UCB and InfoGain Exploration via Q-Ensembles

Richard Y. Chen
OpenAI
richardchen@openai.com

Szymon Sidor
OpenAI
szymon@openai.com

Pieter Abbeel
OpenAI
University of California, Berkeley
pieter@openai.com

John Schulman
OpenAI
joschu@openai.com

Abstract

We show how an ensemble of Q^*-functions can be leveraged for more effective exploration in deep reinforcement learning. We build on well established algorithms from the bandit setting, and adapt them to the Q-learning setting. First we propose an exploration strategy based on upper-confidence bounds (UCB). Next, we define an "InfoGain" exploration bonus, which depends on the disagreement of the Q-ensemble. Our experiments show significant gains on the Atari benchmark.

1 Introduction

Deep reinforcement learning seeks to learn mappings from high-dimensional observations to actions. Deep Q-learning (Mnih et al. [14]) is a leading technique that has been used successfully, especially for video game benchmarks. However, fundamental challenges remain, for example, improving sample efficiency and ensuring convergence to high quality solutions. Provably optimal solutions exist in the bandit setting and for small MDPs, and at the core of these solutions are exploration schemes. However these provably optimal exploration techniques do not extend to deep RL in a straightforward way.

Bootstrapped DQN (Osband et al. [18]) is a previous attempt at adapting a theoretically verified approach to deep RL. In particular, it draws inspiration from *posterior sampling for reinforcement learning* (PSRL, Osband et al. [16], Osband and Van Roy [15]), which has near-optimal regret bounds. PSRL samples an MDP from its posterior each episode and exactly solves Q^*, its optimal Q-function. However, in high-dimensional settings, both approximating the posterior over MDPs and solving the sampled MDP are intractable. Bootstrapped DQN avoids having to establish and sample from the posterior over MDPs by instead approximating the posterior over Q^*. In addition, bootstrapped DQN uses a multi-headed neural network to represent the Q-ensemble. While the authors proposed bootstrapping to estimate the posterior distribution, their empirical findings show best performance is attained by simply relying on different initializations for the different heads, not requiring the sampling-with-replacement process that is prescribed by bootstrapping.

In this paper, we design new algorithms that build on the Q-ensemble approach from Osband et al. [18]. However, instead of using posterior sampling for exploration, we use the uncertainty estimates from the Q-ensemble. Specifically, we propose two exploration strategies. The first strategy is inspired by established UCB algorithms in the bandit setting and constructs uncertainty estimates of the Q-values. In this strategy, agents are optimistic and take actions with the highest UCB. The second strategy defines an "InfoGain" exploration bonus which measures the disagreement of the Q-ensemble. We demonstrate that our algorithms significantly improve performance on the Atari benchmark.

2 Background

2.1 Notation

We model reinforcement learning as an Markov decision process (MDP). We define an MDP as $(\mathcal{S}, \mathcal{A}, T, R, p_0, \gamma)$, in which both the state space $\mathcal{S}$ and action space $\mathcal{A}$ are discrete, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}_+$ is the transition distribution, $R : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the reward function, and $\gamma \in (0, 1]$ is a discount factor, and p_0 is the initial state distribution. We denote a transition experience as $\tau = (s, a, r, s')$ where $s' \sim T(s'|s, a)$ and $r = R(s, a)$. A policy $\pi : \mathcal{S} \mapsto \mathcal{A}$ specifies the action taken after observing a state. We denote the Q-function for policy π as $Q^\pi(s, a) := \mathbb{E}_{s,a,\pi} \left[\sum_{i=1}^{\infty} \gamma^t r_t \right]$. The optimal Q^*-function corresponds to taking the optimal policy

$$Q^*(s, a) := \sup_\pi Q^\pi(s, a)$$

and satisfies the Bellman equation

$$Q^*(s, a) = \mathbb{E}_{s' \sim T(\cdot|s,a)} \left[r + \gamma \cdot \max_{a'} Q^*(s', a) \right].$$

2.2 Exploration in reinforcement learning

A notable early optimality result in reinforcement learning was the proof by Watkins and Dayan [26, 25] that an online Q-learning algorithm is guaranteed to converge to the optimal policy, provided that every state is visited an infinite number of times. However, the convergence of Watkins' Q-learning can be prohibitively slow in MDPs where ϵ-greedy action selection explores state space randomly. Later work developed reinforcement learning algorithms with provably fast (polynomial-time) convergence (Kearns and Singh [11], Brafman and Tennenholtz [5], Strehl et al. [20]). At the core of these provably-optimal learning methods is some exploration strategy, which actively encourages the agent to visit novel state-action pairs. For example, R-MAX optimistically assumes that infrequently-visited states provide maximal reward, and delayed Q-learning initializes the Q-function with high values to ensure that each state-action is chosen enough times to drive the value down.

Since the theoretically sound RL algorithms are not computationally practical in the deep RL setting, deep RL implementations often use simple exploration methods such as ϵ-greedy and Boltzmann exploration, which are often sample-inefficient and fail to find good policies. One common approach of exploration in deep RL is to construct an exploration bonus, which adds a reward for visiting state-action pairs that are deemed to be novel or informative. In particular, several prior methods define an exploration bonus based on a density model or dynamics model. Examples include VIME by Houthooft et al. [10], which uses variational inference on the forward-dynamics model; and Tang et al. [23], Bellemare et al. [3], Ostrovski et al. [19], Fu et al. [9]. While these methods yield successful exploration in some problems, a major drawback is that this exploration bonus does not depend on the rewards, so the exploration may focus on irrelevant aspects of the environment, which are unrelated to reward.

2.3 Bayesian reinforcement learning

Earlier works on Bayesian reinforcement learning include Dearden et al. [7, 8]. Dearden et al. [7] studied Bayesian Q-learning in the model-free setting and learned the distribution of Q^*-values through Bayesian updates. The prior and posterior specification relied on several simplifying assumptions, some of which are not compatible with the MDP setting. Dearden et al. [8] took a model-based approach that updates the posterior distribution of the MDP. The algorithm samples from the MDP posterior multiple times and solving the Q^* values at every step. Strens [21] proposed posterior sampling for reinforcement learning (PSRL). PSRL instead takes a single sample of the MDP from the posterior in each episode and solves the Q^* values. Recent works including Osband et al. [16] and Osband and Van Roy [15] established near-optimal Bayesian regret bounds for episodic RL. The computational cost of these methods is intractable for high dimensional RL.

2.4 Bootstrapped DQN

Inspired by PSRL, but wanting to reduce computational cost, prior work developed approximate methods. Osband et al. [17] proposed randomized least-square value iteration for linearly-

parameterized value functions. Bootstrapped DQN Osband et al. [18] applies to Q^*-functions parameterized by deep neural networks. Bootstrapped DQN (Osband et al. [18]) maintains a Q-ensemble, represented by a multi-head neural net structure to parameterize $K \in \mathbb{N}_+$ Q^*-functions. This multi-head structure shares the convolution layers but includes multiple "heads", each of which defines a Q^*-function Q_k.

Bootstrapped DQN diversifies the Q-ensemble through two mechanisms. The first mechanism is independent initialization. The second mechanism applies different samples to train each Q^*-function. These Q^*-functions can be trained simultaneously by combining their loss functions with the help of a random mask $m_\tau \in \mathbb{R}_+^K$

$$ L = \sum_{\tau \in B_{\mathrm{mini}}} \sum_{k=1}^{K} m_\tau^k \cdot (Q^k(s, a; \theta) - y_\tau^{Q_k})^2, $$

where $y_\tau^{Q_k}$ is the target of the kth Q^*-function. Thus, the transition τ updates Q_k only if m_τ^k is nonzero. To avoid the overestimation issue in DQN, bootstrapped DQN calculates the target value $y_\tau^{Q_k}$ using the approach of Double DQN (Van Hasselt et al. [24]), such that the current $Q_k(\cdot; \theta_t)$ network determines the optimal action and the target network $Q_k(\cdot; \theta^-)$ estimates the value

$$ y_\tau^{Q_k} = r + \gamma \max_a Q^k(s', \operatorname*{argmax}_a Q_k(s', a; \theta_t); \theta^-). $$

In their experiments on Atari games, Osband et al. [18] set the mask $m_\tau = (1, \ldots, 1)$ such that all $\{Q_k\}$ are trained with the same samples and their only difference is initialization. Bootstrapped DQN picks one Q_k uniformly at random at the start of an episode and follows the greedy action $a_t = \operatorname*{argmax}_a Q_k(s_t, a)$ for the whole episode.

3 Approximating Bayesian Q-learning with Q-Ensembles

Ignoring computational costs, the ideal Bayesian approach to reinforcement learning is to maintain a posterior over the MDP. However, with limited computation and model capacity, it is more tractable to maintain a posterior of the Q^*-function. In this section, we first derive a posterior update formula for the Q^*-function, which turns out to depend on the transition Markov chain (Section 3.1). The Bellman equation emerges as an approximation of the log-likelihood. This motivates using a Q-ensemble as a particle-based approach to approximate the posterior over Q^*-function and an Ensemble Voting algorithm (Section 3.2).

3.1 Bayesian update for Q^*

An MDP is specified by the transition probability T and the reward function R. Unlike prior works outlined in Section 2.3 which learned the posterior of the MDP, we will consider the joint distribution over (Q^*, T). Note that R can be recovered from Q^* given T. So (Q^*, T) determines a unique MDP. Based on observed transitions, we update the posterior joint distribution. Recall that the Q^*-function satisfies the Bellman equation

$$ Q(s, a) = r + \mathbb{E}_{s' \sim T(\cdot|s,a)} \left[\gamma \max_{a'} Q(s', a') \right]. $$

Denote the joint prior distribution as $p(Q^*, T)$ and the posterior as $\tilde{p}$. Given transition $\tau = (s, a, r, s')$, we apply Bayes' formula to expand the posterior:

$$ \begin{aligned} \tilde{p}(Q^*, T | \tau) &= \frac{p(\tau | Q^*, T) \cdot p(Q^*, T)}{Z} \\ &= \frac{p(Q^*, T) \cdot p(s, a, s' | Q^*, T) \cdot p(r | Q^*, T, (s, a, s'))}{Z}, \end{aligned} \tag{1} $$

where Z is a normalizing constant. Next, we calculate the two conditional probabilities in (1)

$$ p(s, a, s' | Q^*, T) = p(s, a, s' | T) = T(s' | s, a), \tag{2} $$

and

$$ \begin{aligned} p(r | Q^*, T, (s, a, s')) &= p(r | Q^*, T, (s, a)) \\ &= \mathbb{1}_{\{Q^*(s,a) = r + \gamma \cdot \mathbb{E}_{s'' \sim T(\cdot|s,a)} \max_{a'} Q^*(s'', a')\}} \\ &:= \mathbb{1}(Q^*, T), \end{aligned} \tag{3} $$

3

where $\mathbb{1}_{\{\cdot\}}$ is the indicator function and in the last equation we abbreviate it as $\mathbb{1}(Q, T)$. Substituting (2) and (3) into (1), we obtain the posterior of Q^*

$$\tilde{p}(Q^*, T|\tau) = \frac{p(Q^*, T) \cdot T(s'|s, a) \cdot \mathbb{1}(Q^*, T)}{Z}. \tag{4}$$

We point out that exact Bayesian Q-learning is intractable in high-dimensional RL due to the large space of (Q^*, T).

3.2 Q-learning with Q-ensembles

In this section, we make several approximations to the Q^*-posterior update and derive a tractable algorithm. First, we approximate the prior of Q^* by sampling $K \in \mathbb{N}_+$ independently initialized Q^*-functions $\{Q_k\}_{k=1}^K$. Next, we update them as more transitions are observed. The resulting $\{Q_k\}$ approximate samples drawn from the posterior. The agent chooses the action by taking a majority vote from the actions determined by each Q_k. We display our method, Ensemble Voting, in Algorithm 1.

We derive the update rule for $\{Q_k\}$ after observing each new transition $\tau = (s, a, r, s')$. Given a $Q^* = Q_k$ compatible with T, the prior factors into

$$p(Q^*, T) = p(Q^*|T) \cdot p(T) = \mathbb{1}_{\{Q^*=Q_k\}} \cdot p(T). \tag{5}$$

Substitute (5) into (4) and we obtain the corresponding posterior for each Q_k after observing τ

$$\tilde{p}(Q_k, T|\tau) = \frac{p(T) \cdot T(s'|s, a)}{Z} \cdot \mathbb{1}(Q_k, T). \tag{6}$$

We update Q_k according to

$$Q_k \leftarrow \underset{Q_k}{\operatorname{argmax}} \, \tilde{p}(Q_k, T|\tau). \tag{7}$$

We demonstrate that we can approximate (7) by applying a standard Bellman update to Q_k. Instead of maximizing the posterior in (7), we maximize an arbitrarily tight upper bound. Note that the indicator function is controlled by an exponential

$$\mathbb{1}(Q_k, T) \leq \exp\left(-c \cdot \left[Q_k(s, a) - \left(r + \gamma \cdot \mathbb{E}_{s'' \sim T(\cdot|s, a), T \sim \tilde{p}(T|\tau)} \max_{a'} Q_k(s'', a')\right)\right]^2\right), \tag{8}$$

where $c > 0$ and $\tilde{p}(T|\tau)$ is the posterior of the transition dynamics T. Equality holds in (8) when $c \to +\infty$. Substitute (8) into (6) and we obtain an upper bound of the posterior's log-likelihood

$$\begin{aligned}
\log \tilde{p}(Q_k, T|\tau) &= \log\left(p(T) \cdot T(s'|s, a)\right) + \log \mathbb{1}(Q_k, T) - \log Z \\
&\leq \log\left(p(T) \cdot T(s'|s, a)\right) - \log Z \\
&\quad - c \cdot \left[Q_k(s, a) - \left(r + \gamma \cdot \mathbb{E}_{s'' \sim T(\cdot|s, a), T \sim \tilde{p}(T|\tau)} \max_{a'} Q_k(s'', a')\right)\right]^2.
\end{aligned} \tag{9}$$

By maximizing the upper bound (9), the update of Q_k becomes

$$Q_k \leftarrow \underset{Q_k}{\operatorname{argmax}} \left\{-c \cdot \left[Q_k(s, a) - \left(r + \gamma \cdot \mathbb{E}_{s'' \sim T(\cdot|s, a), T \sim \tilde{p}(T|\tau)} \max_{a'} Q_k(s'', a')\right)\right]^2\right\}. \tag{10}$$

We note that the exact value of c has no impact on the solution of (10). Thus one can take the limit of $c \to +\infty$ to make the upper bound of (9) arbitrarily tight. However, (10) is not tractable because the expectation in (10) is taken with respect to the posterior $\tilde{p}(T|\tau)$ of the transition T. To overcome this challenge, we approximate the posterior distribution of $T \sim \tilde{p}(T|\tau)$ by sampling a minibatch B_{mini} of transitions (s, a, r, s') from an experience replay buffer B that stores observed transitions, such that the update of each Q_k becomes a standard Bellman update

$$Q_k \leftarrow \underset{Q_k}{\operatorname{argmin}} \, \mathbb{E}_{(s, a, r, s') \in B_{\text{mini}}} \left[Q_k(s, a) - \left(r + \gamma \cdot \max_{a'} Q_k(s', a')\right)\right]^2. \tag{11}$$

We point out that the action choice of Algorithm 1 is exploitation only. In the next section, we propose two exploration strategies.

Algorithm 1 Ensemble Voting

1: **Input**: $K \in \mathbb{N}_+$ copies of independently initialized Q^*-functions $\{Q_k\}_{k=1}^K$.
2: Let B be a replay buffer storing transitions for training
3: **for** each episode **do** do
4:     Obtain initial state from environment s_0
5:     **for** step $t = 1, \ldots$ until end of episode **do**
6:         Pick an action according to $a_t = \text{MajorityVote}(\{\text{argmax}_a Q_k(s_t, a)\}_{k=1}^K)$
7:         Execute a_t. Receive state s_{t+1} and reward r_t from the environment
8:         Add (s_t, a_t, r_t, s_{t+1}) to replay buffer B
9:         At learning interval, sample random minibatch and update $\{Q_k\}$ according to (11)
10:     **end for**
11: **end for**

---

4   Exploration Strategies Using *Q*-Ensembles

In this section, we propose two exploration strategies based on uncertainty estimates from the Q-ensemble. In Section 4.1, we construct an upper-confidence bound based on the the Q-ensemble. In Section 4.2, we define an "InfoGain" exploration bonus from the Q-ensemble. We summarize the algorithm in Section 4.3.

4.1   UCB exploration

In this section, we propose optimism-based exploration by adapting the UCB algorithms (Auer et al. [2], Audibert et al. [1]) from the bandit setting. The UCB algorithms maintain an upper-confidence bound for each arm, such that the expected reward from pulling each arm is smaller than this bound with high probability. At every time step, the agent optimistically chooses the arm with the highest UCB. Auer et al. [2] constructed the UCB based on empirical reward and the number of times each arm is chosen. Audibert et al. [1] incorporated the empirical variance of each arm's reward into the UCB, such that at time step t, an arm A_t is pulled according to

$$A_t = \underset{i}{\text{argmax}} \left\{ \hat{r}_{i,t} + c_1 \cdot \sqrt{\frac{\hat{V}_{i,t} \log(t)}{n_{i,t}}} + c_2 \cdot \frac{\log(t)}{n_{i,t}} \right\}$$

where $\hat{r}_{i,t}$ and $\hat{V}_{i,t}$ are the empirical reward and variance of arm i at time t, $n_{i,t}$ is the number of times arm i has been pulled up to time t, and c_1, c_2 are positive constants.

We extend the intuition of UCB algorithms to the RL setting. Using the outputs of the $\{Q_k\}$ functions, we construct a UCB by adding the empirical standard deviation $\tilde{\sigma}(s_t, a)$ of $\{Q_k(s_t, a)\}_{k=1}^K$ to the empirical mean $\tilde{\mu}(s_t, a)$ of $\{Q_k(s_t, a)\}_{k=1}^K$. The agent chooses the action that maximizes this UCB

$$a_t \in \underset{a}{\text{argmax}} \left\{ \tilde{\mu}(s_t, a) + \lambda \cdot \tilde{\sigma}(s_t, a) \right\}, \tag{12}$$

where $\lambda \in \mathbb{R}_+$ is a hyperparameter.

4.2   InfoGain exploration

In this section, we propose an "InfoGain" exploration bonus, which encourages agents to gain information about the Q^*-function. Similar to Sun et al. [22], we define the information gain from observing an additional transition τ_n as

$$H_{\tau_t | \tau_1, \ldots, \tau_{n-1}} = D_{KL}(\tilde{p}(Q^* | \tau_1, \ldots, \tau_n) || \tilde{p}(Q^* | \tau_1, \ldots, \tau_{n-1}))$$

where $\tilde{p}(Q^* | \tau_1, \ldots, \tau_n)$ is the posterior distribution of Q^* after observing a sequence of transitions $(\tau_1, \ldots, \tau_n)$. The total information gain is

$$H_{\tau_1, \ldots, \tau_N} = \sum_{n=1}^N H_{\tau_n | \tau_1, \ldots, \tau_{n-1}}. \tag{13}$$

Our Ensemble Voting, Algorithm 1, does not maintain the posterior $\tilde{p}$, thus we cannot calculate (13) explicitly. Instead, inspired by Lakshminarayanan et al. [13], we define an InfoGain exploration

bonus that measures the disagreement among $\{Q_k\}$. Note that

$$H_{\tau_1,\ldots,\tau_N} + \mathsf{H}(\tilde{p}(Q^*|\tau_1,\ldots,\tau_N)) = \mathsf{H}(p(Q^*)),$$

where $\mathsf{H}(\cdot)$ is the entropy. If $H_{\tau_1,\ldots,\tau_N}$ is small, then the posterior distribution has high entropy and high residual information. Since $\{Q_k\}$ are approximate samples from the posterior, high entropy of the posterior leads to large discrepancy among $\{Q_k\}$. Thus, the exploration bonus is monotonous with respect to the residual information in the posterior $\mathsf{H}(\tilde{p}(Q^*|\tau_1,\ldots,\tau_N))$. We first compute the Boltzmann distribution for each Q_k

$$P_{\mathsf{T},k}(a|s) = \frac{\exp\left(Q_k(s,a)/\mathsf{T}\right)}{\sum_{a'} \exp\left(Q_k(s,a')/\mathsf{T}\right)},$$

where $\mathsf{T} > 0$ is a temperature parameter. Next, calculate the average Boltzmann distribution

$$P_{\mathsf{T},\mathrm{avg}} = \frac{1}{K} \cdot \sum_{k=1}^{K} P_{\mathsf{T},k}(a|s).$$

The InfoGain exploration bonus is the average KL-divergence from $\{P_{\mathsf{T},k}\}_{k=1}^{K}$ to $P_{\mathsf{T},\mathrm{avg}}$

$$b_{\mathsf{T}}(s) = \frac{1}{K} \cdot \sum_{k=1}^{K} \mathrm{D}_{KL}[P_{\mathsf{T},k}||P_{\mathsf{T},\mathrm{avg}}]. \tag{14}$$

The modified reward is

$$\hat{r}(s,a,s') = r(s,a) + \rho \cdot b_{\mathsf{T}}(s), \tag{15}$$

where $\rho \in \mathbb{R}_+$ is a hyperparameter that controls the degree of exploration.

The exploration bonus $b_{\mathsf{T}}(s_t)$ encourages the agent to explore where $\{Q_k\}$ disagree. The temperature parameter T controls the sensitivity to discrepancies among $\{Q_k\}$. When $\mathsf{T} \to +\infty$, $\{P_{\mathsf{T},k}\}$ converge to the uniform distribution on the action space and $b_{\mathsf{T}}(s) \to 0$. When T is small, the differences among $\{Q_k\}$ are magnified and $b_{\mathsf{T}}(s)$ is large.

4.3 Exploration algorithm

We present Algorithm 2, which combines the UCB exploration of Section 4.1 and the InfoGain exploration bonus in Section 4.2. The hyperparemeters λ and ρ control the degrees of exploration jointly and the temperature T specifies the Boltzmann distribution. The optimal exploration hyperparameters vary for each game. In Section 5, we compare the performance of our algorithms on Atari games using a consistent set of parameters.

Algorithm 2 UCB + InfoGain Exploration with Q-Ensembles

1: **Input:** Value function networks Q with K outputs $\{Q_k\}_{k=1}^{K}$. Hyperparameters T, λ, and ρ.
2: Let B be a replay buffer storing experience for training.
3: **for** each episode **do**
4:     Obtain initial state from environment s_0
5:     **for** step $t = 1,\ldots$ until end of episode **do**
6:         Pick an action according to $a_t \in \mathrm{argmax}_a\left\{\tilde{\mu}(s_t,a) + \lambda \cdot \tilde{\sigma}(s_t,a)\right\}$
7:         Receive state s_{t+1} and reward r_t from environment, having taken action a_t
8:         Calculate exploration bonus $b_{\mathsf{T}}(s_t)$ according to (14)
9:         Add $(s_t, a_t, r_t + \rho \cdot b_{\mathsf{T}}(s_t), s_{t+1})$ to replay buffer B
10:        At learning interval, sample random minibatch and update $\{Q_k\}$ according to (11)
11:     **end for**
12: **end for**

---

5 Experiment

In this section, we conduct experiments to answer the following questions:

1. does Ensemble Voting, Algorithm 1, improve upon existing algorithms including Double DQN and bootstrapped DQN?

2. are the proposed exploration strategies of Algorithm 2 effective in improving learning compared to Algorithm 1?

We evaluate the algorithms on each Atari game of the Arcade Learning Environment (Bellemare et al. [4]). We use the multi-head neural net architecture of Osband et al. [18]. We fix the common hyperparameters of all algorithms based on a well-tuned double DQN implementation, which uses the Adam optimizer (Kingma and Ba [12]), different learning rate and exploration schedules compared to Mnih et al. [14]. Appendix A tabulates the hyperparameters. The number of $\{Q_k\}$ functions is $K = 10$. Experiments are conducted on the OpenAI Gym platform (Brockman et al. [6]) and trained with 40 million frames and 2 trials on each game.

We take the following directions to evaluate the performance of our algorithms:

1. we compare Algorithm 1 against Double DQN and bootstrapped DQN,

2. we isolate the impact of UCB exploration by comparing Algorithm 2 with $\lambda = 0.1, \rho = 0$, denoted as `ucb exploration`, against Algorithm 1.

3. we study the benefit of InfoGain exploration. We define `ucb+infogain exploration` as Algorithm 2 with $\lambda = 0.1, \rho = 1$ and a specified value of temperature T. We compare its performance against `ucb exploration`.

Figure 1 compares the normalized learning curves of all algorithms across Atari games. Overall, Algorithm 1 outperforms both Double DQN and bootstrapped DQN. With exploration, `ucb exploration` and `ucb+infogain exploration` improves further by outperforming Ensemble Voting. In Appendix B, we tabulate the maximal mean reward in 100 consecutive episodes for Ensemble Voting, `ucb exploration`, bootstrapped DQN and Double DQN. Without exploration, Ensemble Voting already achieves higher maximal mean reward than both Double DQN and bootstrapped DQN in a majority of Atari games. `ucb exploration` achieves the highest maximal mean reward among these four algorithms in a dominating majority.

Figure 2 displays the learning curves of these five algorithms on a set of six games. Ensemble Voting outperforms Double DQN and bootstrapped DQN. `ucb exploration` outperforms Ensemble Voting. The impact of InfoGain exploration varies. UCB exploration achieves sufficient exploration in games including Demon Attack and Kangaroo and Riverraid, while InfoGain exploration further improves learning on Enduro, Seaquest, and Up N Down. The effect of InfoGain exploration depends on the choice of the temperature T. The optimal temperature varies across games and Appendix C displays the behavior of `ucb+infogain exploration` with different temperature values.

6   Conclusion

We proposed a Q-ensemble approach to deep Q-learning, a computational practical algorithm inspired by Bayesian reinforcement learning that outperforms Double DQN and bootstrapped DQN, as evaluated on Atari. Key ingredients are two exploration strategies: UCB and InfoGain, both inspired by bandit algorithms. Our experiments show that the exploration strategies achieve improved learning performance on the majority of Atari games.

References

[1] Jean-Yves Audibert, Rémi Munos, and Csaba Szepesvári. Exploration–exploitation tradeoff using variance estimates in multi-armed bandits. *Theor. Comput. Sci.*, 410(19):1876–1902, 2009.

[2] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2-3):235–256, 2002.

[3] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *NIPS*, pages 1471–1479, 2016.

[4] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *J. Artif. Intell. Res.*, 47:253–279, 2013.
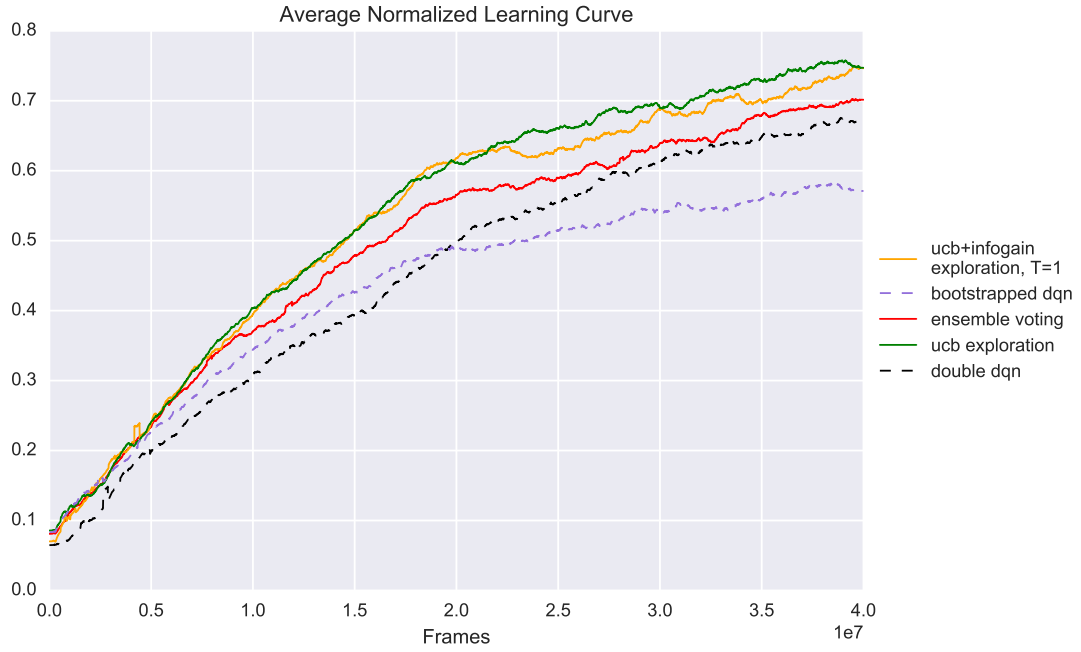
Figure 1: Comparison of all algorithms in normalized curve. The normalized learning curve is calculated as follows: first, we normalize learning curves for all algorithms in the same game to the interval $[0, 1]$; next, average the normalized learning curve from all games for each algorithm.
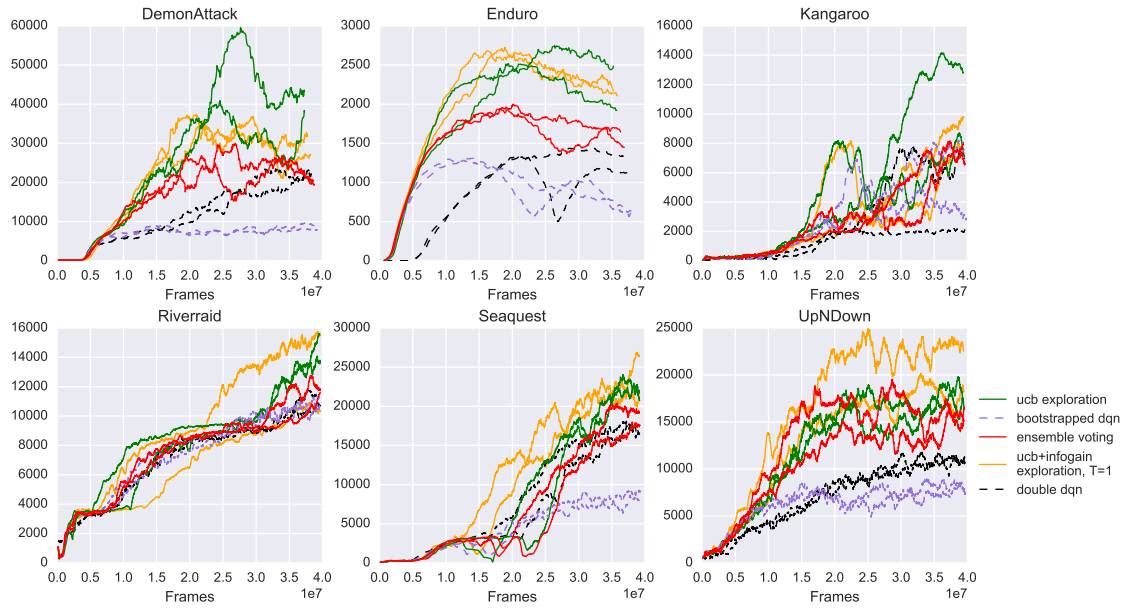

Figure 2: Comparison of algorithms against Double DQN and bootstrapped DQN.

[5] Ronen I Brafman and Moshe Tennenholtz. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *J. Mach. Learn. Res.*, 3(Oct):213–231, 2002.

[6] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.

[7] Richard Dearden, Nir Friedman, and Stuart Russell. Bayesian Q-learning. In *AAAI/IAAI*, pages 761–768, 1998.

[8] Richard Dearden, Nir Friedman, and David Andre. Model based Bayesian exploration. In *UAI*, pages 150–159, 1999.

[9] Justin Fu, John D Co-Reyes, and Sergey Levine. EX2: Exploration with exemplar models for deep reinforcement learning. *arXiv preprint arXiv:1703.01260*, 2017.

[10] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. VIME: Variational information maximizing exploration. In *NIPS*, pages 1109–1117, 2016.

[11] Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Mach. Learn.*, 49(2-3):209–232, 2002.

[12] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv preprint arXiv:1612.01474*, 2016.

[14] 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.

[15] Ian Osband and Benjamin Van Roy. Why is posterior sampling better than optimism for reinforcement learning. *arXiv preprint arXiv:1607.00215*, 2016.

[16] Ian Osband, Dan Russo, and Benjamin Van Roy. (More) efficient reinforcement learning via posterior sampling. In *NIPS*, pages 3003–3011, 2013.

[17] Ian Osband, Benjamin Van Roy, and Zheng Wen. Generalization and exploration via randomized value functions. *arXiv preprint arXiv:1402.0635*, 2014.

[18] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped DQN. In *NIPS*, pages 4026–4034, 2016.

[19] Georg Ostrovski, Marc G Bellemare, Aaron van den Oord, and Remi Munos. Count-based exploration with neural density models. *arXiv preprint arXiv:1703.01310*, 2017.

[20] Alexander L Strehl, Lihong Li, Eric Wiewiora, John Langford, and Michael L Littman. Pac model-free reinforcement learning. In *ICML*, pages 881–888. ACM, 2006.

[21] Malcolm Strens. A Bayesian framework for reinforcement learning. In *ICML*, pages 943–950, 2000.

[22] Yi Sun, Faustino Gomez, and Jürgen Schmidhuber. Planning to be surprised: Optimal Bayesian exploration in dynamic environments. In *ICAGI*, pages 41–51. Springer, 2011.

[23] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. # Exploration: A study of count-based exploration for deep reinforcement learning. *arXiv preprint arXiv:1611.04717*, 2016.

[24] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double Q-learning. In *AAAI*, pages 2094–2100, 2016.

[25] Christopher JCH Watkins and Peter Dayan. Q-learning. *Mach. Learn.*, 8(3-4):279–292, 1992.

[26] Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. PhD thesis, University of Cambridge England, 1989.

A Hyperparameters

We tabulate the hyperparameters in our well-tuned implementation of double DQN in Table 1:

hyperparameter	value	descriptions
total training frames	40 million	Length of training for each game.
minibatch size	32	Size of minibatch samples for each parameter update.
replay buffer size	1000000	The number of most recent frames stored in replay buffer.
agent history length	4	The number of most recent frames concatenated as input to the Q network. Total number of iterations = total training frames / agent history length.
target network update frequency	10000	The frequency of updating target network, in the number of parameter updates.
discount factor	0.99	Discount factor for Q value.
action repeat	4	Repeat each action selected by the agent this many times. A value of 4 means the agent sees every 4th frame.
update frequency	4	The number of actions between successive parameter updates.
optimizer	Adam	Optimizer for parameter updates.
β_1	0.9	Adam optimizer parameter.
β_2	0.99	Adam optimizer parameter.
ϵ	10^{-4}	Adam optimizer parameter.
learning rate schedule	$\begin{cases} 10^{-4} & t \le 10^6 \\ Interp(10^{-4}, 5*10^{-5}) & \text{otherwise} \\ 5*10^{-5} & t > 5*10^6 \end{cases}$	Learning rate for Adam optimizer, as a function of iteration t.
exploration schedule	$\begin{cases} Interp(1, 0.1) & t < 10^6 \\ Interp(0.1, 0.01) & \text{otherwise} \\ 0.01 & t > 5*10^6 \end{cases}$	Probability of random action in ϵ-greedy exploration, as a function of the iteration t .
replay start size	50000	Number of uniform random actions taken before learning starts.

Table 1: Double DQN hyperparameters

These hyperparameters are selected based on performances of seven Atari games: Beam Rider, Breakout, Pong, Enduro, Qbert, Seaquest, and Space Invaders. $Interp(\cdot, \cdot)$ is linear interpolation between two values.

B   Results table

	Bootstrapped DQN	Double DQN	Ensemble Voting	UCB-Exploration
Alien	1445.1	2059.7	2282.8	**2817.6**
Amidar	430.58	667.5	**683.72**	663.8
Assault	2519.06	2820.61	3213.58	**3702.76**
Asterix	3829.0	7639.5	**8740.0**	8732.0
Asteriods	1009.5	1002.3	**1149.3**	1007.8
Atlantis	1314058.0	1982677.0	1786305.0	**2016145.0**
Bank Heist	795.1	789.9	869.4	**906.9**
Battle Zone	26230.0	24880.0	**27430.0**	26770.0
Beam Rider	8006.58	7743.74	7991.9	**9188.26**
Bowling	28.62	30.92	32.92	**38.06**
Boxing	85.91	94.07	94.47	**98.08**
Breakout	400.22	**467.45**	426.78	411.31
Centipede	5328.77	5177.51	6153.28	**6237.18**
Chopper Command	2153.0	3260.0	3544.0	**3677.0**
Crazy Climber	110926.0	124456.0	126677.0	**127754.0**
Demon Attack	9811.45	23562.55	30004.4	**59861.9**
Double Dunk	-10.82	-14.58	-11.94	**-4.08**
Enduro	1314.31	1439.59	1999.88	**2752.55**
Fishing Derby	21.89	23.69	**30.02**	29.71
Freeway	33.57	32.93	33.92	**33.96**
Frostbite	1284.8	529.2	1196.0	**1903.0**
Gopher	7652.2	12030.0	10993.2	**12910.8**
Gravitar	227.5	279.5	**371.5**	318.0
Ice Hockey	-4.62	-4.63	**-1.73**	-4.71
Jamesbond	594.5	594.0	602.0	**710.0**
Kangaroo	8186.0	7787.0	8174.0	**14196.0**
Krull	8537.52	8517.91	8669.17	**9171.61**
Kung Fu Master	24153.0	**32896.0**	30988.0	31291.0
Montezuma Revenge	2.0	4.0	1.0	4.0
Ms Pacman	2508.7	2498.1	3039.7	**3425.4**
Name This Game	8212.4	**9806.9**	9255.1	9570.5
Pitfall	-5.99	-7.57	-3.37	**-1.47**
Pong	21.0	20.67	21.0	20.95
Private Eye	1815.19	788.63	**1845.28**	1252.01
Qbert	10557.25	6529.5	12036.5	**14198.25**
Riverraid	11528.0	11834.7	12785.8	**15622.2**
Road Runner	52489.0	49039.0	**54768.0**	53596.0
Robotank	21.03	29.8	31.83	**41.04**
Seaquest	9320.7	18056.4	20458.6	**24001.6**
Space Invaders	1549.9	1917.5	1890.8	**2626.55**
Star Gunner	20115.0	**52283.0**	41684.0	47367.0
Tennis	-15.11	-14.04	-11.63	**-7.8**
Time Pilot	5088.0	5548.0	6153.0	**6490.0**
Tutankham	167.47	**223.43**	208.61	200.76
Up N Down	9049.1	11815.3	19528.3	**19827.3**
Venture	**115.0**	96.0	78.0	67.0
Video Pinball	364600.85	**374686.89**	343380.29	372564.11
Wizard Of Wor	2860.0	3877.0	5451.0	**5873.0**
Zaxxon	592.0	**8903.0**	3901.0	3695.0

Table 2: Comparison of maximal mean rewards achieved by agents. Maximal mean reward is calculated in a window of 100 consecutive episodes. Bold denotes the highest value in each row.

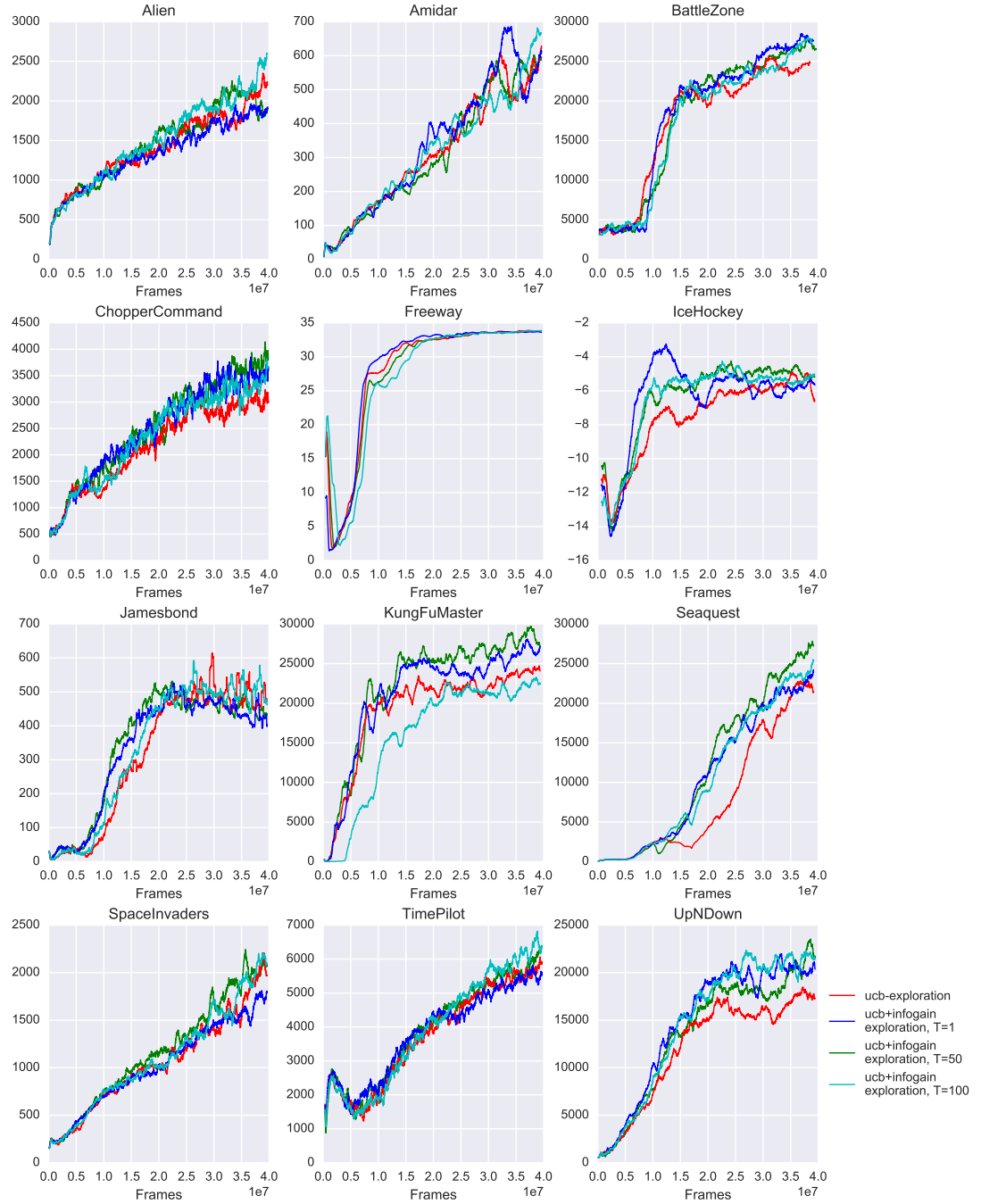# C UCB+InfoGain exploration with different temperatures

Figure 3: Comparison of UCB+InfoGain exploration with different temperatures versus UCB exploration.