Interval Q-Learning: Balancing Deep and Wide Exploration

Grigory Neustroev
Delft University of Technology
Delft, the Netherlands
g.neustroev@tudelft.nl

Canmanie T. Ponnambalam
Delft University of Technology
Delft, the Netherlands
c.t.ponnambalam@tudelft.nl

Mathijs M. de Weerdt
Delft University of Technology
Delft, the Netherlands
m.m.deweerdt@tudelft.nl

Matthijs T. J. Spaan
Delft University of Technology
Delft, the Netherlands
m.t.j.spaan@tudelft.nl

ABSTRACT

Reinforcement learning requires exploration, leading to repeated execution of sub-optimal actions. Naïve exploration techniques address this problem by changing gradually from exploration to exploitation. This approach employs a wide search resulting in exhaustive exploration and low sample-efficiency. More advanced search methods explore optimistically based on an upper bound estimate of expected rewards. These methods employ deep search, aiming to reach states not previously visited. Another deep search strategy is found in action-elimination methods, which aim to discover and eliminate sub-optimal actions. Despite the effectiveness of advanced deep search strategies, some problems are better suited to naïve exploration. We devise a new method, called Interval Q-Learning, that finds a balance between wide and deep search. It assigns a small probability to taking sub-optimal actions and combines both greedy and optimistic exploration. This allows for fast convergence to a near-optimal policy, and then exploration around it. We demonstrate the performance of tabular and deep Q-network versions of Interval Q-Learning, showing that it offers convergence speed-up both in problems that favor wide exploration methods and those that favor deep search strategies.

KEYWORDS

Reinforcement learning, Q-learning, deep learning, guided exploration, action elimination

1 INTRODUCTION

Reinforcement learning methods require agents to repeatedly take bad or irrelevant actions during learning. In simulated environments, the consequences of the exploration-exploitation trade-off are navigated with relative ease; agents can explore freely and exhaustively, eventually converging to the best possible solution. In a real-life scenario, however, the consequences of taking bad actions may be severe, with sub-optimal performance resulting in real losses such as physical damage to a robot. In these situations, exhaustively taking all possible actions is no longer suitable.

A naïve answer to the exploration-exploitation dilemma is the common ϵ-greedy approach; one where the exploration goes *wide*, trying numerous random actions hoping, by chance, to find the best ones [35]. Some have argued that these "dithering" approaches are not reliable in the general case and more strategic methods are needed [26]. Examples of such advanced methods are, for example, those that apply the notion of optimism to address the issue of unnecessary exploration [15, 17, 34]. In these cases, agents explore based on an optimistic estimate of the expected reward, with a bonus applied to encourage exploring state-actions which have not yet been visited. These methods go *deep* in their exploration, persistently trying to reach states that are yet to be seen, rather than searching for the best actions for the states they already encountered. Yet another deep approach to exploration is to maintain both an upper and lower bound over the expected reward and eliminate actions that are sub-optimal, however such 'action elimination' methods [9] can be slow in practice as many samples are needed to confidently deem actions sub-optimal. Furthermore, the benefits of exploration strategies can be tied to the specifics of the problem to which they are applied [21, 28]. Our experiments reveal that some problems favor naïve wide search strategies over more advanced exploration methods. Exploration strategies which can address a wide variety of problems are necessary in order to develop versatile reinforcement learning solutions.

We present a novel method, Interval Q-Learning, that makes use of bounded value functions, action elimination and optimistic exploration to achieve fast convergence, exploring both *wide* and *deep*. Interval Q-Learning directs exploration away from sub-optimal actions, allowing it to converge to a good solution fast. Because it refrains from eliminating actions completely, this exploration around an initial good solution ultimately leads to the optimal.

In this paper, we briefly discuss literature regarding several approaches to exploration in learning algorithms. We then propose our method of guided exploration, Interval Q-Learning, which combines existing exploration approaches to achieve fast convergence. We further include an Interval Q-Learning variant that integrates with deep Q-networks (DQNs) [22], as deep learning techniques are capable of solving problems requiring human-level control [23]. We demonstrate the effectiveness of our approach, comparing its performance to other directed exploration methods for two different types of benchmark problems that are best solved by either wide or deep search. Interval Q-Learning is shown to effectively reduce the number of sub-optimal actions it takes and converge quickly to a good solution.

2 RELATED WORK

The first instance of exploration steered by uncertainty appeared in a method for finite-state MDPs that guides action selection towards the largest confidence interval of estimated rewards for each state-action pair [15]. In addition to subsequent appearances in planning literature (such as [2, 32]) this idea is present in reinforcement learning. Rmax, for example, directs exploration towards parts of the state-action space for which the transition function is not well known [4]. Model-Based Interval Estimation (MBIE) methods [33], as well as the KWIK framework [18] apply similar principles.

Another approach to guiding exploration is to identify sub-optimal actions by maintaining upper and lower bounds on the value of each state-action (the Q-value). Methods that use such bounds to facilitate action elimination appeared in early dynamic programming literature [20] and have shown up in solution techniques for bandit problems [8, 9, 30], Markov decision processes (MDPs) [16], and in more general models [7] since then. Updating an upper bound further allows for exploration according to the notion of optimism. This is applied in the popular Upper Confidence Bound algorithm (UCB) [1] which effectively trades exploration and exploitation by exploring according to the optimistic estimate of expected reward. A Q-Learning variant of UCB [14], as well as a few of its modifications [24, 36], was shown to be provably sample efficient. Another recent method, Optimistic Pessimistically Initialized Q-Learning (OPIQ) extends the principle of optimistic exploration to the deep learning setting [29].

3 BACKGROUND

We use Markov decision processes (MDPs) to model a dynamic environment in which an agent interacts by choosing and executing actions. In an MDP, at each time step t, an agent observes the state of the environment $s_t \in \mathbb{S}$ and chooses an action $a_t \in \mathbb{A}$. The environment transitions to a new state $s_{t+1} \sim p(\cdot|s_t, a_t)$, sampled using a transition function $p : \mathbb{X} \to \Delta(\mathbb{S})$ (where $\mathbb{X} \triangleq \mathbb{S} \times \mathbb{A}$ is the set of all state-action pairs, and $\Delta(\cdot)$ denotes the set of all distributions). The agent observes the next state s_{t+1} and receives a reward $r_t \sim r(\cdot|s_t, a_t)$, where $r : \mathbb{X} \to \Delta(\mathbb{R})$. We denote the expected reward of a state-action pair (s, a) as $r(s, a) \triangleq \mathbb{E}[r(\cdot|s, a)]$.

Given the initial state s, each course of actions, known as policy π, has a value $V_\pi(s)$, equal to the sum of expected discounted rewards:

$$V_\pi(s) \triangleq \mathbb{E}_{\pi, s_0=s} \sum_{t=0}^{\infty} \gamma^t r_t.$$

The goal is to find an optimal policy, that is, a policy π^* with the highest possible value $V_{\pi^*}(s) = V^*(s) \triangleq \max_\pi V_\pi(s)$. This can be done by computing the so-called optimal action values or *optimal Q-values* $Q^*(s, a)$ as the solution of

$$Q^*(s, a) = r(s, a) + \gamma \sum_{s' \in \mathbb{S}} p(s'|s, a)V^*(s') \text{ and } V^*(s) = \max_{a \in \mathbb{A}} Q^*(s, a).$$

A policy π_{Greedy} is called *greedy* with respect to Q-values, if for each state it uses an action with the highest Q-value: $\pi_{\text{Greedy}}(s) \in \arg\max_{a \in \mathbb{A}} Q(s, a)$. A policy that is greedy with respect to Q^* is known to be an optimal policy. Thus, a problem of finding an optimal policy is reduced to a problem of finding optimal Q-values.

3.1 Tabular Q-Learning

In reinforcement learning (RL), the transition and reward functions of a Markov decision process are not necessarily known. A popular method for solving reinforcement learning problems is Q-Learning [37]. Q-Learning is a model-free method, which means that it does not try to estimate the reward and transition functions. Instead, it maintains Q-values for each state-action pair; through collected experiences in the environment, these values are updated over time:

$$Q_{t+1}(s, a) = \big(1 - \alpha_t(s, a)\big)Q_t(s, a) + \alpha_t(s, a)U_t(r_t, s_{t+1}), \quad (1)$$

$$U_t(r, s') \triangleq r + \gamma \max_{a \in \mathbb{A}} Q_t(s', a). \quad (2)$$

For an appropriate choice of α, called the *learning rate*, the sequence $\{Q_t\}_{t=0}^{\infty}$ converges to Q^* with probability 1 if the state-action space $\mathbb{X}$ is finite $|\mathbb{X}| < \infty$, and the rewards function r is uniformly bounded $r_{\min} \le r \le r_{\max}$ [13]. Usually, a learning agent follows a policy π and only the values along the observed path are updated, that is, $\alpha_t(s, a) = 0$ if $s \ne s_t$ or $a \ne a_t = \pi(s_t)$. In this case, each state-action pair needs to be visited infinitely often. One example of a learning policy achieving this is an *ϵ-greedy policy*, where the agent takes a random action with a probability of ϵ, or the action with the highest Q-value in the current state otherwise. As Q-values converge to Q^*, it is common to decrease ϵ to prioritize exploitation over exploration.

3.2 Deep Q-Learning

In an MDP, each state-action pair has an associated Q-value that can be used to find an optimal policy. In practice however, the state space may be immensely large (including the continuous state space case), and learning these values becomes computationally infeasible. Therefore, approximate methods are used instead. One such approximation is known as *deep Q-network* (DQN) [22].

In DQN, a deep neural network is used to approximate the Q-value function. For each state, the neural network returns Q-values of the actions $Q_\theta(s, a) \approx Q^*(s, a)$, where θ is a vector of the weights of the network. The network starts with random weights θ and is trained based on the agent's experiences so that the outputs of the DQN $Q_\theta(s, a)$ converge to the optimal Q-values $Q^*(s, a)$.

One of the techniques used for effectively training DQNs is employing an experience replay buffer [19, 23]. The agent's experiences $e = (s_t, a_t, r_t, s_{t+1})$ are stored in a buffer $\mathbb{D}$. During learning, a batch of experiences $\mathbb{B} = \{e_1, \dots, e_{|\mathbb{B}|}\}$ is sampled uniformly from the experience replay buffer, $e_i \sim \mathcal{U}(\mathbb{D})$. For each experience in the batch $e = (s, a, r, s') \in \mathbb{B}$, target Q-values $U_{\hat{\theta}}$ are bootstrapped similarly to the tabular version (2):

$$U_{\hat{\theta}}(r, s') \triangleq r + \gamma \max_{a' \in \mathbb{A}} Q_{\hat{\theta}}(s', a'). \quad (3)$$

The mean square error $\mathcal{L}(\mathbb{B}|\theta)$ between the bootstrapped values and the network's outputs is used as a loss function in the DQN training. It is given by

$$\mathcal{L}(\mathbb{B}|\theta) \triangleq \frac{1}{|\mathbb{B}|} \cdot \sum_{(s,a,r,s') \in \mathbb{B}} \big(U_{\hat{\theta}}(r, s') - Q_\theta(s, a)\big)^2,$$

Note that bootstrapping is done using previously obtained weights $\hat{\theta}$, which are updated periodically rather than at each step to improve stability of learning.

4 INTERVAL Q-LEARNING

In this section we present a novel method called *Interval Q-Learning* (IQ-Learning), which guides exploration by combining dithering, optimistic exploration and action elimination techniques.

4.1 Tabular Variant

We begin with the tabular version of IQ-Learning. Intuitively, IQ-Learning uses intervals $\big[Q^-(s,a), Q^+(s,a)\big]$ to represent possible ranges of the optimal Q-values $Q^*(s,a)$. It combines ideas from Action-Elimination Q-Learning [9], Optimistic Q-Learning [10], and Upper Confidence Bound Q-Learning [14] (although in the latter two cases only the upper bound is used).

IQ-Learning uses two Q-value tables to store Q^+ and Q^-. The tables are initialized with the upper V^+ and lower V^- value bounds respectively:

$$Q_0^- \leftarrow V^- \triangleq \tfrac{r^-}{1-\gamma} \quad \text{and} \quad Q_0^+ \leftarrow V^+ \triangleq \tfrac{r^+}{1-\gamma}.$$

This initialization is chosen so that each interval $[Q_0^-(s,a), Q_0^+(s,a)]$ is the smallest interval to which the optimal Q-values belong, $Q^*(s,a) \in [Q_0^-(s,a), Q_0^+(s,a)]$, if *no other information is given.* Given each observation, each table is updated independently, using the regular Q-Learning update rule (1). If each state-action pair (s,a) is visited infinitely often, both $Q^-(s,a)$ and $Q^+(s,a)$ will eventually converge to $Q^*(s,a)$.

The use of two Q-tables allows us to guide exploration from actions that appear sub-optimal. To do so, we employ the following observation:

Observation 1. *If for all actions a it holds that*

$$Q^*(s,a) \in \big[Q^-(s,a), Q^+(s,a)\big], \tag{4}$$

and $Q^-(s,a'') > Q^+(s,a')$ for two actions a' and a'', then action a' is not optimal in state s.

This observation has been used before to identify sub-optimal actions to guide exploration. For example, in AEQ-Learning, the upper and lower Q-values Q^+ and Q^- are updated so that the following condition holds at each time step:

$$\mathbb{P}\big[Q^-(s,a) \le Q^*(s,a) \le Q^+(s,a)\ \forall(s,a)\big] \ge 1-\delta. \tag{5}$$

Therefore, permanent action elimination using Observation 1 eliminates the optimal action with probability at most δ. If δ is low enough, the resulting policy is near-optimal.

Maintaining a high-probability bound comes at a cost of slow convergence of Q^- and Q^+ to Q^*. In practice, this means that AEQ-Learning learns much slower than Q-Learning and catches up only after it starts eliminating sub-optimal actions, resulting in low values in the early stages. To improve this, exploration should be steered away from potentially bad actions as early as possible.

In IQ-Learning, this is achieved as follows. We use regular Q-Learning update rules on both interval ends separately, without requiring the probabilistic bounds (5) to hold at all time steps. As the updates are stochastic, this may result in optimal values not belonging to the intervals anymore. However, we know that both $Q^-(s,a)$ and $Q^+(s,a)$ will eventually converge to $Q^*(s,a)$ if all of the actions remain visited, as in the case of Q-Learning. Thus, we maintain convergence to optimality by assigning small probabilities to candidate sub-optimal state-action pairs instead of completely eliminating them. This is the key difference between IQ-Learning and action-elimination methods.

Formally, IQ-Learning is presented in Algorithm 1, and its exploration policy which we call (ϵ,ζ)-greedy in Algorithm 2.

Algorithm 1: Interval Q-Learning (tabular version)

```
Data: MDP, learning rate $\alpha$, exploration rate $\epsilon$, true
      exploration coefficient $\zeta$
Result: near-optimal Q-table $Q \approx Q^*$
1 Initialize Q-table $Q_0^+(s,a) \leftarrow V^+, Q_0^-(s,a) \leftarrow V^-$;
2 for episode $t \leftarrow 0,1,2,\dots,T-1$ do
3     observe current state $s_t$;
4     take action $a_t \leftarrow \pi_{(\epsilon,\zeta)\text{-Greedy}}(s_t, Q_t^-, Q_t^+)$;
5     observe reward $r_t$ and next state $s_{t+1}$;
6     compute updates $U_t^\pm \leftarrow r_t + \gamma \max_{a\in\mathbb{A}} Q_t^\pm(s_{t+1},a)$;
7     update Q-tables:
        $Q_{t+1}^\pm(s_t,a_t) \leftarrow (1-\alpha_t)Q_t^\pm(s_t,a_t) + \alpha_t U_t^\pm$;
8     update $\epsilon$ and $\zeta$;
9 return final Q-tables $Q_t^+$ and $Q_t^-$
```

Algorithm 2: (ϵ,ζ)-Greedy Policy

```
Data: state $s$, Q-tables $Q^+$ and $Q^-$
Result: action $a$
1 with probability $\epsilon$ do exploration
2     with probability $\zeta$ do true exploration
3         use all actions $\mathbb{C}(s) \leftarrow \mathbb{A}$;
4     else do action elimination
5         find candidate actions
            $\mathbb{C}(s) \leftarrow \{a \in \mathbb{A} \mid \forall a' \in \mathbb{A},\ Q^-(s,a') \le Q^+(s,a)\}$.
6     choose one of the candidate actions $a \in \mathbb{C}(s)$;
7 else do exploitation
8     find the best action $a \in \arg\max_{a'\in\mathbb{A}} Q^+(s,a')$;
9 return action a
```

We call this learning policy (ϵ,ζ)-*greedy*, where ϵ is the exploration rate and ζ governs selection of potentially sub-optimal actions. With probability ζ, true unconstrained exploration (step 3 of Algorithm 2) happens and any action can be chosen; otherwise actions that appear to be sub-optimal are eliminated first (step 5). Once the candidate actions have been identified, exploration happens (step 6). We use random exploration, as in the ϵ-greedy case, but alternative strategies are possible. For example, one can always explore the action with the largest interval length, $a \in \arg\max_{a'\in\mathbb{A}}\big(Q^+(s,a') - Q^-(s,a')\big)$.

In exploitation (step 8 of Algorithm 2), the (ϵ,ζ)-greedy policy has access to two Q-tables Q^+ and Q^-. To determine the best action, either one or a combination of both, such as the interval middle $\bar{Q}(s,a) = 0.5\big(Q^+(s,a)+Q^-(s,a)\big)$, can be used. Experiments suggest that the upper bound Q^+ works the best, conforming with the well-known principle of optimism in the face of uncertainty. This optimistic approach is especially beneficial in problems where deep exploration is required; the agent believes that undiscovered states give the largest possible reward and thus it keeps exploring new states instead of learning about the already visited ones.

Finally, the exploration rates are updated (step 8 of Algorithm 1). We use exponentially decaying exploration rate ϵ

$$\epsilon_t = \max\{\epsilon_0 \cdot t^{\eta_\epsilon}, \epsilon_{\mathrm{m}}\}, \tag{6}$$

where ϵ_0 is the initial exploration rate, η_ϵ is the exploration rate decay, and ϵ_m is the minimum exploration rate. To ensure that each action remains chosen infinitely often, we use a constant true exploration rate ζ. Other choices of ϵ and ζ may be suitable as well.

4.2 Deep Learning Variant

A deep learning variant of IQ-Learning is based on Deep Q-Learning described in Section 3.2. In Deep Q-Learning, a neural network (also known as a deep Q-network, DQN) takes a state s as an input and outputs Q-values of each action, $Q_\theta(s,a)$. In IQ-Learning, it must output two values for each action instead, $Q^+_\theta(s,a)$ and $Q^-_\theta(s,a)$ with a restriction $Q^-_\theta(s,a) \leq Q^+_\theta(s,a)$, which poses new challenges.

In the tabular setting, we initialize the Q-table with $Q^\pm_0 \leftarrow V^\pm$ so that the optimal Q-values Q^* satisfy the requirement (4) and the unvisited state-action pairs do not affect action selection. In the deep setting, the initial outputs of the DQN may already break this constraint due to random initialization. In order to maintain an upper and lower bound on Q-values of unvisited state-action pairs, we augment the network output with a special bonus v introduced in OPIQ [29]. This bonus term is referred to as *bonus for optimism* [24], and the *augmented Q-values* are computed as follows:

$$\bar{Q}^\pm_\theta(s,a) \triangleq Q^\pm_\theta(s,a) \pm v\big(\#(s,a)\big), \text{ where } v(t) \triangleq C \cdot (t+1)^{-M}, \quad (7)$$

where $C \geq V^\triangle \triangleq V^+ - V^-$ and $M \geq 1$ are constants, and $\#(s,a)$ is the visitation function. This choice of C guarantees that for unvisited states (i.e., when $\#(s,a) = 0$) the following holds even if the DQN outputs do not satisfy $Q^-_\theta(s,a) \leq Q^+_\theta(s,a)$:

$$\bar{Q}^-_\theta(s,a) = Q^-_\theta(s,a) - C/1^M \leq V^+ - V^\triangle = V^-, \qquad \text{and}$$
$$\bar{Q}^+_\theta(s,a) = Q^+_\theta(s,a) + C/1^M \geq V^- + V^\triangle = V^+.$$

As the states get visited and the DQN-based approximation improves, the bonus for optimism approaches zero with M governing the speed of decay.

In deep learning, and in general when a function approximator is involved, the visitation function $\#(s,a)$ is not available and thus is also approximated, usually by using pseudo-counts. Several techniques exist in the literature for calculating these pseudo-counts [3, 5, 6]; we use a simple count weighed according to the L_1-distance between the current state-action and all state-action pairs in the replay buffer [31] $\mathbb{D}$:

$$\#(s,a) = \sum_{(s',a',r,s'')\in\mathbb{D}} \max\big\{1 - \|s - s'\|, 0\big\}.$$

Another challenge is that, in the tabular setting our choice of initialization and the monotonicity of the updates guarantee that the Q-values satisfy $Q^-_t(s,a) \leq Q^+_t(s,a)$ at each step t; however, in the deep setting, as $Q^\pm_\theta(s,a)$ converge to optimal values $Q^*(s,a)$ and the bonus for optimism vanishes, it becomes harder to satisfy the constraint $Q^-_\theta(s,a) \leq Q^+_\theta(s,a)$. To overcome this problem, we parameterize the value intervals not with their ends $Q^-_\theta(s,a)$ and $Q^+_\theta(s,a)$, but with the middle $Q_\theta(s,a)$ and the scale $\Delta Q_\theta(s,a)$:

$$Q_\theta(s,a) = \frac{Q^+_\theta(s,a) + Q^-_\theta(s,a)}{2}, \quad \Delta Q_\theta(s,a) = \frac{Q^+_\theta(s,a) - Q^-_\theta(s,a)}{V^\triangle}. \quad (8)$$

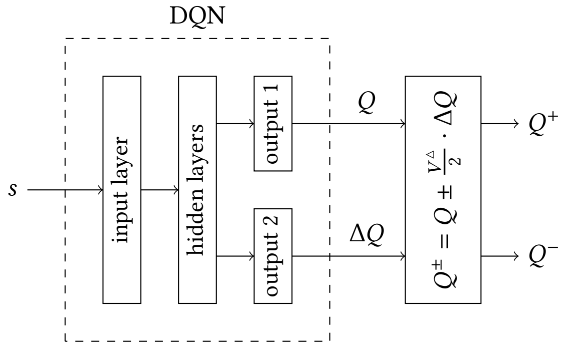

Figure 1: Non-linear approximator used in IQ-Learning.

The denominator in (8) is chosen so that the largest possible interval $[V^-, V^+]$ has the scale of 1, and the smallest one (when $Q^-_\theta(s,a) = Q^+_\theta(s,a)$) has the scale of 0.

Using this interval representation, we split the output layer of the DQN into two. The first one outputs $|\mathbb{A}|$ unconstrained values $Q_\theta(s,a)$ in the exact same way as the regular DQN does. The second one outputs $|\mathbb{A}|$ values between 0 and 1; these values are interpreted as $\Delta Q_\theta(s,a)$. We use a layer with the sigmoidal activation function to guarantee that the outputs are in the desired interval. Given the DQN output, the intervals are then computed as

$$Q^\pm_\theta(s,a) = Q_\theta(s,a) \pm \tfrac{V^\triangle}{2} \cdot \Delta Q_\theta(s,a). \quad (9)$$

The resulting approximator is visualized in Figure 1.

For network training, we first bootstrap Q-values

$$U^\pm_{\hat{\theta}}(r,s') \triangleq r + \gamma \max_{a'\in\mathbb{A}} \Big\{ Q^\pm_{\hat{\theta}}(s',a') \pm v_\mathrm{b}\big(\#(s',a')\big) \Big\} \quad (10)$$

similarly to regular DQN bootstrapping (3). We add a bonus for optimism $v_\mathrm{b}(t) \triangleq C_\mathrm{b} \cdot (t+1)^{-M}$ to maintain optimism in learning [29]. Then we compute target DQN outputs $U_{\hat{\theta}}(r,s')$ and $\Delta U_{\hat{\theta}}(r,s')$ similarly to equation (8). Finally, the training is done by performing a gradient descent with mean squared error loss function $\mathcal{L}(\mathbb{B}|\theta)$:

$$\mathcal{L}(\mathbb{B}|\theta) = \frac{1}{|\mathbb{B}|} \cdot \sum_{e\in\mathbb{B}} \big(\mathcal{L}_Q(e|\theta) + \mathcal{L}_{\Delta Q}(e|\theta)\big), \quad (11)$$

where the losses for outputs $Q_\theta(s,a)$ and $\Delta Q_\theta(s,a)$ are defined as

$$\mathcal{L}_Q(s,a,r,s'|\theta) \triangleq \big(U_{\hat{\theta}}(r,s') - Q_\theta(s,a)\big)^2 \qquad \text{and}$$
$$\mathcal{L}_{\Delta Q}(s,a,r,s'|\theta) \triangleq \big(\Delta U_{\hat{\theta}}(r,s') - \Delta Q_\theta(s,a)\big)^2.$$

Summarizing the aforementioned modifications, a deep learning variant of IQ-Learning is presented in Algorithm 3.

5 EMPIRICAL RESULTS

We conducted experiments in two different environments to investigate the performance of Interval Q-Learning over other action elimination and bounded value function methods. We chose the experiments to play to the strengths of different competing methods, and to test both tabular and deep versions of IQ-Learning. In particular, Q-Learning with our choice of the learning rate performs exceptionally well in the first experiment known as the automobile replacement problem, while UCB-learning lags behind. The second experiment is a slight modification of a well-known problem called

Algorithm 3: Interval Q-Learning (deep version)

```
Data: MDP, exploration rate $\epsilon$, true exploration coefficient $\zeta$
Result: DQN $Q_\theta$ such that $Q_\theta \approx Q^*$
1  Initialize weights $\theta$ randomly and $\hat{\theta} \leftarrow \theta$;
2  for episode $t \leftarrow 0, 1, 2, \dots, T-1$ do
3      observe current state $s_t$;
4      use the DQN to find $Q_\theta(s_t, a)$ and $\Delta Q_\theta(s_t, a)$ for all
         actions $a \in \mathbb{A}$;
5      find the augmented $Q$-values $\bar{Q}^{\pm}_{\theta}(s_t, a)$ using (9) and (7);
6      take action $a_t \leftarrow \pi_{(\epsilon,\,\zeta)\text{-Greedy}}(s_t, \bar{Q}^-_\theta, \bar{Q}^+_\theta)$;
7      observe reward $r_t$ and next state $s_{t+1}$;
8      store the experience $(s_t, a_t, r_t, s_{t+1})$ in the buffer $\mathbb{D}$;
9      sample a batch $\mathbb{B}$ from the buffer $\mathbb{D}$;
10     for experience $e = (s, a, r, s') \in \mathbb{B}$ do
11         bootstrap target $Q$-values $U^{\pm}_{\hat{\theta}}(r, s')$ using (10);
12         compute target DQN outputs $U_{\hat{\theta}} \leftarrow \frac{1}{2}(U^+_{\hat{\theta}} + U^-_{\hat{\theta}})$
             and $\Delta U_{\hat{\theta}} \leftarrow \frac{1}{V^{\Delta}}(U^+_{\hat{\theta}} - U^-_{\hat{\theta}})$;
13     perform a gradient descent with MSE-loss given by (11);
14     update $\epsilon$ and $\zeta$;
15     every $N$ steps, update $\hat{\theta} \leftarrow \theta$;
16 return final DQN
```

the randomized chain. It is a weak spot of Q-Learning, as it requires deep exploration instead, but is not hard for optimistic methods to solve. In these experiments we show that IQ-Learning is able to combine the strengths of both methods and performs well in both problems.

5.1 Tabular Q-Learning

In tabular RL, we test the performance of IQ-Learning on the automobile replacement problem. This problem was originally proposed by Howard [12] and is used as a benchmark of sample efficiency of learning methods [9, 24, 27].

In this problem, an agent operates an automobile. The state of the automobile degrades randomly over time and ranges from 1 (new) to 40 (broken). At each time step the agent observes the state of the automobile and must choose either to continue driving it, or to replace it with another one. In the latter case, the replacement automobile does not have to be a new one and can be in any of the 40 states. The full problem specification, including the rewards and transition probabilities can be found in the original paper [12]. With 40 states and 41 actions, this problem is well-connected but has too many possible policies, making exploration that goes wide more efficient than deep exploration.

We compare IQ-Learning with ϵ-greedy Q-Learning [37], Action Elimination Q-Learning (AEQ) [9], and infinite-horizon UCB-learning (∞-UCB) [36]. We conduct T trials, each lasting for K episodes of H time steps. All methods are initialized with $Q_0 = 0$. This initialization is optimistic, because almost all of the rewards in this problem are negative, thus the optimal values are as well. For IQ-Learning, we use $Q^+_0 = 0$ and $Q^-_0 = V^-$. We use an exploration rate given by (6) and a learning rate $\alpha_t = (h+1)/(h+t^{\omega})$

Problem size		Exploration		Learning		Optimism	
$\lvert\mathbb{S}\rvert$	40	ϵ_0	1	h	320	C^1	4
T	10	ϵ_{m}	0.01	ω	0.8	$c_2{}^2$	4
H	1000	η_ϵ	0.99	γ	0.95	$\varepsilon^{1,2}$	0.05
K	10000	ζ	0.05			$\delta^{1,2}$	0.05

1 Used in AEQ; see [9] for details.
2 Used in ∞-UCB; see [36] for details.

Table 1: Parameters of tabular learning methods.

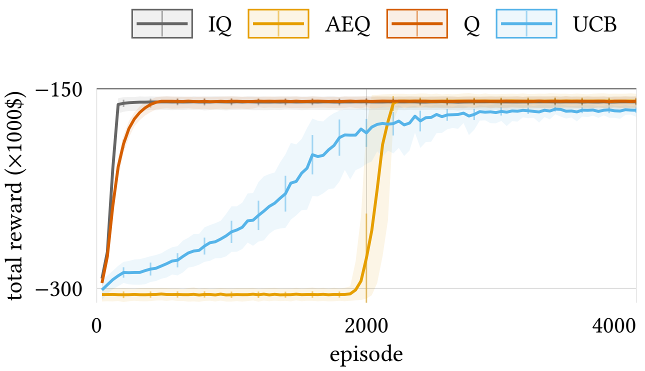

Figure 2: Automobile replacement solved by tabular learning methods, average of 10 trials. Lines, ribbons, and notches represent means, 95%-quantile ranges, and 95%-confidence intervals on means respectively.

suggested in [24] for episodic learning. Note that in finite-horizon models $h = H$, but each state-time combination is modeled separately. Instead, we use an infinite-horizon discounted model with states having the same values at different time steps; details on how to compute this extra parameter h in this case can be found in [36]. This and other hyper-parameters used by the algorithms are presented in Table 1.

The results of this experiment are presented in Figure 2. IQ-Learning is able to identify a near-optimal solution faster than its competitors. It then continues to explore around the found solution, slowly improving its policy. Q-Learning achieves comparable performance in this experiment, but does learn marginally slower than IQ. Action Elimination Q-Learning takes longer to exhaustively explore its options; once a good solution is found it starts exploiting it and converges quickly. ∞-UCB improves its performance gradually, but is much slower that Q-Learning.

5.2 Deep Q-Learning

We compare the performance of IQ-Learning and other deep learning algorithms on a modification of the chain problem used by several authors to test exploration strategies in deep RL [25, 29]. A similar modification can be found in [11].

The problem is presented in Figure 3. The environment consists of multiple states arranged in a chain, and each episode starts in the second state from the left. In each state the agent can go left

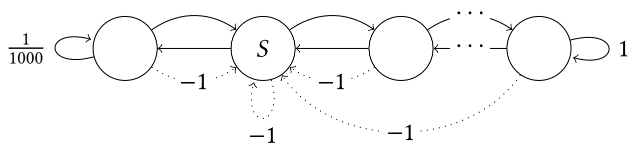

Figure 3: Modified chain. Unlabeled actions yield no reward.

Problem size		Exploration		Learning		Optimism	
$\lvert\mathbb{S}\rvert$	8	ϵ_0	1	$\lvert\mathbb{D}\rvert$	4096	C	2
T	50	ϵ_m	0.001	$\lvert\mathbb{B}\rvert$	32	C_b	0.01
H	16	η_ϵ	0.995	N	4	M	10
K	120	ζ	0.01	γ	0.95	β^1	1

[1] Intrinsic motivation, used in OPIQ; see [29] for details.

Table 2: Parameters of deep learning methods.

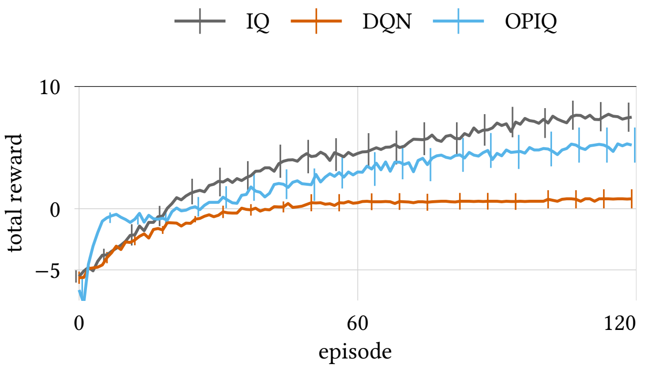

Figure 4: Chain solved by deep learning methods, average of 7 trials. Lines and notches represent means and 95%-confidence intervals on means respectively.

or right. An attempt to go left in the leftmost state does not affect the position, but yields a reward of 0.001; similarly, going right in the rightmost state gives a reward of 1. We add another action, depicted in dotted lines. This action resets the agent's position to the starting state and gives a reward of −1.

The actions are randomly permuted for each state. The agent cannot infer which action is the best for a particular state by considering the experiences with other states; instead, each state-action pair has to be experienced. As a result, finding an optimal strategy requires thorough exploration. At the same time, our addition of a negative-reward action makes exploration prohibitive. Therefore, an efficient algorithm for this problem must achieve a good exploration-exploitation balance and be capable of deep exploration, being persistent in trying to reach undiscovered states.

Instead of observing a state directly, the agent sees a vector of $|\mathbb{S}|$ binary features ϕ known as the thermometer encoding $\phi(s) = [\mathbb{I}\{x \leq s\}]$, where $x \in \mathbb{S}$. This state encoding is known to improve performance of deep learning methods [25, 29].

We compare IQ-Learning with ϵ-greedy DQN [23] and OPIQ [29], which are deep learning extensions of Q-Learning and UCB respectively. All methods use neural networks with two hidden layers of 32 neurons each and rectified linear unit activations. The output layers use the same activation function, except for the second output layer of IQ-Learning, which uses a sigmoidal activation function to output values ΔQ_θ between 0 and 1 as described in Section 4.2.

As in the tabular variant, we conduct T trials of K episodes and H time steps each. The number of steps H is chosen so that the maximum total reward is equal to 10. These and other parameters are given in Table 2. The results are presented in Figure 4.

Our experiments show that IQ-Learning outperforms both DQN and OPIQ, and its behavior is a combination of that seen in the other two methods. OPIQ can easily identify the reset actions and avoid them, quickly escaping the negative reward zone. DQN on the other hand, improves its performance slowly and steadily. IQ-Learning is able to do both. Due to their exploration strategies, IQ-Learning and OPIQ achieve substantially better performance than DQN.

6 CONCLUSION AND FUTURE WORK

In this paper, we proposed Interval Q-Learning, a novel reinforcement learning algorithm that achieves fast convergence to a near-optimal policy. It does so by combining the strengths of both random and optimistic exploration styles. In experiments, we showed that Interval Q-Learning is able to perform well in two different types of problems: those where wide, "dithering" exploration is required, such as the automobile replacement problem, and those where deep exploration is more beneficial, such as the randomized chain. These problems present very different challenges, and in both cases Interval Q-Learning performs on par with or better than the best alternative method which is less versatile in its exploration strategy. Interval Q-Learning finds a near-optimal solution fast using random exploration, then does a more focused search around it based on the principle of optimism in the face of uncertainty. While doing so, it actively reduces the disastrous actions it takes. Versatile approaches such as Interval Q-Learning are especially important in solving real-life problems where the consequences of bad actions are more substantial and random exploration should be limited.

Future work includes several directions. Applications on more complex problems will help to better understand strengths and weaknesses of Interval Q-Learning. Further, improvements can be done by investigating different versions of its sub-procedures and ablations. For example, another pseudo-counter function can be used, and learning can use a procedure other than experience replay. Finally, derivation of theoretical guarantees of worst-case and average-case performance of Interval Q-Learning remains an open question.

ACKNOWLEDGMENTS

This research received funding from the Netherlands Organization for Scientific Research (NWO).

REFERENCES

[1] Peter Auer. 2002. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research* 3 (2002), 397–422.

[2] Andrew G. Barto, Steven J. Bradtke, and Satinder P. Singh. 1995. Learning to act using real-time dynamic programming. *Artificial Intelligence* 72 (1995), 81–138.

[3] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. 2016. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.). Curran Associates, Inc., New York, USA, 1471–1479.

[4] Ronen I. Brafman and Moshe Tennenholtz. 2002. R-MAX – A general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research* 3 (2002), 213–231.

[5] Yuri Burda, Harrison A Edwards, Amos J. Storkey, and Oleg Klimov. 2018. Exploration by random network distillation. In *Proceedings of the International Conference on Learning Representations*, Vol. 1. OpenReview, Vancouver, Canada, 17.

[6] Leshem Choshen, Lior Fox, and Yonatan Loewenstein. 2018. DORA the explorer: Directed outreaching reinforcement action-selection. In *Proceedings of the International Conference on Learning Representations*, Vol. 1. OpenReview, Vancouver, Canada, 17.

[7] Khen Elimelech and Vadim Indelman. 2020. Fast action elimination for efficient decision making and belief space planning using bounded approximations. In *Robotics Research*, Nancy M. Amato, Greg Hager, Shawna Thomas, and Miguel Torres-Torriti (Eds.). Springer International Publishing, Cham, 843–858.

[8] Eyal Even-Dar, Shie Mannor, and Yishay Mansour. 2002. PAC bounds for multi-armed bandit and Markov decision processes. In *Computational Learning Theory*, Jyrki Kivinen and Robert H. Sloan (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 255–270.

[9] Eyal Even-Dar, Shie Mannor, and Yishay Mansour. 2006. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of Machine Learning Research* 7 (Dec. 2006), 1079–1105.

[10] Eyal Even-Dar and Yishay Mansour. 2002. Convergence of optimistic and incremental Q-Learning. In *Advances in Neural Information Processing Systems 14*, T. G. Dietterich, S. Becker, and Z. Ghahramani (Eds.). The MIT Press, Cambridge, MA, USA, 1499–1506.

[11] Zhaohan Daniel Guo. 2019. *Directed exploration for improved sample efficiency in reinforcement learning*. PhD Thesis. School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA.

[12] Ronald A. Howard. 1960. *Dynamic programming and Markov processes*. Technology Press of the Massachusetts Institute of Technology and Wiley, New York, NY, USA. 136 pages.

[13] Tommi Jaakkola, Michael I. Jordan, and Satinder P. Singh. 1994. Convergence of stochastic iterative dynamic programming algorithms. In *Advances in Neural Information Processing Systems 6*, J. D. Cowan, G. Tesauro, and J. Alspector (Eds.). Morgan-Kaufmann, Burlington, MA, USA, 703–710.

[14] Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I. Jordan. 2018. Is Q-Learning provably efficient? In *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.). Curran Associates, Inc., Red Hook, NY, USA, 4863–4873.

[15] Leslie Pack Kaelbling. 1993. *Learning in embedded systems*. MIT Press, Cambridge, MA, USA.

[16] Ugur Kuter and Jiaqiao Hu. 2007. Computing and using lower and upper bounds for action elimination in MDP planning. In *Abstraction, Reformulation, and Approximation*, Ian Miguel and Wheeler Ruml (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 243–257.

[17] T.L Lai and Herbert Robbins. 1985. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics* 6, 1 (1985), 4–22.

[18] Lihong Li, Michael L. Littman, and Thomas J. Walsh. 2008. Knows what it knows: A framework for self-aware learning. In *Proceedings of the 25th International Conference on Machine Learning*. Association for Computing Machinery, New York, NY, USA, 568–575.

[19] Long-Ji Lin. 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning* 8, 3-4 (1992), 293–321.

[20] James B. MacQueen. 1966. A modified dynamic programming method for Markovian decision problems. *J. Math. Anal. Appl.* 14 (1966), 38–43.

[21] Roger McFarlane. 2013. A survey of exploration strategies in reinforcement learning. (2013). https://www.cs.mcgill.ca/~cs526/roger.pdf

[22] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing Atari with deep reinforcement learning. (2013). arXiv:cs.LG/1312.5602

[23] 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. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.

[24] Grigory Neustroev and Mathijs M. de Weerdt. 2020. Generalized optimistic Q-Learning with provable efficiency. In *International Conference on Autonomous Agents and Multi-Agent Systems*. IFAAMAS, Auckland, New Zealand, Article 1674, 8 pages.

[25] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. 2016. Deep exploration via bootstrapped DQN. In *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.). Curran Associates, Inc., Red Hook, NY, USA, 4026–4034.

[26] Ian Osband, Benjamin Van Roy, Daniel J. Russo, and Zheng Wen. 2019. Deep exploration via randomized value functions. *Journal of Machine Learning Research* 20, 124 (2019), 1–62.

[27] Martin L. Puterman. 1994. *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, Inc., Hoboken, NJ, USA. 672 pages.

[28] Sudeep Raja Putta and Theja Tulabandhula. 2017. Efficient reinforcement learning via initial pure exploration. (2017). arXiv:cs.LG/1706.02237

[29] Tabish Rashid, Bei Peng, Wendelin Boehmer, and Shimon Whiteson. 2020. Optimistic exploration even with a pessimistic initialisation. In *International Conference on Learning Representations*. OpenReview, Addis Ababa, Ethiopia, Article 588, 28 pages. https://openreview.net/forum?id=r1xGP6VYwH

[30] Shahin Shahrampour, Mohammad Noshad, and Vahid Tarokh. 2017. On sequential elimination algorithms for best-arm identification in multi-armed bandits. *IEEE Transactions on Signal Processing* 65, 16 (2017), 4281–4292.

[31] Thiago D. Simão and Matthijs T. J. Spaan. 2019. Safe policy improvement with baseline bootstrapping in factored environments. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*. AAAI Press, Honolulu, USA, 4967–4974.

[32] Trey Smith and Reid Simmons. 2004. Heuristic search value iteration for POMDPs. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence (UAI '04)*. AUAI Press, Arlington, Virginia, USA, 520–527.

[33] Alexander L. Strehl and Michael L. Littman. 2004. An empirical evaluation of interval estimation for Markov decision processes. In *16th IEEE International Conference on Tools with Artificial Intelligence*. IEEE, Boca Raton, FL, USA, 128–135.

[34] Richard S Sutton. 1996. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems 8*, D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo (Eds.). MIT Press, Denver, USA, 1038–1044.

[35] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement learning: An introduction* (2nd ed.). The MIT Press, Cambridge, MA, USA. 552 pages.

[36] Yuanhao Wang, Kefan Dong, Xiaoyu Chen, and Liwei Wang. 2020. Q-Learning with UCB exploration is sample efficient for infinite-horizon MDP. In *International Conference on Learning Representations*. OpenReview, Addis Ababa, Ethiopia, Article 509, 18 pages. https://openreview.net/forum?id=BkglSTNFDB

[37] Christopher John Cornish Hellaby Watkins. 1989. *Learning from delayed rewards*. PhD Thesis. King's College, Cambridge, UK.