Parameterized Indexed Value Function for Efficient Exploration in Reinforcement Learning

Tian Tan,[1*†] **Zhihan Xiong,**[2*] **Vikranth R. Dwaracherla**[3]
[1]Department of Civil and Environmental Engineering, Stanford University
[2]Department of Statistics, Stanford University
[3]Department of Electrical Engineering, Stanford University
{tiantan, 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 (**?**), (**?**), (**?**). 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 (**?**), (**?**), (**?**), 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 (**?**), (**?**) and variational inference (**?**), (**?**), these methods are typically not suitable for sequential decision making (**?**). Instead, ensemble sampling (**?**), (**?**), (**?**), 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 (**?**) 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 z, 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, $z \sim \mathcal{N}(0,1)$, $v(x,a)$ 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 $\widetilde{O}\left(H^2\sqrt{|\mathcal{X}|\,|\mathcal{A}|\,L}\right)$ in finite-horizon episodic MDPs, where H is the episode length, L is the number of episodes, and $|\mathcal{X}|$ and $|\mathcal{A}|$ are the cardinality of the state and action spaces. Then, we propose a dual-network architecture *Parameterized Indexed Networks* (PINs) to generalize with complex models in deep reinforcement 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* (**?**). Our open-source implementation of PINs can be found at https://github.com/tiantan522/PINs.

[*]Equal contribution.
[†]Corresponding author.

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 (**?**). 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 (**?**) mainly as a heuristic method. The theoretical aspects of PSRL were poorly understood until very recently. Osband et al. (**?**) established an $\widetilde{O}\left(H^{1.5}\sqrt{|\mathcal{X}|\,|\mathcal{A}|\,L}\right)$ Bayesian regret bound for PSRL in finite-horizon episodic MDPs. Note that this result improves upon the best previous Bayesian regret bound of $\widetilde{O}\left(H^{1.5}\,|\mathcal{X}|\,\sqrt{|\mathcal{A}|\,L}\right)$ 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 (**?**) 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, (**?**) 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 restrains BootDQN from performing well in environments with delayed rewards. A simple remedy was proposed recently in (**?**) 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 (**?**) 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 algorithm for learning a parameterized indexed value function and the corresponding Bayesian regret analysis. The formulation inspired us to design PINs which combines reinforcement learning with deep learning.

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 (**?**), which is stated as the following assumption.

Assumption 1. *The MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, H, R, \mathbb{P}, \rho)$ is finite-horizon time-inhomogeneous such that the state space can be factorized as $\mathcal{S} = \mathcal{S}_0 \cup \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_{H-1}$ and each $s_h \in \mathcal{S}_h$ 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 $|\mathcal{A}| < \infty$ and $\mathbb{P}(s_{h+1} \in \mathcal{S}_{h+1} \mid s_h \in \mathcal{S}_h, a_h) = 1$ for any $a_h \in \mathcal{A}$, $h < H-1$ 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 $s \in \mathcal{S}, a \in \mathcal{A}$.*

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

The Dirichilet 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 (**?**).

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

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

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 $Q^l_{Z,h}(x,a) = \nu^l(h,x,a) + m^l(h,x,a)\,Z_{h,x,a}$, 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 lowercase subscript $Q^l_{z,h}(x,a) = \nu^l(h,x,a) + m^l(h,x,a) z_{h,x,a}$ as a sampled value function, where $z_{h,x,a}$ is a sample from a standard Gaussian. Further, let $\overline{Q} \sim \mathcal{N}\left(\bar{\theta}, \sigma_0^2\right)$ be a prior of the Q function for all (h,x,a). Denote $\mathcal{D}^{l-1}_{h,x,a} = \left\{(r^j_{h+1}, x^j_{h+1}) \mid x^j_h = x, a^j_h = a, j < l\right\}$, which includes all observed transition outcomes for (h,x,a) up to the start of episode l, and $n^l(h,x,a) = \left|\mathcal{D}^{l-1}_{h,x,a}\right|$.

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

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

where $\tilde{z}_{h+1,x',a'} \sim \mathcal{N}(0,1)$ is independently sampled for each $(h+1,x',a')$, σ 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 (**?**) except that Z is a random variable and the added noise σ is divided by $\sqrt{n^l(h,x,a)}$. This decay of noise σ 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:

$$\mathcal{L}_p\left(\nu^l(h,x,a), m^l(h,x,a)\right) = \sum_{(r,x') \in \mathcal{D}^{l-1}_{h,x,a}} W_p\left(Q^l_{Z,h}(x,a),\ y^l(h,x,a,r,x')\right)^2, \quad (2)$$

where $Q^l_{Z,h}(x,a) = \nu^l(h,x,a) + m^l(h,x,a) Z_{h,x,a}$. Here, $W_p(\cdot,\cdot)$ is the p-Wasserstein distance between two distributions. Similarly, we can define a regularization to the prior as $\psi_p(\nu,m) = \beta W_p\left(\nu + mZ,\ \overline{Q}\right)^2$, where $\beta = \frac{\sigma^2}{\sigma_0^2}$. Therefore, when updating the parameters (ν,m), one need to solve $\min_{(\nu,m)}\{\mathcal{L}_p + \psi_p\}$.

For two Gaussian random variables $X \sim \mathcal{N}\left(\mu_X, \sigma_X^2\right)$ and $Y \sim \mathcal{N}\left(\mu_Y, \sigma_Y^2\right)$ 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 $\mathcal{L}_2 + \psi_2$ exactly by

$$\nu^l(h,x,a) = \frac{\sum_{(r,x') \in \mathcal{D}^{l-1}_{h,x,a}} \left(r + \max_{a' \in \mathcal{A}} Q^l_{\tilde{z},h+1}(x',a')\right) + \beta\bar{\theta}}{n^l(h,x,a) + \beta} \quad (3)$$

$$m^l(h,x,a) = \frac{\sqrt{n^l(h,x,a)}\sigma + \beta\sigma_0}{n^l(h,x,a) + \beta} \quad (4)$$

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

Algorithm 1: Tabular Wasserstein TD (WTD)

For all $l \in \{0,1,\ldots,L\}$, $h \in \{0,\ldots,H-1\}$, $x \in \mathcal{X}$ and $a \in \mathcal{A}$, initialize $\mathcal{D}^l_{h,x,a} = \emptyset$ and $\nu^l(H,x,a) = m^l(H,x,a) = 0$
for $l = 1,\ldots,L$ **do**
  Sample $\tilde{z}_{h,x,a} \sim \mathcal{N}(0,1)$ for all $x \in \mathcal{X}$, $a \in \mathcal{A}$ and $h \in \{0,\ldots,H-1\}$
  for $h = H-1,\ldots,0$ **do**
    for $x \in \mathcal{X}$, $a \in \mathcal{A}$ **do**
      Update $\nu^l(h,x,a)$ and $m^l(h,x,a)$ using equations (3) and (4)
    end
  end
  Observe x_0
  $\mathcal{D}^l_{h,x,a} \leftarrow \mathcal{D}^{l-1}_{h,x,a}$ for all $x \in \mathcal{X}$, $a \in \mathcal{A}$ and $h \in \{0,\ldots,H-1\}$
  for $h = 0,\ldots,H-1$ **do**
    Take action $a_h = \operatorname{argmax}_{a \in \mathcal{A}} Q^l_{\tilde{z},h}(x_h,a)$ and observe (r_{h+1}, x_{h+1})
    $\mathcal{D}^l_{h,x_h,a_h} \leftarrow \mathcal{D}^l_{h,x_h,a_h} \cup \{(r_{h+1}, x_{h+1})\}$
  end
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 $\sigma^2 = 3H^2$, $\bar{\theta} = H$ and $\frac{\sigma^2}{\sigma_0^2} = \beta$, for any number of episodes $L \in \mathbb{N}$, we have*

$$\begin{aligned}&\text{BayesRegret}(\mathsf{WTD}, L)\\ &\leq 5H^2\sqrt{\beta\,|\mathcal{X}|\,|\mathcal{A}|\,L\log_+(2\,|\mathcal{X}|\,|\mathcal{A}|\,HL)}\log_+\left(1 + \frac{L}{|\mathcal{X}|\,|\mathcal{A}|}\right)\\ &= \widetilde{O}\left(H^2\sqrt{|\mathcal{X}|\,|\mathcal{A}|\,L}\right)\end{aligned}$$

where $\widetilde{O}(\cdot)$ 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 RLSVI in (**?**) 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 $Q \in \mathbb{R}^{|\mathcal{X}||\mathcal{A}|}$, Algorithm 1 defines a functional operator $F_{l,h}$:

$$F_{l,h}Q(x,a) = \frac{\sum_{(r,x')\in\mathcal{D}_{h,x,a}^{l-1}}\left(r+\max_{a'\in\mathcal{A}}Q(x',a')\right)+\beta\bar{\theta}}{n^l(h,x,a)+\beta} + \frac{\sqrt{n^l(h,x,a)}\sigma+\beta\sigma_0}{n^l(h,x,a)+\beta}\cdot Z_{h,x,a}$$

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

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

To prove Theorem 1, we resort to the following key lemma proved in (**?**).

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

$$\mathbb{E}\left[\max_{a'\in\mathcal{A}}Q_{Z,0}^l\left(x_0^l,a'\right)\right] \geq \mathbb{E}\left[\max_{a'\in\mathcal{A}}Q_{\mathcal{M},0}^*\left(x_0^l,a'\right)\right] \quad (5)$$

then, by defining $\Delta_l = V_{\mathcal{M},0}^*\left(x_0^l\right) - V_{\mathcal{M},0}^{\pi^l}\left(x_0^l\right)$, we can have

$$\mathbb{E}\left[\Delta_l\right] \leq \mathbb{E}\left[\sum_{h=0}^{H-1}\left(\left(F_{l,h}-F_{\mathcal{M},h}\right)Q_{Z,h+1}^l\right)\left(x_h^l,a_h^l\right)\right] \quad (6)$$

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 $\mathbb{E}\left[u(X)\right] \geq \mathbb{E}\left[u(Y)\right]$ holds for all convex increasing function $u : \mathbb{R} \mapsto \mathbb{R}$.

If Assumption 2 holds, by additionally assuming some technical conditions on parameters σ^2, $\bar{\theta}$ and β, it is possible to show that

$$Q_{Z,0}^l(x,a) \mid \mathcal{H}_{l-1} \geq_{SO} Q_{\mathcal{M},0}^*(x,a) \mid \mathcal{H}_{l-1}$$

for any history $\mathcal{H}_{l-1} = \bigcup_{k=1}^{l-1}\left\{\left(h,x_h^k,a_h^k,r_{h+1}^k\right) \mid h < H\right\}$ and $(x,a) \in \mathcal{X}\times\mathcal{A}$.

By using Lemma 2 in (**?**) on preservation of optimism, we can then further show that

$$\mathbb{E}_{\mathcal{H}_{l-1}}\left[\max_{a'\in\mathcal{A}}Q_{Z,0}^l\left(x_0^l,a'\right)\right] \geq \mathbb{E}_{\mathcal{H}_{l-1}}\left[\max_{a'\in\mathcal{A}}Q_{\mathcal{M},0}^*\left(x_0^l,a'\right)\right]$$

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

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

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

Finally, after some algebraic manipulations, it is possible to bound this term by $\widetilde{O}\left(H^2\sqrt{L\left|\mathcal{X}\right|\left|\mathcal{A}\right|}\right)$, 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 σ^2, $\bar{\theta}$ 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 $\tilde{z}$'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 (**?**), 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* (**?**) for balancing between exploration and exploitation, and it amounts to sampling one single z^l 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^l that is fixed within an episode, applying the updating rules for action-value function (3) and (4) in the tabular case then gives

$$\begin{aligned}Q_{z^l,h}^l(x,a) &= \frac{\sum_{(r,x')\in\mathcal{D}_{h,x,a}^{l-1}}\left(r+Q_{z^l,h+1}^l(x',\tilde{a})\right)+\beta\bar{\theta}}{n^l(h,x,a)+\beta} + \frac{\sqrt{n^l(h,x,a)}\sigma+\beta\sigma_0}{n^l(h,x,a)+\beta}z^l \\ &= \frac{\sum_{(r,x')\in\mathcal{D}_{h,x,a}^{l-1}}\left(r+\nu^l(h+1,x',\tilde{a})\right)+\beta\bar{\theta}}{n^l(h,x,a)+\beta} \\ &+ \frac{\sqrt{n^l(h,x,a)}\sigma+\beta\sigma_0+\sum_{(r,x')\in\mathcal{D}_{h,x,a}^{l-1}}m^l(h+1,x',\tilde{a})}{n^l(h,x,a)+\beta}z^l\end{aligned}$$

Algorithm 2: live_IS

Input: agent, environment

for l *in* $(1, 2, \dots)$ **do**
  Sample $z^l \sim \mathcal{N}(0, 1)$
  agent.learn_from_buffer()
  history $\leftarrow$ environment.reset()
  while history *is not* ***terminal*** **do**
    action $\leftarrow$ agent.act(history) $\left(z^l\right)$
    history $\leftarrow$ environment.step(action)
  end
  agent.update_buffer(history)
end

where $\tilde{a} = \text{argmax}_{a' \in \mathcal{A}} Q^l_{z^l, h+1}(x', a')$ 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:

$$\left(\nu^l(h, x, a), m^l(h, x, a)\right) = \min_{\nu, m} \sum_{(r, x') \in \mathcal{D}^{l-1}_{h,x,a}} W_2\left(\nu + mZ, \tilde{y}^l\right)^2 + \psi_2(\nu, m),$$

where

$$\tilde{y}^l(h, x, a, r, x') = r + \frac{\sigma Z}{\sqrt{n^l(h, x, a)}} + Q^l_{Z,h+1}(x', \tilde{a})$$
$$= \underbrace{r + \nu^l(h+1, x', \tilde{a})}_{\text{mean target}} + \underbrace{\left(\frac{\sigma}{\sqrt{n^l(h, x, a)}} + m^l(h+1, x', \tilde{a})\right) Z}_{\text{uncertainty target}}$$

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 (**?**).

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 lookhead 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 ϕ, ω 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 (**?**). Then, in learn_from_buffer() of Algorithm 2, the agent is essentially trying to minimize the following distributional loss:

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

where $\overline{Q}$ is a prior distribution, β is a prior scale which is treated as a hyperparameter, $Z \sim \mathcal{N}(0, 1)$, γ is a discount factor, $\tilde{\mathcal{D}}$ 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 the $\tilde{a}$, obtained by taking argmax on the sampled value function, can sometimes affect learning stability. Therefore, we instead use $\bar{a} \in \text{argmax}_{a' \in \mathcal{A}} \mathbb{E}\left[Q_{\tilde{\theta},Z'}(s', a')\right] = \text{argmax}_{a' \in \mathcal{A}} \nu_{\tilde{\phi}}(s', a')$ 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.

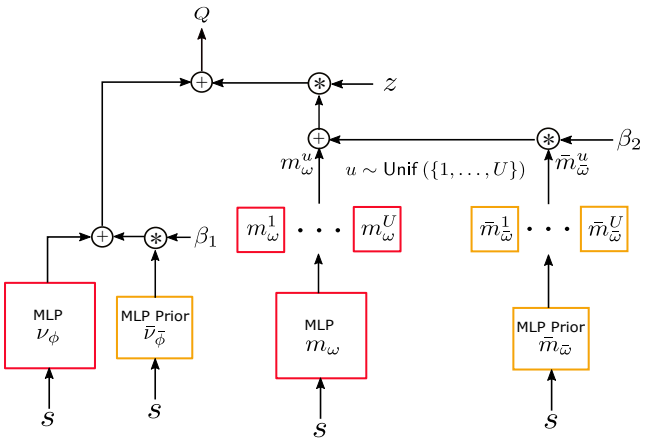

Figure 1: Parameterized Indexed Networks

The overall architecture of PINs is depicted in Figure 1. We incorporate a prior mechanism by following (**?**), where each trainable network is paired with an *additive* prior network. Instead of explicitly regularizing to a prior, i.e. the $\beta W_2\left(Q_{\theta,Z}, \overline{Q}\right)^2$ term in $\mathcal{L}_{\text{dist}}$, we *add* a randomized prior function to the Q function (**?**). 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 (**?**) 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 $u \in \{1, \dots, U\}$ maps the input state s to uncertainty estimates $m^u_\omega(s, a)$ for all $a \in \mathcal{A}$. Despite that all but the output layer are shared among different heads, we promote diversity by (1) applying a bootstrap mask $\text{Ber}(0.5)$

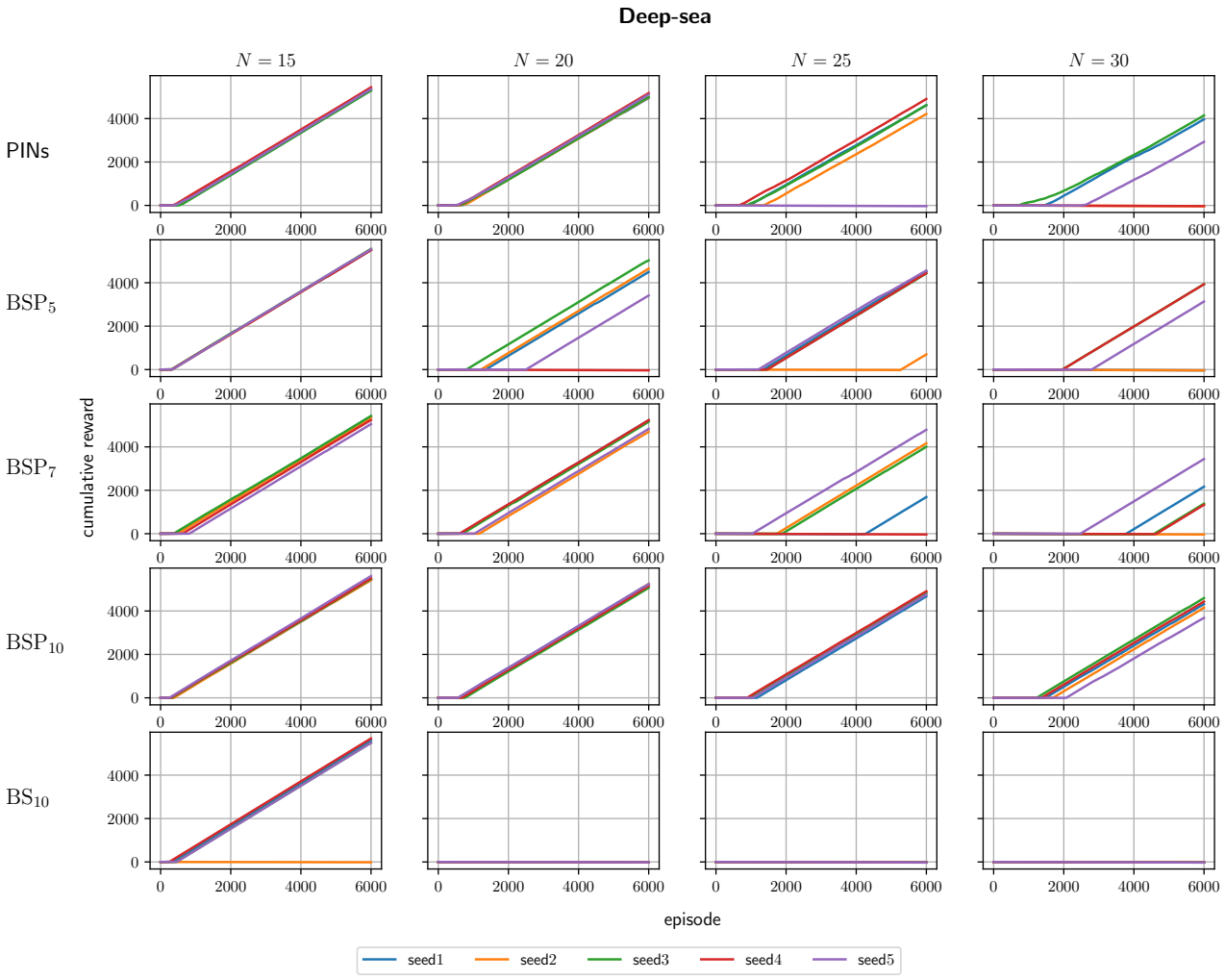

Figure 2: Comparison on cumulative reward with different problem size Ns among PINs, BSP_5, BSP_7, BSP_{10} and BS_{10}

to each output head so that data in buffer is not completely shared among heads (**?**), and (2) each head is paired and trained together with a slightly different prior $\bar{m}^u_{\bar{\omega}}(s,a)$.

For each episode l, we first sample an index $z^l \sim \mathcal{N}(0,1)$ and an output head $u \sim \text{Unif}(\{1,\dots,U\})$. The agent then acts greedily with respect to the sampled value function with *additive* priors $Q^u_{z^l} = \nu + m^u z^l + \beta_1 \bar{\nu} + \beta_2 \bar{m}^u z^l$ for consistent exploration in episode l, where β_1, β_2 are scaling hyperparameters for the mean prior and the uncertainty prior, respectively. Here the additive prior distribution can be seen as $\overline{Q}^u_{z^l} = \beta_1 \bar{\nu} + \beta_2 \bar{m}^u z^l$. 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* (**?**), and compare it with the state-of-the-art ensemble sampling methods: the bootstrapped DQN with additive prior networks (BSP_K) (**?**) and the bootstrapped DQN without prior mechanism (BS_K) (**?**), where K denotes the number of networks in the ensemble.

Deep-sea

Deep-sea is a family of grid-like deterministic environments (**?**), which are indexed by a problem size $N \in \mathbb{N}$, with $N \times N$ cells as states, and sampled action mask $M_{ij} \sim \text{Ber}(0.5)$, $i,j \in \{1,\dots,N\}$. Action set $\mathcal{A} = \{0,1\}$, 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 cell immediately to the left (right) and below. Thus, each episode lasts exactly N time steps and the agent can never revisit the same state within an episode. No cost or reward is associated with action "left". However, taking action "right" results in a cost of $0.01/N$ in cells along the main diagonal except the lower-right-most cell where a reward of 1 is given for taking action "right". Therefore, the optimal policy is picking action "right" at

each step giving an episodic reward of 0.99. All other policies generate zero or negative rewards. The usual dithering methods will need $\Omega\left(2^N\right)$ 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 (**?**), each single network in the ensemble contains 50 hidden units, and we set prior scale $\beta = 10$ for BSP as recommended in (**?**). 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 value 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 now present our results on a classic benchmark problem: *Cartpole Swing-up* (**?**), (**?**), 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 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.

[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 $|\dot{\theta}| < 1$.

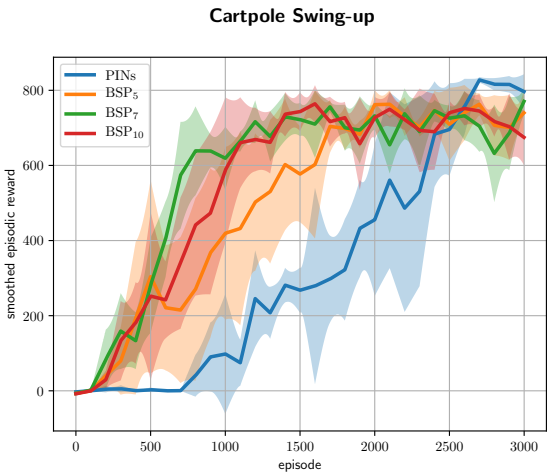

Figure 3: Comparison on smoothed episodic reward: each point is the maximum episodic reward within the most recent 100 episodes. We plot the average performance over 5 random seeds for each method and the shaded area represents $+/-$ standard deviation.

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 (**?**). 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

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 very7 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

7 Acknowledgement

We thank Benjamin Van Roy, Chengshu Li for the insightful