h}$ and $Q_{Z,h+1}$ are independent. Therefore, we can have

$$\mathbb{E} [\Delta_l] \leq \mathbb{E} \left[ \sum_{h=0}^{H-1} \frac{\beta \hat{\theta} + \|V_Q^l\|_\infty}{n^l (h, x_h^l, a_h^l) + \beta} + w^l (h, x_h^l, a_h^l) \right]$$

As a result, we have

$$\mathbb{E} \left[ \sum_{l=1}^{L} \Delta_l \right] \leq \mathbb{E} \left[ \beta \left( \hat{\theta} + \max_{l \leq L, h < H} \|V_{Q_{Z,h+1}}^l\|_\infty \right) \sum_{l \leq L, h < H} \frac{1}{n^l (h, x_h^l, a_h^l) + \beta} \right] + \mathbb{E} \left[ \sum_{l \leq L, h < H} w^l (h, x_h^l, a_h^l) \right]$$

These two terms can be bounded separately. To bound the first term, we have

$$\mathbb{E} \left[ \beta \left( \hat{\theta} + \max_{l \leq L, h < H} \|V_{Q_{Z,h+1}}^l\|_\infty \right) \sum_{h < H, l \leq L} \frac{1}{n^l (h, x_h^l, a_h^l) + \beta} \right] \leq \beta \left( H + \mathbb{E} \left[ \max_{l \leq L, h < H} \|V_{Q_{Z,h+1}}^l\|_\infty \right] H |X| |A| \log \left( 1 + \frac{L}{|X| |A|} \right) \right)$$

(By Lemma 6 and $\hat{\theta} = H$)
When we have \( \sum_{l=1}^{n} \sigma^L (\ell, x^l, a^l_h) \leq \sqrt{2n} \sum_{i=1}^{n} a_i \) (By equation (13))

\[
\sum_{h<H, l\leq L} \sqrt{E \left[ \sigma^L (h, x^l_h, a^l_h)^2 \right]} \leq \sqrt{HL} \sqrt{E \left[ \sum_{h<H, l\leq L} \sigma^L (h, x^l_h, a^l_h)^2 \right]}
\]

\[
\leq \sqrt{HL} \sqrt{E \left[ \sum_{h<H, l\leq L} \frac{2\sigma^2}{n^l (h, x^l_h, a^l_h) + \beta} \right]}
\]

\[
\leq \sqrt{6H^4L |\mathcal{X}| |\mathcal{A}| \log \left( 1 + \frac{L}{|\mathcal{X}| |\mathcal{A}|} \right)}
\]

Therefore, we have the regret bound

\[
E \left[ \sum_{l=1}^{L} \Delta_l \right] \leq 5\beta H^3 |\mathcal{X}| |\mathcal{A}| \sqrt{\log_+ \left( 2 |\mathcal{X}| |\mathcal{A}| HL \right) \log \left( 1 + \frac{L}{|\mathcal{X}| |\mathcal{A}|} \right) + 2H^2 \sqrt{3L |\mathcal{X}| |\mathcal{A}| \log (|\mathcal{X}| |\mathcal{A}|) \log \left( 1 + \frac{L}{|\mathcal{X}| |\mathcal{A}|} \right)}
\]

We then consider two separate cases.

When \( L \geq 25\beta H^2 |\mathcal{X}| |\mathcal{A}| \), which means \( 5H \sqrt{\beta |\mathcal{X}| |\mathcal{A}|} \leq \sqrt{L} \), we can have \( 5\beta H^3 |\mathcal{X}| |\mathcal{A}| \leq H^2 \sqrt{\beta L |\mathcal{X}| |\mathcal{A}|} \). Therefore, we have

\[
E \left[ \sum_{l=1}^{L} \ell_l \right] \leq \left( \sqrt{\beta + 2\sqrt{3}} \right) H^2 \sqrt{L |\mathcal{X}| |\mathcal{A}| \log_+ \left( 2 |\mathcal{X}| |\mathcal{A}| HL \right) \log_+ \left( 1 + \frac{L}{|\mathcal{X}| |\mathcal{A}|} \right) \leq 5H^2 \sqrt{\beta L |\mathcal{X}| |\mathcal{A}| \log_+ \left( 2 |\mathcal{X}| |\mathcal{A}| HL \right) \log_+ \left( 1 + \frac{L}{|\mathcal{X}| |\mathcal{A}|} \right)
\]

When \( L \leq 25\beta H^2 |\mathcal{X}| |\mathcal{A}| \), we can just use the naive bound

\[
E \left[ \sum_{l=1}^{L} \Delta_l \right] \leq HL \leq H \sqrt{L \sqrt{25\beta H^2 |\mathcal{X}| |\mathcal{A}|} = 5H^2 \sqrt{\beta L |\mathcal{X}| |\mathcal{A}|}
\]
In either case, we can conclude that

\[
\text{BayesRegret (WTD, } L) = \mathbb{E} \left[ \sum_{l=1}^{L} \Delta_l \right] \leq 5H^2 \sqrt{\beta L |X| |A| \log_+ \left( 2 |X| |A| H L \right) \log_+ \left( 1 + \frac{L}{|X| |A|} \right)}
\]

\[
= \tilde{O} \left( H^2 \sqrt{L |X| |A|} \right),
\]

which completes the proof.

\[
\square
\]

**Proof of Lemma 8**

**Lemma 8.** If Algorithm 1 is applied with parameters \( \sigma^2 = 3H^2 \), \( \bar{\theta} = H \) and \( \frac{\sigma^2}{\bar{\theta}^2} = \beta \geq 3 \), we can have

\[
\mathbb{E} \left[ \max_{l \leq L, h \leq H} \left\| V_{Q_{Z,l+1}}^l \right\|_\infty \right] \leq 2H + 2H^2 \sqrt{\log (2 |X| |A| H L)}
\]

**Proof.** Define \( w_{\text{max}} = \max_{h \leq H, l \leq L, a \in A, x \in X} \left| w^l (h, x, a) \right| \). Then, for any fixed state-action value function \( Q \in \mathbb{R}^{|X||A|} \), by equation (11), we can have

\[
\left\| F_{l,h} Q \right\|_\infty \leq \max \left\{ \bar{\theta}, \left\| V_Q \right\|_\infty \right\} + \max_{x \in X, a \in A} w^l (h, x, a)
\]

\[
\leq \max \left\{ \bar{\theta}, \left\| Q \right\|_\infty \right\} + 1 + w_{\text{max}} \quad \text{(By Assumption 1, we have reward } r \in \{0, 1\})
\]

Therefore, we can have

\[
\left\| Q_{Z,H-1}^l \right\|_\infty = \left\| F_{l,H-1} 0 \right\|_\infty \leq \bar{\theta} + 1 + w_{\text{max}}
\]

\[
\implies \left\| Q_{Z,H-2}^l \right\|_\infty \leq \max \left\{ \bar{\theta}, \left\| Q_{Z,H-1}^l \right\|_\infty \right\} + 1 + w_{\text{max}} \leq \bar{\theta} + 2 (1 + w_{\text{max}})
\]

By keeping this procedure and applying that \( \bar{\theta} = H \), we can have

\[
\mathbb{E} \left[ \max_{l \leq L, h \leq H} \left\| V_{Q_{Z,l+1}}^l \right\|_\infty \right] \leq H + H \left( 1 + \mathbb{E} |w_{\text{max}}| \right) = 2H + HE |w_{\text{max}}|
\]

For the last term above, we can have

\[
\mathbb{E} |w_{\text{max}}| = \mathbb{E} \left[ \max_{h \leq H, l \leq L, x \in X} \left| \frac{\sigma^l (h, x, a)}{\sigma^l (h, x, a)} \cdot w^l (h, x, a) \right| \right]
\]

\[
\leq \mathbb{E} \left[ \max_{h \leq H, l \leq L, x \in X} \left| w^l (h, x, a) \right| \right] \mathbb{E} \left[ \max_{h \leq H, l \leq L, x \in X} \left| \frac{\sigma^l (h, x, a)}{\sigma^l (h, x, a)} \right| \right]
\]

\[
\overset{(a)}{\leq} \frac{4\sigma^2}{\beta} \log (2 |X| |A| H L)
\]

\[
\leq 2H \sqrt{\log (2 |X| |A| H L)} \quad \text{(Since } \beta \geq 3 \text{ and } \sigma^2 = 3H^2\text{)}
\]

The inequality (a) above is true for two reasons. First, by equation (14), we can have \( \sigma^l (h, x, a) \leq \sqrt{\frac{2\sigma^2}{\beta}} \) for any \( l \in \mathbb{N}, h \in \{0, \ldots, H - 1\} \) and \( (x, a) \in X \times A \). Second, for independent \( X_1, \ldots, X_n \sim \mathcal{N} (0, 1) \), we can have \( \mathbb{E} |X_i| \leq \sqrt{2 \log (2n)} \). Therefore, we can have the following bound

\[
\mathbb{E} \left[ \max_{l \leq L, h \leq H} \left\| V_{Q_{Z,l+1}}^l \right\|_\infty \right] \leq 2H + HE |w_{\text{max}}| \leq 2H + 2H^2 \sqrt{\log (2 |X| |A| H L)}
\]

\[
\square
\]