

---

# Scalable and Parameter-Free Optimal Regret Reinforcement Learning

---

## Abstract

## 1 BACKGROUND

A reinforcement learning environment is modelled as a Markov decision process (MDP)  $M = \langle \mathcal{S}, \mathcal{A}, r, P, P_0, \gamma \rangle$ , where  $\mathcal{S}$  is the state space and  $\mathcal{A}$  is the set of available actions. At time  $t = 0$  a state  $s_0$  is sampled from the distribution  $P_0(\cdot)$ . At each timestep an action  $a_t \sim \pi(\cdot|s_t)$  is selected and the agent transitions to a new state  $s_{t+1} \sim P(\cdot|s_t, a_t)$ . A scalar reward  $r_{t+1} = r(\cdot|s_{t+1}, s_t, a_t)$  is observed. As the agent and environment interact in a sequence of time steps, a history of observations  $\mathcal{H}_t = (s_0, a_0, s_1, r_1, a_1, s_2, r_2, \dots, s_t, r_t)$  is collected. The goal is to find a policy  $\pi^*$ , such that sampling actions  $a \sim \pi^*(\cdot|s)$  maximizes the expected accumulated and discounted future reward,  $J^\pi := \mathbb{E}_\pi [\sum_{t=1}^{\infty} \gamma^t r_t]$ . An efficient agent must be able to learn from the data it collects, but since the data is dependent on the policy, it must also prioritize to explore states and actions that the agent can learn a lot from.

The efficiency of exploration can be measured in regret. The regret of a policy is the difference in the expected reward obtained by following that policy, and the expected reward for an optimal policy. An learning algorithm's efficiency in exploration can be measured by its cumulative regret over time. One of the simplest reinforcement learning problems is the multi-arm bandit problem. This is an MDP with no state and no transition probabilities. One exploration strategy employed in most Q-learning algorithms to date,  $\epsilon$ -greedy is provably inefficient, and has a regret bound that grows linearly with time. There are several optimal algorithms for this problem, but perhaps the simplest one is called Thompson sampling. Thompson sampling approximates a posterior distribution of the expected reward for each action. The next action is decided by sampling rewards for each action from the posterior distribution, and selecting the action that gave the highest reward. Bayesian methods have also been shown to behave efficiently with respect to

cumulative regret Osband et al. [2016] on general MDPs.

The  $Q$ -function,  $Q^\pi(s, a)$ , is defined as the expected reward of taking action  $a$  in state  $s$  and then following policy  $\pi$  thereafter:  $Q^\pi := \mathbb{E}_\pi [\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a]$ . The Bellman operator on  $Q^\pi$  is defined as  $\mathcal{B}[Q^\pi](s_t, a_t) := \mathbb{E}_{P(s_{t+1}|s_t, a_t)\pi(a_{t+1}|s_t)} [r(s_{t+1}, s_t, a) + \gamma Q^\pi(s_{t+1}, a_{t+1})]$ . With the  $Q$ -function we can define a policy that always picks the action with the highest  $Q$ -value in any state. We can't find  $Q$  exactly, but with a good approximation this policy,  $\pi(a|s) = \arg \max_a Q(s, a)$ , should hopefully be close to optimal. In deep RL, we model  $Q$  with a neural network  $\hat{Q}_\omega$ . One way to facilitate exploration in this policy is to introduce uncertainty in the  $Q$ -value. Fortunato et al. [2019] introduced NoisyNet for exploration. These are networks with stochastic weights, where each weight  $w_{ij}^{(l)}$  has an added perturbation sampled from a noise distribution with standard deviation  $\sigma_{ij}^{(l)}$ . After initializing  $\omega$  and  $\sigma$  such that the network has sufficient stochasticity for exploration, both parameters are learned using standard backpropagation. The approach is similar to variational inference schemes such as Bayes by backprop [Blundell et al., 2015] where the weights of a neural network model are assumed to be normally distributed with mean  $\mu_{ij}^{(l)}$  and standard deviation  $\sigma_{ij}^{(l)}$ . The objective of Bayes by backprop, however, is to approximate the posterior distribution  $p(\omega|\mathcal{D})$  in a task with a fixed dataset  $\mathcal{D}$ . The parameter distribution in NoisyNet does not necessarily converge to an approximate posterior. This means that it does not have the same guarantee on total regret as methods that approximate a posterior over the value functions [Osband et al., 2016]. They do, however, have experimental results which shows that the function does not always converge to a deterministic solution, but it is unclear why the network learns to introduce more noise into the network parameters.

Fortunato et al. [2019] apply NoisyNet to three reinforcement learning algorithms, DQN, Dueling DQN, and A3C, and show improved performance on all of them. Later NoisyNet was used in the Rainbow algorithm [Hessel et al.,

2017], a combination of six extensions to the DQN algorithms [Fortunato et al., 2019, Bellemare et al., 2017, Wang et al., 2016, van Hasselt et al., 2015, Schaul et al., 2016, Mnih et al., 2016], that shows state-of-the-art performance across 57 Atari games.

In NoisyNet [Fortunato et al., 2019], the Q-function space can be seen as a stochastic process on the probability space  $(\Omega, \mathcal{F}, P)$ . The stochastic process can simply be written as  $\{Q(s, a) : (s, a) \in \mathcal{S} \times \mathcal{A}\}$ . For any  $\omega \in \Omega$ ,  $Q(\cdot, \cdot, \omega)$  is a sample function mapping  $\mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ . To simplify notation we will denote sample functions as  $f$ . NoisyNet, specifically, is a stochastic process parameterized by its weights and their standard deviation. We will define this parameterized function as  $g_\phi$ , and their sampling distribution as  $q_\phi \in \mathcal{Q}$ . Given a noise vector  $\xi$ , we have  $f(s, a) = g_\phi(a, s, \xi)$ .

Functional variational Bayesian neural networks [Sun et al., 2019] is a variational inference method for neural networks that approximates the posterior distribution in function space. Sun et al. [2019] show that for two stochastic processes  $P$  and  $Q$ :

$$\text{KL}[P\|Q] = \sup_{n \in \mathbb{N}, \mathbf{X} \in \mathcal{X}^n} \text{KL}[P_{\mathbf{X}}\|Q_{\mathbf{X}}], \quad (1)$$

where  $\mathbf{X} \in \mathcal{X}^n$  is a finite measurement set and  $P_{\mathbf{X}}, Q_{\mathbf{X}}$  are the marginal distributions at  $\mathbf{X}$ . They further show that if  $\mathbf{f}^{\mathbf{X}}$  are the function values at points  $\mathbf{X}$ , then

$$\begin{aligned} \nabla_\phi \text{KL}[q_\phi(\mathbf{f}^{\mathbf{X}})\|p(\mathbf{f}^{\mathbf{X}})] \\ = \mathbb{E} [\nabla_\phi \mathbf{f}^{\mathbf{X}} (\nabla_{\mathbf{f}} \log q(\mathbf{f}^{\mathbf{X}}) - \nabla_{\mathbf{f}} \log p(\mathbf{f}^{\mathbf{X}}))]. \end{aligned} \quad (2)$$

The difficult part in (2) is to estimate  $\nabla_{\mathbf{f}} \log q(\mathbf{f}^{\mathbf{X}})$  and  $\nabla_{\mathbf{f}} \log p(\mathbf{f}^{\mathbf{X}})$ .  $\nabla_{\mathbf{f}} \log q(\mathbf{f}^{\mathbf{X}})$  is likely intractable, considering  $q_\phi$  is a neural network with stochastic weights. Depending on how we define the prior, however,  $\nabla_{\mathbf{f}} \log p(\mathbf{f}^{\mathbf{X}})$  can be easy to compute analytically. To reduce variance in the gradients we define tractable priors in this paper. To estimate the log-density gradient  $\nabla_{\mathbf{f}} \log q(\mathbf{f}^{\mathbf{X}})$ , they use a spectral Stein gradient estimator [Shi et al., 2018].

## 2 METHOD

We will present a method that allows the following:

- Optimal regret **TODO: define optimal regret**
- Incorporate domain knowledge

A natural step for finding the optimal sampling distribution  $q_\phi$  is to minimize the KL-divergence between it and the true Q-value function. This is different from the optimization scheme in NoisyNet, which only tries to maximize the likelihood that samples from  $q_\phi$  comes from the true Q-value function. This should probably go under method

We would like to approximate the posterior distribution of the Q value function. This would (1) give optimal regret, (2) let us incorporate domain knowledge through an appropriate prior distribution. We can use the functional variational bayesian neural network (FVBNN) [Sun et al., 2019] framework discussed earlier. NoisyNet uses one sample from  $Q$  for each optimization step. We will instead use  $N$  samples from  $Q$ . This lets us add the KL-term from FVBNN to the loss function that encourages stochasticity. The maximization target becomes:

$$\log p(\mathcal{D}_\omega^N | f) - \text{KL} [q(\mathbf{f}^{\mathcal{D}_\omega^N}, \mathbf{f}^M) \| p(\mathbf{f}^{\mathcal{D}_\omega^N}, \mathbf{f}^M)], \quad (3)$$

where  $\mathbf{f}^{\mathcal{D}_\omega^N}$  is  $f$  applied to the dataset  $\mathcal{D}_\omega^N$ , and  $\mathbf{f}^M$  is  $f$  applied to a set of i.i.d. random points  $M = \{m_i \sim c \mid i = 1, \dots, k\} \subseteq \mathcal{S} \times \mathcal{A}$  with full support. We will let

$$M = \{(s_i, a_j) \mid \forall s_i \in \mathcal{D}_\omega^N, \forall a_j \in \mathcal{A}\}. \quad (4)$$

Since we are modeling  $q_\phi$  with a Bayesian neural network, and  $Q_\omega$  with a neural network, we have a bilevel optimization problem. To solve this, we employ a similar strategy to Fellows et al. [2021], where we have a two-timescale gradient update.  $\phi$  and  $\omega$  are updated using stochastic gradient descent at different timescales to ensure stable convergence.

---

### Algorithm 1 NoisyNet RL

---

```

 $\mathcal{D} \leftarrow \emptyset$ 
 $s \sim P_0$ 
Initialize  $\phi, \omega$ 
while not converged do
   $f \sim q_\phi$ 
   $a \sim \arg \max_a f(s, a)$ 
   $s' \sim P(\cdot | s, a)$ 
   $r = r(s', a, s)$ 
   $\mathcal{D} \leftarrow \mathcal{D} \cup \{s, a, r, s'\}$ 
  UPDATENOISYNET( $\phi, \omega, \mathcal{D}$ )
end while

function UPDATENOISYNET( $\phi, \omega, \mathcal{D}$ )
   $T \sim \mathcal{D}$ 
   $f \sim q_\phi$ 
   $b \sim q_\omega$ 
   $\Delta_{\mathcal{L}} \leftarrow 0$ 
  for  $\{s, a, r, s'\} \in T$  do
     $q \leftarrow \max_a \mathbf{f}(s, a)$ 
     $\mathcal{F} \leftarrow \mathcal{F} \cup \mathbf{f}(s, a)$ 
     $G \leftarrow r + \gamma b(s, a)$ 
     $\Delta_{\mathcal{L}} \leftarrow \Delta_{\mathcal{L}} - \frac{1}{|T|} \nabla_\phi \log p(q|G)$ 
  end for
   $\phi \leftarrow \phi + \eta_\phi \Delta_{\mathcal{L}}$ 
   $\omega \leftarrow \omega + \eta_\omega (\omega - \phi)$ 
end function

```

---

## 3 EXPERIMENTS

### 3.1 UNINFORMATIVE PRIOR

### 3.2 INFORMATIVE PRIOR

## 4 DISCUSSION

### References

- Marc G. Bellemare, Will Dabney, and Rémi Munos. A Distributional Perspective on Reinforcement Learning. *arXiv:1707.06887 [cs, stat]*, July 2017. URL <http://arxiv.org/abs/1707.06887>. arXiv: 1707.06887.
- Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight Uncertainty in Neural Network. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1613–1622, Lille, France, July 2015. PMLR. URL <http://proceedings.mlr.press/v37/blundell115.html>.
- Matthew Fellows, Kristian Hartikainen, and Shimon Whiteson. Bayesian Bellman Operators. *arXiv:2106.05012 [cs]*, June 2021. URL <http://arxiv.org/abs/2106.05012>. arXiv: 2106.05012.
- Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. Noisy Networks for Exploration. *arXiv:1706.10295 [cs, stat]*, July 2019. URL <http://arxiv.org/abs/1706.10295>. arXiv: 1706.10295.
- Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining Improvements in Deep Reinforcement Learning. *arXiv:1710.02298 [cs]*, October 2017. URL <http://arxiv.org/abs/1710.02298>. arXiv: 1710.02298.
- Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. *arXiv:1602.01783 [cs]*, June 2016. URL <http://arxiv.org/abs/1602.01783>. arXiv: 1602.01783.
- Ian Osband, Benjamin Van Roy, and Zheng Wen. Generalization and Exploration via Randomized Value Functions. *arXiv:1402.0635 [cs, stat]*, February 2016. URL <http://arxiv.org/abs/1402.0635>. arXiv: 1402.0635.

---

#### Algorithm 2 Functional Bayesian RL

---

```

 $\mathcal{D} \leftarrow \emptyset$ 
 $s \sim P_0$ 
Initialize  $\phi, \omega$ 
while not converged do
     $f \sim q_\phi$ 
     $a \sim \arg \max_a f(s, a)$ 
     $s' \sim P(\cdot | s, a)$ 
     $r = r(s', a, s)$ 
     $\mathcal{D} \leftarrow \mathcal{D} \cup \{s, a, r, s'\}$ 
    UPDATEPOSTERIOR( $\phi, \omega, \mathcal{D}$ )
end while

function UPDATEPOSTERIOR( $\phi, \omega, \mathcal{D}$ )
     $T \sim \mathcal{D}$ 
     $f \leftarrow \{f_i \sim q_\phi \mid i = 1, \dots, k\}$ 
     $\mathbf{b} \leftarrow \{b_i \sim q_\omega \mid i = 1, \dots, k\}$ 
     $\mathcal{F} \leftarrow \emptyset$ 
     $\Delta_{\mathcal{L}} \leftarrow 0$ 
    for  $\{s, a, r, s'\} \in T$  do
         $\mathbf{q} \leftarrow \max_a \mathbf{f}(s, a)$ 
         $\mathcal{F} \leftarrow \mathcal{F} \cup \mathbf{f}(s, a)$ 
         $\mathbf{G} \leftarrow \{r + \gamma b_i(s, a) \mid i = 1, \dots, k\}$ 
         $\Delta_{\mathcal{L}} \leftarrow \Delta_{\mathcal{L}} - \frac{1}{|T|} \nabla_\phi \log p(\mathbf{q} | \mathbf{G})$ 
    end for
     $\Delta_{\text{KL}} \leftarrow \text{SSGE}(p, \mathcal{F})$ 
     $\phi \leftarrow \phi + \eta_\phi (\Delta_{\mathcal{L}} - \lambda \Delta_{\text{KL}})$ 
     $\omega \leftarrow \omega + \eta_\omega (\omega - \phi)$ 
end function

```

---

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized Experience Replay. *arXiv:1511.05952 [cs]*, February 2016. URL <http://arxiv.org/abs/1511.05952>. arXiv: 1511.05952.

Jiabin Shi, Shengyang Sun, and Jun Zhu. A Spectral Approach to Gradient Estimation for Implicit Distributions. June 2018. URL <https://arxiv.org/abs/1806.02925v1>.

Shengyang Sun, Guodong Zhang, Jiabin Shi, and Roger Grosse. Functional Variational Bayesian Neural Networks. *arXiv:1903.05779 [cs, stat]*, March 2019. URL <http://arxiv.org/abs/1903.05779>. arXiv: 1903.05779.

Hado van Hasselt, Arthur Guez, and David Silver. Deep Reinforcement Learning with Double Q-learning. *arXiv:1509.06461 [cs]*, December 2015. URL <http://arxiv.org/abs/1509.06461>. arXiv: 1509.06461.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. Dueling Network Architectures for Deep Reinforcement Learning. *arXiv:1511.06581 [cs]*, April 2016. URL <http://arxiv.org/abs/1511.06581>. arXiv: 1511.06581.