NeurIPS 2023 Workshop on Adaptive Experimental Design and Active Learning in the Real World

Efficient and scalable reinforcement learning via Hypermodel

Yingru Li YINGRULI@LINK.CUHK.EDU.CN
Jiawei Xu JIAWEIXU@LINK.CUHK.EDU.CN
Zhi-Quan Luo LUOZQ@CUHK.EDU.CN
The Chinese University of Hong Kong, Shenzhen, China

Abstract

Data-efficient reinforcement learning(RL) requires deep exploration. Thompson sampling is a principled method for deep exploration in reinforcement learning. However, Thompson sampling need to track the degree of uncertainty by maintaining the posterior distribution of models, which is computationally feasible only in simple environments with restrictive assumptions. A key problem in modern RL is how to develop data and computation efficient algorithm that is scalable to large-scale complex environments. We develop a principled framework, called HyperFQI, to tackle both the computation and data efficiency issues. HyperFQI can be regarded as approximate Thompson sampling for reinforcement learning based on hypermodel. Hypermodel in this context serves as the role for uncertainty estimation of action-value function. HyperFQI demonstrates its ability for efficient and scalable deep exploration in DeepSea benchmark with large state space. HyperFQI also achieves super-human performance in Atari benchmark with 2M interactions with low computation costs. We also give a rigorous performance analysis for the proposed method, justifying its computation and data efficiency. To the best of knowledge, this is the first principled RL algorithm that is provably efficient and also practically scalable to complex environments such as Arcade learning environment that requires deep networks for pixel-based control.

1. Introduction

In reinforcement learning (RL), intelligent exploration relies on decisions that are driven not only by expectations but also by epistemic uncertainty (Osband et al., 2019b). Actions are taken to resolve epistemic uncertainty not only based on immediate consequences but also on what will be observed over subsequent time periods, a concept known as deep exploration (Osband et al., 2019b). One popular exploration scheme in RL is Thompson Sampling (TS), which makes decisions based on a posterior distribution over models (Thompson, 1933; Strens, 2000; Russo et al., 2018). A basic form of TS involves sampling a model from the posterior and selecting an action that optimizes the sampled model.

However, generating exact posterior samples is computationally tractable only for simple environments, such as tabular MDPs with Dirichlet priors over transition probability vectors (Strens, 2000; Osband et al., 2013). For complex domains, approximations are necessary (Russo et al., 2018). In order to address this need, Osband et al. (2019b) developed randomized least-squares value iteration (RLSVI). RLSVI aims to approximate sampling from the posterior over the optimal value function without explicitly representing the distribution. The algorithm achieves this by randomly perturbing a prior and an accumulated

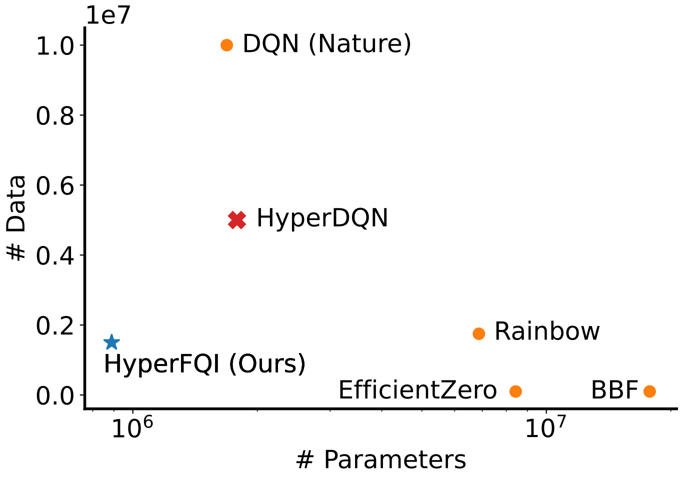


Figure 1: This figure investigates the relationship between the required training data and the model parameters for achieving human-level performance using various algorithms. The evaluation is conducted using IQM (Agarwal et al., 2021) on 26 Atari games. The x-axis represents the amount of training data required to achieve human-level performance, measured in 1.0 IQM. The $\times$ indicates that the algorithm fails to reach human-level performance with the corresponding amount of training data. The $\star$ denotes that our proposed algorithm, `HyperFQI`, achieves human-level performance using only 15% of the training data compared with DQN (Nature) (Van Hasselt et al., 2016) and 5% of the model parameters compared with BBF (Schwarzer et al., 2023).

dataset and fitting a point estimate of the value function to this perturbed prior and data. The induced randomness from these perturbations leads to deep exploration, improving data efficiency (Osband et al., 2019b).

While RLSVI avoids the explicit maintenance of a posterior distribution, it still requires computationally intensive operations to generate a new point estimate for each episode. These computations do not leverage previously computed point estimates and therefore cannot be incrementally updated. Ensemble sampling has been proposed as an alternative approach to approximate RLSVI's performance. It involves maintaining a set of point estimates, with each estimate updated incrementally as data accumulates (Osband et al., 2016, 2018, 2019b). Nevertheless, maintaining an ensemble of complex models can be computationally burdensome. Furthermore, to obtain a good approximation of the posterior distribution, the ensemble size needs to grow significantly with the complexity of the distribution (Dwaracherla et al., 2020; Osband et al., 2021; Li et al., 2022a; Qin et al., 2022).

Alternatively, instead of maintaining an ensemble of models, one can learn a hypermodel. A hypermodel can be used to generate approximate posterior samples, as discussed in prior works (Dwaracherla et al., 2020; Li et al., 2022a). This approach shows promise, but it requires a representation that can be more complex than a point estimate of the value function. The computational requirements and the number of parameters needed for this representation, however, lack theoretical understanding.

None of these algorithms have been shown to be computationally efficient, data efficient and scalable at the same time. In particular, RLSVI is data efficient but it is neither computationally efficient nor scalable. Ensemble (Osband et al., 2016, 2018, 2019b) or previous hypermodel-related approaches (Li et al., 2022a) are computationally tractable to scale to complex environment with deep networks but they are not efficient enough and also have no theoretical guarantees. This paper aims to develop a principled RL algorithm that is both computationally and data efficient and also scalable to large-scale complex environments.

1.1 Contributions

We propose a novel algorithm, called HyperFQI, that combines the benefits of RLSVI and hypermodel-based approaches. HyperFQI can be regarded as an online version of the fitted Q-iteration (FQI) algorithm (Ernst et al., 2005; Mnih et al., 2015). FQI is originally a batch RL algorithm that learns a Q-function by fitting a regression model to a dataset of state-action-reward-state tuples. HyperFQI maintains a hypermodel that can be used to generate approximate posterior samples and carefully designs a way to sample from hypermodel for both training and action selection.

- HyperFQI is the first algorithm in the literature solving DeepSea at a large scale up to 100^2 states, see details in Section 4.1.
- HyperFQI is also the first one achieving human-level performance in Atari games, when considering both data and computation efficiency, see details in Figure 1 and Section 4.2.
- We provide a rigorous performance analysis for the proposed method in Section 5, justifying its computation and data efficiency. HyperFQI achieves the Bayesian regret bound of order $\tilde{O}(H^2\sqrt{|\mathcal{X}||\mathcal{A}|K})$ for finite horizon MDPs with H horizon, $|\mathcal{X}|$ states, $|\mathcal{A}|$ actions, and K episodes, sharing the same order with famous RLSVL (Osband et al., 2019b) and PSRL (Osband and Van Roy, 2017). The additional computation burden of HyperFQI than single point estimate is only logarithmic in $|\mathcal{X}|$ and $|\mathcal{A}|$ and episode number K, i.e. the additional dimension is $M = \tilde{O}(\log(|\mathcal{X}||\mathcal{A}|K))$. The analysis is enabled by our novel probability tools in Appendix G, which maybe of independent interest.

To the best of knowledge, this is the first principled RL algorithm that is provably efficient and also practically efficiently scalable to complex environments such as DeepSea with large state space and Arcade learning environment. We believe this work serves as a bridge for theory and practice in reinforcement learning.

2. Preliminary

2.1 Reinforcement Learning

We consider the episodic RL setting in which an agents interacts with an unknown environment over a sequence of episodes. We model the environment as a Markov decision problem (MDP) $M = (\mathcal{S}, \mathcal{A}, R, P, s_{\text{terminal}}, \rho)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, terminal $\in \mathcal{S}$ is the terminal state, and ρ is the initial state distribution. For each episode, the initial state S_0 is drawn from the distribution ρ. In each time period $t = 1, 2, \ldots$ within an episode, the agent observes a state $S_t \in \mathcal{S}$. If $S_t \neq s_{\text{terminal}}$, the agent selects an action $A_t \in \mathcal{A}$, receives a reward $R_{t+1} \sim R(\cdot \mid S_t, A_t)$, and transitions to a new state $S_{t+1} \sim P(\cdot \mid S_t, A_t)$. An episode terminates once the agent arrives at the terminal state. Let τ be the termination time of a generic episode, i.e., $S_\tau = s_{\text{terminal}}$. Note that τ is a stopping time in general. To illustrate, we denote the sequence of observations in episode k by $\mathcal{O}_k = (S_{k,0}, A_{k,0}, R_{k,1}, \ldots, S_{k,\tau_k-1}, A_{k,\tau_k-1}, R_{k,\tau_k})$ where $S_{k,t}, A_{k,t}, R_{k,t+1}$ are the state, action, reward observed at t-th time period of the k-th episode and τ_k is the termination time at episode k. We denote the history of observations made prior to episode k by $\mathcal{H}_k = (\mathcal{O}_1, \ldots, \mathcal{O}_{k-1})$.

A policy $\pi : \mathcal{S} \to \mathcal{A}$ maps a state $s \in \mathcal{S}$ to an action $a \in \mathcal{A}$. For each MDP M with state space $\mathcal{S}$ and action space $\mathcal{A}$, and each policy π, we define the associated state-action value function as:

$$Q_M^\pi(s,a) := \mathbb{E}_{M,\pi}\left[\sum_{t=1}^{\tau} R_t \mid S_0 = s, A_0 = a\right]$$

where the subscript π next under the expectation is a shorthand for indicating that actions over the whole time periods are selected according to the policy π. Let $V_M^\pi(s) := Q_M^\pi(s, \pi(s))$. We say a policy π^M is optimal for the MDP M if $\pi^M(s) \in \arg\max_\pi V_M^\pi(s)$ for all $s \in \mathcal{S}$. To simplify the exposition, we assume that under any MDP M and any policy π, the termination time $\tau < \infty$ is finite with probability 1.

The agent is given knowledge about $\mathcal{S}, \mathcal{A}, s_{\text{terminal}}$, and ρ, but is uncertain about R and P. The unknown MDP M, together with reward function R and transition function P, are modeled as random variables with a prior belief. The agent's behavior is governed by a RL algorithm alg which uses the history of observations $\mathcal{H}_k$ to select a policy $\pi_k = \text{alg}(\mathcal{S}, \mathcal{A}, \mathcal{H}_k)$ for the k-th episode. The design goal of RL algorithm is to maximize the expected total reward up to episode K

$$\mathbb{E}_{M,\text{alg}}[\sum_{k=1}^{K}\sum_{t=1}^{\tau_k} R_{k,t}] = \mathbb{E}_{M,\text{alg}}[\sum_{k=1}^{K} V_M^{\pi_k}(s_{k,0})]. \tag{1}$$

where the subscript alg under the expectation indicates that policies are generated through algorithm alg. Note that the expectations in both sides of Equation (1) is over the stochastic transitions and rewards under the MDP M, the possible randomization in the learning algorithm alg. The expectation in the LHS of Equation (1) is also over the randomness in the termination time τ_k.

2.2 Hypermodel

We build RL agents based on the hypermodel framework (Li et al., 2022a; Dwaracherla et al., 2020; Osband et al., 2021). Hypermodel is a function f parameterized with θ, receiving input $x \in \mathbb{R}^d$ and an random index $\xi \in \mathbb{R}^M$ from reference distribution P_ξ, making predictions $f_\theta(x, \xi) \in \mathbb{R}$. We aim to capture the uncertainty via the variation over the hypermodel predictions by the random index ξ. Hypermodel parameter θ is trainable to adjust its uncertainty representation when seeing more data. The reference distribution P_ξ remain fixed throughout the training. For example, linear-Gaussian model is a special case of hypermodel with parameter $\theta = (\boldsymbol{A}, \mu)$ with reference distribution $P_\xi = N(0, I_M)$, where $f_\theta(x, \xi) = \langle x, \mu + \boldsymbol{A}\xi\rangle$. In this case f_θ follows a Gaussian distribution with mean μ

and covariance $\boldsymbol{A}\boldsymbol{A}^\top$. Ensemble of M neural networks $g_{\theta_1}, \dots, g_{\theta_M}$ is also a special case of hypermodel with parameter $\theta = (\theta_1, \dots, \theta_M) \in \mathbb{R}^{d \times M}$ with reference distribution $P_\xi = \mathcal{U}(e_1, \dots, e_M)$ being uniform distribution over one-hot vectors, where $f_\theta(x, \xi) = g_{\langle \theta, \xi \rangle}(x)$. In general, the hypermodel $f_\theta(\cdot)$ can be any function, e.g. neural networks, transforming the reference distribution P_ξ to arbitrary distribution. We adopt a class of hypermodel that can be represented as an additive function

$$\underbrace{f_\theta(x, \xi)}_{\textbf{“Posterior” Hypermodel}} = \underbrace{f^L_\theta(x, \xi)}_{\textbf{Learnable function}} + \underbrace{f^P(x, \xi)}_{\textbf{Fixed prior model}} \tag{2}$$

The prior model f^P represents the prior bias and uncertainty and has NO trainable parameters. The learnable function is initialized to output value near zero and is then trained by fitting the data. The resultant sum f_θ produces reasonable predictions for all probable values of ξ. Variations of a prediction $f_\theta(x, \cdot)$ as a function of ξ indicate the epistemic uncertainty estimation. The prior model $f^P(\cdot, \xi)$ can be viewed as a prior distribution of the true model f^*, which is the true function that generates the data. The hypermodel $f_\theta(\cdot, \xi)$ can be viewed as a trained approximate posterior distribution of the true model f^* given the data. Similar decomposition in Equation (2) is also used in (Dwaracherla et al., 2020; Osband et al., 2021; Li et al., 2022a). We will discuss the implementation details and clarify the importance of difference between our work and prior works in Appendix A.

3. Algorithm

We now develop a novel DQN-type algorithm for large-scale RL problems with value function approximation, called `HyperFQI`. `HyperFQI` uses a hypermodel to maintain a probability distribution over the action-value function and aims to approximate the posterior distribution of $Q^* := Q^{\pi^*}_M$. The hypermodel in this context is a function $f_\theta : \mathcal{S} \times \mathcal{A} \times \Xi \to \mathbb{R}$ parameterized by $\theta \in \Theta$ and Ξ is the index space. Hypermodel is then trained by minimizing the loss function motivated by fitted Q-iteration (FQI), a classical method (Ernst et al., 2005) for value function approximation. `HyperFQI` selects the action based on sampling indices from reference distribution P_ξ and then taking the action with the highest value from hypermodels applying these indices. This can be viewed as an value-based approximate Thompson sampling via Hypermodel.

Alongside the learning process, `HyperFQI` maintains two hypermodels, one for the current value function f_θ and the other for the target value function f_{θ^-}. `HyperFQI` also maintains a buffer of transitions $D = \{(s, a, r, s', \mathbf{z})\}$, where $\mathbf{z} \in \mathbb{R}^M$ is the algorithm-generated perturbation random vector sampled from the perturbation distribution $P_{\mathbf{z}}$. For a transition tuple $d = (s, a, r, s', \mathbf{z}) \in D$ and given index ξ, the temporal difference (TD) error for hypermodel is

$$\ell^{\gamma, \sigma}(\theta; \theta^-, \boldsymbol{\xi}^-, \xi, d) = \left(f_\theta(s, a, \xi) - (r + \sigma \xi^\top \mathbf{z} + \gamma \max_{a' \in \mathcal{A}} f_{\theta^-}(s', a', \boldsymbol{\xi}^-(s'))) \right)^2 \tag{3}$$

where θ^- is the target parameters, and the σ is a hyperparameter to control the injected noise by algorithm. $\boldsymbol{\xi}^-$ is the target index mapping such that $\boldsymbol{\xi}^-(s)$ one-to-one maps each state $s \in \mathcal{S}$ to a random vector from P_ξ, all of which are independent with ξ.[1] The algorithm update the hypermodel for value function by minimizing

$$L^{\gamma,\sigma,\beta}(\theta;\theta^-,\boldsymbol{\xi}^-,D) = \mathbb{E}_{\xi\sim P_\xi}[\sum_{d\in D}\frac{1}{|D|}\ell^{\gamma,\sigma}_{z^-}(\theta;\theta^-,\boldsymbol{\xi}^-,\xi,d)] + \frac{\beta}{|D|}\|\theta\|^2 \tag{4}$$

where $\beta \geq 0$ is the prior regularization parameter. Note the target hypermodel is necessary for stabilizing the optimization and reinforcement learning process, as discussed in target Q-network literature (Mnih et al., 2015; Li et al., 2022b). We optimize the loss function Equation (4) using stochastic gradient descent (SGD) with a mini-batch of data $\tilde{D}$ and a batch of indices $\tilde{\Xi}$ from P_ξ. That is, we take gradient descent with respect to the sampled version of loss

$$\tilde{L}(\theta;\theta^-,\boldsymbol{\xi}^-,\tilde{D}) = \frac{1}{|\tilde{\Xi}|}\sum_{\xi\in\tilde{\Xi}}\left(\sum_{d\in\tilde{D}}\frac{1}{|\tilde{D}|}\ell^{\gamma,\sigma}(\theta;\theta^-,\boldsymbol{\xi}^-,\xi,d)\right) + \frac{\beta}{|D|}\|\theta\|^2 \tag{5}$$

We summarize the `HyperFQI` algorithm: At each episode k, `HyperFQI` samples an index mapping $\boldsymbol{\xi}_k$ from the index distribution P_ξ and then take action by maximizing the associated hypermodel $f_\theta(\cdot,a,\boldsymbol{\xi}_k(\cdot))$, which we call index sampling (IS) action selection.[2] This can be viewed as an value-based approximate Thompson sampling. The algorithm updates the hypermodel parameters θ in each episode according to Equation (5), and updates the target hypermodel parameters θ^- periodically. The algorithm also maintains a replay buffer of transitions D, which is used to sample a mini-batch of data $\tilde{D}$ for training the hypermodel.

Algorithm 1 HyperFQI for RL

```
1: Input: Initial parameter $\theta_{\text{init}}$, Hypermodel for value $f_\theta(s=\cdot,a=\cdot,\boldsymbol{\xi}=\cdot)$ with dist. $P_\xi$.
2: Initialize $\theta=\theta^-=\theta_{\text{init}}$, train step $j=0$ and buffer $D$
3: for each episode $k=1,2,\ldots$ do
4:     Sample index mapping $\boldsymbol{\xi}_k \sim P_\xi$
5:     Set $t=0$ and Observe $S_{k,0}\sim\rho$
6:     repeat
7:         Select $A_{k,t}=\arg\max_{a\in\mathcal{A}} f_\theta(S_{k,t},a,\boldsymbol{\xi}_k(S_{k,t}))$
8:         Observe $R_{k,t+1}$ and $S_{k,t+1}$ from environment
9:         Sample $\mathbf{z}_{k,t+1}\sim P_{\mathbf{z}}$ and $D$.add$((S_{k,t},A_{k,t},R_{k,t+1},S_{k,t+1},\mathbf{z}_{k,t+1}))$
10:        Increment step counter $t\leftarrow t+1$
11:        $\theta,\theta^-,j \leftarrow$ update$(D,\theta,\theta^-,\boldsymbol{\xi}^-=\boldsymbol{\xi}_k,t,j)$
12:    until $S_t=s_{\text{terminal}}$
13: end for
```

This algorithm offers several advantages over existing methods. First, it is computationally efficient due to the nature of incremental update and scalable to large-scale problems.

1. To clarify, the random vector $\boldsymbol{\xi}^-(s)$ remains the same vector if we do not resample the mapping $\boldsymbol{\xi}^-$.
2. To clarify, the random vector $\boldsymbol{\xi}_k(s)$ remains the same vector within the episode k.

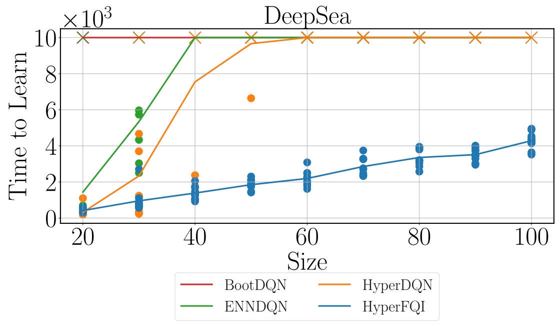
Second, it is compatible with existing deep RL algorithms and can be used as a drop-in replacement for the Q-network in DQN-type methods. Finally, it is easy to implement and can be applied to a wide range of problems.

4. Experimental studies

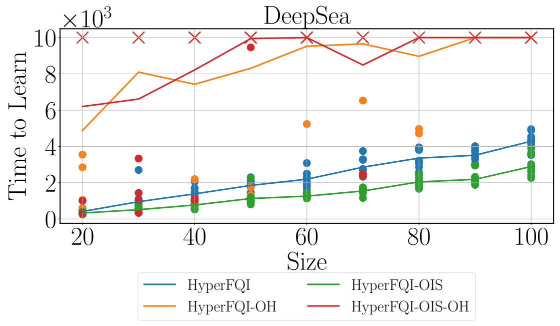
This section evaluates the efficiency and scalability of our HyperFQI. Our experiments on the DeepSea demonstrate its high data and computation efficiency, achieving polynomial performance. We also showcase the scalability of our approach by successfully processing large-scale states with a size of 100^2. In addition, we evaluate the scalability using the Atari games, where our HyperFQI performs exceptionally well in processing states with pixels. Furthermore, our approach can achieve human-level performance with remarkable data and computation efficiency in Atari games. **To highlight**, HyperFQI is (1) the first algorithm in the literature solving DeepSea at a large scale up to 100^2 states, and (2) also the first one achieving human-level performance in Atari games, considering both data and computation efficiency.

4.1 Computational results for deep exploration

We demonstrate the exploration effectiveness of our HyperFQI using DeepSea, a reward-sparse environment that demands deep exploration. DeepSea offers only two actions: moving left or right; see Appendix B. The agent receives a reward of 0 for moving left, and a penalty of $-(0.01/N)$ for moving right, where N denotes the size of DeepSea. The agent earns a reward of 1 upon reaching the lower-right corner of the DeepSea, making it optimal for the agent to continuously move towards the right and get the total return 0.99.



(a)



(b)

Figure 2: Experimental results on DeepSea. The y-axis represents the number of episodes required to learn the optimal policy for a specific problem size. The symbol $\times$ indicates that the algorithm was unable to solve the problem within $10e3$ episodes. (a) The performance of various baselines. (b) The performance of different variants with our HyperFQI.

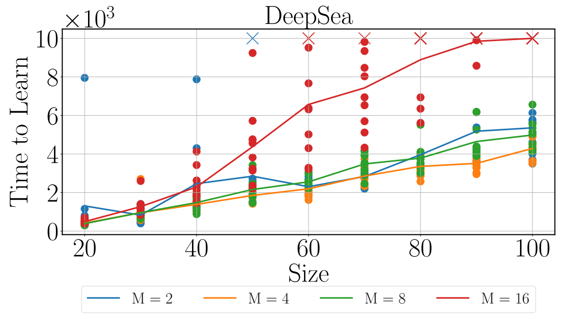
Baselines Result: We define the *Time to Learn*$(N) := mean\{K|\bar{R}_K \geq 0.99\}$, which serves as an evaluation metric for algorithm performance on DeepSea with size N. The $\bar{R}_K$ represents the return obtained by the agent after K episodes of interaction, and $\bar{R}_K$ is the average return over 100 evaluation instances. Overall, the *Time to Learn*(N) indicates the number of episodes needed to learn the optimal policy. As shown in Figure 2(a), our HyperFQI can achieve superior performance compared to other baselines. Based on the structure of DeepSea, we can deduce that discovering the optimal policy requires at least $N^2/2$ episodes, as all accessible states must be traversed. Our experiments demonstrate that our HyperFQI can achieve the optimal policy within the aforementioned episodes range, which fully demonstrates the data efficiency of our algorithm.

For all baselines in Figure 2(a), we set the index dimension to 4 (and set the number of ensemble networks to 4 for BootDQN), and conduct experiments with 10 different seeds. Notably, BootDQN (Osband et al., 2018), known for its effective exploration in chain environments of considerable size, fails to solve the DeepSea with a size of 20 within $10e3$ episodes. Furthermore, HyperDQN (Li et al., 2022a), which has demonstrated effective deep exploration, cannot learn the optimal policy when the size of DeepSea exceeds 50 within $10e3$ episodes. These results provide evidence for the effectiveness of our network structure and suggest that the update method used in HyperFQI enhances the ability to capture uncertainty and promote effective exploration.

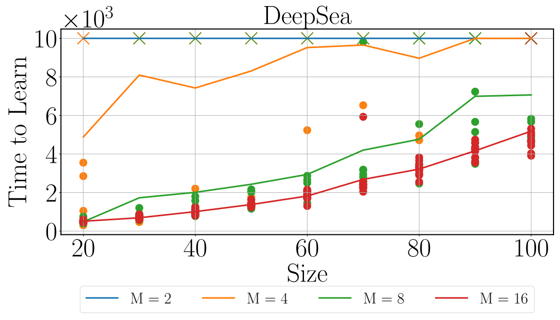
The ENNDQN, which was adapted from Osband et al. (2023), struggles to solve DeepSea as its size increases. Compared to our approach, ENNDQN includes the original input as a component of the final ENN layer's input. Both HyperFQI and ENNDQN share the same feature network, and the parameters in our output layer (hypermodel) remain constant when scaling up the problem. However, the ENN layer requires a greater number of parameters and computational workload, especially as the problem's scale increases. In the case of DeepSea with of size of 20, the number of parameters in the ENN layer is almost 20 times larger than our hypermodel. These findings demonstrate that the network architecture of our HyperFQI can enhance both computational efficiency and scalability when dealing with large scale problem.

Variants of HyperFQI: We can produce various variants based on the framework of HyperFQI, including HyperFQI-OIS, which applies optimistic index sampling (OIS) to action selection and computation of Q-target (refer to Appendix A.1.2 for details). Furthermore, we substitute the Gaussian distributional index with a one-hot index under two different action selections, resulting in HyperFQI-OH and HyperFQI-OIS-OH. In Figure 2(b), we compare the performance of different variants of our approach. Our HyperFQI-OIS impressively outperforms HyperFQI by utilizing optimistic index sampling to achieve more optimistic estimates, which can enhance exploration. The OIS method does not increase much computation as we set the dimension M to 4. We observed that HyperFQI-OH is not effective in DeepSea as the Gaussian distributional index provides superior expectation estimation compared to the one-hot index. However, subsequent experiments show that increasing the dimension of the one-hot index can improve exploration.

Ablation Study: We consider how the dimension M of the index affects our methods. Figure 3 demonstrates that increasing the M of the one-hot index can lead to improved estimation of expectations, which in turn can enhance exploration. HyperFQI-OIS also can result in better performance when using the one-hot index with $M = 16$, which is shown in Appendix C.1. On the other hand, increasing the dimension of the Gaussian distributional index can actually hurt the algorithm's performance because it becomes more difficult to estimate the expectation in Equation (5) under P_ξ higher index dimension. However, there are ways to mitigate this problem. For a given dimension M, increasing the number of indices $|\tilde{\Xi}|$ of Equation (5) during the update phase can result in more accurate estimates, as demonstrated in Appendix C.1. However, this comes at the cost of increased computation, which slows down the algorithm. To strike a balance between performance and computation, we have chosen $M = 4$ and $|\tilde{\Xi}| = 20$ as our default hyper-parameters. In addition, we have also investigated the effect of other hyper-parameters on our methods, as shown in Appendix C.1.



(a) HyperFQI



(b) HyperFQI-OH

Figure 3: Ablation results under different index dimension M.

4.2 Atari Results

We assess the computational complexity of various methods on the Arcade Learning Environment (Bellemare et al., 2013) using IQM (Agarwal et al., 2021) as the evaluation criterion. An IQM score of 1 indicates that the algorithm's performance is comparable to that of a human. We examine our HyperFQI with six baselines: DQN (Nature) (Van Hasselt et al., 2016), Rainbow (Hessel et al., 2018), HyperDQN (Li et al., 2022a), BBF (Schwarzer et al., 2023) and EfficientZero (Ye et al., 2021). Specially, the EfficientZero is a state-of-the-art model-based method, while the others are value-based methods. Following the established practice in popular and widely accepted research (Van Hasselt et al., 2019; Ye et al., 2021), the results is compared on 26 Atari games in Arcade Learning Environment. Figure 1 illustrates the relationship between model parameters and the amount of training data required to achieve human-level performance. Our HyperFQI achieves human-level performance with minimal parameters and relatively little training data, outperforming other methods. Notably, the Convolutional layers in our model are the same as those in Rainbow and account for only a small fraction (about 13%) of the overall model parameters. This suggests that the Fully Connected layers dominate the computational complexity of the model, and as we know, the computational complexity of a Fully Connected layer is directly proportional to the number of parameters. In fact, our model employs the first Fully Connected layer with just 256 units, which is even fewer than DQN (Nature). Consequently, our HyperFQI offers superior computational performance due to having fewer parameters in the Fully Connected layer than other baselines.

We also assessed various variants of our HyperFQI on 26 Atari games using 2 million training data, and we presented their performance in Table 1. In addition, we implemented our version of DDQN, named DDQN(ours), using the same hyper-parameters and network

	IQM	Median	Mean
DDQN†	0.13 (0.11, 0.15)	0.12 (0.07, 0.14)	0.49 (0.43, 0.55)
DDQN(ours)	0.70 (0.69, 0.71)	0.55 (0.54, 0.58)	0.97 (0.95, 1.00)
HyperFQI	1.22 (1.15, 1.30)	1.07 (1.03, 1.14)	1.97 (1.89, 2.07)
HyperFQI-OH	1.28 (1.21, 1.35)	1.13 (1.10, 1.18)	2.03 (1.93, 2.15)
HyperFQI-OIS	1.15 (1.09, 1.22)	1.12 (1.02, 1.18)	2.02 (1.91, 2.16)
HyperFQI-OIS-OH	1.25 (1.18, 1.32)	1.10 (1.04, 1.17)	2.02 (1.93, 2.12)

Table 1: Performance profiles of our HyperFQI with different variant.

initialization scheme as our HyperFQI. In comparison, We report the results of vanilla DDQN† from Hessel et al. (2018). Our results show that DDQN(ours) outperforms DDQN† due to the increased data efficiency provided by our hyper-parameters and network initialization. Furthermore, our HyperFQI demonstrates superior performance compared to DDQN, as HyperFQI includes an additional hypermodel that enables deep exploration in Atari games. Additionally, our findings indicate that all methods performed similarly, implying that the OIS method or one-hot index do not generate significant differences in complex networks such as Convolutional layers. More detailed results for each Atari game are available in Appendix C.2, where we visualize the relative improvement compared to other baselines and the learning curve of our variants. The results demonstrate the better exploration efficiency of our HyperFQI than all baselines and the robustness of all our variants across all Atari games.

Furthermore, we demonstrated the superiority of our HyperFQI on the 8 hardest exploration Atari games with more baselines. We utilized the released results of AdamLMCDQN (Ishfaq et al., 2023), LangevinAdam (Ishfaq et al., 2023), and HyperDQN (Li et al., 2022a) on Atari games. Additionally, we adopted SANE (Aravindan and Lee, 2021) as our new baseline, which leverages a variational distribution to approximate the posterior. To reproduce the outcomes, we employed the official implementation of SANE. We trained both HyperFQI and SANE with 5 different seeds, up to a limit of 2M steps. The Figure 4 indicates that our HyperFQI outperformed other baselines on 5 out of 8 games. For Solaris and Venture, we anticipate that providing more training time can further improve the performance of our HyperFQI. Overall, these results demonstrate the effectiveness and exploration ability of our HyperFQI in environments with complex observation.

5. Analysis

In this section, we try to explain the intuition behind the HyperFQI algorithm and how it achieves efficient deep exploration. We also provide a regret bound for HyperFQI in finite horizon time-inhomoegeneous MDPs. First, we describe the HyperFQI algorithm in Algorithm 1 when specified to tabular problems.

Tabular HyperFQI. Let $f_\theta(s,a,\xi) = \mu_{sa} + m_{sa}^\top \xi + \mu_{0,sa} + \sigma_0 \mathbf{z}_{0,sa}^\top \xi$ where $\theta = (\mu \in \mathbb{R}^{SA}, m \in \mathbb{R}^{SA\times M})$ are the parameters to be learned, and $\mathbf{z}_{0,sa} \in \mathbb{R}^M$ is a random vector from $P_{\mathbf{z}}$ and $\mu_{0,sa}, \sigma_0$ is a prior mean and prior variance for each (s,a). The regularizer in

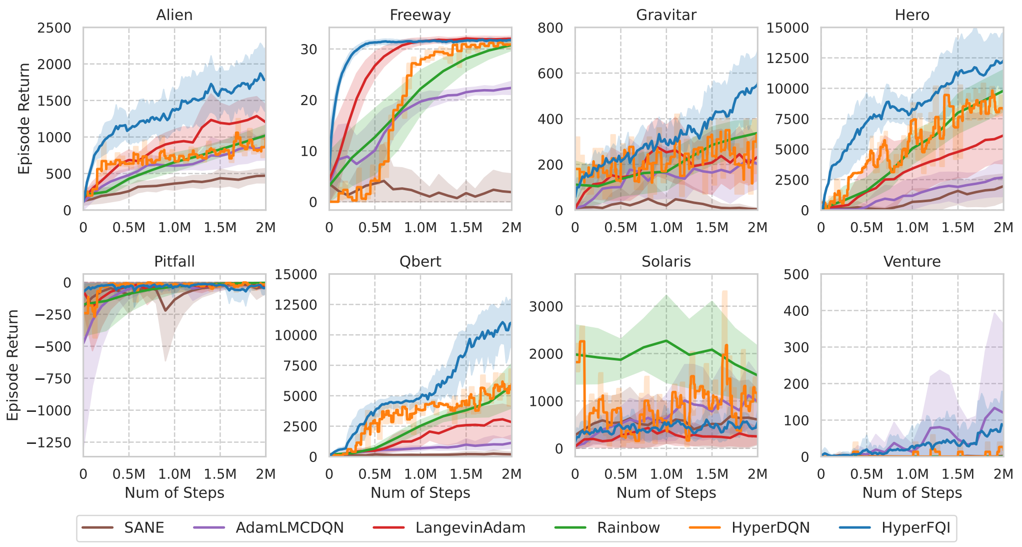


Figure 4: Comparative results on 8 hardest exploration games. HyperFQI shows significant performance and data efficiency gain compared with baselines.

Equation (4) becomes $\beta\|\theta\|^2 = \beta\sum_{s,a}\left(\mu_{sa}^2 + \|m_{sa}\|^2\right)$. Let the set $E_{k,sa}$ record the time index the agent encountered (s,a) in the k-th episode $E_{k,sa} = \{t : (S_{k,t}, A_{k,t}) = (s,a)\}$. Let $N_{k,sa} = \sum_{\ell=1}^{k-1}\sum_{t=0}^{H-1}\mathbb{1}_{(S_{\ell,t},A_{\ell,t})=(s,a)}$ denoting the counts of visitation for state-action pair (s,a) prior to episode k.

Closed-form incremental update. Let $\beta = \sigma^2/\sigma_0^2$. Then, given the dataset $D = H_k$ and target noise mapping $\boldsymbol{\xi}^- = \boldsymbol{\xi}_k$ at the beginning of the episode k, tabular `HyperFQI` with hyper-parameters specified in Table 3 would yield the following closed-form iterative procedure $\theta_k^{(i)} = (\mu_k^{(i)}, m_k) \to \theta_k^{(i+1)} = (\mu_k^{(i+1)}, m_k)$ for all $i = 1, 2, \ldots, H$: for all $(s,a) \in \mathcal{S}\times\mathcal{A}$

$$m_{k,sa} = \frac{\sigma\sum_{\ell=1}^{k-1}\sum_{t\in E_{\ell,sa}}\mathbf{z}_{\ell,t+1} + \beta\sigma_0\mathbf{z}_{0,sa}}{N_{k,sa}+\beta}, \tag{6}$$

$$\mu_{k,sa}^{(i+1)} = \frac{\sum_{\ell=1}^{k-1}\sum_{t\in E_{\ell,sa}} y_{\ell,t+1}(\theta^- = \theta_k^{(i)}, \boldsymbol{\xi}^- = \boldsymbol{\xi}_k) + \beta\mu_{0,sa}}{N_{k,sa}+\beta}, \tag{7}$$

where $y_{\ell,t+1}(\theta^-, \boldsymbol{\xi}^-) = R_{\ell,t+1} + \gamma\max_{a'\in\mathcal{A}} f_{\theta^-}(S_{\ell,t+1}, a', \boldsymbol{\xi}^-(S_{\ell,t+1}))$.

5.1 How does HyperFQI drives efficient deep exploration?

In this section, we highlight the key components of HyperFQI that enable efficient deep exploration. We consider a simple example (adapted from (Osband et al., 2019b)) to understand the HyperFQI's learning rule in Equations (4) and (5) and the role of hypermodel, and how they together drive efficient deep exploration.

Example 5.1. Consider a fixed horizon MDP $\mathcal{M}$ with four states $\mathcal{S} = \{1, 2, 3, 4\}$, two actions $\mathcal{A} = \{up, down\}$ and a horizon of $H = 6$. Let $\mathcal{H}$ be the list of all transitions observed so far, and let $\mathcal{H}_{s,a} = ((\hat{s}, \hat{a}, r, s') \in \mathcal{H} : (\hat{s}, \hat{a}) = (s, a))$ contain the transitions from state-action pair (s, a). Suppose $|\mathcal{H}_{4,down}| = 1$, while for every other pair $(s, a) \neq (4, down)$, $|\mathcal{D}_{s,a}|$ is very large, virtually infinite. Hence, we are highly certain about the expected immediate rewards and transition probabilities except for $(4, down)$. Assume that this is the case for all time periods $t \in \{0, 1, \dots, 5\}$.

HyperFQI produces a sequence of action-value functions $f_0, f_1, \dots f_5$. In Figure 5, each triangle in row s and column t contains two smaller triangles that are associated with action-values of *up* and *down* actions at state s. The shade on the smaller triangle shows the uncertainty estimates in the $f_t(s, a, \xi)$, specifically the variance $\mathrm{Var}_\xi\left(f_t(s, a, \xi)\right)$. The dotted lines show plausible transitions, except at $(4, down)$. Since we are uncertain about $(4, down)$, any transition is plausible. We will show how HyperFQI efficiently computes the

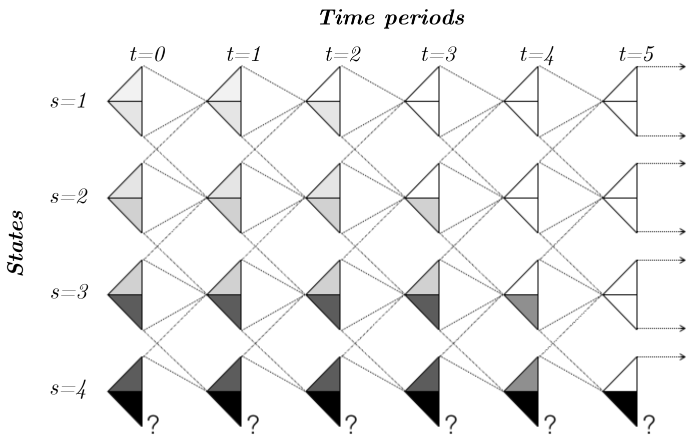


Figure 5: Example to illustrate how HyperFQI achieves deep exploration. We can see the propagation of uncertainty from later time period to earlier time period in the figure. Darker shade indicates higher degree of uncertainty.

uncertainty propagation backward in time, which can be visualized as progressing leftward in Figure 5. Also, we will show the ability to estimate the degree of uncertainty drives deep exploration. This can be explained by the incremental closed-form update in tabular setting described in Equations (12) and (13). A key property is that, with logarithmically small additional dimension M, hypermodel can approximate the posterior distribution of the optimal Q^*-values with low computation cost. This is formalized in the following Lemma 5.2. To prove Lemma 5.2, we develop a new probability tools for sequential random projection in Appendices F and G and a new probability tool for random projection Appendix H. These are novel technical contributions, which maybe of independent interests.

Lemma 5.2 (Approximate posterior variance)**.** *For m_k defined in Equation* (12) *with* $\mathbf{z} \sim \mathrm{Uniform}(\mathbb{S}^{M-1})$*. For any $k \geq 1$, a good event $\mathcal{G}_k(s, a)$ is defined as*

$$\mathcal{G}_k(s, a) = \left\{ \|m_k(s, a)\|^2 \in \left(\frac{\sigma^2}{N_{k,sa} + \beta}, \frac{3\sigma^2}{N_{k,sa} + \beta} \right) \right\}.$$

Then the joint event $\cap_{(s,a,k)\in\mathcal{S}\times\mathcal{A}\times[K]}\mathcal{G}_k(s,a)$ *holds w.p. at least* $1-\delta$ *if* $M \simeq \log(SAK/\delta)$.

5.2 Regret bound

Denote the regret of a policy π_k over episode k by $\Delta_k := \mathbb{E}_{M,\mathrm{alg}}[V_M^{\pi^*}(s_{k,0}) - V_M^{\pi_k}(s_{k,0})]$, where π^* is an optimal policy for M. The goal of the agent is equivalent to minimizing the expected total regret up to episode K, $\mathrm{Regret}(K, \mathrm{alg}) := \mathbb{E}_{\mathrm{alg}} \sum_{k=1}^{K} \Delta_k$, where the subscript alg under the expectation indicates that policies are generated through algorithm alg. Note that the expectation in Equation (17) is over the random transitions and rewards, the possible randomization in the learning algorithm alg, and also the unknown MDP M based on the agent designer's prior beliefs. Finally, we show that, with the help of hypermodel approximation property in Lemma 5.2, HyperFQI achieves efficient deep exploration in finite horizon time-inhomogeneous MDPs. This is formalized in the following theorem. The detail of analysis is in Appendix E.

Theorem 5.3 (Regret bound of HyperFQI)**.** *Consider an HyperFQI with an infinite buffer, greedy actions and with tabular representation. Under Assumptions E.1 and E.2 with* $\beta \geq 3$, *if the tabular* `HyperFQI` *is applied with planning horizon* H, *and parameters with* $(M, \mu_0, \sigma, \sigma_0)$ *satisfying* $M \simeq \log(|\mathcal{X}||\mathcal{A}|HK)$, $(\sigma^2/\sigma_0^2) = \beta$, $\sigma \geq \sqrt{3}H$ *and* $\mu_{0,s,a} = H$, *then for all* $K \in \mathbb{N}$,

$$\mathrm{Regret}(K, \textit{HyperFQI}) \leq 18H^2\sqrt{\beta|\mathcal{X}||\mathcal{A}|K \log_+(1 + |\mathcal{X}||\mathcal{A}|HK)}\log_+\left(1 + \frac{K}{|\mathcal{X}||\mathcal{A}|}\right), \quad (8)$$

where $\log_+(x) = \max\{1, \log(x)\}$.

Remark 5.4. The Assumption E.1 is common in the literature of regret analysis, e.g. (Osband et al., 2019b; Jin et al., 2018). The relationship between two set $\mathcal{S}$ and $\mathcal{X}$ is described in Assumption E.1. The Assumption E.2 is common in the Bayesian regret literature (Osband et al., 2013, 2019b; Osband and Van Roy, 2017; Lu and Van Roy, 2019). Our regret bound $\mathcal{O}(H^2\sqrt{|\mathcal{X}||\mathcal{A}|K})$ matches the best known Bayesian regret bound in the literature, say RLSVI (Osband et al., 2019b) and PSRL (Osband and Van Roy, 2017) which while our HyperFQI provide the computation and scalability benefit that RLSVI and PSRL do not have. We believe the `HyperFQI` algorithm provides a bridge for the theory and practice in RL.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Siddharth Aravindan and Wee Sun Lee. State-aware variational thompson sampling for deep q-networks. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pages 124–132, 2021.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Victor H de la Peña, Michael J Klass, and Tze Leung Lai. Self-normalized processes: Exponential inequalities, moment bounds and iterated logarithm laws. *Annals of Probability*, pages 1902–1933, 2004.

Vikranth Dwaracherla, Xiuyuan Lu, Morteza Ibrahimi, Ian Osband, Zheng Wen, and Benjamin Van Roy. Hypermodels for exploration. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=ryx6WgStPB`.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6, 2005.

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. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Haque Ishfaq, Qiwen Cui, Viet Nguyen, Alex Ayoub, Zhuoran Yang, Zhaoran Wang, Doina Precup, and Lin Yang. Randomized exploration in reinforcement learning with general value function approximation. In *International Conference on Machine Learning*, pages 4607–4616. PMLR, 2021.

Haque Ishfaq, Qingfeng Lan, Pan Xu, A. Rupam Mahmood, Doina Precup, Anima Anandkumar, and Kamyar Azizzadenesheli. Provable and practical: Efficient exploration in reinforcement learning via langevin monte carlo, 2023.

Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I Jordan. Is q-learning provably efficient? *Advances in neural information processing systems*, 31, 2018.

William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. In *Conference on Modern Analysis and Probability*, volume 26, pages 189–206. American Mathematical Society, 1984.

Tze Leung Lai. Martingales in sequential analysis and time series, 1945-1985. In *Electronic Journal for history of probability and statistics*, 2009.

Ziniu Li, Yingru Li, Yushun Zhang, Tong Zhang, and Zhi-Quan Luo. HyperDQN: A randomized exploration method for deep reinforcement learning. In *International Conference on Learning Representations*, 2022a. URL `https://openreview.net/forum?id=X0nrKAXu7g-`.

Ziniu Li, Tian Xu, and Yang Yu. A note on target q-learning for solving finite mdps with a generative oracle. *arXiv preprint arXiv:2203.11489*, 2022b.

Xiuyuan Lu and Benjamin Van Roy. Information-theoretic confidence bounds for reinforcement learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/411ae1bf081d1674ca6091f8c59a266f-Paper.pdf`.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Ian Osband and Benjamin Van Roy. Why is posterior sampling better than optimism for reinforcement learning? In *International conference on machine learning*, pages 2701–2710. PMLR, 2017.

Ian Osband, Benjamin Van Roy, and Daniel Russo. (more) efficient reinforcement learning via posterior sampling. In *Advances in Neural Information Processing Systems*, pages 3003–3011, 2013.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. *Advances in neural information processing systems*, 29, 2016.

Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 31, 2018.

Ian Osband, Yotam Doron, Matteo Hessel, John Aslanides, Eren Sezener, Andre Saraiva, Katrina McKinney, Tor Lattimore, Csaba Szepesvari, Satinder Singh, et al. Behaviour suite for reinforcement learning. *arXiv preprint arXiv:1908.03568*, 2019a.

Ian Osband, Benjamin Van Roy, Daniel J. Russo, and Zheng Wen. Deep exploration via randomized value functions. *Journal of Machine Learning Research*, 20(124):1–62, 2019b. URL `http://jmlr.org/papers/v20/18-339.html`.

Ian Osband, Zheng Wen, Seyed Mohammad Asghari, Vikranth Dwaracherla, Morteza Ibrahimi, Xiuyuan Lu, and Benjamin Van Roy. Epistemic neural networks. *arXiv preprint arXiv:2107.08924*, 2021.

Ian Osband, Zheng Wen, Seyed Mohammad Asghari, Vikranth Dwaracherla, Morteza Ibrahimi, Xiuyuan Lu, and Benjamin Van Roy. Approximate thompson sampling via epistemic neural networks. *arXiv preprint arXiv:2302.09205*, 2023.

Victor H Peña, Tze Leung Lai, and Qi-Man Shao. *Self-normalized processes: Limit theory and Statistical Applications*. Springer, 2009.

Chao Qin, Zheng Wen, Xiuyuan Lu, and Benjamin Van Roy. An analysis of ensemble sampling. *arXiv preprint arXiv:2203.01303*, 2022.

John Quan and Georg Ostrovski. DQN Zoo: Reference implementations of DQN-based agents, 2020. URL `http://github.com/deepmind/dqn_zoo`.

Herbert Robbins and David Siegmund. Boundary crossing probabilities for the wiener process and sample sums. *The Annals of Mathematical Statistics*, pages 1410–1429, 1970.

Mark Rudelson and Roman Vershynin. Hanson-wright inequality and sub-gaussian concentration. 2013.

Daniel J Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, Zheng Wen, et al. A tutorial on thompson sampling. *Foundations and Trends® in Machine Learning*, 11(1): 1–96, 2018.

Max Schwarzer, Johan Samir Obando Ceron, Aaron Courville, Marc G Bellemare, Rishabh Agarwal, and Pablo Samuel Castro. Bigger, better, faster: Human-level atari with human-level efficiency. In *International Conference on Machine Learning*, pages 30365–30380. PMLR, 2023.

Maciej Skorski. Bernstein-type bounds for beta distribution. *Modern Stochastics: Theory and Applications*, 10(2):211–228, 2023.

Malcolm Strens. A bayesian framework for reinforcement learning. In *International Conference on Machine Learning*, pages 943–950, 2000.

William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Hado P Van Hasselt, Matteo Hessel, and John Aslanides. When to use parametric models in reinforcement learning? *Advances in Neural Information Processing Systems*, 32, 2019.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering atari games with limited data. *Advances in Neural Information Processing Systems*, 34: 25476–25488, 2021.

Appendix A. `HyperFQI` algorithm details

In this section, we describe more details of the proposed `HyperFQI`. First, we describe the general treatment for the `update` function (in line 11 of `HyperFQI`) in the following Algorithm 2. Then, in Appendix A.1, we provide the implementation details of `HyperFQI` with deep neural network (DNN) function approximation. We want to emphasize that all experiments done in this article is using **Option 1** with DNN value function approximation. In Appendix A.2, we describe the closed-form update rule (**Option 2**) when the tabular representation of the value function is exploited. Note that the tabular version of `HyperFQI` is only for the clarity of analysis and understanding.

Algorithm 2 update

```
1: Input: buffer $D$, $\theta, \theta^-, \boldsymbol{\xi}^-$, agent step $t$, train step $j$
2: if $t \mod \text{training\_freq} = 0$ then
3:   repeat
4:     Obtain $\theta$ by optimizing the loss $L^{\gamma,\sigma,\beta}(\theta; \theta^-, \boldsymbol{\xi}^-, D)$ in Equation (4):
       – Option (1) with gradient descent w.r.t. the mini-batch sampled loss Equation (5);
       – Option (2) with closed-form solution in Equations (12) and (13).
5:     Increment $j \leftarrow j + 1$
6:     if $(j \mod \text{target\_update\_freq}) = 0$ then
7:       $\theta^- \leftarrow \theta$
8:     end if
9:   until $(j \mod \text{sample\_update\_ratio} \times \text{training\_freq}) = 0$
10: end if
11: Return: $\theta, \theta^-, j$.
```

Notice that in `update`, there are three important hyper-parameters (target_update_freq, sample_update_ratio, training_freq), which we will specify in Table 2 for practical implementation of `HyperFQI` for all experiments; and in Table 3 for understanding and regret analysis.

A.1 Function approximation with deep neural networks

Here we describe the implementation details of HyperFQI with deep neural networks and the main difference compared to baselines.

A.1.1 Network architecture of HyperFQI

In our implementation, we only apply hypermodel to the output layer of our network, which will not result in much parameters and provide better expectation estimate. Suppose the hidden layers in neural networks forms the nonlinear feature mapping $\phi_w(\cdot)$ with parameters w. Our hypermodel takes the random index $\xi \in \mathbb{R}^M$ from reference distribution P_ξ and outputs the weights for output layer. It's worth to note that our hypermodel only outputs the weights but not bias for output layer, which is indicated by following equation with trainable $\theta = \{\boldsymbol{A}, b, w\}$ and fixed parameters $\{\boldsymbol{A}_0, b_0, w_0\}$

$$\begin{aligned} f_\theta(x, \xi) &= \underbrace{\langle \boldsymbol{A}\xi + b, \phi_w(x) \rangle}_{\textbf{Learnable } f_\theta^L(x,\xi)} + \underbrace{\langle \boldsymbol{A}_0\xi + b_0, \phi_{w_0}(x) \rangle}_{\textbf{Fixed prior } f^P(x,\xi)} \\ &= \underbrace{\langle \boldsymbol{A}\xi, \phi_w(x) \rangle}_{\sigma_\theta^L(x,\xi)} + \underbrace{\langle \boldsymbol{A}_0\xi, \phi_{w_0}(x) \rangle}_{\sigma^P(x,\xi)} + \underbrace{\langle b, \phi_w(x) \rangle}_{\mu_\theta^L(x)} + \underbrace{\langle b_0, \phi_w(x) \rangle}_{\mu^P(x)}. \end{aligned} \tag{9}$$

Through our formulation in Equation (9), HyperFQI can accurately estimate the learnable mean $\mu_\theta^L(x)$, which relies solely on the original input x, and the variation prediction $\sigma_\theta^L(x, \xi)$, which is dependent on both the original input x and random index ξ. This allows our hypermodel to capture uncertainty better, without being influenced by other components that may only depend on the random index ξ like HyperDQN (Li et al., 2022a). The fixed prior model also offers prior bias and prior variation through the functions $\mu^P(x)$ and $\sigma^P(x, \xi)$. This prior function is NOT trainable so that it will not bring much computation, and designed to provide better exploration in the early stage of training. We use Xavier Normal method to initialize our entire network except the part of prior function which is corresponding to hypermodel. For the initialization of prior model, we follow the method described in (Li et al., 2022a; Dwaracherla et al., 2020). In this way, each row of prior function is sampled from the unit hypersphere, which guarantees that the output of prior function can follow a desired Gaussian distribution.

In the context of reinforcement learning, we define the action-value function with hypermodel and DNN approximation as following. For each action $a \in \mathcal{A}$, there is a set of trainable parameters $\{\mathbf{A}^a, b^a\}$ and fixed parameters $\{\mathbf{A}_0^a, b_0^a\}$, i.e., the trainable set of parameters $\theta = \{w, (\mathbf{A}^a, b^a) : a \in \mathcal{A}\}$ and the fixed one $\{w_0, (\mathbf{A}_0^a, b_0^a) : a \in \mathcal{A}\}$ with action-value function

$$f_\theta(s, a, \xi) = \underbrace{\langle \boldsymbol{A}^a\xi + b^a, \phi_w(s) \rangle}_{\textbf{Learnable } f_\theta^L(s,a,\xi)} + \underbrace{\langle \boldsymbol{A}_0^a\xi + b_0^a, \phi_{w_0}(s) \rangle}_{\textbf{Fixed prior } f^P(s,a,\xi)}.$$

A.1.2 Training details for HyperFQI

For the estimation of expectation in Equation (4), we sample multiple random indices for each data tuple in the mini-batch and compute the empirical average, as described in Equation (5). Recall that $|\tilde{\Xi}|$ is the number of random indices for each state.

For the detail of sampling scheme for $\boldsymbol{\xi}_k(s)$, we have two options:

1. **State-dependent sampling.** As for implementation, especially for continuous or uncountable infinite state space: in the interaction, $\boldsymbol{\xi}_k(s)$ in the line 7 of `HyperFQI` is implemented as independently sampling $\xi \sim P_\xi$ for each encountered state; for the target computation in Equation (3), $\boldsymbol{\xi}_k(s')$ is implemented as independently sampling $\xi \sim P_\xi$ for each tuple $d = (s, a, r, s', \mathbf{z})$ in the every sampled mini-batch.

2. **State-independent sampling.** The implementation of state-independent $\boldsymbol{\xi}_k(s) = \xi_k$ is straightforward as we independently sample ξ_k in the beginning of each episode k and use the same ξ_k for each state s encountered in the interaction and for each target state s' in target computation.

We tuned some hyper-parameters on our HyperFQI and listed them in Table 2, and other hyper-parameters for Atari games are the same as Rainbow Hessel et al. (2018). Notice that we use a single configuration for all Atari games we test. Also we use a single configuration for the deepsea environments with variant sizes.

Hyper-parameters	Atari Setting	DeepSea Setting
discount factor γ	0.99	0.99
learning rate	0.001	0.001
minibatch size $\lvert\tilde{D}\rvert$	32	128
index dim M	4	4
# Indices $\lvert\tilde{\Xi}\rvert$ for approximation	20	20
n-step target	5	1
target_update_freq in `update`	5	4
sample_update_ratio in `update`	1	1
training_freq in `update`	1	1
hidden units	256	64
min replay size for sampling	2,000 steps	128 steps
memory size	500,000 steps	1000000 steps

Table 2: Hyper-parameters of our HyperFQI

As per the framework of HyperFQI, we can generate various variants. By default, a Gaussian distribution is used for P_ξ, but this can be changed to a one-hot index, referred to as HyperFQI-OH.

The action selection and computation of Q-target can also be modified by sampling multiple indexes and computing multiple Q-values for each action under one exact state. The optimal action is then selected based on these multiple Q-values. This variant is called HyperFQI-OIS and is described in the following.

Optimistic Index Sampling. To make agent's behavior more optimistic, in each episode k, we can sample N_{OIS} indices $\xi_{k,1}, \ldots, \xi_{k,N_{\mathrm{OIS}}}$ and take the greedy action according to the associated hypermodel

$$a_k = \arg\max_{a\in\mathcal{A}} \max_{n\in[N_{\mathrm{OIS}}]} f_\theta(s_k, a, \xi_{k,n}), \tag{10}$$

which we call optimistic index sampling (OIS) action selection scheme.

In the hypermodel training part, for any transition tuple $d = (s, a, r, s', \mathbf{z})$, we also sample multiple indices $\xi_1^-, \ldots, \xi_{N_{\mathrm{OIS}}}^-$ independently and modify the target computation in Equation (3) as

$$r + \sigma\xi^\top \mathbf{z} + \gamma \max_{a'\in\mathcal{A}} \max_{n\in[N_{\mathrm{OIS}}]} f_{\theta^-}(s', a', \xi_n^-(s')). \tag{11}$$

This modification in target computation boosts the propagation of uncertainty estimates from future states to earlier states, which is beneficial for deep exploration. We call this variant **HyperFQI-OIS**. For the completeness, we also specify the sampling scheme for

$\boldsymbol{\xi}_k(s)$ in the line 7 of `HyperFQI` and the target computation in Equation (3) for HyperFQI-OIS in the following.

1. **State-dependent sampling.** In the interaction, $\boldsymbol{\xi}_k(s)$ in the line 7 of `HyperFQI` is implemented as independently sampling $\xi_{k,n} \sim P_\xi$ for each encountered state s and for each index $n \in [N_{\text{OIS}}]$; for the target computation in Equation (11), $\boldsymbol{\xi}_{k,n}(s')$ is implemented as independently sampling $\xi_n \sim P_\xi$ for each tuple $d = (s, a, r, s', \mathbf{z})$ in the every sampled mini-batch.

2. **State-independent sampling.** For each $n \in N_{\text{OIS}}$, the implementation of state-independent $\boldsymbol{\xi}_{k,n}(s) = \xi_{k,n}$ is straightforward as we independently sample $\xi_{k,n}$ in the beginning of each episode k and use the same $\xi_{k,n}$ for each state s encountered in the interaction and for each target state s' in target computation.

A.1.3 Difference compared to prior works

Several related work can be included in the hypermodel framework introduced in Section 2.2. We will discuss the structural differences under the unified framework in this sections. Furthermore, we performs ablation studies concerning these mentioned differences in Figure 11.

Difference with Hypermodel (Dwaracherla et al., 2020). Hypermodel (Dwaracherla et al., 2020) employs hypermodels to represent epistemic uncertainty and facilitate exploration. However, the use of hypermodels over entire networks leads to an extensive number of parameters and optimization challenges. As a result, applying HyperModel (Dwaracherla et al., 2020) to address large-scale problems such as Atari games can be extremely difficult. Additionally, HyperModel also encounters challenges in addressing the DeepSea problem due to its substantial state space, even when the space is relatively small. Our HyperFQI only apply hypermodel to the output layer, which provide better exploration and efficient computation in large scale problem.

Difference with BootDQN (Osband et al., 2018). BootDQN (Osband et al., 2018) also applies an ensemble method to the entire network, utilizing a random mask from a Bernoulli distribution to update the network. When combined with prior network, it has demonstrated effective exploration in chain environments with large sizes and other tasks that necessitate exploration. However, BootDQN requires a relatively large ensemble size to achieve effective exploration, which presents similar challenges to the HyperModel, namely, dealing with an extensive number of parameters and optimization issues. BootDQN is akin to a variant of HyperModel (Dwaracherla et al., 2020) in which the index is sampled from a Bernoulli distribution. Typically, an ensemble size of 10 is chosen for BootDQN to ensure effective exploration. However, even with an ensemble size of 16, BootDQN fails to efficiently solve the DeepSea, as depicted in Figure 9. Our HyperFQI applies the hypermodel solely to the final output layer of the network, with our index dimension set at 4, leading to a reduction in network parameters.

Difference with HyperDQN (Li et al., 2022a). HyperDQN (Li et al., 2022a) shares a similar structure with our HyperFQI and has shown promising results in exploration. Nevertheless, it falls short in handling the DeepSea environment, which demands deep exploration. We have improved HyperDQN by simplifying our hypermodel, as shown in equation (9). Our HyperFQI estimates the mean μ solely based on the original input x, and estimates the variation σ based on both the original input x and a random index ξ. However, HyperDQN use hypermodel to generate the both weights and bias for output layer, resulting in some redundant components, such as functions that depend solely on the random index ξ or functions that depend only on the parameters of the hypermodel. These components lack a clear semantic explanation, rendering HyperFQI unsuitable for estimating uncertainty.

Additionally, we apply multiple indexes $|\tilde{\Xi}|$ to each transition in update stage to improve the expectation estimate, whereas HyperDQN only applies them to batch transitions. We have found that initializing the hypermodel with Xavier Normal can improve optimization. The combination of these factors leads to our HyperFQI outperforming HyperDQN on both DeepSea and Atari, as demonstrated in Section 4.

Differnce with Episdemic Neural Networks (ENN) (Osband et al., 2021, 2023). The ENN approach, as described in Osband et al. (2023), shows promise for capturing epistemic uncertainty and has demonstrated efficiency on various tasks. We have implemented the ENNDQN by their description on bsuite (Osband et al., 2019a). Except for the update method and network structure, other settings are the same as our HyperFQI. In the update stage, they use "stop gradient" between feature layers and final ENN layers. For the network structure, they concatenate the original input x, feature $\phi(x)$ and random index ξ as the input for the ENN layer, and use ensemble prior function for ENN layer but don't have prior function for feature network. The network structure leads to larger parameters when processing tasks at a large scale, causing significant computation and optimization challenges. As shown in Section 4.1, ENNDQN performs well on DeepSea-20 but struggles with larger scale of the problem. This difficulty arises because the input x in DeepSea has a dimension of N^2, which is too large for the ENN layer to handle. We designed our HyperFQI to take only a random index ξ as input, resulting in a more efficient computation with fewer parameters.

A.2 Tabular algorithm

To understand and analyze the behavior of `HyperFQI`, we specify the algorithm in the tabular setups.

Tabular HyperFQI. Let the tabular hypermodel be $f_\theta(s,a,\xi) = \mu_{sa} + m_{sa}^\top \xi + \mu_{0,sa} + \sigma_0 \mathbf{z}_{0,sa}^\top \xi$ where $\theta = (\mu \in \mathbb{R}^{SA}, m \in \mathbb{R}^{SA \times M})$ are the parameters to be learned, and $\mathbf{z}_{0,sa} \in \mathbb{R}^M$ is a independent random vector sampled from $P_{\mathbf{z}}$ and $\mu_{0,sa}, \sigma_0$ is a prior mean and prior variance for each $(s,a) \in \mathcal{S} \times \mathcal{A}$. The regularizer in Equation (4) then becomes $\beta \|\theta\|^2 = \beta \sum_{s,a} \left(\mu_{sa}^2 + \|m_{sa}\|^2\right)$. Let the set $E_{k,sa}$ record the time index the agent encountered (s,a) in the k-th episode $E_{k,sa} = \{t : (S_{k,t}, A_{k,t}) = (s,a)\}$. Let $N_{k,sa} = \sum_{\ell=1}^{k-1} \sum_{t=0}^{H-1} \mathbb{1}_{(S_{\ell,t}, A_{\ell,t}) = (s,a)}$ denoting the counts of visitation for state-action pair (s,a) prior to episode k.

Closed-form incremental update. Let $\beta = \sigma^2/\sigma_0^2$. Then, given the dataset $D = H_k$ and target noise mapping $\boldsymbol{\xi}^- = \boldsymbol{\xi}_k$ at the beginning of the episode k, `HyperFQI` with hyperparameters specified in Table 3 would yield the following closed-form iterative procedure

$\theta_k^{(i)} = (\mu_k^{(i)}, m_k) \to \theta_k^{(i+1)} = (\mu_k^{(i+1)}, m_k)$ for all $i = 1, 2, \dots, H$: for all $(s, a) \in \mathcal{S} \times \mathcal{A}$

$$m_{k,sa} = \frac{\sigma \sum_{\ell=1}^{k-1} \sum_{t \in E_{\ell,sa}} \mathbf{z}_{\ell,t+1} + \beta \sigma_0 \mathbf{z}_{0,sa}}{N_{k,sa} + \beta}, \tag{12}$$

$$\mu_{k,sa}^{(i+1)} = \frac{\sum_{\ell=1}^{k-1} \sum_{t \in E_{\ell,sa}} y_{\ell,t+1}(\theta^- = \theta_k^{(i)}, \boldsymbol{\xi}^- = \boldsymbol{\xi}_k) + \beta \mu_{0,sa}}{N_{k,sa} + \beta}, \tag{13}$$

where $y_{\ell,t+1}(\theta^-, \boldsymbol{\xi}^-) = R_{\ell,t+1} + \gamma \max_{a' \in \mathcal{A}} f_{\theta^-}(S_{\ell,t+1}, a', \boldsymbol{\xi}^-(S_{\ell,t+1}))$. The derivation of

Hyper-parameters	Finite MDP with Horizon H
discount factor γ	1
target_update_freq	1
sample_update_ratio	1
training_freq	H

Table 3: Hyper-parameters of our Tabular-HyperFQI

Equations (12) and (13) is mainly from the separability of optimization problem in tabular setup. Let $\theta_{sa} = (\mu_{sa}, m_{sa})$ be the optimization variable for specific $(s, a) \in \mathcal{S} \times \mathcal{A}$. Let the optimal solution $\theta_{k,sa} = \arg\min_{\theta_{sa}} L(\theta; \theta^-, \boldsymbol{\xi}^-, H_k)$. In tabular setting, by the separability of the objective function in Equation (4), we have $\theta_{k,sa} = (\mu_{k,sa}, m_{k,sa}) = \arg\min_{\theta_{sa}} L_{sa}(\theta_{sa}; \theta^-, \boldsymbol{\xi}^-, H_k)$ which is defined as

$$\begin{aligned}
&\mathbb{E}_{\xi \sim P_\xi}[\sum_{\ell=1}^{k-1} \sum_{t \in E_{\ell,sa}} (f_\theta(S_{\ell,t}, A_{\ell,t}, \xi) - (\sigma \xi^\top \mathbf{z}_{\ell,t+1} + y_{\ell,t+1}(\theta^-, \boldsymbol{\xi}^-)))^2] + \beta(\mu_{sa}^2 + \|m_{sa}\|^2) \\
&= \mathbb{E}_{\xi \sim P_\xi}[\sum_{\ell=1}^{k-1} \sum_{t \in E_{\ell,sa}} (f_\theta(s, a, \xi) - (\sigma \xi^\top \mathbf{z}_{\ell,t+1} + y_{\ell,t+1}(\theta^-, \boldsymbol{\xi}^-)))^2] + \beta(\mu_{sa}^2 + \|m_{sa}\|^2).
\end{aligned}$$

A.2.1 Understanding through Bellman equation

In this section, we provide a deeper understanding of `HyperFQI` via the lens of Bellman equations. First, we introduce the empirical transition operators that depends on the collected data by algorithm.

Empirical transition. For every pair (s, a) with $N_{k,sa} > 0$, $\forall s' \in \mathcal{S}$, the empirical transition probabilities up to pseudo-episode k are

$$\hat{P}_k(s' \mid s, a) = \frac{1}{N_{k,sa}} \sum_{\ell=1}^{k-1} \sum_{t \in E_{\ell,sa}} \mathbb{1}_{(S_{\ell,t}, A_{\ell,t}, S_{\ell,t+1}) = (s, a, s')}.$$

If the pair (s, a) has never been sampled before pseudo-episode k, we define $\hat{P}_k(s' \mid s, a) = 1$ for any selected $s' \in \mathcal{S}$, and $\hat{P}_k(s'' \mid s, a) = 0$ for $s'' \in \mathcal{S} \setminus \{s'\}$.

Next, we introduce some short notations for the convenience of specifying bellman operators.

Short notation. Let V_Q be a vector of S-dim such that $V_Q(s) = \max_{a \in \mathcal{A}} Q(s,a)$ for all $s \in \mathcal{S}$, which is the greedy value with respect to Q. Let $P_{sa} = [P(s' \mid s,a)]_{s' \in \mathcal{S}}$ and $\hat{P}_{k,sa} = [\hat{P}_k(s' \mid s,a)]_{s' \in \mathcal{S}}$ be the vectors of S-dim. Denote the short notation $r_{sa} = r(s,a)$. We also use the short notation $\hat{P}_{k,sa} = [\hat{P}_k(s' \mid s,a)]_{s' \in \mathcal{S}}$ which is S-dim vector. Now we are ready to introduce the Bellman operator underlying the true environment $\mathcal{E}$ and a notion of a Bellman operator that represents the iterative procedure induced by Tabular `HyperFQI`.

True Bellman Operator. For any $\mathcal{E} = (\mathcal{S}, \mathcal{A}, P)$,

$$F_{\mathcal{E}}^{\gamma} Q(s,a) = r_{sa} + \gamma V_Q^\top P_{sa}, \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A}. \tag{14}$$

Bellman operator of HyperFQI. We define the bellman operator according to HyperFQI in Algorithm 1 as

$$F_k^{\gamma} Q(s,a) := \frac{\beta \mu_{0,sa} + N_{k,sa}(r_{sa} + \gamma V_Q^\top \hat{P}_{k,sa})}{N_{k,sa} + \beta} + m_{k,sa}^\top \boldsymbol{\xi}_k, \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A}. \tag{15}$$

With the following observation,

$$\sum_{\ell=1}^{k-1} \sum_{t \in E_{\ell,sa}} y_{\ell,t+1}(\theta^-, \boldsymbol{\xi}^-) = N_{k,sa} \left(r_{sa} + \gamma \sum_{s' \in \mathcal{S}} \hat{P}_{k,sa}(s') (\max_{a' \in \mathcal{A}} f_{\theta^-}(s', a', \boldsymbol{\xi}^-(s'))) \right)$$

Then Equation (13) has a simple expression with the bellman operator of HyperFQI,

$$f_{\theta_k^{(i+1)}, \boldsymbol{\xi}_k} = F_k^{\gamma} f_{\theta_k^{(i)}, \boldsymbol{\xi}_k} \tag{16}$$

where $f_{\theta^-, \boldsymbol{\xi}^-} = f_{\theta^-}(\boldsymbol{\xi}^-(\cdot), \cdot, \cdot)$.

Lemma A.1 (Contraction mapping). *Let $B(\mathcal{S} \times \mathcal{A})$ be the space of bounded functions $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and ρ be the distance metric $\rho(Q, Q') = \sup_{(s,a) \in \mathcal{S} \times \mathcal{A}} |Q(s,a) - Q'(s,a)|$. For all $k \in \mathbb{Z}_{++}$ the Bellman operator of HyperFQI $F_k^{\gamma} : B(\mathcal{S} \times \mathcal{A}) \to B(\mathcal{S} \times \mathcal{A})$ is a contraction mapping with modulus $\gamma \in [0,1)$ in metric space $(B(\mathcal{S} \times \mathcal{A}), \rho)$.*

By Lemma A.1, since contraction mapping, the bellman operator of HyperFQI F_k^{γ} has a unique fixed point and the iterative process in Equation (16) can converge.

Appendix B. Environment Settings

In this section, we describe our environment used in experiments. We firstly use the DeepSea to demonstrate the exploration efficiency of our HyperFQI. DeepSea is a reward-sparse environment that demands extensive exploration (see Figure 6). The environment under consideration has a discrete action space consisting of two actions: moving left or right. During each run of the experiment, the action for moving right is randomly sampled from Bernoulli distribution for each row. Specifically, the action variable takes binary values of 1 or 0 for moving right, and the action map is different for each run of the experiment. The agent receives a reward of 0 for moving left, and a penalty of $-(0.01/N)$ for moving right, where N denotes the size of DeepSea. The agent will earn a reward of 1 upon reaching the lower-right corner. The optimal policy for the agent is to learn to move continuously to the right. The sparse rewards and states present in this environment effectively showcase the exploration efficiency of our method without any additional complexity.

For the experiments on the Atari games, we evaluate our HyperFQI on 26 of the 55 games from the full ALE suite. We utilized the standard wrapper provided by OpenAI gym. For example, we terminated each environment after a maximum of 108K steps without using sticky actions. For further details on the settings used for the Atari games, please refer to the Table 4.

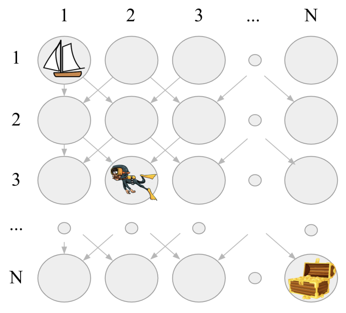


Figure 6: Illustration for DeepSea.

Hyper-parameters	Setting
Grey-scaling	True
Observation down-sampling	(84, 84)
Frames stacked	4
Action repetitions	4
Reward clipping	[-1, 1]
Terminal on loss of life	True
Max frames per episode	108K

Table 4: Detailed settings for Atari games

Appendix C. Additional Results

This section presents additional results for our HyperFQI algorithm. We demonstrate its robustness through ablation experiments on DeepSea. Moreover, we provide detailed results for each environment of Atari, highlighting the superiority of our approach over other baselines.

C.1 Results on DeepSea

In Section 4.1, we noted that a larger M in the Gaussian distributional index can harm the algorithm's performance due to increased difficulty in estimating the expectation. To address this, we can increase the number of indices $|\tilde{\Xi}|$ of Equation (5) during the update stage for each state in the batch, thereby improving the estimation of expectation. As shown in Figure 7 for $M = 16$, increasing $|\tilde{\Xi}|$ can lead to better performance. However,

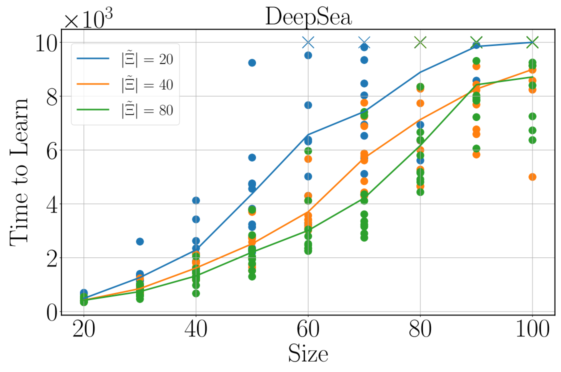


Figure 7: Ablation results under different $|\tilde{\Xi}|$.

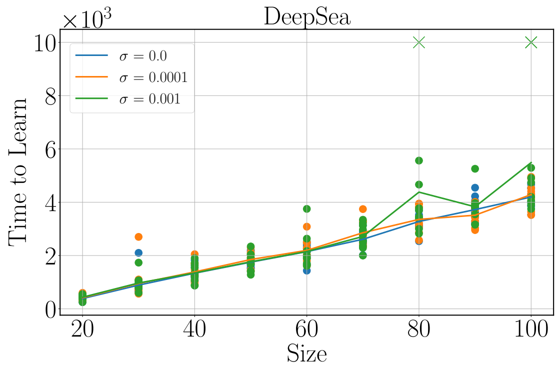


Figure 8: Ablation results under different σ.

some seeds with $|\tilde{\Xi}| = 80$ still do not work well for $M = 16$. To achieve accurate estimation of expectation, $|\tilde{\Xi}|$ should grow exponentially with M, but this comes at the cost of increased computation, slowing down the algorithm. To balance performance and computation, we have chosen $M = 4$ and $|\tilde{\Xi}| = 20$ as our default hyper-parameters, which have demonstrated superior performance in Figure 2.

In addition, we have also investigated the effect of the σ of Equation (3) on our methods, as shown in Figure 8. Our HyperFQI is not sensitive to this hyper-parameter, and we have selected $\sigma = 0.0001$ as our default hyper-parameters.

We observed that HyperFQI-OIS method performs better when using Gaussian distributional index, but not with one-hot index in Figure 2(b). This is because $M = 4$ cannot accurately estimate the expectation when using one-hot index. However, We observed that HyperFQI with one-hot index achieves good performance when $M = 16$ in Figure 3(b). Thus, we evaluated the efficiency of HyperFQI-OH and HyperFQI-OIS-OH with one-hot index when $M = 16$. Furthermore, we incorporated optimistic index sampling (OIS) into BootDQN, labeling the modified version as BootDQN-OIS. Subsequently, we compared our HyperFQI with both BootDQN and BootDQN-OIS, each employing an ensemble size of 16. It's worth noting that BootDQN-OIS can be regarded as an implementation of LSVI-PHE (Ishfaq et al., 2021) with DNN.[3]

As depicted in Figure 9, BootDQN still struggled to efficiently solve the DeepSea with large ensemble size of 16. Nonetheless, the integration of the OIS method enabled BootDQN-OIS to effectively tackle DeepSea with a smaller size, showcasing the efficacy of our OIS approach in exploration. Additionally, with a relatively large $M = 16$ for one-hot index, our HyperFQI-OH actually achieves better performance. In contrast, our HyperFQI approach efficiently solves DeepSea with index dimension $M = 4$, demonstrating its superiority. These experiments again justify that `HyperFQI` can efficiently solve large-scale complex environment with low computation cost.

We also conduct an ablation experiment concerning the $\boldsymbol{\xi}_k$ using DeepSea. In our implementation, we employ the state-dependent $\boldsymbol{\xi}_k$, where we independently sample $\xi \sim P_\xi$

3. As we do not find the official implementation of LSVI-PHE (Ishfaq et al., 2021) anywhere.We use results of BootDQN-OIS to represent the performance of LSVI-PHE.

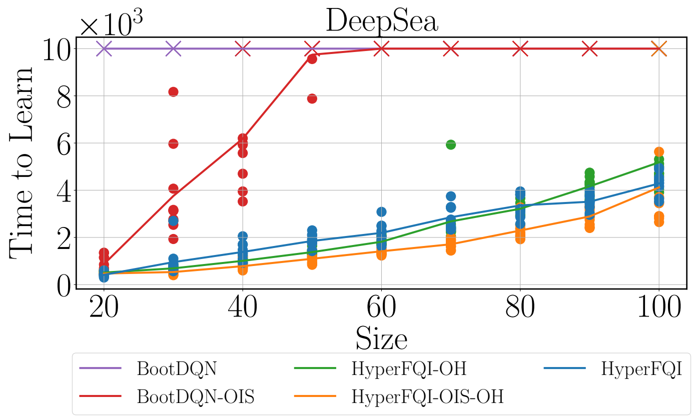


Figure 9: Results on DeepSea with additional baselines. In this experiment, we set $M = 16$ for all algorithms except for HyperFQI.

for each transition. Furthermore, we assess our HyperFQI with state-independent $\boldsymbol{\xi}_k$, where we sample ξ_k at the start of each episode k and use the same ξ_k for all transitions within the episode, referring to as HyperFQI-SAME. As illustrated in Figure 10, these two distinct sampling schemes for $\boldsymbol{\xi}_k$ exhibit nearly identical performance.

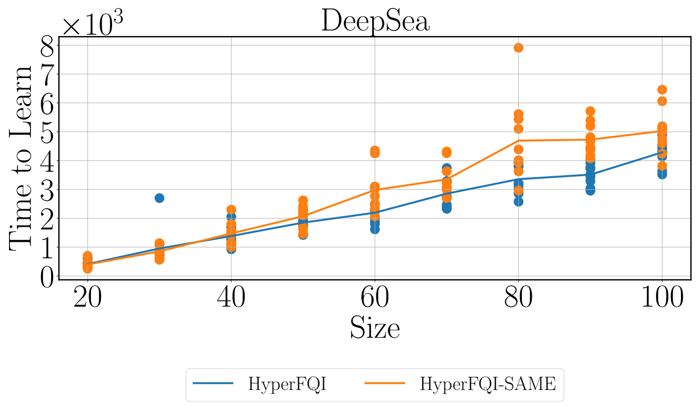


Figure 10: Ablation results about $\boldsymbol{\xi}_k$ on DeepSea.

We perform an ablation experiment examining various network structures outlined in Appendix A.1.3 with the same hyper-parameters, algorithmic update rule and action selection rule. Initially, we compare our HyperFQI with HyperModel (Dwaracherla et al., 2020), which integrates the hypermodel across the entire network. Subsequently, we compare it with ENNDQN, which applies the ENN layer (Osband et al., 2021) to the final output layer, requiring the original input x as the input for the ENN layer. Both of the baselines encounter challenges due to their network architectures, possibly due to extensive number of parameters and the accompanying optimization issues. The comparison results are presented in Figure 11. It is clear that HyperModel is incapable of solving DeepSea even with a size of 20. In contrast, the network structure of ENNDQN contains fewer parameters than HyperModel, as it only applies the ENN layer to the final output layer. However, ENNDQN is unable to solve DeepSea with a larger size, as the ENN layer necessitates the use of the original input x. Overall, our HyperFQI demonstrates superb efficiency and scalability, as it efficiently solves DeepSea instances of size 100^2 that previous literatures have never achieved.

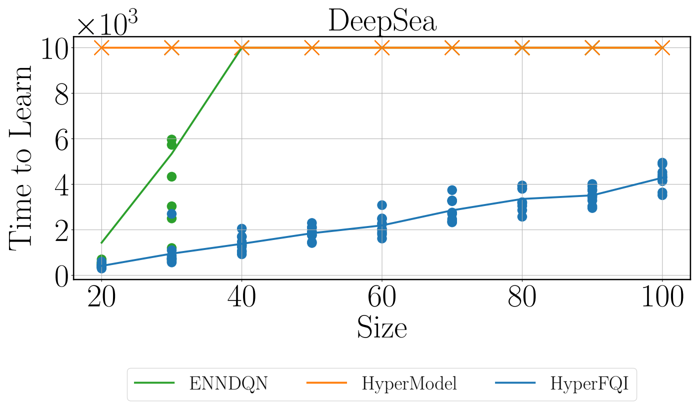


Figure 11: Ablation experimental results regarding the network structure.

C.2 Results on Atari

We demonstrated the efficiency of our HyperFQI in handling data and computation in Section 4.2. Here, we present comprehensive results on each environment to further establish the superiority of our approach.

In Table 5, we present the best score achieved in each environment with 2M steps. Our training and evaluation protocol follows the baseline works (Li et al., 2022a; Mnih et al., 2015; Van Hasselt et al., 2016). To be more specific, the protocol includes the following steps: (1) for each Atari game, the algorithm is performed with 20 different initial random seeds; (2) The program with one particular random seed will produce one best model in hindsight, leading to 20 different models for each Atari game; (3) We then evaluate all 20 models, each for 200 times; 4) We calculate the average score from these 200 evaluations as the score for each model associated with each seed. 5) Finally, we calculate and report the average score across 20 seeds as the final score for each Atari game. The scores for Rainbow and DDQN are obtained from Hessel et al. (2018), which were based on 200M Frames. Specifically, we extracted the first 20M steps from these results to compare them with our HyperFQI. For HyperDQN, we refer to the results from (Li et al., 2022a) and similarly extracted the first 20M steps for comparison purposes. The DER (Van Hasselt et al., 2019) was executed using the popular implementation available at `https://github.com/Kaixhin/Rainbow`. We conducted all experiments with 20 different seeds and computed the average best score with the best policy during training.

We also present the relative improvement of our HyperFQI in comparison to other baselines for each game, which is determined by the given following equation as per (Wang et al., 2016).

$$\text{relative improvement} = \frac{\text{proposed} - \text{baseline}}{\max(\text{human}, \text{baseline}) - \text{human}}$$

Game	Random	Human	DDQN	DER	Rainbow	HyperDQN	HyperFQI
Alien	227.8	7127.7	722.7	1642.2	1167.1	862.0	**1830.2**
Amidar	5.8	1719.5	61.4	476.0	374.0	140.0	**800.4**
Assault	222.4	742.0	815.3	488.3	2725.2	494.2	**3276.2**
Asterix	210.0	8503.3	2471.1	1305.3	3213.3	713.3	2370.2
BankHeist	14.2	753.1	7.4	460.5	411.1	272.7	430.3
BattleZone	2360.0	37187.5	3925.0	19202.5	19379.7	11266.7	**29399.0**
Boxing	0.1	12.1	26.7	1.7	69.9	6.8	**74.0**
Breakout	1.7	30.5	2.0	6.5	137.3	11.9	54.8
ChopperCommand	811.0	7387.8	354.6	1488.9	1769.4	846.7	**2957.2**
CrazyClimber	10780.5	35829.	53166.5	36311.1	110215.8	42586.7	**121855.8**
DemonAttack	152.1	1971.0	1030.8	955.3	45961.3	2197.7	**5852.0**
Freeway	0.0	29.6	5.1	32.8	32.4	30.9	32.2
Frostbite	65.2	4334.7	358.3	3628.3	3648.7	724.7	**4583.9**
Gopher	257.6	2412.5	569.8	742.1	4938.0	1880.0	**7365.8**
Hero	1027.0	30826.4	2772.9	15409.4	11202.3	9140.3	12324.7
Jamesbond	29.0	302.8	15.0	462.1	773.1	386.7	**951.6**
Kangaroo	52.0	035.0	134.9	8852.3	6456.1	3393.3	8517.1
Krull	1598.0	2665.5	6583.3	3786.7	8328.5	5488.7	8222.6
KungFuMaster	258.5	22736.3	12497.2	15457.0	25257.8	12940.0	23821.2
MsPacman	307.3	6951.6	1912.3	2333.7	1861.1	1305.3	**3182.3**
Pong	-20.7	14.6	-15.4	20.6	5.1	20.5	20.5
PrivateEye	24.9	69571.3	37.8	900.9	100.0	64.5	171.9
Qbert	163.9	13455.0	1319.4	12345.5	7885.3	5793.3	12021.9
RoadRunner	11.5	7845.0	3693.5	14663.0	33851.0	7000.0	28789.4
Seaquest	68.4	42054.7	367.6	662.0	1524.7	370.7	**2732.4**
UpNDown	533.4	11693.2	3422.8	6806.3	39187.1	4080.7	19719.2

Table 5: The best score over 200 evaluation episodes for the best policy in hindsight (after 2M steps) for Atari games. The performance of the random policy and the human expert is from dqn_zoo (Quan and Ostrovski, 2020).

Our classification of environments into three groups, namely “hard exploration (dense reward)”, “hard exploration (sparse reward)” and “easy exploration”, is based on the taxonomy proposed by Bellemare et al. (2016). The overall results are illustrated in Figure 12, Figure 13, Figure 14 and Figure 15.

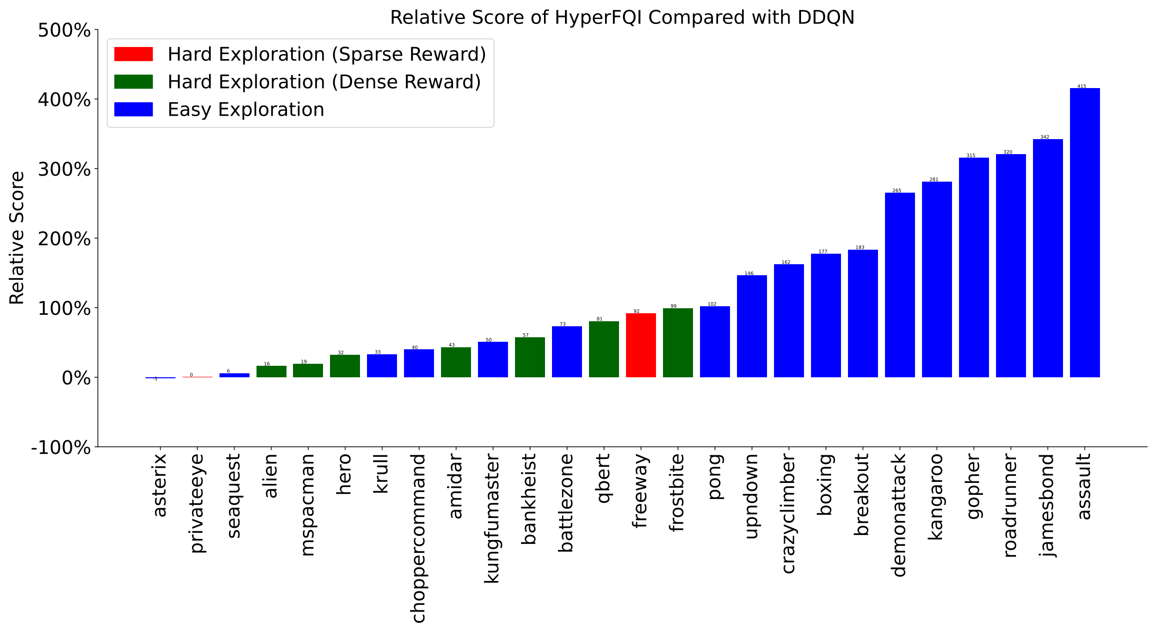


Figure 12: Relative improvement of HyperFQI compared with DDQN

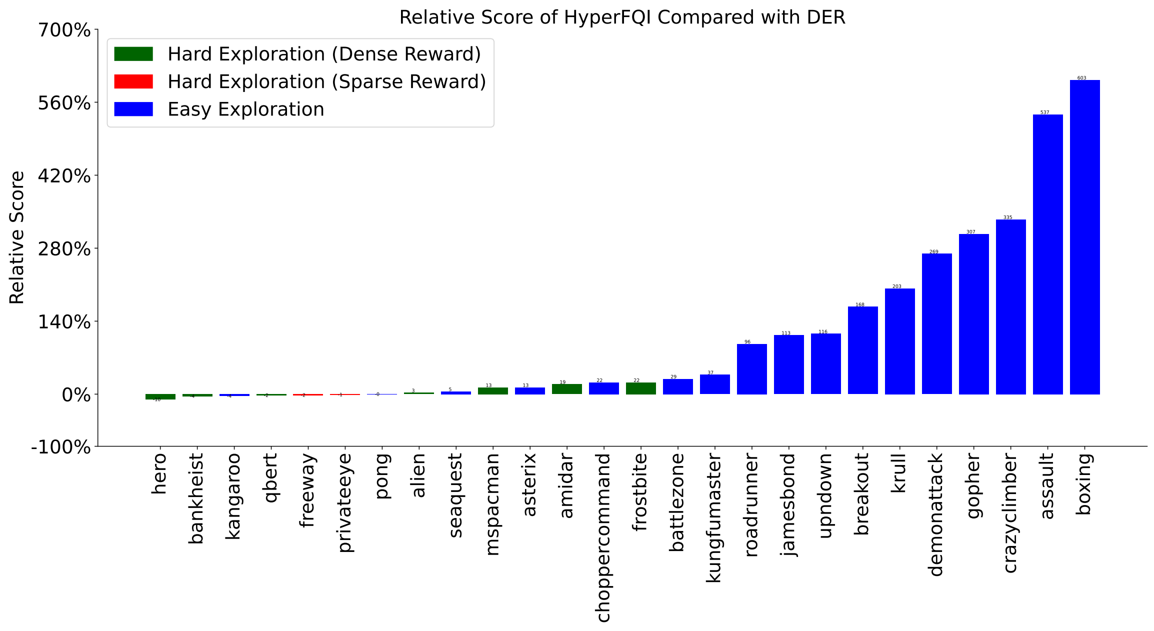


Figure 13: Relative improvement of HyperFQI compared with DER

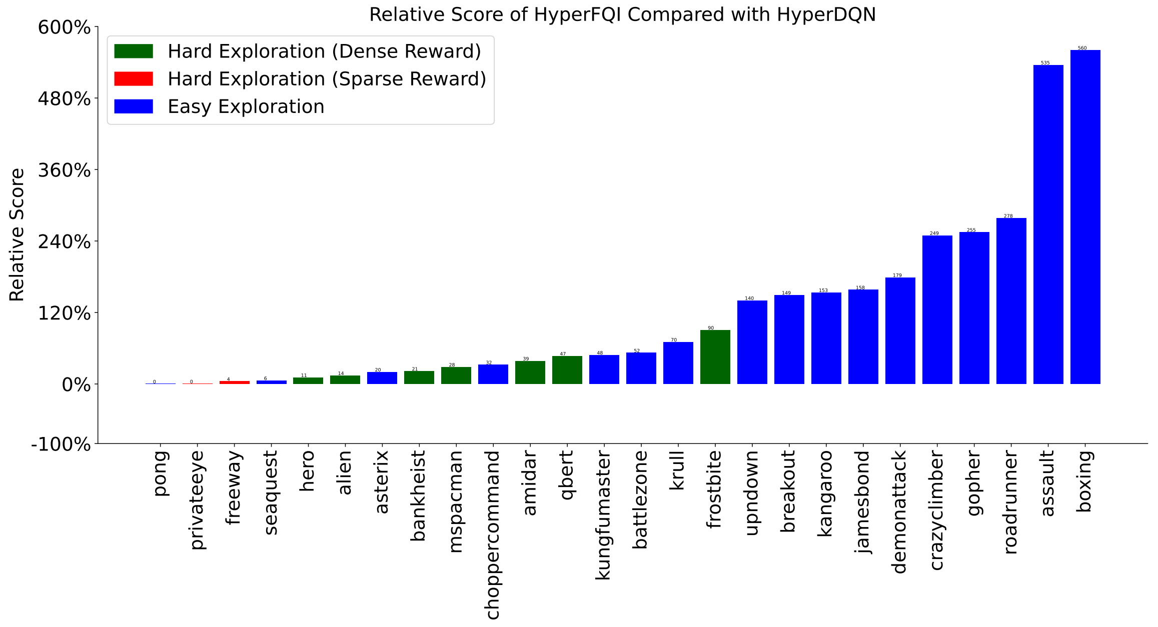


Figure 14: Relative improvement of HyperFQI compared with HyperDQN

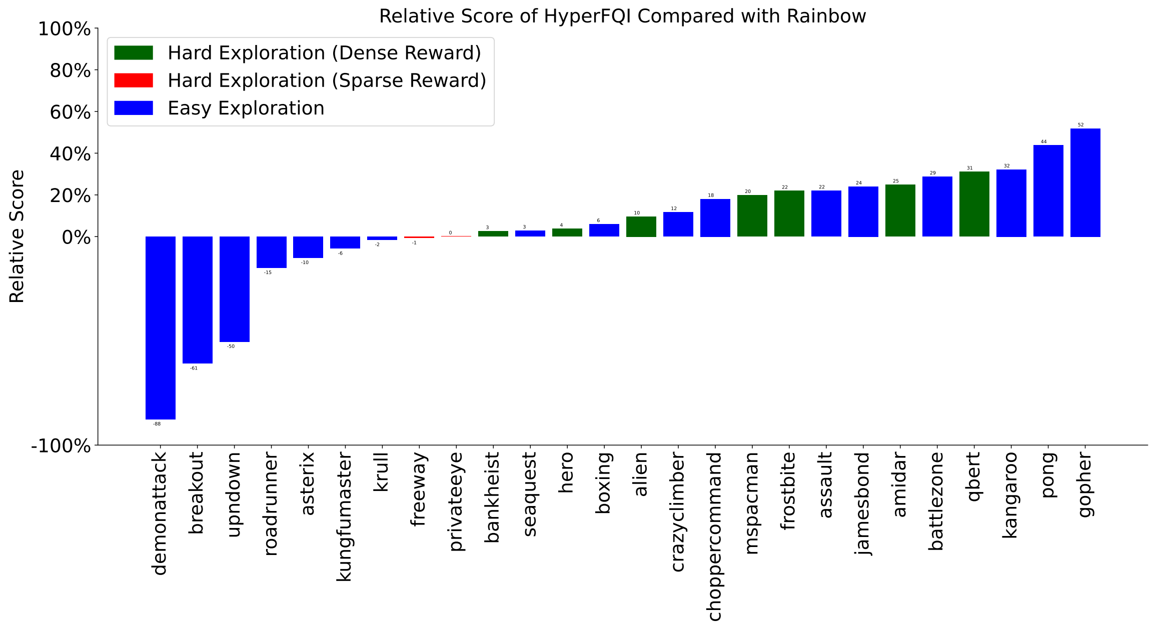


Figure 15: Relative improvement of HyperFQI compared with Rainbow

Our HyperFQI algorithm exhibits significant improvement compared to DDQN, DER, and HyperDQN in environments with "easy exploration", and overall it performs better in all environments. This indicates that HyperFQI has better generalization and exploration abilities. On the other hand, when compared to Rainbow, our algorithm performs better in environments which are in the group of "hard exploration (dense reward)", demonstrating our superior deep exploration capabilities. However, in the case of Freeway, which belongs to the "hard exploration (sparse reward)" group, both HyperFQI and Rainbow achieve similar optimal scores (as shown in Table 5), suggesting no significant improvement in this environment. Overall, our HyperFQI showcases better generalization and exploration efficiency.

Figure 16 illustrates the learning curve for each game. Our HyperFQI has shown superior performance in comparison to DDQN (ours), attributed to the incorporation of a hypermodel that enhances exploration in Atari games. Additionally, our HyperFQI variants demonstrated stable and efficient learning, as indicated by the results. The learning curves of these variants exhibit remarkable similarity, indicating the robustness of our HyperFQI on Atari games. However, our experiments have demonstrated that the HyperFQI-OIS outperforms the others in DeepSea, which necessitates deep exploration. Furthermore, it is worth highlighting that the learning curve of our algorithm continues to rise in certain environments, indicating that our HyperFQI can achieve even better performance with additional training.

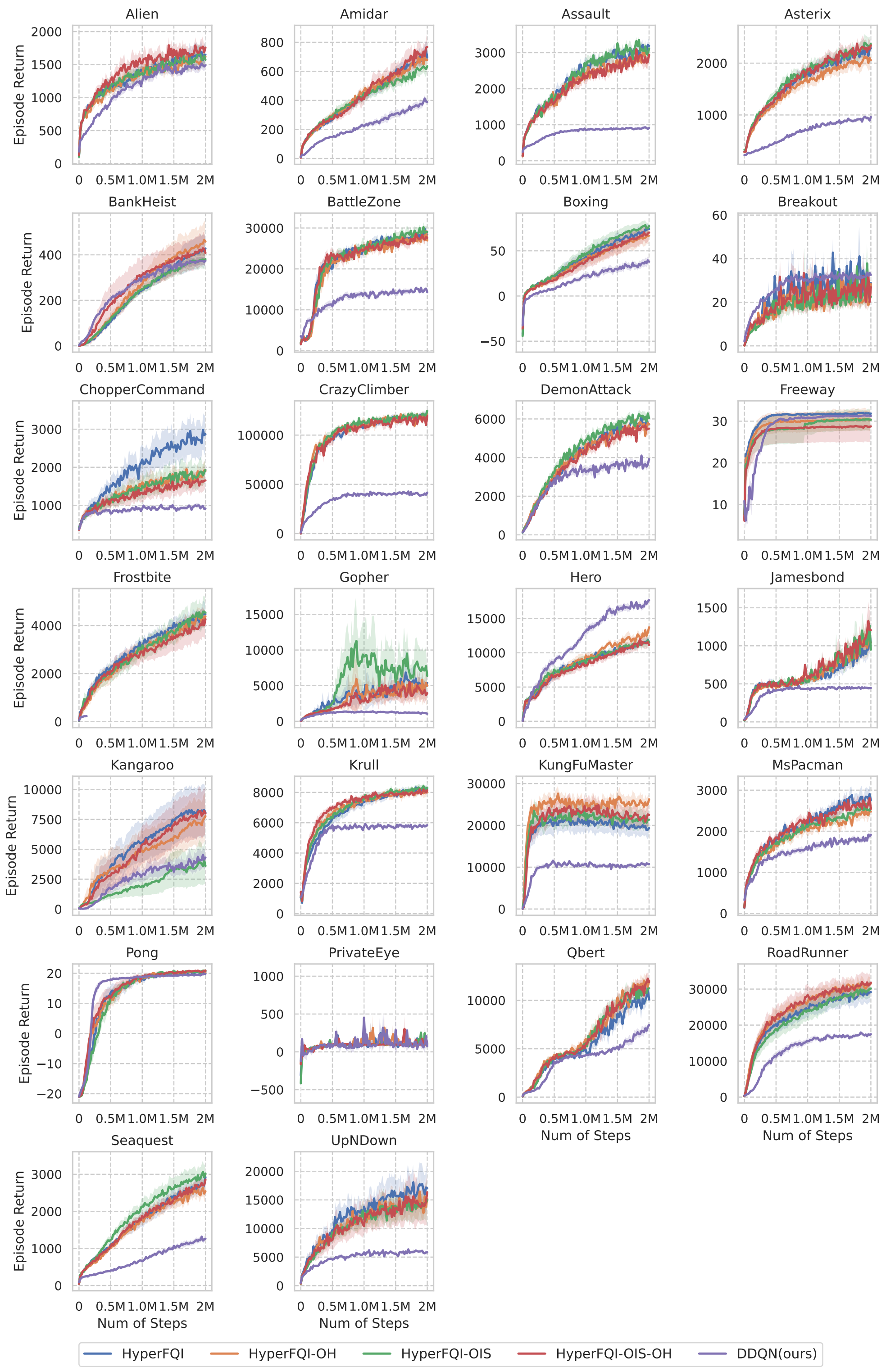


Figure 16: Learning curve for each game. All variants exhibit remarkable similarity, indicating the robustness of our HyperFQI on Atari games.

Appendix D. Probabilistic formalism

One of the difficulties in the analysis is to deal with the sequential dependence structure among the random variables generated from the reinforcement learning problems. We define some important concept that would be useful in the analysis.

Let $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\geq 0}, \mathbb{P})$ be a complete filtered probability space.

Definition D.1 (Adapted process)**.** For an index set I of the form $\{t \in \mathbb{N} : t \geq t_0\}$ for some $t_0 \in \mathbb{N}$, we say a stochastic process $(\mathrm{x}_t)_{t\in I}$ is adapted to the filtration $(\mathcal{F}_t)_{t\in I}$ if each x_t is $\mathcal{F}_t$-measurable.

Definition D.2 ((Conditionally) σ-sub-Gaussian)**.** We first describe the property associated with one-dimensional random variable. Second, we describe the generalization in high-dimension random vector.

- Random variables
  - We say a random variable x is σ-sub-Gaussian if

    $$\mathbb{E}[\exp(\lambda \mathrm{x})] \leq \exp\left(\frac{\lambda^2\sigma^2}{2}\right), \qquad \forall \lambda \in \mathbb{R}.$$

  - Let $(\mathbf{x}_t)_{t\geq 1} \subset \mathbb{R}^M$ be a stochastic process adapted to filtration $(\mathcal{F}_t)_{t\geq 1}$. Let $\sigma = (\sigma_t)_{t\geq 0}$ be a stochastic process adapted to filtration $(\mathcal{F}_t)_{t\geq 0}$. We say the process is $(\mathbf{x}_t)_{t\geq 1}$ is conditionally σ-sub-Gaussian if

    $$\mathbb{E}[\exp(\lambda \mathbf{x}_t) \mid \mathcal{F}_{t-1}] \leq \exp\left(\frac{\lambda^2\sigma_{t-1}^2}{2}\right), \quad a.s. \quad \forall \lambda \in \mathbb{R}.$$

    Specifically for the index $t+1$, we can say $\mathbf{x}_{t+1}$ is ($\mathcal{F}_t$-conditionally) σ_t-sub-Gaussian. If σ_t is a constant σ for all $t \geq 0$, then we just say (conditionally) σ-sub-Gaussian.
- Random vectors
  - For random vector $\mathbf{x}$ or vector process $(\mathbf{x}_t)_{t\geq 1}$, we say it is σ-sub-Gaussian is for every fixed $v \in \mathbb{S}^{M-1}$ if the random variable $\langle v, \mathbf{x}\rangle$ or stochastic process $\langle v, \mathbf{x}_t\rangle$ is σ-sub-Gaussian.

Definition D.3 (Almost sure unit-norm)**.** We say a random vector $\mathbf{x}$ is almost sure unit-norm if $\|\mathbf{x}\|_2 = 1$ almost surely.

Definition D.4 (c_x-bounded process)**.** For an index set I of the form $\{t \in \mathbb{N} : t \geq t_0\}$ for some $t_0 \in \mathbb{N}$, the stochastic process $(\mathrm{x}_t)_{t\in I}$ is c_x-bounded if $\mathrm{x}_t^2 \leq c_x$ almost surely for all $t \in I$.

Appendix E. Regret bound

Denote the regret of a policy π_k over episode k by

$$\Delta_k := \mathbb{E}_{M,\mathrm{alg}}[V_M^{\pi^*}(s_{k,0}) - V_M^{\pi_k}(s_{k,0}],$$

where π^* is an optimal policy for M. The goal of the agent is equivalent to minimizing the expected total regret up to episode K

$$\mathrm{Regret}(K, \mathrm{alg}) := \mathbb{E}_{\mathrm{alg}} \sum_{k=1}^{K} \Delta_k, \tag{17}$$

where the subscript alg under the expectation indicates that policies are generated through algorithm alg. Note that the expectation in Equation (17) is over the random transitions and rewards, the possible randomization in the learning algorithm alg, and also the unknown MDP M based on the agent designer's prior beliefs.

Assumption E.1 (Finite-horizon time-inhomogeneous MDPs)**.** We consider a problem class that can be formulated as a special case of the general formulation in Section 2. Assume the state space factorizes as $\mathcal{S} = \mathcal{S}_0 \cup \mathcal{S}_1 \cup \mathcal{S}_2 \cup \cdots \cup \mathcal{S}_{H-1}$ where the state always advances from some state $s_t \in \mathcal{S}_t$ to $s_{t+1} \in \mathcal{S}_{t+1}$ and the process terminates with probability 1 in period H. For notational convenience, we assume each set $\mathcal{S}_0 = \ldots = \mathcal{S}_{H-1} = \mathcal{X}$ contains an equal number of elements that is $\mathcal{X}$. That is $|\mathcal{S}| = |\mathcal{X}|H$.

We study the regret of HyperFQI under the following Bayesian model for the MDP M.

Assumption E.2 (Independent Dirichlet prior for outcomes)**.** For each $(s, a) \in \mathcal{S} \times \mathcal{A}$, the outcome distribution is drawn from a Dirichlet prior

$$P_{sa} \sim \mathrm{Dirichlet}(\alpha_{0,sa})$$

for $\alpha_{0,sa} \in \mathbb{R}_+^{\mathcal{S}}$ and each P_{sa} is drawn independently across (s, a). Assume there is $\beta \geq 3$ such that $\mathbf{1}^\top \alpha_{0,sa} = \beta$ for all (s, a).

A key observation that enable the regret analysis is that hypermodel can approximate the posterior distribution of the optimal Q^*-values with low computation cost. This is formalized in the following lemma.

Lemma E.3 (Approximate posterior variance (restated for Lemma 5.2))**.** *For m_k defined in Equation* (12) *with* $\mathbf{z} \sim \mathrm{Uniform}(\mathbb{S}^{M-1})$*. For any $k \geq 1$, a good event $\mathcal{G}_k(s,a)$ is defined as*

$$\mathcal{G}_k(s,a) = \left\{ \|m_k(s,a)\|^2 \in \left(\frac{\sigma^2}{N_{k,sa} + \beta}, \frac{3\sigma^2}{N_{k,sa} + \beta} \right) \right\}.$$

Then the joint event $\cap_{(s,a,k) \in \mathcal{S} \times \mathcal{A} \times [K]} \mathcal{G}_k(s,a)$ holds w.p. at least $1 - \delta$ if $M \simeq \log(SAK/\delta)$.

To prove Lemma E.3, we develop a new probability tools for sequential random projection in Appendices F and G and a new probability tool for random projection Appendix H.

These are novel technical contributions, which maybe of independent interests. Once the approximation lemma Lemma E.3 is established, the rest proof can be reduced to the Bayesian regret analysis of RLSVI in Osband et al. (2019b), which will be appeared in the full version later. Still, we want to emphasize a key argument that enables efficient deep exploration is the stochastic optimism of HyperFQI.

Definition E.4 (Stochastic optimism)**.** A random variable X is stochastically optimistic with respect to another random variable Y, written $X \geq_{SO} Y$, if for all convex increasing functions $u : \mathbb{R} \to \mathbb{R}$

$$\mathbb{E}[u(X)] \geq \mathbb{E}[u(Y)].$$

We show that HyperFQI is stochastic optimistic in the sense that it overestimates the value of each action in expectation. This is formalized in the following lemma.

Proposition E.5. *If Assumption E.2 holds and tabular* `HyperFQI` *is applied with parameters parameters* $(M, \mu_0, \sigma, \sigma_0)$ *satisfying* $M \simeq \log(SAK)$, $(\sigma^2/\sigma_0^2) = \beta$, $\sigma \geq \sqrt{3}H$ *and* $\min_{s,a} \mu_{0,s,a} \geq H$*,*

$$f_{\theta_k^H}(s, a, \boldsymbol{\xi}_k)\,|\mathcal{H}_k \geq_{SO} Q_M^*(s,a)|\,\mathcal{H}_k. \tag{18}$$

for any history $\mathcal{H}_k$ *and state-action pair* $(s, a) \in \mathcal{S}_0 \times \mathcal{A}$ *given the event* $\mathcal{G}_k$ *defined in Lemma E.3 holds.*

Appendix F. Proof of the key approximate posterior Lemma 5.2

Proof [Proof of Lemma 5.2] We first show the prior approximation by the fixed prior model using the tool from Appendix H. By Lemma H.4, we show $\|\beta\sigma_0\mathbf{z}\| \approx_{\varepsilon/2} \beta\sigma_0^2$ with high probability.

To handle the posterior approximation, take a look at the formula in Equation (12) and apply Theorem G.1 with sequence $(\mathbf{z}_{\ell,t})_{\ell\geq 1, H-1\geq t\geq 0}$ and $(x_{\ell,t})_{\ell\geq 1,H-1\geq t\geq 0}$ s.t. $x_{\ell,t} = \sigma \mathbb{1}_{t\in E_{\ell,sa}}$ and $\mathbf{s} = \beta\sigma_0\mathbf{z}_{0,sa}$, $s = \beta\sigma_0$ for each state action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$. Then taking union bound over the set $\mathcal{S} \times \mathcal{A}$ yields the results. ■

Appendix G. Sequential random projection

In this section, we describe our technical innovation in a probability statement for sequential random projection. Based on a novel and careful construction of stopped process that controls the deviation behavior of the good event on concentration, we adopt the method of mixtures (Peña et al., 2009) in self-normalized process to derive a probability tool stated in Theorem G.1. This bound is new to the whole literature of random projection and also sequential analysis, which may be of independent interests.

Now we provide the key theorem that enables the analysis of approximate posterior argument in Appendix F. We use short notation for $[n] = \{1, 2, \ldots, n\}$ and $\mathcal{T} = \{0, 1, \ldots, T\} = \{0\} \cup [T]$.

Theorem G.1 (Sequential random projection in adaptive process)**.** *Let $(\mathcal{F}_t)_{t\geq 0}$ be a filtration. For any fixed $\varepsilon \in (0,1)$ any fixed $s \in \mathbb{R}_+$, let $\mathbf{s} \in \mathbb{R}^M$ be an $\mathcal{F}_0$-measurable random vector satisfies $\mathbb{E}[\|\mathbf{s}\|^2] = s^2$ and $|\|\mathbf{s}\|^2 - s^2| \leq (\varepsilon/2)s^2$. Let $(\mathbf{z}_t)_{t\geq 1} \subset \mathbb{R}^M$ be a stochastic process adapted to filtration $(\mathcal{F}_t)_{t\geq 1}$ such that it is $\sqrt{c_0/M}$-sub-Gaussian and each $\mathbf{z}_t$ is unit-norm. Let $(x_t)_{t\geq 1} \subset \mathbb{R}$ be a stochastic process adapted to filtration $(\mathcal{F}_{t-1})_{t\geq 1}$ such that it is c_x-bounded. Here, c_0 and c_x are absolute constants.*

If the following condition is satisfied

$$M \geq \frac{16c_0(1+\varepsilon)}{\varepsilon^2}\left(\log\left(\frac{1}{\delta}\right) + \log\left(1 + \frac{c_x T}{s^2}\right)\right),$$

we have, with probability at least $1-\delta$

$$\forall t \in \mathcal{T}, \quad (1-\varepsilon)\left(s^2 + \sum_{i=1}^t x_i^2\right) \leq \|\mathbf{s} + \sum_{i=1}^t x_i \mathbf{z}_i\|^2 \leq (1+\varepsilon)\left(s^2 + \sum_{i=1}^t x_i^2\right).$$

Remark G.2. We say this is an "sequential random projection" argument because one can relate Theorem G.1 to the traditional random projection setting where $\Pi = (\mathbf{z}_1, \ldots, \mathbf{z}_T) \in \mathbb{R}^{M\times T}$ is a random projection matrix and $\boldsymbol{x} = (x_1, \ldots, x_T)^\top \in \mathbb{R}^T$ is the vector to be projected. When $s = 0$ and $\mathbf{s} = 0$, this is essentially an analog of distributional JL lemma (discribed in Lemma H.3) while the traditional JL lemma are NOT handle the sequential dependence structure in our setup. Therefore, Theorem G.1 also an innovation in the literature of random projection and sequential analysis.

Remark G.3. The unit-norm condition in the Theorem G.1 is easy to remove. Then, more distribution of random vectors can be covered in our probability framework. We leave it for the future work.

Example G.4 (Stylized stochastic process satisfying the condition in Theorem G.1.)**.** If $\mathbf{s}$ is a random vector that is independent with all following random variables and $(\mathbf{z}_t)_{t\geq 1}$ are i.i.d random vectors, each sampled from $\mathcal{U}(\mathbb{S}^{M-1})$. The stochastic process $(x_t)_{t\geq 1}$ has the following dependence structure with the process $(\mathbf{z}_t)_{t\geq 1}$:

- x_t is dependent on $\mathbf{s}, x_1, \mathbf{z}_1, \ldots, x_{t-1}, \mathbf{z}_{t-1}$.
- $\mathbf{z}_t$ is independent of $\mathbf{s}, x_1, \mathbf{z}_1, \ldots, x_{t-1}, \mathbf{z}_{t-1}, x_t$

Define the filtration $(\mathcal{F}_t)_{t\geq 0}$ where $\mathcal{F}_t = \sigma(\mathbf{s}, x_1, \mathbf{z}_1, \ldots, x_t, \mathbf{z}_t, x_{t+1})$. From Example I.2, we notice $\mathbf{z} \sim \mathcal{U}(\mathbb{S}^{M-1})$ is $(1/\sqrt{M})$-sub-Gaussian random vector. Thus, $(\mathbf{z}_t)_{t\geq 1} \overset{i.i.d.}{\sim} \mathcal{U}(\mathbb{S}^{M-1})$ is a stochastic process adapted to $(\mathcal{F}_t)_{t\geq 1}$ and is $1/\sqrt{M}$-sub-Gaussian and unit-norm.

First, we state a Peña et al. (2009) type self-normalized bound that would be useful to prove our main theoretical contribution of sequential random projection in Theorem G.1.

Theorem G.5 (Any-time self-normalized concentration bound)**.** *Let $(\mathcal{F}_t)_{t\geq 0}$ be a filtration and $\{(A_t, B_t), t \geq 1\}$ be a sequence of pairs of random variables satisfying that for all $\lambda \in \mathbb{R}$*

$$\left\{\exp\left(\lambda A_t - \frac{\lambda^2}{2}B_t^2\right), \mathcal{F}_t, t \geq 1\right\} \text{ is a supermartingale with mean} \leq 1. \tag{19}$$

Then, for any fixed positive sequence $(L_t)_{t\geq 1}$*, with probability at least* $1-\delta$

$$\forall t \geq 1, \quad |A_t| \leq \sqrt{2\left(B_t^2+L_t\right)\log\left(\frac{1}{\delta}\frac{\left(B_t^2+L_t\right)^{1/2}}{L_t^{1/2}}\right)} \tag{20}$$

The proof of Theorem G.5 can be found in Appendix G.2.

G.1 Sequential random projection argument in Theorem G.1

Before dig into the proof, we identify some important sequence structure and also clarity our proof idea in a intuitive level.

G.1.1 Preparation for the proof of Theorem G.1

For $t \in \mathcal{T}$, let the short notation be

$$Y_t = \|\mathbf{s} + \sum_{i=1}^{t} x_i \mathbf{z}_i\|^2 - \left(s^2 + \sum_{i=1}^{t} x_i^2\right).$$

and $S_t = s^2 + \sum_{i=1}^t x_i^2$. Our *key observation* is that for any $t \in [T]$

$$\begin{aligned}
\|\mathbf{s} + \sum_{i=1}^{t} x_i \mathbf{z}_i\|^2 &= \|\mathbf{s} + \sum_{i=1}^{t-1} x_i \mathbf{z}_i + x_t \mathbf{z}_t\|^2 \\
&= \|\mathbf{s} + \sum_{i=1}^{t-1} x_i \mathbf{z}_i\|^2 + 2\left(\mathbf{s} + \sum_{i=1}^{t-1} x_i \mathbf{z}_i\right)^\top x_t \mathbf{z}_t + x_t^2 \|\mathbf{z}_t\|^2
\end{aligned}$$

and thus we have the following relationship between Y_t and Y_{t-1}

$$Y_t - Y_{t-1} = 2x_t \mathbf{z}_t^\top (\mathbf{s} + \sum_{i=1}^{t-1} x_i \mathbf{z}_i) + x_t^2\left(\|\mathbf{z}_t\|^2 - 1\right).$$

Since $\mathbf{z}_t$ is unit-norm, we can further simplify the exposition

$$Y_t - Y_{t-1} = 2x_t \mathbf{z}_t^\top (\mathbf{s} + \sum_{i=1}^{t-1} x_i \mathbf{z}_i). \tag{21}$$

Another key observation is that the difference term in Equation (21) depends on the $(\mathbf{s} + \sum_{i=1}^{t-1} x_i \mathbf{z}_i)$ that is F_{t-1}-measurable. We can control the deviation of the difference $Y_t - Y_{t-1}$ by if we already have information in the history-dependent term. Intuitively, once the concentration behavior is bad, it is highly possible to be bad for the later time index. To mathematically formalize this intuition, we introduce a definition of good event and stopping time for analysis.

Definition G.6 (Good event)**.** For each time $t \in \mathcal{T}$, we introduce the good event E_t under which the strongly concentration behavior is guaranteed, suppose $\varepsilon \in (0, 1)$,

$$E_t(\varepsilon) = \left\{ (1-\varepsilon) \left(s^2 + \sum_{i=1}^{t} x_i^2 \right) \leq \|\mathbf{s} + \sum_{i=1}^{t} x_i \mathbf{z}_i\|^2 \leq (1+\varepsilon) \left(s^2 + \sum_{i=1}^{t} x_i^2 \right) \right\}.$$

With short notation,

$$E_t(\varepsilon) = \{|Y_t| \leq \varepsilon S_t\}.$$

We also define the stopping time as the first time the bad event happens, i.e. the good event in Definition G.6 violates.

Definition G.7 (Stopping time)**.** For any fixed ε, we define the stopping time

$$\tau(\varepsilon) = \min\{t \in \mathcal{T} : \neg E_t(\varepsilon)\}.$$

Based on the stopping time, we construct a **stopped** process. For $t \in [T]$, define the stopped difference term

$$X_t^\tau = (Y_t - Y_{t-1})\mathbb{1}_{t \leq \tau} \tag{22}$$

such that the process $(X_t^\tau)_{t \geq 1}$ is adapted to the filtration $(\mathcal{F}_t)_{t \geq 1}$.

Claim G.8. *The stopping time τ defined in Definition G.7. Let $(X_t^\tau)_{t \geq 1}$ be the stochastic process defined in Equation (22) which is adapted to $(\mathcal{F}_t)_{t \geq 0}$. Let $A_t^\tau = \sum_{i=1}^{t} X_i^\tau$. Further denote $(B_t^\tau)^2 = \sum_{i=1}^{t} (C_i^\tau)^2$ with*

$$(C_t^\tau)^2 := \frac{4c_0}{M} x_t^2 (1+\varepsilon) S_{t-1} \mathbb{1}_{t \leq \tau}.$$

If the $(\mathcal{F}_t)_{t \geq 1}$-adapted process $(\mathbf{z}_t)_{t \geq 1}$ is $(\sqrt{\frac{c_0}{M}})$-sub-Gaussian and each $\mathbf{z}_t$ is unit-norm, then for any fixed $\lambda \in \mathbb{R}$

$$M_t^\tau(\lambda) = \exp\left(\lambda A_t^\tau - \frac{\lambda^2}{2} (B_t^\tau)^2 \right), \quad t \geq 1$$

is a supermartingale.

Proof [Proof of Claim G.8] Note $\mathbb{1}_{t \leq \tau} = 1 - \mathbb{1}_{\tau \leq t-1}$ is $\mathcal{F}_{t-1}$-measurable. Thus, the vector $(\mathbf{s} + \sum_{i=1}^{t-1} x_i \mathbf{z}_i)\mathbb{1}_{t \leq \tau} x_t$ is $\mathcal{F}_{t-1}$-measurable. By the condition on process $(\mathbf{z}_t)_{t \geq 1}$, we conclude from the definition of conditionally sub-Gaussian from Definition D.2 that

$$\begin{aligned}
\mathbb{E}[\exp(\lambda X_t^\tau) \mid \mathcal{F}_{t-1}] &= \mathbb{E}[\exp(2\lambda x_t \langle \mathbf{z}_t, \mathbf{s} + \sum_{i=1}^{t-1} x_i \mathbf{z}_i \rangle \mathbb{1}_{t \leq \tau}) \mid \mathcal{F}_{t-1}] \\
&\leq \exp\left(\frac{\lambda^2}{2} (4c_0/M) x_t^2 \|\mathbf{s} + \sum_{i=1}^{t-1} x_i \mathbf{z}_i\|^2 \mathbb{1}_{t \leq \tau} \right) \\
&\leq \exp\left(\frac{\lambda^2}{2} (4c_0/M) x_t^2 (1+\varepsilon) S_{t-1} \mathbb{1}_{t \leq \tau} \right) \\
&= \exp\left(\frac{\lambda^2}{2} (C_t^\tau)^2 \right)
\end{aligned}$$

where the last inequality is because of the stopping time argument. ■

We also need the following lemma in the intial treatment of the proof of Theorem G.1.

Lemma G.9 (Trigger lemma). *For any sequence of event $(\mathcal{E}_t, t \in \mathcal{T})$, define the stopping time τ as the first time t the event $\mathcal{E}_t$ is violated, i.e.*

$$\tau = \min\{t \in \mathcal{T} : \neg\mathcal{E}_t\}.$$

Then, the following equality holds for all $t \in \mathcal{T}$,

$$\{\tau \le t\} = \neg\mathcal{E}_{t\wedge\tau}. \tag{23}$$

G.1.2 Proof of Theorem G.1

Now we are ready to the proof.

Proof [Proof of Theorem G.1] We apply Lemma G.9 for $\mathcal{E}_t = E_t(\varepsilon)$ and it follows

$$\begin{aligned}\mathbb{P}\left(\exists t \in \mathcal{T}, \neg E_t(\varepsilon)\right) &= \mathbb{P}(\tau \le T) = \mathbb{P}\left(\neg E_{T\wedge\tau}(\varepsilon)\right)\\ &= \mathbb{P}\left(|Y_{T\wedge\tau}| \ge \varepsilon S_{T\wedge\tau}\right)\\ &= \mathbb{P}\left(|Y_0 + \sum_{t=1}^{T}(Y_t - Y_{t-1})\mathbb{1}_{t\le\tau}| \ge \varepsilon S_{T\wedge\tau}\right)\end{aligned} \tag{24}$$

By the construction of stopped process $Y_{T\wedge\tau} - Y_0 = \sum_{t=1}^{T} X_t^\tau = A_T^\tau$. Then, *our goal*, from Equation (24), becomes to upper bound the RHS of Equation (25),

$$\mathbb{P}\left(\exists t \in \mathcal{T}, (\neg E_t)\right) = \mathbb{P}\left(|Y_0 + A_T^\tau| \ge \varepsilon S_{T\wedge\tau}\right) \tag{25}$$

By Claim G.8, the process $(A_t^\tau, B_t^\tau)_{t\ge1}$ with $A_t^\tau = \sum_{i=1}^{t} X_i^\tau = \sum_{i=1}^{t}(Y_t - Y_{t-1})\mathbb{1}_{t\le\tau}$ and $(B_t^\tau)^2 = \sum_{i=1}^{t}(4c_0/M)x_t^2(1+\varepsilon)S_{t-1}\mathbb{1}_{t\le\tau}$ satisfy the condition of Theorem G.5. Then we can apply the Theorem G.5: with probability at least $1-\delta$,

$$\forall t \ge 1, |A_t^\tau| \le \sqrt{2\left((B_t^\tau)^2 + L_t\right)\log\left(\frac{1}{\delta}\frac{((B_t^\tau)^2 + L_t)^{1/2}}{L_t^{1/2}}\right)}$$

Since by the condition in Theorem G.1, we have $|Y_0| \le (\varepsilon/2)s^2$. Now we want to argue that for any fixed $\varepsilon \in (0,1)$, with suitable choice of L_T and M, we have with probability at least $1-\delta$

$$|Y_0 + A_T^\tau| \le \underbrace{\sqrt{2\left((B_T^\tau)^2 + L_T\right)\log\left(\frac{1}{\delta}\frac{\left((B_T^\tau)^2 + L_T\right)^{1/2}}{L_T^{1/2}}\right)} + (\varepsilon/2)s^2}_{(I)} \le \varepsilon S_{T\wedge\tau}. \tag{26}$$

Claim G.10. *With some computations, we found the following configuration suffices for Equation (26):*

$$L_T \le \frac{4(1+\varepsilon)s^4}{2M} \quad \textit{and} \quad M \ge (16(1+\varepsilon)/\varepsilon^2)\left(\log\left(\frac{1}{\delta}\right) + \log\left(1 + \frac{T}{s^2}\right)\right).$$

Proof [Proof of Claim G.10] Recall the definition

$$S_t = s^2 + \sum_{t=1}^{t} x_i^2$$

We first calculate the term $(B_T^\tau)^2$ by our construction:

$$\begin{aligned}
(B_T^\tau)^2 &\leq \frac{4c_0}{M} \sum_{t=1}^{T\wedge\tau} x_t^2 \left((1+\varepsilon)S_{t-1}\right) \\
&= \frac{4c_0(1+\varepsilon)}{M} \sum_{t=1}^{T\wedge\tau} x_t^2 \left(S_{T\wedge\tau} - (S_{T\wedge\tau} - S_{t-1})\right) \\
&\leq \frac{4c_0(1+\varepsilon)}{M} (S_{T\wedge\tau} - s^2) S_{T\wedge\tau}
\end{aligned}$$

Then, the almost sure upper bound of $(B_T^\tau)^2$ assuming $x_t^2 \leq c_x$ is

$$(B_T^\tau)^2 \leq \frac{4c_0(1+\varepsilon)}{M} \sum_{t=1}^{T} (s^2 + (t-1)c_x) \leq \frac{4c_0(1+\varepsilon)}{M}(s^2 T + c_x T^2/2)$$

Since $(a+b)^2 \leq (1+\lambda)(a^2 + (1/\lambda)b^2)$ for all λ,

$$(I)^2 \leq (1+\lambda)\left(2\left(B_T^2 + L_T\right)\log\left(\frac{1}{\delta}\frac{\left(B_T^2 + L_T\right)^{1/2}}{L_T^{1/2}}\right) + \frac{\varepsilon^2 s^4}{4\lambda}\right)$$

Let $L_T = c\ell/M$ and $c = 4c_0(1+\varepsilon)$ and ℓ to be determined.

$$\begin{aligned}
(I)^2 &\leq (1+\lambda)\left(2\left(B_T^2 + L_T\right)\log\left(\frac{1}{\delta}\frac{\left(B_T^2 + L_T\right)^{1/2}}{L_T^{1/2}}\right) + \frac{\varepsilon^2 s^4}{4\lambda}\right) \\
&\leq (1+\lambda)\left(\frac{2c}{M}\left((S_{T\wedge\tau} - s^2)S_{T\wedge\tau} + \ell\right)\log\left(\frac{1}{\delta}\sqrt{\frac{(s^2T + c_xT^2/2 + \ell)}{\ell}}\right) + \frac{\varepsilon^2 s^4}{4\lambda}\right)
\end{aligned}$$

Let $M = (2c/m)\log\left(\frac{1}{\delta}\sqrt{\frac{(s^2T+c_xT^2/2)+\ell}{\ell}}\right)$, we can simplify

$$(I)^2 \leq (1+\lambda)\left(m((S_{T\wedge\tau} - s^2)S_{T\wedge\tau} + \ell) + \frac{\varepsilon^2 s^4}{4\lambda}\right)$$

Let $\ell = s^4/2c_x$, $m = \varepsilon^2/(1+\lambda)$ and $\lambda = 1$, we have

$$(I)^2 \leq \varepsilon^2((S_{T\wedge\tau} - s^2)S_{T\wedge\tau} + s^4/2 + s^4/2) \leq \varepsilon^2 S_{T\wedge\tau}^2$$

where the last inequality is because $s^2 = S_0 \leq S_{T\wedge\tau}$ and $s^4 \leq s^2 S_{T\wedge\tau}$. The conclusion is that we could select

$$\begin{aligned}
M &\geq (16c_0(1+\varepsilon)/\varepsilon^2)\log\left(\frac{1}{\delta}\sqrt{\frac{2s^2c_xT + c_x^2T^2 + s^4}{s^4}}\right) \\
&= (16c_0(1+\varepsilon)/\varepsilon^2)\left(\log\left(\frac{1}{\delta}\right) + \log\left(1 + \frac{c_xT}{s^2}\right)\right)
\end{aligned}$$

and the auxiliary variable

$$L_T \leq \frac{4c_0(1+\varepsilon)s^4}{2Mc_x}.$$

■

■

G.2 Proof of Theorem G.5: Method of mixtures

Robbins-Siegmund method of mixtures (Robbins and Siegmund, 1970) originally is developed to evaluate boundary crossing probabilities for Brownian motion. The method was further developed in the general theory for self-normalized process (de la Peña, Klass, and Lai, 2004; Peña, Lai, and Shao, 2009; Lai, 2009).

Remark G.11 (Essential idea of Laplace approximation). If we integrate the exponential of a function that has a pronounced maximum, then we can expect that the integral will be close to the exponential function of the maximum. In our case, let

$$M_t(\lambda) = \exp\left(\lambda A_t - \frac{\lambda^2}{2} B_t^2\right)$$

Informally, with this principle of Laplace approximation, we would have

$$\max_\lambda M_t(\lambda) \approx \int_\Omega M_t(\lambda) dh(\lambda)$$

where h is some measure on Ω.

The main benefit of replacing the maximum $\max_\lambda M_t(\lambda)$ *with an integral* $\bar{M}_t := \int_\Omega M_t(\lambda) dh(\lambda)$ *is that we can handle the expectation* $\mathbb{E}[\bar{M}_t]$ *easier while we don't know the upper bound on* $\mathbb{E}[\max_\lambda M_t(\lambda)]$. This is formalized in the following lemma.

Lemma G.12. *Let* (h_t) *be a sequence of probability measures on* Ω*. If* $(M_t(\lambda), \mathcal{F}_t, t \geq 1)$ *is a supermartingale with* $\mathbb{E}[M_1(\lambda)] \leq 1$ *for all* $\lambda \in \Omega$*, then for any* $t \geq 1$*, the integrated random variable* $\bar{M}_t = \int_\Omega M_t(\lambda) dh_t(\lambda)$ *has expectation* $\mathbb{E}[\bar{M}_t] \leq 1$*.*

Further, let τ *be a stopping time with respect to filtration* $(\mathcal{F}_t)_{t \geq 0}$*, i.e.* $\{\tau \leq t\} \in \mathcal{F}_t, \forall t \geq 0$*. Then* $M_\tau(\lambda)$ *is almost surely well-defined with expectation* $\mathbb{E}[M_\tau(\lambda)] \leq 1$ *as well as* $\mathbb{E}[\bar{M}_\tau] \leq 1$*.*

Proof Using Fubini's theorem and the fact that $M_t(\lambda)$ is a supermartingale with $\mathbb{E}[M_t(\lambda)] \leq \mathbb{E}[M_1(\lambda)] = 1$, we have

$$\mathbb{E}[\bar{M}_t] = \int_\Omega \mathbb{E}[M_t(\lambda)] dh_t(\lambda) \leq 1.$$

For the expectation of stopped version $M_\tau(\lambda)$ and $\bar{M}_\tau$, we apply (supermartingale) optional sampling theorem. ■

Finally, we are comfortable to drive the proof of the self-normalized concentration bounds.
Proof [Proof of Theorem G.5] Let $\Lambda = (\Lambda_t)$ be a sequence of independent Gaussian random variable with densities

$$f_{\Lambda_t}(\lambda) = c(L_t)\exp(-\frac{1}{2}L_t\lambda^2)$$

where $c(A) = \sqrt{A/2\pi}$ is a normalizing constant. We explicitly calculate $\bar{M}_t$ for any $t \geq 1$,

$$\begin{aligned}
\bar{M}_t &= \int_{\mathbb{R}} \exp\left(\lambda A_t - \frac{\lambda^2}{2}B_t^2\right) f_{\Lambda_t}(\lambda)d\lambda \\
&= \int_{\mathbb{R}} \exp\left(-\frac{1}{2}\left(\lambda - \frac{A_t}{B_t^2}\right)^2 B_t^2 + \frac{1}{2}\frac{A_t^2}{B_t^2}\right) f_{\Lambda_t}(\lambda)d\lambda \\
&= \exp\left(\frac{1}{2}\frac{A_t^2}{B_t^2}\right)\int_{\mathbb{R}} \exp\left(-\frac{1}{2}\left(\lambda - \frac{A_t}{B_t^2}\right)^2 B_t^2\right) f_{\Lambda_t}(\lambda)d\lambda \\
&= c\left(L_t\right)\exp\left(\frac{1}{2}\frac{A_t^2}{B_t^2}\right)\int_{\mathbb{R}} \exp\left(-\frac{1}{2}\left(\left(\lambda - A_t/B_t^2\right)^2 B_t^2 + \lambda^2 L_t\right)\right) d\lambda.
\end{aligned}$$

Completing the square yields

$$\left(\lambda - \frac{A_t}{B_t^2}\right)^2 B_t^2 + \lambda^2 L_t = \left(\lambda - \frac{A_t}{L_t + B_t^2}\right)^2 (L_t + B_t^2) + \frac{A_t^2}{B_t^2} - \frac{A_t^2}{L_t + B_t^2}.$$

By the change of variables $\lambda' = \lambda - A_t/(L_t + B_t^2)$ in the following (i),

$$\begin{aligned}
\bar{M}_t &= c\left(L_t\right)\exp\left(\frac{1}{2}\frac{A_t^2}{L_t + B_t^2}\right)\int_{\mathbb{R}} \exp\left(-\frac{1}{2}\left(\lambda - \frac{A_t}{L_t + B_t^2}\right)^2 \left(L_t + B_t^2\right)\right) d\lambda \\
&\overset{(i)}{=} c\left(L_t\right)\exp\left(\frac{1}{2}\frac{A_t^2}{L_t + B_t^2}\right)\int_{\mathbb{R}} \exp\left(-\frac{1}{2}\left(\lambda^2\left(L_t + B_t^2\right)\right)\right) d\lambda \\
&= \frac{c\left(L_t\right)}{c\left(L_t + B_t^2\right)}\exp\left(\frac{1}{2}\frac{A_t^2}{L_t + B_t^2}\right).
\end{aligned}$$

A final application of Markov's inequality yields

$$\begin{aligned}
&\mathbb{P}\left[|A_\tau| \geq \sqrt{2\left(L_\tau + B_\tau^2\right)\log\left(\frac{1}{\delta}\frac{\left(L_\tau + B_\tau^2\right)^{1/2}}{L_\tau^{1/2}}\right)}\right] \\
&= \mathbb{P}\left[\frac{c\left(L_\tau\right)}{c\left(L_\tau + B_\tau^2\right)}\exp\left(\frac{1}{2}\frac{A_\tau^2}{L_\tau + B_\tau^2}\right) \geq \frac{1}{\delta}\right] \\
&\leq \delta \cdot \mathbb{E}\left[\frac{c\left(L_\tau\right)}{c\left(L_\tau + B_\tau^2\right)}\exp\left(\frac{1}{2}\frac{A_\tau^2}{L_\tau + B_\tau^2}\right)\right] \\
&\overset{\text{i)}}{\leq} \delta \cdot \mathbb{E}\left[\bar{M}_\tau\right] \overset{\text{(ii)}}{\leq} \delta,
\end{aligned}$$

where (i) uses the inequality for $\bar{M}_\tau$ derived above, and (ii) follows from Lemma G.12.

To get the anytime result in Theorem G.5, we define the stopping time

$$\tau = \min\left\{t \geq 1 : |A_t| \geq \sqrt{2\left(L_t + B_t^2\right)\log\left(\frac{1}{\delta}\frac{\left(L_t + B_t^2\right)^{1/2}}{L_t^{1/2}}\right)}\right\}$$

With an application of extended version of Lemma G.9, and applying the previous inequality yields

$$\begin{aligned}
&\mathbb{P}\left[\exists t \geq 1, |A_t| \geq \sqrt{2\left(L_t + B_t^2\right)\log\left(\frac{1}{\delta}\frac{\left(L_t + B_t^2\right)^{1/2}}{L_t^{1/2}}\right)}\right] \\
&= \mathbb{P}\left[\tau < \infty, |A_\tau| \geq \sqrt{2\left(L_\tau + B_\tau^2\right)\log\left(\frac{1}{\delta}\frac{\left(L_\tau + B_\tau^2\right)^{1/2}}{L_\tau^{1/2}}\right)}\right] \\
&\leq \mathbb{P}\left[|A_\tau| \geq \sqrt{2\left(L_\tau + B_\tau^2\right)\log\left(\frac{1}{\delta}\frac{\left(L_\tau + B_\tau^2\right)^{1/2}}{L_\tau^{1/2}}\right)}\right] \\
&\leq \delta.
\end{aligned}$$

This completes the proof. ■

Appendix H. A new proof of random projection for fixed data

This section provide a new tool for random projection, which maybe of independent interest. In this work, it is used to deal with the initial approximation in the prior model in Appendix F.

Theorem H.1 (High-dimensional Hanson-Wright inequality)**.** *Let $X_1, \ldots, X_n$ be independent, mean zero random vectors in $\mathbb{R}^M$, each X_i is K_i-subGaussian. Let $A = (a_{ij})$ be an $n \times n$ matrix. Then for any $t \geq 0$, we have*

$$\mathbb{P}\left(|\sum_{i,j:i\neq j}^{n} a_{ij}\langle X_i, X_j\rangle| \geq t\right) \leq 2\exp\left(-\min\left\{\frac{t^2}{64K^4M\|A\|_F^2}, \frac{t}{8K^2\|A\|_2}\right\}\right)$$

where $K = \max_i K_i$.

Remark H.2. This is an high-dimension extension of famous Hanson-Wright inequality (Rudelson and Vershynin, 2013). The Theorem H.1 with exact constant is new in the literature, which maybe of independent interest. Our proof technique generalizes from (Rudelson and Vershynin, 2013) with new treatments on the diagnolization.

Lemma H.3 (Distributional JL lemma (Johnson and Lindenstrauss, 1984))**.** *For any $0 < \varepsilon, \delta < 1/2$ and $d \geq 1$ there exists a distribution $\mathcal{D}_{\varepsilon,\delta}$ on $\mathbb{R}^{M\times d}$ for $M = O\left(\varepsilon^{-2}\log(1/\delta)\right)$ such that for any $x \in \mathbb{R}^d$*

$$\mathbb{P}_{\Pi\sim\mathcal{D}_{\varepsilon,\delta}}\left(\|\Pi x\|_2^2 \notin \left[(1-\varepsilon)\|x\|_2^2, (1+\varepsilon)\|x\|_2^2\right]\right) < \delta$$

Lemma H.4. *We claim that the following construction of the random projection matrix $\Pi \in \mathbb{R}^{M\times d}$ with $M \geq 64\varepsilon^{-2}\log(2/\delta)$ satisfy the Lemma H.3: Let $\Pi = (\mathbf{z}_1, \dots, \mathbf{z}_d)$ be a random matrix with each $\mathbf{z}_i \sim P_{\mathbf{z}}$, i.e., uniformly sampled over the unit sphere $\mathbb{S}^{M-1}$.*

Proof From Example I.2, we know that $\mathbf{z}_i \sim P_{\mathbf{z}} = \mathcal{U}(\mathbb{S}^{M-1})$ is a $\frac{1}{\sqrt{M}}$-sub-Gaussian random vector with mean zero. Let $x \in \mathbb{R}^d$ be the vector to be projected. By the construction of Π,

$$\|\Pi x\|^2 - \|x\|^2 = \underbrace{\sum_{1\leq i\neq j\leq d} x_i x_j \langle \mathbf{z}_i, \mathbf{z}_j\rangle}_{\text{off-diagonal}} + \underbrace{\sum_{i=1}^{d} x_i^2(\|\mathbf{z}_i\|^2 - 1)}_{\text{diagonal}}$$

The diagonal term is zero due to the unit sphere $\mathbb{S}^{M-1}$. The JL lemma is then a consequence of bounding the following

$$\mathbb{P}_{\Pi}\left(|\text{off-diagonal}| \geq \varepsilon\|x\|^2\right)$$

We apply Theorem H.1 with $A = xx^\top$ and $t = \varepsilon\|x\|^2$. Since $K = 1/\sqrt{M}$ and $\|A\|_F = \mathrm{tr}(xx^\top) = \|x\|^2$, $\|A\|_2 = \|x\|^2$, then

$$\begin{aligned}\mathbb{P}\left(|\sum_{1\leq i\neq j\leq d} x_i x_j\langle \mathbf{z}_i, \mathbf{z}_j\rangle| \geq \varepsilon\|x\|^2\right) &\leq 2\exp\left(-\min\left\{\frac{\varepsilon^2\|x\|^4}{64K^4M\|A\|_F^2}, \frac{\varepsilon\|x\|^2}{8K^2\|A\|_2}\right\}\right)\\ &\leq 2\exp\left(-M\min\left\{\varepsilon^2/64, \varepsilon/8\right\}\right).\end{aligned}$$

This implies that to get the RHS upper bound by δ, we need

$$M \geq 64\varepsilon^{-2}\log(2/\delta).$$

■

H.1 Proof of Theorem H.1

Proof We prove the one-side inequality and the other side is similar by replacing A with $-A$. Let

$$S = \sum_{i,j:i\neq j}^{n} a_{ij}\langle X_i, X_j\rangle. \tag{27}$$

Step 1: decoupling. Let $\iota_1, \dots, \iota_d \in \{0,1\}$ be symmetric Bernoulli random variables, (i.e., $\mathbb{P}(\iota_i = 0) = \mathbb{P}(\iota_i = 1) = 1/2$) that are independent of $X_1, \dots, X_n$. Since

$$\mathbb{E}[\iota_i(1-\iota_i)] = \begin{cases} 0, & i = j,\\ 1/4, & i \neq j,\end{cases}$$

we have $S = 4\mathbb{E}_\iota[S_\iota]$, where

$$S_\iota = \sum_{i,j=1}^{n} \iota_i(1-\iota_j)a_{ij}\langle X_i, X_j\rangle$$

and the expectation $\mathbb{E}_\iota[\cdot]$ is the expectation taken with respect to the random variables ι_i. By Jensen's inequality, we have

$$\mathbb{E}[\exp \lambda S] \le \mathbb{E}_{X,\iota}[\exp 4\lambda S_\iota].$$

Let $\Lambda_\iota = \{i \in [d] : \iota_i = 1\}$. Then we write

$$S_\iota = \sum_{i\in\Lambda_\iota}\sum_{j\in\Lambda_\iota^c} a_{ij}\langle X_i, X_j\rangle = \sum_{j\in\Lambda_\iota^c}\langle \sum_{i\in\Lambda_\iota} a_{ij}X_i, X_j\rangle.$$

Taking expectation over $(X_j)_{j\in\Lambda_\iota^c}$ (i.e., conditioning on $(\iota_i)_{i=1,\ldots,d}$ and $(X_i)_{i\in\Lambda_\iota}$), it follows that

$$\mathbb{E}_{(X_j)_{j\in\Lambda_\iota^c}}[\exp 4\lambda S_\iota] = \prod_{j\in\Lambda_\iota^c}\mathbb{E}_{(X_j)_{j\in\Lambda_\iota^c}}[\exp 4\lambda\langle \sum_{i\in\Lambda_\iota} a_{ij}X_i, X_j\rangle]$$

by the independence among $(X_j)_{j\in\Lambda_\iota}$. By the assumption that X_i are independent sub-Gaussian with mean zero, we have

$$\mathbb{E}_{(X_j)_{j\in\Lambda_\iota^c}}[\exp 4\lambda S_\iota] \le \exp\left(\sum_{j\in\Lambda_\iota^c} 8\lambda^2 K_j^2 \|\sum_{i\in\Lambda_\iota} a_{ij}X_i\|^2\right) =: \exp\left(8\lambda^2\sigma_\iota^2\right).$$

Thus we get

$$\mathbb{E}_X[\exp 4\lambda S_\iota] \le \mathbb{E}_X[\exp 8\lambda^2\sigma_\iota^2].$$

Step 2: reduction to Gaussian random variables. For $j = 1,\ldots,n$, let g_j be independent $N\left(0, 16K_j^2\boldsymbol{I}\right)$ random variables in $\mathbb{R}^M$ that are independent of $X_1,\ldots,X_n$ and $\iota_1,\ldots,\iota_n$. Define

$$T := \sum_{j\in\Lambda_\iota^c}\langle g_j, \sum_{i\in\Lambda_\iota} a_{ij}X_i\rangle.$$

Then, by the definition of Gaussian random variables in $\mathbb{R}^M$, we have

$$\begin{aligned}\mathbb{E}_g[e^{\lambda T}] &= \prod_{j\in\Lambda_\iota^c}\mathbb{E}_g[\exp\langle g_j, \lambda\sum_{i\in\Lambda_\iota} a_{ij}X_i\rangle]\\ &= \exp\left(8\lambda^2\sum_{j\in\Lambda_\iota^c} K_j^2\|\sum_{i\in\Lambda_\iota} a_{ij}X_i\|^2\right) = \exp\left(8\lambda^2\sigma_\iota^2\right)\end{aligned}$$

So it follows that

$$\mathbb{E}_X[\exp 4\lambda S_\iota] \le \mathbb{E}_{X,g}[\exp \lambda T].$$

Since $T = \sum_{i\in\Lambda_\iota}\langle\sum_{j\in\Lambda_\iota^c} a_{ij}g_j, X_i\rangle$, by the assumption that X_i are independent sub-Gaussian with mean zero, we have

$$\mathbb{E}_{(X_i)_{i\in\Lambda_\iota}}[\exp\lambda T] \le \exp\left(\frac{\lambda^2}{2}\sum_{i\in\Lambda_\iota} K_i^2\|\sum_{j\in\Lambda_\iota^c} a_{ij}g_j\|^2\right),$$

which implies that

$$\mathbb{E}_X[\exp 4\lambda S_\iota] \le \mathbb{E}_g[\exp\left(\lambda^2\tau_\iota^2/2\right)] \tag{28}$$

where $\tau_\iota^2 = \sum_{i\in\Lambda_\iota} K_i^2 \|\sum_{j\in\Lambda_\iota^c} a_{ij} g_j\|^2$. Note that τ_ι^2 is a random variable that depends on $(\iota_i)_{i=1}^d$ and $(g_j)_{j=1}^n$.

Step 3: diagonalization. We have $g_j = \sum_{k=1}^M \langle g_j, e_k\rangle e_k$ and

$$\begin{aligned}
\tau_\iota^2 &= \sum_{i\in\Lambda_\iota} K_i^2 \left\|\sum_{j\in\Lambda_\iota^c} a_{ij} g_j\right\|^2 = \sum_{i\in\Lambda_\iota} K_i^2 \left\|\sum_{k=1}^M \left(\sum_{j\in\Lambda_\iota^c} a_{ij}\langle g_j, e_k\rangle\right) e_k\right\|^2 \\
&= \sum_{k=1}^M \sum_{i\in\Lambda_\iota} \left(\sum_{j\in\Lambda_\iota^c} K_i a_{ij}\langle g_j, e_k\rangle\right)^2 \\
&= \sum_{k=1}^M \|P_\iota \tilde{A}(I-P_\iota)G_k\|^2
\end{aligned}$$

where the last second step follows from Parseval's identity. $G_{jk} := \langle g_j, e_k\rangle, j = 1,\ldots,n$, are independent $N\left(0, 16K_j^2\right)$ random variables. $G_k = (G_{1k},\ldots,G_{nk})^\top \in \mathbb{R}^n$. $\widetilde{A} = (\tilde{a}_{ij})_{i,j=1}^n$ with $\tilde{a}_{ij} = K_i a_{ij}$. Let $P_\iota \in \mathbb{R}^{n\times n}$ be the restriction matrix such that $P_{\iota,ii} = 1$ if $i \in \Lambda_\iota$ and $P_{\iota,ij} = 0$ otherwise.

Define normal random variables $Z_k = (Z_{1k},\ldots,Z_{nk})^\top \sim N(0,I)$ for each $k = 1,\ldots,M$. Then we have $G_k \overset{D}{=} \Gamma^{1/2} Z_k$ where $\Gamma = 16\,\mathrm{diag}(K_1^2,\ldots,K_n^2)$.

Let $\tilde{A}_\iota := P_\iota\tilde{A}(I-P_\iota)$. Then by the rotational invariance of Gaussian distributions, we have

$$\sum_{k=1}^M \|\tilde{A}_\iota G_k\|^2 \overset{D}{=} \sum_{k=1}^M \|\tilde{A}_\iota \Gamma^{1/2} Z_k\|^2 \overset{D}{=} \sum_{k=1}^M\sum_{j=1}^n s_j^2 Z_{jk}^2$$

where $s_j^2, j = 1,2,\ldots,n$ are the eigenvalues of $\Gamma^{1/2}\tilde{A}_\iota^\top\tilde{A}_\iota\Gamma^{1/2}$.

Step 4: bound the eigenvalues. It follows that

$$\max_{j\in[n]} s_j^2 = \|\tilde{A}_\iota\Gamma^{1/2}\|_{\mathrm{op}}^2 \le 16K^4\|A\|_2^2.$$

In addition, we also have

$$\sum_{j=1}^n s_j^2 = \mathrm{tr}(\Gamma^{1/2}\tilde{A}_\iota^\top\tilde{A}_\iota\Gamma^{1/2}) \le 16K^4\|A\|_F^2$$

and $\sum_{k=1}^M\sum_{j=1}^n s_j^2 \le 16MK^4\|A\|_F^2$. Invoking Equation (28), we get

$$\mathbb{E}_X\left[e^{4\lambda S_\iota}\right] \le \prod_{k=1}^M\prod_{j=1}^n \mathbb{E}_Z\left[\exp\left(\lambda^2 s_j^2 Z_{jk}^2/2\right)\right]$$

Since Z_{jk}^2 are i.i.d. χ_1^2 random variables with the moment generating function $\mathbb{E}[e^{tZ_{jk}^2}] = (1-2t)^{-1/2}$ for $t < 1/2$, we have

$$\mathbb{E}_X\left[e^{4\lambda S_\iota}\right] \le \prod_{k=1}^{M}\prod_{j=1}^{n}\frac{1}{\sqrt{1-\lambda^2 s_j^2}} \quad \text{if } \max_j \lambda^2 s_j^2 < 1.$$

Using $(1-z)^{-1/2} \le e^z$ for $z \in [0,1/2]$, we get that if $16K^4\|A\|_2^2\lambda^2 < 1$, then

$$\mathbb{E}_X\left[e^{4\lambda S_\iota}\right] \le \exp\left(\lambda^2\sum_{k=1}^{M}\sum_{j=1}^{n}s_j^2\right) \le \exp\left(16\lambda^2K^4\|A\|_F^2\right).$$

Note that the last inequality is uniform in ι. Taking expectation with respect to δ, we obtain that

$$\mathbb{E}_X\left[e^{\lambda S}\right] \le \mathbb{E}_{X,\iota}\left[e^{4\lambda S_\iota}\right] \le \exp\left(16\lambda^2MK^4\|A\|_F^2\right)$$

whenever $0 < \lambda < (4K^2\|A\|_2)^{-1}$.

Step 5: Conclusion. Step 5: conclusion. Now we have

$$\mathbb{P}(S \ge t) \le \exp\left(-\lambda t + 16\lambda^2MK^4\|A\|_F^2\right) \quad \text{for } 0 < \lambda \le \left(4K^2\|A\|_2\right)^{-1}$$

Optimizing in λ, we deduce that there exists a universal constant $C > 0$ such that

$$\mathbb{P}(S \ge t) \le \exp\left[-\min\left(\frac{t^2}{64MK^4\|A\|_F^2}, \frac{t}{8K^2\|A\|_2}\right)\right].$$

■

Appendix I. Verify the assumption for some typical distributions

Lemma I.1 (MGF of Beta distribution)**.** *For any $\alpha, \beta \in \mathbb{R}_+$ with $\alpha \le \beta$. Random variable $X \sim \mathrm{Beta}(\alpha,\beta)$ has variance* $\mathrm{Var}(X) = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$ *and the centered MGF* $\mathbb{E}[\exp(\lambda(X - \mathbb{E}[X]))] \le \exp(\frac{\lambda^2\mathrm{Var}(X)}{2})$.

Example I.2 (Uniform over sphere $\mathcal{U}(\mathbb{S}^{M-1})$)**.** Given a random vector $\mathbf{z} \sim \mathcal{U}(\mathbb{S}^{M-1})$, for any $v \in \mathbb{S}^{M-1}$, we have

$$\langle \mathbf{z}, v\rangle \sim 2\,\mathrm{Beta}\left(\frac{M-1}{2}, \frac{M-1}{2}\right) - 1.$$

Thus, by Lemma I.1, we confirm $\mathbf{z}$ is $(1/\sqrt{M})$-sub-Gaussian random vector.

Proof [Proof of Lemma I.1]

For $X \sim \mathrm{Beta}(\alpha, \beta)$, Skorski (2023) gives a novel Order 2 Recurrence for Central Moments.

$$\begin{aligned}\mathbb{E}\left[(X-\mathbb{E}[X])^p\right] =& \frac{(p-1)(\beta-\alpha)}{(\alpha+\beta)(\alpha+\beta+p-1)} \cdot \mathbb{E}\left[(X-\mathbb{E}[X])^{p-1}\right] \\ &+ \frac{(p-1)\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+p-1)} \cdot \mathbb{E}\left[(X-\mathbb{E}[X])^{p-2}\right]\end{aligned}$$

Let $m_p := \frac{\mathbb{E}[(X-\mathbb{E}[X])^p]}{p!}$, When $\alpha \leq \beta$, it follows that m_p is non-negative when p is even, and negative otherwise. Thus, for even p,

$$m_p \leq \frac{1}{p} \cdot \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+p-1)} m_{p-2} \leq \frac{\mathrm{Var}\,(X)}{p} \cdot m_{p-2}.$$

Repeating this $p/2$ times and combining with $m_p \leqslant 0$ for odd p, we obtain

$$m_p \leqslant \begin{cases} \frac{\mathrm{Var}(X)^{\frac{p}{2}}}{p!!} & p \text{ even} \\ 0 & d \text{ odd} \end{cases}.$$

Using $p!! = 2^{p/2}(p/2)!$ for even p, for $t \geqslant 0$ we obtain

$$\mathbb{E}[\exp(\lambda[X-\mathbb{E}[X]])] \leqslant 1 + \sum_{p=2}^{+\infty} m_p \lambda^p = 1 + \sum_{p=1}^{+\infty} (\lambda^2 \mathrm{Var}\,(X)/2)^p/p! = \exp\left(\frac{\lambda^2 \mathrm{Var}\,(X)}{2}\right)$$

■