Asking for Help with a Cost in Reinforcement Learning

by

Colin Vandenhof

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2020

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Reinforcement learning (RL) is a powerful tool for developing intelligent agents, and the use of neural networks makes RL techniques more scalable to challenging real-world applications, from task-oriented dialogue systems to autonomous driving. However, one of the major bottlenecks to the adoption of RL is efficiency, as it often takes many time steps to learn an acceptable policy. To address this problem, we investigate the idea of allowing the agent to ask for action advice from a teacher. We formalize this concept in a framework called ask-for-help RL, which entails augmenting a Markov decision process with a teacher-query action that can be taken at a fixed cost in any state. In this task, the agent faces a dilemma between exploration, exploitation, and teacher-querying. To make this trade-off, we propose an action selection strategy that is rooted in the classical notion of value-of-information, and suggest a practical implementation that is based on deep Q-learning. This algorithm, called VOE/Q, can jointly decide between taking a particular environment action or querying the teacher, and is sensitive to the query cost. We perform experiments in two domains: a maze navigation task and the Atari game Freeway. When the teacher is excluded, the algorithm shows substantial gains over many other exploration strategies from the literature. With the teacher included, we again find that the algorithm outperforms baselines. By taking advantage of the teacher, higher cumulative reward can be achieved than with standard RL alone. Together, our results point to a promising approach to both RL and ask-for-help RL.

Acknowledgements

I would first like to thank my advisor, Edith Law, for her guidance and support throughout my Master's studies. Thank you for giving me the freedom to pursue the research questions that interested me, wherever they led.

I would also like to thank all of the students and faculty that make up the HCI lab at Waterloo. The lab is very special place, and you have all made my time at Waterloo a memorable and enriching experience.

Thank you to Kate Larson and Pascal Poupart for reading my thesis and providing helpful feedback. It is worth noting that this thesis grew out of a project that was done for Pascal's reinforcement learning course.

Finally, thank you to my family and friends for all of their support along the way.

Dedication

To my family and friends.

Table of Contents

List of Tables ix

List of Figures x

1 Introduction 1

2 Background 4

2.1 Reinforcement Learning 4

2.1.1 Q-learning 5

2.1.2 Deep Q-learning 6

2.1.3 Bayesian Q-learning 7

2.1.4 Exploration Strategies in Bayesian Q-Learning 8

2.2 Learning from Demonstration 11

2.3 Teacher Advice in Reinforcement Learning 12

2.3.1 Teacher Monitoring 13

2.3.2 Asking for Help 14

2.3.3 Other Approaches 15

3 Asking for Help with a Cost using Value-of-information 16

3.1 The Ask-for-help RL Framework 16

3.1.1 Ask-for-help RL 16

3.1.2 Example: Chain Problem . . . 18
3.1.3 Optimal Policies with the Teacher . . . 18
3.1.4 Balancing Exploration, Exploitation, and Asking for Help . . . 19
3.2 Ask-for-help RL with VOE/Q Action Selection . . . 20
3.2.1 VOE/Q . . . 21
3.2.2 Optimality of VOE/Q . . . 23
3.3 Deep Q-learning Implementation . . . 23
3.3.1 Uncertainty-aware Deep Q-learning . . . 23
3.3.2 Sample VOE/Q . . . 25
3.3.3 Algorithm Summary . . . 27

4 Experiments **30**
4.1 Domains . . . 30
4.2 Implementation Details . . . 33
4.3 Experiment: Exploration . . . 34
4.4 Experiment: Asking for Help . . . 36
4.4.1 Alternative Approaches . . . 36
4.4.2 Baselines . . . 37
4.5 Experiment: Adjusting Query Cost . . . 40
4.6 Experiment: Varying Advice Optimality . . . 42
4.7 Experiment: Adding the Supervised Learning Update . . . 43

5 Discussion **45**
5.1 VOE exploration for RL . . . 45
5.2 VOE/Q action selection for ask-for-help RL . . . 45
5.3 Other considerations . . . 47

6 Conclusion **49**
6.1 Future work . . . 50

References **52**

APPENDICES **62**

A Proofs **62**

B Cumulative Reward Tables **64**

List of Tables

4.1 Atari environment settings. 33

4.2 Hyperparameters for Freeway and maze domains. 34

B.1 Cumulative reward using various exploration strategies. 64

B.2 Cumulative reward using various action selection strategies. 64

B.3 Cumulative reward using VOE/Q action selection, varying the query cost. 65

B.4 Cumulative reward using VOE/Q action selection, varying the probability of optimal advice. 65

B.5 Cumulative reward using VOE/Q action selection, with and without the supervised update. 65

List of Figures

3.1 MDP for the "Chain" problem with a teacher. 17
3.2 Bootstrapped DQN architecture. 24

4.1 Frame from maze navigation task. 31
4.2 Frame from Atari game Freeway. 32
4.3 Learning curves using various exploration strategies. 35
4.4 Learning curves using various action selection schemes. 38
4.5 Heat map for one run of VOE/Q-querying and π-querying. 39
4.6 Learning curves using VOE/Q action selection, varying the query cost. . . 41
4.7 Query frequency over time using VOE/Q action selection, varying the query cost. 41
4.8 Learning curves using VOE/Q action selection, varying the probability of optimal advice. 42
4.9 Learning curves using VOE/Q, with and without the supervised learning update. 44

Chapter 1

Introduction

Reinforcement learning (RL) is increasingly being proposed for training real-world agents, from task-oriented dialogue systems to autonomous vehicles [44, 60]. However, it is often hindered by poor sample efficiency. Even state-of-the-art deep RL algorithms may require many training steps to learn an acceptable policy, or worse, may converge to unacceptably bad policies. For agents trained in simulation, poor sample efficiency can make the use of RL time-consuming, expensive, and ineffective. For agents trained in real-world environments, it can result in harmful real-world mistakes.

A fundamental limitation of RL is that the optimal policy must be inferred through self-directed interaction with the environment. This ignores the fact that in many tasks, it may be possible to leverage existing expertise from artificial agents or humans. For example, a human expert might possess a good policy for playing the video game Tetris. How can an RL agent take advantage of this expertise? For a human, a natural way of relaying knowledge is though advice: when the agent needs help in a particular game state, the human can provide a recommended move. Still, coming up with this advice may carry a cost – the human has the burden of assessing the game state, deciding on a recommended move, and relaying it to the agent. Even if the advice-giver is another artificial agent, there may still be a computational cost to generating advice and a communication cost to relaying it. Accordingly, an agent should have the capacity to weigh the cost of advice with its potential benefit, and request help only when it is most useful.

We formalize this intuition with a framework called ask-for-help RL. The ask-for-help RL framework allows for a natural integration of apprenticeship learning with RL by treating the request for advice as an additional action that can be taken in each state at a known cost. This approach has many of the same benefits of RL, allowing agents to learn in a

self-directed manner, but with the potential to vastly speed up learning through teacher advice. The problem of ask-for-help RL is to find the best trade-off between pure RL and pure apprenticeship learning, given the cost of teacher queries.

Besides introducing the framework, we devise an action selection strategy to decide whether to query the teacher or perform a particular environment action in the current state. To this end, we revisit the classical notion of *value-of-information* from decision theory [34, 16]. Roughly speaking, value-of-information attempts to quantify the expected improvement in decision quality that immediately arises after performing a particular action. In addition to estimating the value-of-information for each environment action, we show how it can be estimated for the query action. By weighing the value-of-information with the expected costs of performing each action, the agent can effectively balance exploration, exploitation, and teacher-querying.

We then apply this action selection strategy to deep Q-learning. In particular, we build off of the Bootstrapped DQN algorithm that was proposed by Osband et al. [50]. Bootstrapped DQN uses a Q-network with multiple heads to produce uncertainty estimates over the Q-function.

We experimentally test our approach on several challenging tasks. We use a maze navigation problem to help illustrate the general behavior of the algorithm and show how it can scale to playing the Atari game Freeway.

There are three main contributions of this thesis which can be summarized as follows:

- We introduce the ask-for-help RL framework, in which a Markov decision process (MDP) is augmented by adding a teacher-query action to each state. Taking this action incurs a fixed cost that is accounted like a negative reward.
- By ignoring the teacher-query action, we introduce a novel exploration strategy for standard RL called VOE. On the RL tasks tested, we find that VOE surpasses other exploration strategies from the literature.
- By including the teacher-query action, we introduce an action selection strategy for ask-for-help RL called VOE/Q. This strategy is sensitive to query cost, and when querying is free, it matches the optimal policy to query in every state. We again compare VOE/Q to several baselines experimentally, and find substantial performance gains.

The rest of the thesis is organized into the following chapters. Chapter 2 provides some background information and reviews the related work. Chapter 3 introduces the ask-for-help RL framework. A novel exploration strategy for standard RL is then proposed,

and we show how it can naturally extend to ask-for-help RL. We then outline a practical implementation of the algorithm based on deep Q-learning. Chapter 4 details a set of experiments that are performed to test the algorithm. We discuss the experimental results in Chapter 5 and offer some concluding remarks and future directions in Chapter 6.

Chapter 2

Background

In this chapter, we review some of the preliminaries of RL, with a focus on Q-learning. We discuss the use of Q-learning with neural networks, as well as Bayesian Q-learning, and exploration methods that are compatible with Bayesian Q-learning. Next, we discuss the topic of learning from a teacher. Finally, we review work that combines RL and teacher advice.

2.1 Reinforcement Learning

The standard RL task entails sequentially making decisions in an environment in order to maximize some scalar reward signal [73]. At time step t, the agent is provided its current state, S_t, and must select an action, A_t. Then, the agent receives a reward R_{t+1} from the environment, and transitions to the next state S_{t+1}.

RL is usually formalized as a discrete MDP which is the 5-tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$. $\mathcal{S}$ is a finite set of states, and $\mathcal{A}$ is a finite set of actions. The transition function is $p(s, a, s') = P[S_{t+1} = s' | S_t = s, A_t = a]$, which gives the probability of transitioning from state s' to state s by taking action a, and the reward function is $r(s, a) = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$, which gives the expected reward after taking action a in state s. $\gamma \in [0, 1]$ is a discount factor which is used to discount rewards that are obtained in the future. In RL, the transition and reward functions are initially unknown, and the agent must learn about them through interaction with the environment.

The agent chooses actions according to a policy $\pi(a|s) = P[A_t = a | S_t = s]$ which defines a probability distribution over actions, given the current state. From the state

at time t, we define the discounted return as $G_t^\pi = \sum_{i=0}^{\infty} \gamma_t^{(i)} R_{t+i+1}$, which is the sum of discounted future rewards if actions are selected according to the policy π. Here, γ_t refers the discount factor at time t. In this thesis, we consider the episodic case where $\gamma_t = \gamma$ unless the episode terminates at time t, in which case $\gamma_t = 0$. $\gamma_t^{(i)}$ is the discount applied to a reward i steps into the future, $\gamma_t^{(i)} = \gamma_t \cdot ... \cdot \gamma_{t+i}$.

The goal of RL is to find an optimal policy π^*, which is a policy that maximizes the expected discounted return. Various approaches exist to finding such a policy, but this thesis will focus on Q-learning, which is explained below.

2.1.1 Q-learning

Some RL methods entail learning an action-value function (i.e., the Q-function), which gives the expected discounted return after taking action a in state s and thereafter following the policy π, $q^\pi(s,a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$. Considering an optimal policy π^*, the corresponding Q-function can be written in a special recursive form known as the Bellman equation. This equation states that the value of taking action a in state s, and thereafter following the optimal policy, is equal to the reward we are expected to immediately receive, plus the discounted maximum action value in the state we are expected to visit next:

$$q^*(s,a) = r(s,a) + \sum_{s'} p(s,a,s')\gamma \max_{a'} q^*(s',a')$$

q^* is called the optimal action-value function. With q^*, an optimal policy can be recovered by taking the maximizing action in the every state, $\pi^*(s) = \text{argmax}_a\, q^*(s,a)$. The Bellman equation described above is the basis of the Q-learning update rule proposed by Watkins [84]:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)]$$

Here, α is a learning rate parameter. Q is a tabular approximation of the optimal action-value function, with an Q-value entry for every state-action pair. This update can be understood as incrementing the current Q-value estimate towards a target Q-value. The Q-learning algorithm is simply to apply this update for each new transition $(s_t, a_t, r_{t+1}, s_{t+1})$ that is encountered by the agent in the environment. As long as all state-action pairs are visited infinitely often, and the learning rate is properly reduced over time, Q will converge to q^*.

There are several advantages of Q-learning over other RL methods. First, it is *off-policy*, meaning that the policy being learned does not need to match the policy used to select actions in the environment. It will always learn q^*, regardless of the policy being followed. Additionally, it is *model-free*, which means that there is no need to learn a model of the underlying MDP, i.e., the reward and transition functions. By learning q^*, an optimal policy can be directly obtained.

2.1.2 Deep Q-learning

In recent years, deep neural networks have found extensive use within the field of machine learning. Advances in both the structure and optimization of neural networks has made the training process faster and more effective. As a result, there have been efforts to develop RL algorithms that use neural networks to approximate policies, dynamics models, and value functions. One of the first successful combinations of RL with neural networks was Deep Q-Network (DQN), which was able to successfully learn to play a suite of Atari games [47]. DQN approximates q^* with a convolutional neural network $Q(\cdot;\theta)$, and trains it via stochastic gradient descent with a loss function based on the Q-learning update rule. The loss function is the following:

$$L(\theta) = \sum_{\tau \in B} (r_{t+1} + \gamma \max_a Q(s_{t+1}, a; \theta^-) - Q(s_t, a_t; \theta))^2$$

This loss function computes the squared error between a target value for $Q(s_t, a_t)$ and its current estimate value, and it sums over a minibatch B of transitions. There are two important tricks to improve the stability of DQN. The first is to compute the loss over a minibatch of transitions sampled from a large *replay buffer* containing the last N transitions, rather than only the most recent transition like in the original Q-learning algorithm. This serves to reduce correlations between successive updates to the network that can destabilize learning. The second trick is to compute the target value using a so-called *target network* $Q(\cdot;\theta^-)$. The target network is a copy of the main network whose parameters are periodically copied from the main network, but otherwise kept frozen. This mitigates the divergence issue that arises when the same network is used to compute the target and estimate values.

Since the original work, various extensions to DQN have been proposed [79, 64, 7, 83, 3]. Many of these extensions have been effectively combined to achieve state-of-the-art performance in Atari game-playing [30].

2.1.3 Bayesian Q-learning

Many RL algorithms rely on point estimates of quantities like Q-values. These point estimates may be incorrect, and therefore the actions that have the highest estimated value may in fact be suboptimal. Because of this discrepancy, RL agents must balance exploitation with exploration – the selection of actions currently estimated to have the highest value must be balanced with the selection of other actions that have a chance of being better.

There is also a branch of algorithms that seek to explicitly represent uncertainty about estimated quantities like Q-values. Most of these algorithms fall under the umbrella of Bayesian RL [27]. In the Bayesian approach, probability distributions are maintained over parameters of the model rather than point estimates. The posterior distribution is meant to represent the subjective uncertainty about the true values of the parameters, given prior beliefs and the observed data. Policies can then expressed over the *information state* – the physical state combined with a parameterized posterior. The major benefit of this framework is that it provides a natural way of tackling the exploration-exploitation dilemma. The optimal policy over information states (i.e., the Bayes-optimal policy) would select actions based not only on immediate rewards, but also on how much information it provides about the environment. In theory, an MDP could be constructed over information states that could be solved exactly to yield a Bayes-optimal policy [19]. In practice, it is computationally intractable.

As an alternative, Q-learning algorithms have been proposed that maintain an approximate posterior over Q-functions given the current data, $p(Q^*|H_t)$, and then select actions in an online manner according to the current posterior. Early work from Dearden et al. accomplishes this by some simplifying assumption, including that the posterior for each Q-value has a Gaussian distribution and the prior has a Normal-Gamma distribution [16]. Two procedures are proposed for updating the posterior. Engel et al. instead define a Gaussian process over the Q-function [20].

More recently, there have been attempts to use neural networks for uncertainty-aware Q-learning. Methods for obtaining uncertainty estimates from neural networks has been a long-standing area of study [40, 45]. BBQ-Networks applies the Bayes-by-Backprop method to a DQN in order to generate samples from an approximate posterior [44, 8]. Other work has investigated applying the dropout technique to a DQN as a way of obtaining approximate posterior samples [23]. There has also been work that applies Bayesian linear regression to the last layer of a DQN [6]. Finally, some work has been done on training ensembles of DQNs, and then treating each ensemble member like a sample from the posterior [53, 50]. One of these ensemble methods, called Bootstrapped DQN, is the basis

of the method that we describe in Chapter 3.

Given a posterior over Q-functions, there are various methods by which actions might be selected so as to balance exploration with exploitation. Several of these exploration strategies will be discussed in the next section.

2.1.4 Exploration Strategies in Bayesian Q-Learning

Any Q-learning algorithm requires a method of selecting actions at each step. With standard Q-learning, one of the most popular exploration methods is ϵ-greedy, which means that the action that with current highest estimated value is performed, except with probability ϵ, an action is chosen uniformly at random. The value of ϵ is typically reduced over time. This means that the initial selection of actions is very random, exploring many possible actions since their values are still highly uncertain. Over time, as Q converges to the optimal action-value function, ϵ is reduced so that actions are increasingly selected according to the optimal policy.

Although ϵ-greedy exploration is sometimes effective for small tasks, it is generally very sample inefficient. Because exploration is random, this strategy will repeatedly select actions that are suboptimal with high certainty, rather than directing exploration at the most promising actions. Furthermore, the performance is highly dependent on how ϵ is adjusted over time. If ϵ is reduced too quickly, Q will be slow to converge to the optimal Q-function as there will be too few exploratory actions taken. On the other hand, if ϵ if reduced too slowly, a lot of unnecessary exploration will occur.

Maintaining a posterior distribution over Q^* allows for *directed* exploration methods that can take uncertainty into account. Rather than exploring randomly, actions are chosen based on the possibility that they are optimal. We will discuss three such strategies below: Bayes upper-confidence bound, Thompson sampling, and value-of-information. We will discuss value-of-information in the most detail, as it is the basis of the method that we introduce in Chapter 3.

Bayes Upper-confidence Bound

Upper-confidence bound (UCB) is an exploration method that operates by the principle of "optimism in the face of uncertainty". It originates from the multi-armed bandit setting, and works by setting an upper bound on each arm, so that the expected value of the arm is lower than the bound with high probability, and then selecting the arm with the highest

upper bound at each time step. There exist both frequentist [42, 5] and Bayesian [37] versions of this algorithm with known regret bounds in the multi-armed bandit setting.

Bayes-UCB can be applied to the full RL problem by using Bayesian Q-Learning. At each time step, we find the mean and the standard deviation of the posterior distribution for each action, $\mu_a = \mathbb{E}[q^*(s_t, a))], \sigma_a = SD[q^*(s_t, a)]$, and then select the action with the highest Q-value upper bound:

$$a_t = \underset{a}{\operatorname{argmax}}\, \mu_a + \lambda \sigma_a$$

where $\lambda \in \mathbb{R}_+$ is a hyperparameter that controls how high to set the upper bound. This approach was successfully implemented for deep Q-learning by Chen et al. [10].

Thompson Sampling

Thompson sampling is another action selection algorithm that originates from the multi-armed bandit setting [77, 59]. It operates by the principle of probability matching – selecting actions according to the probability that they are optimal. This is achieved by sampling from the posterior at each step, and then picking the action that has maximal value according to the sample. Regret bounds have also been derived for this multi-armed bandit algorithm [1].

Like Bayes-UCB, Thompson sampling can also be applied to the full RL problem using Bayesian Q-Learning. At each time step, this entails drawing a sample from the posterior distribution $p(Q^*|H_t)$, and then taking the greedy action according to the sampled Q-function:

$$\text{Sample } q^* \sim p(Q^*|H_t)$$

$$a_t = \underset{a}{\operatorname{argmax}}\, q^*(s_t, a)$$

This algorithm was first proposed by Wyatt [86] and a practical deep Q-learning implementation was given by Osband et al. [50]. A model-based version of this algorithm, which samples from a posterior over MDPs, has also been proposed [15, 70] and further analyzed by Osband et al. [51]. In [70, 51, 50], they modify the algorithm by using the same posterior sample over many time steps, rather than re-sampling every time step. They argue that this allows the exploration strategy to remain consistent over a period of time, promoting "deep" exploration.

Value-of-information

A heuristic based on the classical notion of value-of-information was introduced by Dearden et al. [16]. The idea is to compare the immediate expected information gain of performing some action with the immediate expected cost. The information gain is how much the return stands to improve by the discovery that a particular action is optimal. The cost is the potential reduction in return that is incurred by performing a suboptimal action.

It works by considering what is to be gained by knowing the true value of action a in state s. If knowing $q^*(s,a)$ does not change the agent's policy, it is not useful information. In light of this, there are two cases when knowing $q^*(s,a)$ is indeed useful:

Case A: If action a is considered suboptimal, but $q^*(s,a)$ indicates that it is optimal.

Case B: If action a is considered optimal, but $q^*(s,a)$ indicates that it is suboptimal.

These two cases yield the following piecewise formula for value-of-information:

$$VOI(q^*_{s,a}) = \begin{cases} q^*_{s,a} - \mathbb{E}[q^*(s,a^{1st})] & a \neq a^{1st},\ q^*_{s,a} > \mathbb{E}[q^*(s,a^{1st})] \\ \mathbb{E}[q^*(s,a^{2nd})] - q^*_{s,a} & a = a^{1st},\ \mathbb{E}[q^*(s,a^{2nd})] > q^*_{s,a} \\ 0 & \text{otherwise} \end{cases}$$

where a^{1st} and a^{2nd} are the actions with highest and second highest expected Q-value respectively. The top line is case A, where we compute how much higher the true Q-value of action a is compared to the expected Q-value of the action that was previously considered optimal (a^{1st}). The second line is case B, where we compute how much higher the expected Q-value of the second highest action is compared to the true Q-value of the action we previously considered optimal (a^{1st}).

Since the true value $q^*(s,a)$ is not known to the agent, an *expected* value-of-information over the posterior must be computed instead. The expected value-of-information for action a in state s is:

$$EVOI(s,a) = \mathbb{E}[VOI(q^*_{s,a})]$$

To decide what action to take, an action's expected value-of-information must be weighed against its expected cost, which is difference between the action's expected value and the highest expected value over all actions (EV^*). We refer to this is as the expected cost of exploration (ECOE).

$$ECOE(s,a) = EV^* - \mathbb{E}[q^*(s,a)]$$

At each time step, we then select whatever action has the largest expected value-of-information minus expected cost:

$$a_t = \underset{a}{\text{argmax}}[EVOI(s_t, a) - ECOE(s_t, a)]$$

Note that the term EV^* can be dropped from this calculation since it is present for all actions. Like the other exploration strategies, this approach is heuristic. The value-of-information is an upper bound approximation to the true utility of taking action a because in reality, taking an action only provides sample information about its value, rather than perfect information. Also, it is myopic in the sense that it only considers the immediate expected improvement in return. Nevertheless, Dearden et al. find that value-of-information is an effective heuristic, substantially outperforming Thompson sampling in their experiments [16]. One intuitive explanation for this finding is that while Thompson sampling only considers the probability that an action is optimal, value-of-information considers the *amount* by which an action stands to improve the return.

2.2 Learning from Demonstration

Learning from demonstration, also referred to as apprenticeship learning, imitation learning, and behavior cloning, entails learning a policy in a supervised manner [4]. Labelled data is provided, with each example consisting of a state and a recommended action. The goal is to learn a policy that fits the provided data and also generalizes well. For discrete action spaces, learning can be accomplished using any classification algorithm, such as decision trees [61]. For continuous action spaces, any regression technique could be used, such as locally-weighted regression [35]. One key consideration is whether the algorithm employs eager or lazy learning - that is, whether a policy is learned from the data prior to execution (e.g. neural network [55]) or whether a policy must be inferred from the demonstrations during execution (e.g. k-nearest neighbors [62]). The latter requires that the agent store all of training data, which may be impractical in some applications, especially if there is a large amount of demonstrations.

If the demonstration data does not cover all possible states, than the agent policy must generalize to unseen states. In stochastic environments, the agent may enter regions of the state space that are not covered well by any demonstrations, which could lead to catastrophic failure. This highlights the need to not only collect training data prior to execution, but also interactively query the teacher during execution whenever the current

training set is insufficient. In this regard, one very popular algorithm is DAGGER [58]. DAGGER is an algorithm that seeds the training data with expert demonstrations, then iteratively (1) trains a novice policy on the current training set, (2) rolls out a hybrid policy that queries the expert some fraction of the time, and otherwise uses the novice policy, and (3) adds the rollout data to the training set. The authors show that this algorithm is no-regret in the online learning sense. Some variations have since been proposed, such as SafeDagger, which only uses the novice policy if the novice's action choice is similar to the expert [88]. Closer to our work is work by Chernova and Veloso, who suggest that the novice request action advice from the expert in states with high uncertainty [11]. Whenever a state is sufficiently far from a previously visited state in Euclidean distance, or whenever the action given by the notice policy has sufficiently low confidence, action advice is requested from the teacher. The downside to this approach is that it requires that all previously visited states to be kept in memory, which limits its scalability, and it is also unclear how to set the distance and confidence thresholds.

A clear limitation of imitation learning is that the agent policy will end up replicating the teacher policy. For most practical cases, we can assume the teacher possesses a good but suboptimal policy, so the resulting agent policy will also not be optimal. To address this problem, a common technique is to learn an initial policy via demonstration, and then to fine-tune this policy via RL [46]. This idea has been effective in playing video games [24], building dialogue systems [85], and mastering Go [67]. In more recent work, the line between imitation learning and RL has become more blurred. For example, various works have proposed including both demonstration data and experience data in a shared replay buffer, and training simultaneously from both [80, 30].

While combining RL with imitation learning allows for the agent policy to improve over the teacher, there are still some remaining issues. First, it is unclear how much demonstration data should be provided, which is problematic since there are usually costs associated with collecting the data. Second, in large-scale problems where the amount of demonstration data cannot possibly cover the entire state space, it is also not obvious what states should be included in the demonstrations. In this thesis, we address these problems by querying the teacher in an online manner, but only in states when the advice is expected to be useful.

2.3 Teacher Advice in Reinforcement Learning

Various forms of teacher advice have been proposed to improve safety and efficiency in RL [56]. Most approaches fit into one of two categories: a teacher can monitor the agent and

preemptively decide when to provide advice to the agent, or the agent can ask for advice from a teacher themselves. The latter approach is the focus of this thesis. The agent then learns from both their interaction with the environment and the advice provided by the teacher. Advice can include an action or a set of acceptable actions for the agent to take next [13, 76, 9], a sequence of suggested actions [18, 69], or an external reward for the agent's behavior [39]. We review the many approaches to integrating teacher advice below.

2.3.1 Teacher Monitoring

Some work has studied the situation in which a teacher monitors the agent and chooses when to provide advice. The teacher may provide an external reward signal [57, 72] or a recommended action [12, 78]. In [78], the authors propose several heuristics for deciding when to provide an action recommendation. *Importance advising* assumes that the teacher has access to the optimal action-value function, and computes the difference between the maximum and minimum Q-values for actions in each state, giving advice whenever the difference is above some threshold. The justification is that advice is most useful when there is large difference in value between best and worst actions. They also propose *predictive advising*, which learns a model of the agent's policy and gives advice whenever they predict the agent will make a mistake. Both of these heuristics can run into problems. Importance sampling assumes access to the optimal Q-function, which would be unlikely for a human teacher. Predictive advising may fail since (1) the agent policy may rapidly change during the learning process and (2) the policy may be highly stochastic. Walsh et al. instead analyze the return of the agent for each episode, and opt to provide a demonstration of the full episode whenever the return is too low [82]. This is very simple and straightforward, but lacks specific targeting of the states where advice is most useful, and can only provide advice after the mistakes have been made. Ideally, advice should prevent mistakes from happening in the first place.

A significant drawback to any teacher-monitoring approach is that they would be potentially very costly and time-consuming to implement in practice. In complex RL tasks, it might not be feasible for a teacher to actively monitor the entire learning process. Another consideration is whether the teacher has enough information to give advice when it is most useful. For example, the teacher may not have access to the the agent's policy, and therefore, it may not be able to accurately predict what (potentially incorrect) actions the agent is going to take next.

2.3.2 Asking for Help

Some previous papers have also investigated the approach in which the agent requests help from the teacher. Similar to the heuristic described in the previous section, Clouse proposes computing the difference between minimum and maximum value estimates for actions in the current state, and asking for help whenever the difference is below some threshold [13]. A smaller difference indicates that the Q-values have a smaller spread, and therefore the optimal action is less certain. This is a somewhat flawed notion of uncertainty, as in some cases there may be little difference in the true Q-values of the best and worst actions. It is also unclear how the threshold should be set *a priori*. García and Fernández instead distinguish between known and unknown states [25, 26]. They build and update the policy using a case-base of previously-seen states. If a new state is encountered, it is considered unknown if it sufficiently far (in Euclidean distance) from any state in the case-base. The agent seeks help from the teacher in unknown states. This approach is restricted to small and discrete state spaces that can adequately covered by such a case-base. Closer to our work is a recent paper by Silva et al., who propose an ask-for-help algorithm for uncertainty-aware deep Q-learning [14]. The decision to ask for help is based on the uncertainty about Q-values, and the teacher is queried whenever the variance of the value estimates is above some predefined threshold. Again, this raises the question of how such a threshold should be set *a priori*. Our work is distinguished from all previous work by the notion that a query to the teacher may carry a cost, and that its cost should be weighed against the information that the teacher provides.

Amir et al. look at combining the teacher-directed and agent-directed approaches: the agent decides whether or not to ask for the teacher's attention in a particular state, and the teacher in turn decides whether or not to provide action advice in that state [2]. This bidirectional decision-making process seems very natural. However, the ask-for-help and provide-help decisions are made using the same heuristics described above, so the same issues apply.

In many of the works discussed, we note that the teacher is treated as separate from the environment. As a result, it is not captured by the RL framework, and hence it becomes difficult to define optimal behavior for an RL agent that incorporates teacher advice. In our work, we incorporate the teacher into the RL framework, treating queries like actions and accounting for their costs in the reward function. Consequently, it is possible to define a Bayes-optimal policy for balancing teacher queries with exploration.

2.3.3 Other Approaches

There is a great deal of research that incorporates teacher knowledge into RL but does not fit into the paradigm of asking for or giving advice. Instead, the teacher knowledge is used to bias the reinforcement learning process. One way that this can be achieved is with reward shaping – the practice of modifying the reward function without changing the optimal policy, with the hope that the modified rewards can better guide the agent's behavior [49]. Suay et al. propose using inverse reinforcement learning to recover a linear function from expert demonstrations, and then use this function to shape rewards [71].

Alternatively, advice can be used as a kind of initialization procedure. Some authors have proposed using the teacher policy to initialize value functions [68], or to perform initial action selection [22, 21]. This teacher policy might be provided to the agent or learned from demonstrations [75]. Similarly, Schaal et al. use demonstration data to initialize dynamics models for model-based RL [63]. In the Bayesian framework, demonstrations can be used to help generate priors. Doshi-Velez et al. incorporate expert demonstrations into a prior over models for model-based RL [17]. If evolutionary techniques are used for RL, then the initial population could be seeded with teacher policies [66]. This approach was successful in learning to control helicopters [28].

There is also the idea of using the teacher to modify the starting state of the agent, with the intent to promote exploration and bring agents closer to goals. Along these lines, some research has looked at using checkpoints sampled from teacher demonstrations as starting states for episodes [32, 48]. While effective in simulated environments, it is often difficult or impossible to change the starting state of agents in real-world environments.

A common theme among many previous approaches to teacher advice in RL is that there is either a fixed set of demonstrations provided from the teacher, or, at the other extreme, we have unconstrained access to the teacher. Both of these options seem unrealistic when considering agents that operate in the real world, particularly with human teachers. Clearly, an agent should be able to request help from the teacher when they are unsure how to act, much like an employee might request help from their supervisor. However, there are clearly costs to obtaining this advice, such as the time and effort to generate the advice and the communication cost to relay it. As a result, we are interested in the problem of weighing the possible costs and benefits of asking for help, alongside the possible costs and benefits of trying other actions in the environment.

Chapter 3

Asking for Help with a Cost using Value-of-information

In this chapter, we introduce the ask-for-help RL framework, which casts teacher-querying as an additional action that can be taken in each state at a fixed cost. To address the ask-for-help RL problem, we introduce an action selection strategy that is based on the value-of-information principle called VOE/Q. Finally, we walk through a concrete implementation of the algorithm using deep Q-learning.

3.1 The Ask-for-help RL Framework

The ask-for-help RL framework formalizes the notion of asking for help from a teacher with a cost in RL. This can be done by pulling the teacher into the MDP and adding teacher-querying as an additional action in each state that, if taken, incurs a fixed query cost. In this section, we introduce this framework in more detail, and provide an example using the "Chain" problem described by Strens [70]. We then describe how the optimal policies change with the addition of the teacher-query action. Finally, we discuss the problem of balancing exploration, exploitation and teacher-querying.

3.1.1 Ask-for-help RL

The ask-for-help RL task can be described as an RL task that results by modifying some base MDP (M) to form an augmented MDP (M'). M' has identical states to M, all

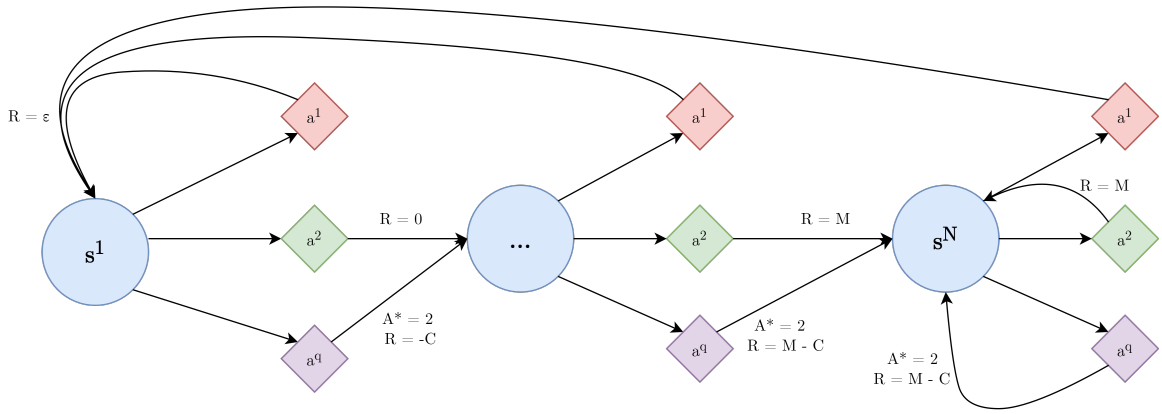


Figure 3.1: MDP for the "Chain" problem with a teacher. The teacher-query action (a^q) results in the same transition as the optimal action (a^2) and also the same reward, minus the query cost C. Taking the query action also returns the index of the optimal action (2).

of the actions of M, and the same reward and transition probabilities associated with states and actions of M. The difference is that we add a teacher-query action a^q to M': $\mathcal{A}_{M'} = \mathcal{A}_M \cup \{a^q\}$. This action can be executed in all states. Taking the teacher-query action incurs a fixed cost $C \in \mathbb{R}_{\geq 0}$, but otherwise behaves like an optimal action in the current state has been performed in M, i.e., $a^*_{t,M} = \text{argmax}_{a \in \mathcal{A}_M} q^*_M(s_t, a)$, where q^*_M is an optimal action-value function of M. In other words, taking the teacher-query action results in the same transition probabilities and rewards as taking $a^*_{t,M}$, except that a query cost C is subtracted from the reward. This is meant to reflect the idea of paying a cost to let the teacher execute an optimal action in the current state.

The MDP with a teacher (M') differs from a standard MDP in one important way. When the query action is taken in s_t, the index of optimal action $a^*_{t,M}$ is also returned. This is important since it allows the agent to learn in a supervised manner from the teacher.

Like in standard RL, the transition and reward functions are initially unknown in ask-for-help RL, and the agent must learn through interaction with the environment. However, the query cost C is known, as is the fact that taking the query action behaves like taking an optimal action (aside from cost C). That is, it is known that $r(s_t, a^q) = r(s_t, a^*_{t,M}) - C$ and that $p(s_t, a^q, s') = p(s_t, a^*_{t,M}, s')\ \forall s' \in \mathcal{S}_{M'}$, even though the actual index of $a^*_{t,M}$ is initially unknown, as are the rewards and transition probabilities.

The goal of the ask-for-help RL agent is still to find a policy that maximizes the expected discounted return, but this return now includes the cost of any teacher queries.

3.1.2 Example: Chain Problem

Figure 3.1 shows an example of an MDP that has been augmented with the teacher-query action. It is similar to the "Chain" problem given by Strens [70]. The starting state is s^1. We will distinguish the non-query actions as "environment actions". In each state, there are two environment actions available, a^1 and a^2. Action a^1 causes a slippage back to s^1 with a small reward of ϵ. Action a^2 causes a transition to the next state in the chain with a reward of 0, until the last state in the chain is reached, at which point there is a large reward of $X >> \epsilon$. Without the teacher-query action, the optimal policy is to choose a^2 in every state so that the large reward can eventually be reached.

If we now consider the teacher-query action, we see that its transition and reward mirrors that of the optimal action, a^2, in every state. The only difference is that the query cost of C is subtracted from the reward when taking the query action.

From this example, we can see that if querying the teacher has a non-zero cost ($C > 0$), the optimal policy does not change with the addition of a teacher. Action a^2 should still be taken in every state. On the other hand, if querying is free ($C = 0$), action a^2 and the query action a^q are interchangeable, and both are optimal. These notions are formalized in the next section.

3.1.3 Optimal Policies with the Teacher

When a teacher is added to an existing MDP, it is useful to know how the optimal policies change. There are two cases to consider: when the query cost is non-zero and when the query cost is zero. A non-zero query cost is considered first.

Theorem 1. *The optimal policies of an MDP (M) are the same as the optimal policies of the MDP with a teacher (M') if query cost $C > 0$.*

Proof. Let $\pi^*_{M'}$ be an optimal policy of M'. Suppose that $\pi^*_{M'}(a^q|s) > 0$ for some state s. Then, the expected return could be increased by $C \cdot \pi^*_{M'}(a^q|s)$ if all of the probability of taking the teacher-query action in state s is shifted to an optimal action in M, i.e., $\text{argmax}_{a \in \mathcal{A}_M} q^*_M(s, a)$. This forms a contradiction. Hence, an optimal policy of the MDP with a teacher assigns zero probability to taking the teacher-query action in any state. For all non-query actions, the transition probabilities and rewards are the same in every state for M and M'. Therefore, $\pi^*_{M'}$ must match an optimal policy of M. □

Next, we consider how optimal policies change with the addition of a teacher with a query cost of zero.

Theorem 2. *If $C = 0$, all optimal policies of an MDP (M) are also optimal policies of the MDP with a teacher (M'). Additionally, the teacher-query action is an optimal action in every state of M': $q^*_{M'}(s, a^q) = \max_{a \in \mathcal{A}_{M'}} q^*_{M'}(s, a)\ \forall s \in \mathcal{S}_{M'}$.*

Proof. For all non-query actions, the transition probabilities and rewards are the same in every state of M and M'. Since $C = 0$, the query action in every state $s \in S_{M'}$ has the same reward and transition probabilities as an optimal action in M, and hence must also be optimal. □

These two facts set some guidelines on how an ask-for-help RL algorithm should behave. Namely, any policy that is optimal without the teacher remains optimal with the addition of a teacher. Additionally, if the query cost is non-zero, querying is not an optimal action, and therefore an algorithm that converges to the optimal policy will eventually stop querying. This means that the total number of queries will be finite. On the other hand, if queries are free, querying is always an optimal action, and therefore an optimal policy can be followed by simply querying at every time step.

If any policy that is optimal without the teacher remains optimal with the teacher, what is the purpose of adding a teacher? A teacher provides an opportunity for more efficient learning, allowing the agent to converge to the optimal policy faster by avoiding costly exploration. Still, the potential benefit of asking for help must be weighed against its cost.

3.1.4 Balancing Exploration, Exploitation, and Asking for Help

In the standard RL problem, an agent must carefully balance exploration with exploitation. Exploration entails information-gathering actions that improve the agent's knowledge of the environment dynamics. In constrast, exploitation entails the greedy choice of whatever action maximizes expected returns according to the agent's current knowledge. Without exploration, an agent that might settle on a suboptimal action because there is not enough information about the optimal action to know that it is better.

In the ask-for-help RL problem, the agent is now tasked with balancing exploration, exploitation and teacher-querying. At each step, the agent must choose whether to take an exploratory action in the environment, exploit their current knowledge, or to ask the teacher for help. By returning an optimal action, the teacher provides more useful information towards learning an optimal policy than could be obtained by exploration. However, it may come at a high query cost C. To decide whether to query the teacher or explore the

environment, the agent must weigh the information that can be provided by the teacher with its cost.

As previously mentioned, there exists an optimal solution to the trade-off between exploration and exploitation under the Bayesian framework [19]. A prior can be expressed over the environment dynamics (i.e., the transition and reward functions), and the state can be augmented by a parameterized posterior to form an information state. An MDP can then be defined over these information states (M_I) that can be solved.

The Bayesian framework would also give an optimal solution to the trade-off between exploration, exploitation and asking for help. The only difference is in how the posterior is updated for the teacher-query action. Suppose the teacher-query action at step t results in a transition $(s_t, a^q, r_{t+1}, s_{t+1})$ and returns the index i of an optimal action. Not only can we use the transition $(s_t, a^i, r_{t+t} + C, s_{t+1})$ as sample information about action a^i to update the posterior, but we also update the posterior to reflect that action a^i is optimal (i.e., all MDPs where a^i is not optimal in state s_t are given probability 0). Like in the standard Bayesian RL problem defined above, the resulting MDP over information states with a teacher (M'_I) could also be explicitly solved. Furthermore, since any policy over M_I could also be executed in M'_I and achieve the same expected return, the Bayes-optimal policy with the teacher must be at least as good as the Bayes-optimal policy without the teacher.

Unfortunately, calculating such a policy is prohibitively expensive for large problems. Therefore, approximate solutions must be used. In the next section, we will propose an action selection method that is compatible with Bayesian Q-learning which naturally extends to ask-for-help RL.

3.2 Ask-for-help RL with VOE/Q Action Selection

In this section, we present a strategy for selecting actions in ask-for-help RL. It is compatible with Bayesian Q-learning, in which a posterior is maintained over the Q-function given the current history, $P(Q^*|H_t)$. Note that in the description of this action selection strategy, the posterior over Q^* relates to the original MDP, rather than the augmented MDP with the teacher-query action added.

3.2.1 VOE/Q

We propose a strategy for ask-for-help RL based on the value-of-information heuristic proposed by Dearden et al. [16]. This strategy was described in detail in Section 2.1.4.

Since the teacher provides the optimal action, it is appropriate to consider the expected value of perfect information (VPI), a term from decision theory [34]. Loosely speaking, the expected VPI measures the price that one would be willing to pay for perfect information about the value of one or more random variables. In this case, consider the value of each of the actions in the current state s as a random vector: $Q^*(s) = [Q^*(s, a^1), \ldots, Q^*(s, a^N)]$. With perfect information about $Q^*(s)$, the maximum expected return is clearly achieved by taking whatever action has maximal Q-value, i.e., $\max_a Q^*(s, a)$. Importantly, the teacher provides information about this optimal action. In the absence of such information, the highest return is expected by taking the action with highest *expected* Q-value, i.e., $\max_a \mathbb{E}[q^*(s, a)]$. The VPI is the difference between these two quantities. We are interested in the expected VPI, which in this context we refer to as the expected value-of-querying (EVOQ):

$$\begin{aligned} EVOQ(s) &= \mathbb{E}[VPI(s)] \\ &= \mathbb{E}[\max_a q^*(s, a) - \max_a \mathbb{E}[q^*(s, a)]] \\ &= \mathbb{E}[\max_a q^*(s, a)] - \max_a \mathbb{E}[q^*(s, a)] \end{aligned}$$

This is an approximation of the value of a teacher query. This must be weighed against the cost of querying the teacher. The expected cost of querying is the difference between the highest expected value over all actions (EV^*) and the expected value of querying the teacher:

$$ECOQ = EV^* - (\mathbb{E}[\max_a q^*(s, a)] - C)$$

Notice that the expected value of querying is the expected value with the perfect information assumption, minus the query cost C. The overall utility of teacher-querying is the difference between the expected value and expected cost of querying:

$$u(s, a^q) = EVOQ(s) - ECOQ(s)$$

To compare the utility of querying with the utility of performing an environment action, we consider a simplification to value-of-information heuristic described in Section

2.1.4. The simplification is to only consider the value-of-information for exploratory actions (i.e., case A in the description). We call this variant the value-of-exploration (VOE) to distinguish it:

$$VOE(q^*_{s,a}) = \begin{cases} q^*_{s,a} - \mathbb{E}[q^*(s, a^{1st})] & a \neq a^{1st},\ q^*_{s,a} > \mathbb{E}[q^*(s, a^{1st})] \\ 0 & \text{otherwise} \end{cases}$$

$$EVOE(s, a) = \mathbb{E}[VOE(q^*_{s,a})]$$

The expected value of exploration (EVOE) for some environment action a must be weighed against the expected cost of exploration (ECOE) which is the difference between the highest expected value over all actions (EV^*) and the expected value of action a:

$$ECOE(s, a) = EV^* - \mathbb{E}[q^*(s, a)]$$

This leads to an algorithm for jointly deciding whether to query the teacher or perform a particular environment action:

$$u(s, a) = \begin{cases} EVOQ(s) - ECOQ(s) & a = a_q \\ EVOE(s, a) - ECOE(s, a) & \text{otherwise} \end{cases}$$

$$a_t = \operatorname*{argmax}_a u(s_t, a)$$

We call this action selection strategy VOE/Q. Again, the term EV^* is present for all actions (it is found in both ECOQ and ECOE), so it can be dropped from the calculation. Dropping this term means that the utility for environment actions can be re-written as $\mathbb{E}[q^*(s, a)] + EVOE(s, a)$, which is the expected Q-value of the action, plus a value-of-information bonus that is 0 for the greedy action. This bonus attempts to capture the information that can be gained via exploration. As the agent becomes more confident in the estimated Q-values, this bonus approaches 0 for all actions. The original heuristic by Dearden et al. is slightly different since it also applies the value-of-information bonus to the greedy action. Such a bonus would cause the greedy action to be chosen more often and potentially lead to inadequate exploration. In light of this, we hypothesize that removing this bonus will lead to improved learning. However, another reason for removing this bonus is to achieve optimal behavior when querying is free, which we will discuss next.

3.2.2 Optimality of VOE/Q

The behavior of VOE/Q can be split into two cases: when querying is free ($C = 0$) and when querying carries a cost ($C > 0$). If querying is free, VOE/Q will immediately follow an optimal policy of querying in every state.

Theorem 3. *If $C = 0$, VOE/Q will immediately follow an optimal policy of querying in every state.*

Proof: See Appendix A.

If querying carries a cost, then assuming that the posterior converges to the optimal action-value function, VOE/Q will follow an optimal policy.

Theorem 4. *If $C > 0$ and the Q^* posterior converges to the optimal action-value function, VOE/Q will follow an optimal policy.*

Proof: See Appendix A.

These two theorems correspond to the Theorems 1 and Theorems 2 of Section 3.1.3, which describe optimal policies in ask-for-help RL.

3.3 Deep Q-learning Implementation

In this section, we propose a practical implementation of VOE/Q with deep Q-learning. The basis of our approach is Bootstrapped DQN, which maintains an ensemble of Q-functions that is intended to approximate samples from the Q^* posterior [50]. The ensemble is updated with a reinforcement loss, and optionally, a supervised loss on the advice given by the teacher. Both the experience data and the teacher advice are stored together in a shared replay buffer.

3.3.1 Uncertainty-aware Deep Q-learning

Our algorithm is based on Bootstrapped DQN by Osband et al., which maintains an Q-function ensemble parameterized by deep neural networks [50]. The ensemble shares the initial layers of the network, but then splits into k separate "heads" that each define a distinct Q-function, $\{Q_k\}$ (see Figure 3.2). Diversity is maintained among members

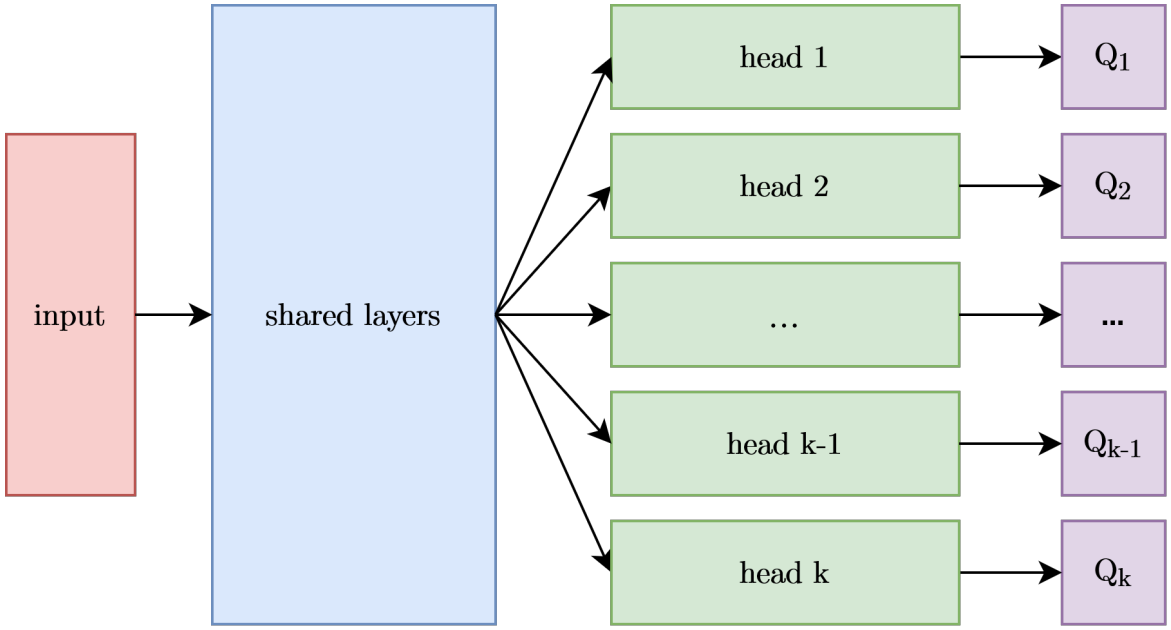


Figure 3.2: Bootstrapped DQN architecture. The initial hidden layers are shared, but split into k heads that each produce separate Q-value estimates.

of the ensemble via (1) random initialization and (2) bootstrapping. The parameters of the network are initialized randomly so that each head initially outputs different Q-value estimates [29]. Bootstrapping entails training each Q-function on different subsets of the data. The entire ensemble is trained simultaneously with a combined loss function, however the loss for any transition τ is only applied to a random subset of the Q-functions according a bootstrap mask $m_\tau \in \{0,1\}^k$. The value of $m_{\tau,i}$ decides whether or not a particular function Q_i should train on the transition τ. If $m_{\tau,i} = 1$, the loss is computed as usual, but if $m_{\tau,i} = 0$, the loss is zeroed-out. This is a form of bootstrapping, such that each ensemble member is only trained on a random subset of the data. Each mask value is set as 0 or 1 with equal probability, which corresponds to "double-or-nothing" bootstrapping [52]. The Bootstrapped DQN loss function is:

$$L(\theta) = \sum_{\tau \in B} \sum_{i=1}^{k} m_{\tau,i}(r_{t+1} + \gamma Q_i(s_{t+1}, \operatorname*{argmax}_a Q_i(s_{t+1}, a; \theta); \theta^-) - Q_i(s_t, a_t; \theta))^2$$

where B is a batch of transitions from the replay buffer, $\tau = (s_t, a_t, r_{t+1}, s_{t+1})$ is a transition, θ parameterizes the Q-ensemble and θ^- parameterizes the target network, which

is used to compute target values. Just like the original DQN algorithm, the target network parameters are updated to match the Q-ensemble parameters periodically and otherwise frozen between successive updates to improve learning stability. The transition τ is only applied to Q_i if it is not masked ($m_{\tau,i} = 1$). This loss also incorporates the trick proposed by Hasselt et al. to mitigate overestimation bias [79].

Uncertainty about the true Q-value is estimated by such a ensemble. Although such an ensemble approach to estimating uncertainty would not generally be considered Bayesian, each initial Q-function can be considered as a sample from the prior. Previous work has derived an update rule that would allow such an ensemble of Q-functions to approximate samples drawn from the posterior [10]. While the exact update is intractable, it can be approximated by the standard Q-learning update.

Randomly initializing the parameters of each network and bootstrapping is used to maintain a diverse set of models that are all consistent with the observed data and statistically plausible. As training progresses, the variance of the models decreases, which reflects a concentration of the posterior.

Experimentally, Bootstrapped DQN seems to perform well with methods for uncertainty-aware exploration. In the original paper, the authors pair Bootstrapped DQN with a form of Thompson sampling for action selection [50]. Each episode, one of the Q-functions is selected uniformly at random. All actions are then picked greedily according to the Q-function for the entire episode. More recent work paired Bootstrapped DQN with UCB action selection and showed further performance improvements [10]. Following this work, we choose to pair VOE/Q action selection with Bootstrapped DQN, although in principle, it could be used with other approaches for uncertainty-aware Q-learning.

3.3.2 Sample VOE/Q

It is straightforward to combine VOE/Q action selection with Bootstrapped DQN. Each member of the ensemble is a Q-function that produces value estimates for the environment actions. Since we treat each Q-function as a sample from the posterior, we can replace all expectations over the posterior with sample approximations. For the environment actions, we compute $\widehat{EVOE}$ and $\widehat{ECOE}$:

$$\widehat{VOE}(q^*_{s,a}) = \begin{cases} q^*_{s,a} - \max_{a'} \frac{1}{k} \sum_{i=1}^{k} Q_i(s,a') & a \neq a^{1st},\ q^*_{s,a} > \max_{a'} \frac{1}{k} \sum_{i=1}^{k} Q_i(s,a') \\ 0 & \text{otherwise} \end{cases}$$

$$\widehat{EVOE}(s,a) = \frac{1}{k} \sum_{i=1}^{k} \widehat{VOE}(Q_i(s,a))$$

$$\widehat{ECOE}(s,a) = EV^* - \frac{1}{k} \sum_{i=1}^{k} Q_i(s,a)$$

For the teacher-query action, we compute $\widehat{EVOQ}$ and $\widehat{ECOQ}$:

$$\widehat{EVOQ}(s) = \frac{1}{k} \sum_{i=1}^{k} \max_a Q_i(s,a) - \max_a \frac{1}{k} \sum_{i=1}^{k} Q_i(s,a)$$

$$\widehat{ECOQ}(s) = EV^* - (\frac{1}{k} \sum_{i=1}^{k} \max_a Q_i(s,a) - C)$$

Like previously, we construct a single function that returns the utility for both the environment actions and the teacher-query action, and then simply select the action with highest utility at each time step:

$$\hat{u}(s,a) = \begin{cases} \widehat{EVOQ}(s) - \widehat{ECOQ}(s) & a = a^q \\ \widehat{EVOE}(s,a) - \widehat{ECOE}(s,a) & \text{otherwise} \end{cases}$$

$$a_t = \operatorname*{argmax}_a \widehat{u}(s_t, a)$$

Once again, EV^* is treated as a placeholder that is present for all actions and can be dropped from the final calculation. This is an efficient algorithm that only requires taking two averages (over VOE and Q-values) for each action. Without the additional teacher-query action, we find it is slightly faster ($\sim 5\%$) than sample UCB action selection. Computing the utility of the teacher-query action requires taking only one additional average (over max Q-values), which does not add significant overhead.

Note that Theorems 3 and 4 still apply to this sample version of VOE/Q. Namely, if queries are free, then the algorithm will immediately follow the optimal policy of querying in every state. If queries carry a cost and all ensemble members converge to the optimal Q-function, the algorithm will follow the corresponding optimal policy.

3.3.3 Algorithm Summary

Algorithm 1 VOE/Q deep Q-learning.

Input: Q-ensemble $\{Q_k\}$, target network, environment $\mathcal{E}$, teacher $\mathcal{T}$ with query cost C, replay buffer B
while $\{Q_k\}$ not converged **do**
  $a_t \leftarrow \text{argmax}_a\, \hat{u}(s_t, a)$
  if $a_t = a^q$ **then**
    Assign a_t as the action given by $\mathcal{T}$ at cost C
  end if
  Execute action a_t, observe reward r_{t+1} and next state s_{t+1} from $\mathcal{E}$
  Update replay buffer B with transition $(s_t, a_t, r_{t+1}, s_{t+1})$
  Sample minibatch of transitions from B and update $\{Q_k\}$ with L_{RL}
  Sample minibatch of teacher-advised transitions from B and update $\{Q_k\}$ with L_{SL}
  Periodically copy parameters of $\{Q_k\}$ to target network
end while

Algorithm 1 gives a short description of the VOE/Q deep Q-learning algorithm. A Q-ensemble is initialized that consists of a neural network with k heads. Each head outputs value estimates for the environment actions. At step t, we use the output of Q-ensemble on the current state s_t to pick an action a_t according to sample VOE/Q (Section 3.3.2).

If the selected action is to query the teacher ($a_t = a^q$), we ask for help from the teacher at cost C, and assign a_t to the teacher-recommended action. We then take action a_t, and receive a reward and the next state from the environment. In this description, the teacher is treated as separate from the environment, but the effect is the same as if the teacher were incorporated into the MDP as described in Section 3.1.

We then add the current transition $(s_t, a_t, r_{t+1}, s_{t+1})$ to the replay buffer, along with the bootstrap mask m.

Each time step, we may sample a batch of transitions from the replay buffer and update the ensemble according to a reinforcement learning loss L_{RL}. We may also opt to sample

a batch of transitions that correspond to teacher-advised actions and update the ensemble according to a supervised learning loss L_{SL}. These two updates are described in more detail below.

Finally, we also periodically copy the network parameters to the target network, which is needed to ensure the stability of the reinforcement learning update.

Reinforcement Learning Update

We sample a batch of transitions from the replay buffer and train the network with a reinforcement learning loss L_{RL} that is similar to the loss described for Bootstrapped DQN (Section 3.3.1), except that we also implement prioritized experience replay [64].

Prioritized experience replay allows more important transitions to be sampled more frequently, which serves to dramatically speed up training compared to uniform sampling [30]. The probability of sampling a particular transition i is proportional to its priority : $P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$. The priority of transition i is given by $p_i = |d_i| + \epsilon$, where d_i is the most recent temporal difference error calculated for that sample, and ϵ is a small constant that ensures all transitions have a non-zero chance of being sampled. By prioritizing samples with high temporal difference error, we more frequently train on samples from which there is more to learn.

To offset the sampling bias, the loss for each sampled transition is multiplied by an importance sampling weight, $w_i = (\frac{1}{N \cdot P(i)})^\beta$, where N is the size of the replay buffer and β is a parameter that determines the degree of importance sampling, from $\beta = 0$ (none) to $\beta = 1$ (full). Unbiased updates are most critical near convergence, so β is linearly increased from some starting value to 1 over the course of training.

Supervised Learning Update

Supervised learning can also be performed on the Q-ensemble by applying the large margin classification loss [31, 54]:

$$L_{SL}(\theta) = \sum_{\tau \in B_{SL}} m_{\tau,i} \sum_{i=1}^{k} \max_a (Q_i(s_t, a) + \mathbb{1}_{\{a \neq a_t^T\}} \cdot \epsilon) - Q_i(s_t, a_t^T)$$

where B_{SL} is a batch of teacher-advised transitions, and a transition consists of a state s_t and the action recommended by the teacher, a_t^T. $m_{\tau,i} \in \{0, 1\}$ is the bootstrap mask

which controls whether the loss is counted for Q_i. This loss function enforces that the Q-value of a_t^T to be at least a margin (ϵ) higher than the Q-values of all other actions. When the Q-value of a_t^T is highest by at least this margin, the loss becomes zero. This means that, assuming the teacher-recommended action is always optimal, and the margin is sufficiently small, this loss has no affect once Q_i has converged to the optimal Q-function.

We should note that in practice, we may only have access to a suboptimal teacher. If the teacher sometimes returns suboptimal actions, the supervised learning update will force these suboptimal actions to have the highest Q-value, which may impede learning. In this case, the supervised update should be omitted and only the reinforcement update should be applied. Without the supervised update, it is possible for the agent to follow the advice of the teacher, but also learn to take actions that are better than what the teacher would recommend in a particular state.

How can the supervised data be efficiently stored and sampled? Instead of separately storing states along with actions that have been recommended by the teacher, we can simply maintain a list of references to the transitions in the replay buffer that correspond to actions recommended by the teacher. Whenever a transition is added to the replay buffer, we flag whether the action has been advised by the teacher or not. For any such "teacher-advised transition", we append its reference to the list. When the replay buffer reaches maximum capacity, then the transition being added might be replacing the oldest teacher-advised transition at index 0. If this is the case, then we simply pop that entry from the list. To sample a minibatch of transitions for supervised learning, all that is needed is to uniformly sample from the list.

Chapter 4

Experiments

We perform a variety of experiments to test the method. The first experiment investigates how well VOE action selection performs exploration, in the absence of a teacher. Then we add a teacher, and study how VOE/Q action selection compares to other teacher-querying baselines. Next, we observe how the behavior of the algorithm adapts to query cost. We then vary the optimality of the teacher advice to observe how performance is affected. All of these experiments are performed with only the reinforcement learning update, omitting the supervised learning update. At the end, we look at the effect of adding the supervised update.

4.1 Domains

We test our algorithm in two domains. The first is a maze navigation task, and the second is the video game Freeway for Atari.

Maze navigation: This is an episodic task taking place on a 15x15 grid-world as shown in Figure 4.1. The agent starts at one corner of the maze, and must navigate to the goal state in the opposite corner to receive a reward of +1 and terminate the episode. Scattered around the maze are dangerous holes, which if hit, result in a reward of -1 and cause the episode to terminate. All other states result in 0 reward.

The agent has four actions, which correspond to the four cardinal directions. Whenever an action will make the agent hit a wall, the agent slides to a random adjacent square that is empty. Otherwise, an action will move the agent one cell in that direction. The only

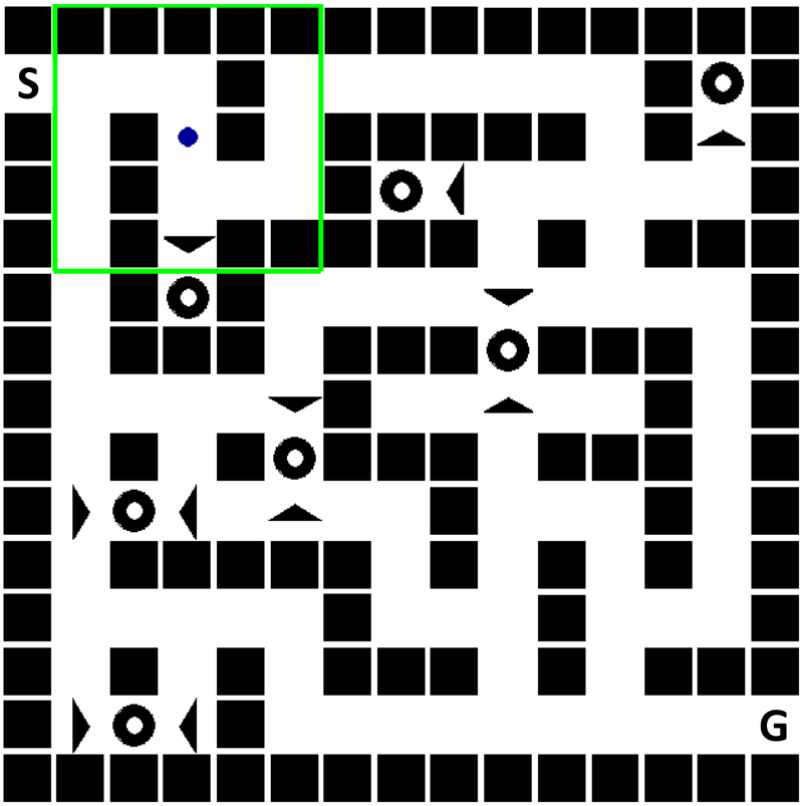


Figure 4.1: Frame from maze navigation task. The agent (blue circle) begins at the start cell (S) and must navigate to the goal cell (G). Holes (black circles) are scattered throughout the maze. The agent has a 50% chance of falling in if they are adjacent to a hole (triangles). The agent's state is represented by the 5x5 surrounding area of the maze (green box). The optimal path through the maze can be traced along the top and right edges.

exception is if the agent is beside a hole. In this case, they have a 50% probability of "falling in". In other words, they will move in the intended direction with probability 0.5, and move to the hole with probability 0.5.

The agent always occupies one cell of the grid, and its state is represented by the nxn surrounding box. This box was chosen to be width $n = 5$ so that holes could be within view of the agent, without the agent being in danger of falling in. Therefore, the state is of size fxnxn, where f is the number of maze features, of which there are 5 (empty, wall, agent, goal, hole). All features are represented with a 1-hot encoding and the state is flattened into a single vector. For additional randomness, Gaussian noise ($\mu = 0, \sigma = 0.3$)

Figure 4.2: Frame from Atari game Freeway. The agent (yellow) must cross the road while avoiding vehicles. Getting to the other side of the road results in a reward of +1, and causes the agent to reset to its start position. Hitting a vehicle causes the episode to terminate.

is added to this state representation.

Freeway: This is a classic Atari game in which the agent must cross multiple lanes of traffic while avoid getting hit by vehicles, as shown in Figure 4.2. The agent is given a reward of +1 for reaching the other side of the road, at which point the agent resets to its start position, and they must cross again. The game is set to the highest difficulty, which means that whenever the agent touches a vehicle, a life is lost and the agent is reset to its start position.

The agent can choose to move up, down or remain stationary. Minimal preprocessing of the raw pixel input is performed. The environment settings are summarized in Table

frame skip	{2,3,4}
sticky action probability	0.25
noops max	30
screen size	84
terminal on life loss	True
greyscale observations	True
normalized observations	True
frames per observation	4

Table 4.1: Atari environment settings.

4.1 which were chosen to align with prior work [79].

4.2 Implementation Details

There are many hyperparameters that need to be chosen for the algorithm, which precludes the possibility of an exhaustive hyperparameter search. Therefore, we only performed a search on hyperparameters for which performance is especially sensitive: target update frequency and learning rate. For Freeway, we matched other hyperparameters with prior work [50, 30, 79] and kept them the same for the maze problem with a few exceptions: we reduced the max number of steps, removed the steps of delay before updates begin, and increased the frequency of updates from every 4 steps to every step. A summary of the hyperparameters chosen is given in Table 4.2.

The neural network architecture was slightly different for the two domains. For Freeway, the neural network is identical to the one proposed by Osband et al. [50]. The input to the network is an 84x84x4 tensor composed of the rescaled, greyscale version of the last 4 frames. The first convolutional layer has 32 filters of size 8 and stride 4, the second layer has 64 filters of size 4 and stride 2, and the final convolution layer has 64 filter of size 3 and stride 1. After this layer, the network splits into 10 heads, which each contain a fully-connected hidden layer of size 512. The activation for each of these layers is a rectified linear unit (ReLu). A final fully-connected layer projects to the Q-value output. A simpler network is used in the maze domain, consisting of a fully-connected hidden layers of size 128, which again splits into 10 heads, each of which contains another fully-connected layer of size 128. All layers are also separated by ReLu activations. The networks are trained with the Adam optimizer using the suggested default settings ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$)[38].

	Freeway	Maze
max steps	10^7	10^6
discount rate	0.99	0.99
train delay	80000	0
train frequency	4	1
target update frequency	30000	1000
batch size (RL loss)	32	32
batch size (SL loss)	32	32
mask probability	0.5	0.5
ensemble size	10	10
replay buffer size	10^6	10^6
β (prioritized replay)	$0.4 \rightarrow 1$	$0.4 \rightarrow 1$
α (prioritized replay)	0.5	0.5
learning rate	10^{-4}	10^{-4}
margin size (SL loss)	10^{-6}	10^{-6}

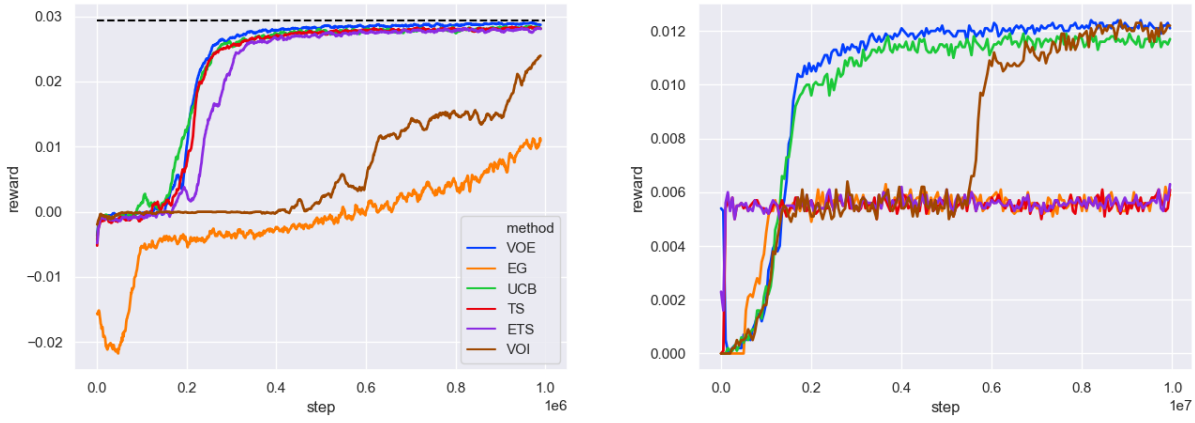
Table 4.2: Hyperparameters for Freeway and maze domains.

We implement a teacher in the maze domain by solving the underlying the MDP using policy iteration [33]. This allows the optimal action to be given by the teacher for every query.

4.3 Experiment: Exploration

If we ignore the teacher-query action, VOE/Q action selection is a novel exploration method for standard RL. Since there is no querying, we call this version VOE. We first study how this method compares to other proposed exploration methods for RL in the literature. VOE is a simplification of the VOI algorithm proposed by Dearden et al. [16]. We are unaware of any work that studies VOI action selection in the context of deep Q-learning, so this is also serves as an important baseline. We list all of the compared methods below.

- VOE: The exploration strategy that we we propose in Section 3.3.2, ignoring the teacher-query action.
- ϵ-greedy: The exploration strategy proposed in the original DQN paper and used in most of the subsequent work [47, 30, 64]. ϵ-greedy entails picking an action uniformly



(a) Maze

(b) Freeway

Figure 4.3: Learning curves for (a) maze and (b) Freeway domains using various exploration strategies. A running average of size 10^4 is used for smoothing. Median reward over 10 repetitions is shown. Optimal performance is indicated with a dotted black line.

at random with probability ϵ, and otherwise picking the greedy action. The value of ϵ is typically decreased linearly over time. For Freeway, we linearly decrease ϵ from 1 to 0.01 over the first 10^6 steps like in prior work [30]. For the maze, we decrease ϵ from 1 to 0.01 over the first 10^5 steps. The greedy action is the action with the highest mean Q-value over the ensemble.

- Bayes-UCB: UCB strategy that was implemented by Chen et al. [10]. This strategy is the same as described in Section 2.1.4, except that means and standard deviations are replaced by sample means and sample standard deviations of the Q-ensemble. The original paper uses $\lambda = 0.1$, but we used $\lambda = 2$, as we found it produced better results.

- Thompson sampling (TS): At each time step, one ensemble member is randomly sampled and the greedy action according to that Q-function is taken.

- Episodic Thompson sampling (ETS): The variant of Thompson sampling proposed by Osband et al., in which an ensemble member is randomly sampled at the beginning of each episode, and actions are taken greedily with respect to that Q-function for the entire episode [50].

- VOI: The exploration strategy proposed by Dearden et al. (referred to in that work

as "myopic VPI") [16]. All expectations are replaced by sample approximations using the Q-ensemble.

We compare all exploration methods on Freeway and the maze domain, and repeated each run 10 times with different random seeds. The median learning curves for these games are compared in Figure 4.3. We also report the median cumulative reward in Table B.1.

VOE achieves the highest cumulative reward in both tasks, and is also amongst the highest in final performance. This is in stark contrast with the VOI method from which it is based. VOI is very slow to improve in the maze and remains at 0 reward for nearly half of the experiment. This is presumably because it fails to discover the goal. Similarly, VOI remains stuck at a suboptimal policy in Freeway for roughly half of the experiment.

The second highest cumulative reward in both tasks is achieved by UCB, although in Freeway, it seems to plateau at slightly lower reward. Both Thompson sampling methods perform similarly. For the maze problem, performance is adequate and both variants end up at close to optimal performance. However, they both struggle with Freeway, and are stuck with a poor policy for the duration of the experiment. Perhaps unsurprisingly, ϵ-greedy performs quite poorly in both tasks. In the maze problem, it is very slow to improve, with the worst performance over the entire experiment. In Freeway, it seems to get stuck similarly to Thompson sampling.

4.4 Experiment: Asking for Help

In the remaining experiments, we incorporate a teacher into the maze domain and assess how well VOE/Q performs in ask-for-help RL. The query cost C is fixed at 0.02. To assess this algorithm, we needed to come up with some reasonable baselines for the ask-for-help RL problem. We will discuss how these baselines are created in the next section.

4.4.1 Alternative Approaches

Our algorithm jointly decides to take the query action or one of the environment actions, but it is also possible to separate these decisions. We could treat the decision to ask for help as separate from the decision about which environment action to take. If separated, these two decision could be made in two orderings. This leads to the following decision-making possibilities:

(a) `Joint`: We jointly decide between all actions (including the teacher-query action).

(b) `Query-Env`: We first decide whether to query the teacher or not. If not, we decide which environment action to take.

(c) `Env-Query`: We first decide which environment action to take. We then decide whether to replace it with the query action.

The joint decision approach is natural in our framework, because querying is considered an action, and therefore its utility can be considered alongside the environment actions. However, one advantage of separating the exploration and query decisions is that it allows us to leverage existing exploration methods for RL, like ϵ-greedy.

To make the query decision, the simplest alternatives would be (1) random querying (2) always querying and (3) never querying.

We call the random querying strategy π-query. With some probability π, the teacher is queried. This is analogous to ϵ-greedy for exploration. In fact, if we pair π-query with ϵ-greedy exploration and set $\pi = \frac{\epsilon}{n+\epsilon}$, where n is the number of environment actions, it would be equivalent to performing ϵ-greedy over all actions (querying being considered as one of the non-greedy actions). To follow the optimal policy, querying must eventually stop, so the value of π must eventually reach 0. Therefore, we use a linear schedule in which π is reduced to 0 over the training period. This strategy is effectively the same as probabilistic policy reuse [22]. Always querying means that help is requested from the teacher at every time step. If querying carries a cost, this means that the cost will be incurred forever and the optimal policy will never actually be followed. At the other extreme, never querying means that the environment is explored without any teacher assistance – this is the same as standard RL.

4.4.2 Baselines

With the information above, we can generate several baselines to test VOE/Q against. We test against the strategy to never query and do only VOE exploration, as well as the strategy to always query. We then pair ϵ-greedy exploration with VOE/Q querying in both orderings (`Env-Query` and `Query-Env`). Finally, we pair π-querying with VOE exploration. All of these baselines are described in more detail below:

- VOE/Q: the action selection algorithm that we propose in Section 3.3.2. [`Joint`]

Figure 4.4: Learning curves for maze domains using various action selection schemes. A running average of size 10^4 is used for smoothing. Median reward over 10 repetitions is shown. Optimal performance is indicated with a dotted black line.

- VOE (never query): VOE/Q action selection, with the query action removed.
- always query: the strategy to query the teacher at every time step.
- ϵ-greedy-VOE/Q: An environment action is picked by ϵ-greedy. If the teacher-query action has higher utility than the selected action (VOE/Q), the teacher is queried instead. The schedule for ϵ is the same as in the previous experiment. [`Env-Query`]
- VOE/Q-ϵ-greedy: If the teacher-query action has the highest utility (VOE/Q), the teacher is queried. Otherwise, an environment action is picked by ϵ-greedy. The schedule for ϵ is the same as in the previous experiment. [`Query-Env`]
- π-query-VOE: With probability π, the teacher is queried. Otherwise, an environment action is picked according to VOE. π is linearly reduced to 0 over the training period. We choose the initial value of π so that the expected number of queries over the training period would be equal to the number of queries performed by VOE/Q. This allows us to directly compare the querying strategies given the same query budget.

The learning curves for all querying methods in the maze domain are shown in Figure 4.4. Cumulative reward is reported in Table B.2, along with the total contribution of query costs.

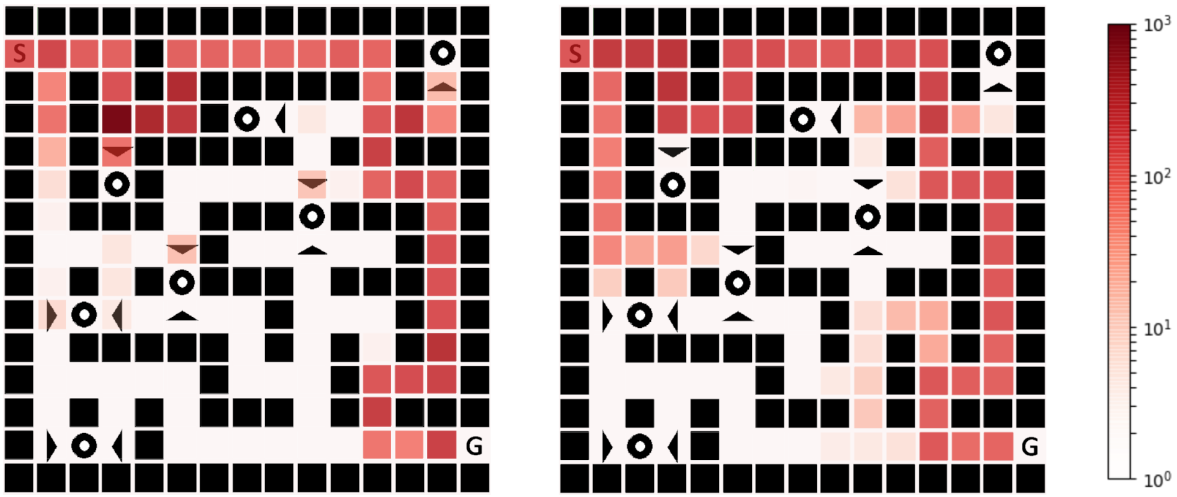


Figure 4.5: Heat map for one run of VOE/Q-querying (left) and π-querying (right) with $C = 0.02$. Darker red indicates a higher frequency of teacher queries when the agent occupies that cell. The mapping of color to number of queries is shown on the right.

We find that VOE/Q outperforms all baselines in terms of cumulative reward. Closest to its performance is ϵ-greedy-VOE/Q. This method ended up with nearly fives times as many queries to the teacher, which allowed it to more quickly converge on an optimal policy, but the cost of all of these extra queries outweighed its benefit. VOE/Q-ϵ-greedy also performed more queries to the teacher than VOE/Q, paying a higher query penalty. A higher query penalty, paired with less effective ϵ-greedy exploration, lead to very poor performance. Interestingly, this method performs even worse than never querying and exploring with VOE. This points to how much more effective VOE exploration is than ϵ-greedy. The random querying approach, δ-query-VOE, performs better than VOE exploration alone, but still substantially worse than VOE/Q. This suggests the importance of *directed* querying – asking for help in the states that it is expected to be the most useful. The last strategy is to always query, which can be observed as a flat line. Since a query cost is incurred at every time step, this strategy will forever obtain an average reward that is $C = 0.02$ lower than optimal, so it is not viable.

Taken together, these comparisons highlight the delicate balance that must be struck between exploration, exploitation and teacher-querying. The methods that rely too greatly on the teacher accumulate heavy query costs, while the methods that do not rely enough on the teacher must perform too much costly exploration. VOE/Q seems to effective at making this trade-off.

In the maze domain, we are able to visualize VOE/Q by overlaying a heat map on the maze as shown in Figure 4.5. Cells that are darker red indicate states in which the agent performed more queries to the teacher. VOE/Q-querying concentrates many queries in the state near the beginning of the maze where the agent is two cells away from a hole. Since there is a 0.5 probability of "falling in" when directly adjacent to the hole, this represents a critical state, in the sense that the action chosen in this states is highly consequential to the expected return. As a result, this is a state in which asking for help carries very high utility. In contrast, we observe that virtually no queries occur in areas of the maze without any nearby holes, such as the corrodors on the top and right of the maze. This is reasonable since none of the actions in these states have much effect on the expected return. In this case, the information gained by querying does not outweigh its cost. We compare to the heat map with random querying. Since queries are randomly performed, they are more spread out, without as much concentration in particular states. We also see more queries in parts of the maze that do not lie along the optimal path. This suggests that VOE/Q does a better job of using queries to direct its exploration at the onset of learning, and does not waste as much time exploring other parts of the maze.

4.5 Experiment: Adjusting Query Cost

Unlike approaches like π-query, VOE/Q querying takes query cost into account. To study the effect of query cost, the algorithm was ran with a range of query costs: 0, 0.00125, 0.0025, 0.005, 0.01, 0.02, 0.04, and 0.08. Learning curves for the maze domain are shown in Figure 4.6 and cumulative rewards are reported in Table B.3.

There is a clear ordering of performance by query cost. As the cost goes down, the cumulative reward increases. When queries are cheaper, VOE/Q takes advantage by querying more often. The difference is most clear near the beginning of learning. At high query costs, the agent must forgo asking for help and do costly exploration to find the goal state. On the other hand, at lower query costs, the agent can use the teacher to quickly identify the path to the goal state, so learning is accelerated. At a cost of 0, VOE/Q ensures that the teacher is always queried, such that the optimal policy is immediately followed and maximum possible cumulative reward is attained.

Figure 4.7 shows how the frequency of querying changes over time, varying the query cost. There is again a rough ordering by query costs. At a cost of 0, the teacher is always queried so the frequency stays at 1. At non-zero cost, the frequency begins high but decreases exponentially as uncertainty is reduced and the ensemble converges. At costs

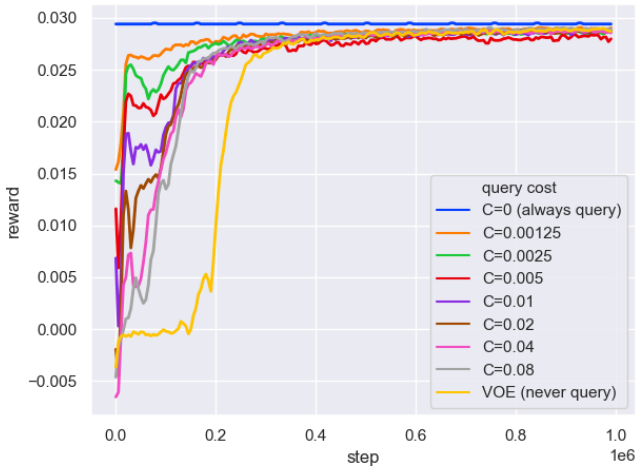


Figure 4.6: Learning curves for maze domain using VOE/Q action selection, varying the query cost. A running average of size 10^4 is used for smoothing. Median reward over 10 repetitions is shown.

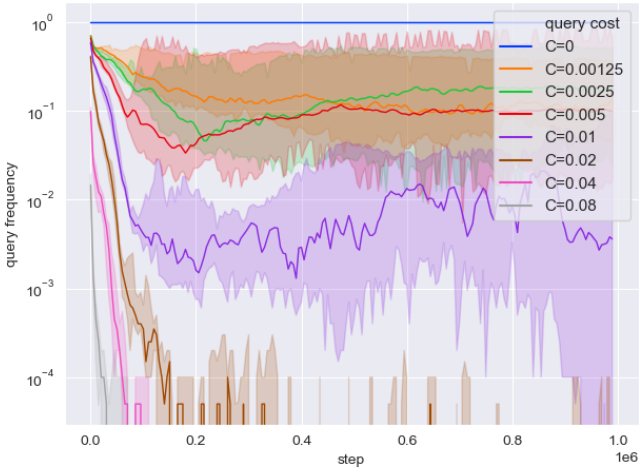


Figure 4.7: Query frequency over time for maze domain using VOE/Q action selection, varying the query cost. Y-axis is logarithmic. Frequency is estimated by the number of queries in a moving window of size 10^4, with the median over 10 repetitions shown. Shaded region corresponds to 90% confidence interval.

Figure 4.8: Learning curves for maze domain using VOE/Q action selection, varying the probability of optimal advice. A running average of size 10^4 is used for smoothing. Median reward over 10 repetitions is shown. Optimal performance is indicated with a dotted black line.

below 0.01, there is still a small frequency of queries at the end of training, but we would expect this to eventually drop off as the ensemble converges to the optimal Q-function.

4.6 Experiment: Varying Advice Optimality

VOE/Q estimates the utility of querying with the assumption of an optimal teacher. However, in many practical scenarios, a perfect teacher would not be readily available. Indeed, for many tasks, a human teacher would not be guaranteed to always give the optimal action. We therefore investigate how suboptimal teacher advice affects the performance of the method in the maze domain, given a fixed query cost of $C = 0.02$.

We model a suboptimal teacher as recommending an optimal action with probability p, and otherwise, recommending some suboptimal action uniformly at random. We experiment with values of p ranging from 100% to 0%. 100% corresponds to a perfect teacher that always gives an optimal action, and 0% represents an adversarial teacher that always gives a suboptimal action. Since there are 4 actions in the maze domain, 25% corresponds to a teacher that gives random advice. Learning curves are shown in Figure 4.8 and cumulative rewards are reported in Table B.4. As a baseline, we also compare with VOE action

selection, which ignores the teacher and performs standard RL.

The results are largely as expected, but with a few surprises. As expected, performance is best with the perfect teacher and worst with the adversarial teacher. Interestingly, we also find that VOE/Q does better than VOE for all teachers that provide better-than-random advice ($> 25\%$).

The finding that any better-than-random advice is advantageous for learning is somewhat unexpected, considering that random advice is quite uninformative and each query incurs a relatively high cost. However, even random advice can serve to promote exploration, and help the agent to discover the optimal policy. Counterintuitively, we do find that performance is not perfectly sorted in order of advice optimality. Indeed, 37.5% optimal advice performs second best (after 100% optimal). We are unsure why this occurs, although it might simply be due to random chance, and performances might sort as expected if we were to conduct more repetitions of the experiment. Taken together, this experiment suggests that VOE/Q is still an effective method when paired with teachers that frequently give suboptimal advice.

4.7 Experiment: Adding the Supervised Learning Update

All previous experiments used only the reinforcement learning update and not the supervised learning update (described in Section 3.3.3). We next investigate the effect of adding the supervised update. As shown in Algorithm 1, we are not limited to just performing reinforcement updates on the network, but may also perform supervised updates using the state-action pairs in the replay buffer that correspond to teacher-advised actions.

We set the frequency of the supervised updates to be the same as the reinforcement updates (every step). We also make the batch size the same (32). Just like with the reinforcement update, we update the network using the Adam optimizer with the recommended default settings.

We compare the performance of our algorithm in both domains, with and without the supervised update. The query cost is again fixed at $C = 0.02$.

Learning curves are shown in Figure 4.9 and cumulative rewards are reported in Table B.5. As expected, adding the supervised update causes substantial gains in learning speed and cumulative reward. Supervised learning allows the agent to quickly replicate the teacher's actions, rather than relying solely on learning from the delayed reward signal.

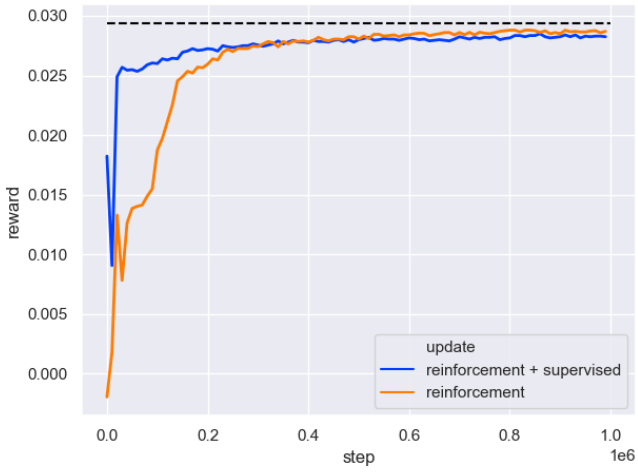


Figure 4.9: Learning curves for maze domain using VOE/Q action selection and $C = 0.02$, with and without the supervised learning update. A running average of size 10^4 is used for smoothing. Median reward over 10 repetitions is shown. Optimal performance is indicated with a dotted black line.

The advantage of including the supervised update is apparent from the learning curves – the agent learning with supervision arrives very quickly at a policy that attains high reward, and it needs only a small amount of refinement over the remainder of the learning period. Since the network converges more quickly, it also means that fewer queries to the teacher are needed. With the supervised update, nearly 10 times fewer teacher-query actions are taken.

Chapter 5

Discussion

The experiments conducted shed light on the use of VOE/Q for both standard and ask-for-help RL.

5.1 VOE exploration for RL

By removing the teacher-query action, we can investigate the use of VOE for standard RL. We compare against many other directed (UCB, VOI, Thompson sampling) and undirected (ϵ-greedy) exploration strategies. The domains chosen, maze navigation and Freeway, are challenging exploration problems because of the long horizon and sparse reward. In both tasks, a positive reward is only achieved after successfully executing a long sequence of actions and avoiding many obstacles that will terminate the episode. Hence, the methods that can effectively explore are distinguished from those that cannot. We find that VOE achieves the best performance in both domains. This suggests that it is a promising method that is worthy of further study for RL. In contrast, the VOI method from which VOE is based does relatively poorly on both tasks. This adds credence to our hypothesis that because VOI gives a value-of-information bonus to the greedy action, it is unable to explore as effectively.

5.2 VOE/Q action selection for ask-for-help RL

Next, we turn to the use of VOE/Q for ask-for-help RL. While VOE/Q is an approach that jointly decides whether to query the teacher or explore the environment, we are able to

generate several baselines by treating the query and exploration decisions separately. First, we consider fixing the querying strategy to VOE/Q but changing the exploration strategy to ϵ-greedy. The two can be combined in two different decision orderings: `Env-Query` and `Query-Env`. Both result in inferior performance. The obvious explanation is that ϵ-greedy is not as effective as an exploration method, which would be consistent with our earlier findings. However, there is also an interplay with the querying behavior - we find that both strategies that use ϵ-greedy query the teacher more often. This is initially surprising when considering that query behavior of VOE/Q-ϵ-greedy is identical to VOE/Q, but the difference could be attributed to the distribution of states that are visited. For example, in the maze domain, ϵ-greedy might randomly move the agent beside a hole. Due to the stochastic nature in which the agent "falls" into the hole, the value of actions in this state might be highly uncertain, so the agent is more likely to ask for help. However, if VOE was used instead of ϵ-greedy, this state might be better avoided, resulting in fewer teacher queries, lower query costs, and better performance.

We also look at fixing exploration to VOE, and varying the querying strategy. There are three obvious alternative querying strategies: to never query, to always query, and to randomly query. Never querying is akin to standard RL, and results in lower cumulative reward because it fails to take advantage of the teacher. At the other extreme, always querying is also problematic if the query cost is non-zero, since the agent accumulates query costs at every time step and never follows the optimal policy. A viable alternative is π-querying, which randomly queries with probability π but reduces π to 0 over the training period so that teacher-querying eventually stops. To choose the initial value of π, we opt to set it so that the expected number of queries is the same as VOE/Q. Comparing these two methods, we find that it is not just the amount of queries that matters, it is also when the teacher is queried and in what states. This can be seen in the heap maps – π-querying uniformly spreads out queries along all visited states, while VOE/Q concentrates queries in particular regions of the environment. The temporal aspect of querying is also important. While we choose to reduce the value of π over time on a linear schedule, the choice of schedule is ad-hoc. On the other hand, VOE/Q automatically reduces querying in correspondence with the uncertainty of the Q-ensemble. This means that querying is initially performed in virtually every state but drops off as learning progresses and as the ensemble converges.

We next examine how VOE/Q adapts to query costs. The general trend is as expected – querying occurs more frequently as costs decrease, and furthermore, performance improves. With cheaper queries, the agent relies less on costly exploration and instead takes advantage of teacher advice. Interestingly, the change in querying frequency is not proportional to the change in cost, i.e., the overall spend on querying does not stay the same (Table B.3).

Rather, the total amount spent on querying seems to have a single peak at some cost, and then decreases in either direction (as cost goes to 0 and infinity).

Then, we consider the effect of a suboptimal teacher by varying the probability with which the teacher returns an optimal action. As expected, the best performance is with the perfect teacher (100% optimal) and the worst performance is with the adversarial teacher (0% optimal). However, we find that any teacher that provides better-than-random advice results in better performance than is obtained by foregoing the teacher. The key takeaway from this experiment is that VOE/Q can still be effective with a suboptimal teacher. This is because the teacher may still occasionally provide optimal advice, and even when it does not, may promote exploration of the environment. One caveat to this finding is that there are many possible suboptimal policies that a teacher could possess. We only consider the class of suboptimal policies that randomly selects a suboptimal action with some fixed probability. However, the probability of suboptimal advice is not the only important factor for assessing the usefulness of a teacher – the *degree* of suboptimality should also be considered (i.e., the Q-value of the action). In other words, a distinction should be made between slightly suboptimal advice and catastrophically bad advice.

Finally, we investigate the addition of the supervised update. Adding the supervised update causes learning to speed up considerably and cumulative reward to be much higher. This suggests that, if we can assume that the teacher has an optimal policy, the supervised update should be included. In many settings, this assumption may not be warranted. For example, an optimal policy is difficult to obtain in Atari video games. In many real-world scenarios, there is no readily-available optimal policy. Instead, we might consider a human teacher to provide advice, but this advice is not guaranteed to be optimal. In these cases, the supervised update may hinder learning progress by pushing a suboptimal action that is recommended by the teacher to have the highest Q-value.

5.3 Other considerations

One of the advantages to VOE/Q action selection is that there are no hyperparameters that need to be chosen. This is in contrast to many other possible approaches to exploration or teacher querying. For example, the UCB algorithm requires λ to be set, which controls the extent of exploration. The δ-query approach requires δ, which controls the probability of querying. It is difficult to determine such hyperparameters *a priori*, which limits its applicability to online learning systems. With that said, it might be possible to improve the performance of VOE/Q by weighting value-of-information estimates by some constant,

and possibly weighting the value-of-information for environment actions differently than for the teacher-query action.

One factor that would strongly affect the performance of VOE/Q, but which we did not investigate in this thesis, is the initialization of the ensemble. We opt to randomly initialize the parameters of each neural network according to Kaiming initialization (the default in PyTorch) [29]. However, since we treat the ensemble as samples from a prior, we would ideally initialize it to more closely mirror any prior beliefs about the optimal Q-function. This could be done in several ways. For example, the ensemble could be pre-trained on a set of existing demonstrations from the teacher. If the algorithm is intended to run in a real-world environment, another possibility would be to pre-train the network in a simulator.

Chapter 6

Conclusion

The issue of poor sample efficiency is persistent within RL. It takes many time steps to converge to an optimal policy, and moreover, the learning process may entail repeatedly making costly mistakes. This problem ties into the issue of safety - how can RL systems learn in the real world while avoiding catastrophic errors? To some extent, this issue is insurmountable within the standard RL framework, since in order to determine if an action is dangerous, it must be repeatedly performed to realize the negative consequences.

In an effort to improve sample efficiency and safety, there has been considerable attention on methods that incorporate teacher advice into RL. By and large, these methods rely on one of two assumptions: there is either a fixed set of demonstrations provided by the teacher at the onset of learning, or there is unrestricted access to the teacher policy throughout learning. Both of these assumptions are far from ideal when considering real teachers (e.g. humans). Often, it is possible to request advice on-the-fly from a teacher, but this advice is not free. It comes at a cost that must be weighed against its potential benefit.

In Chapter 3, we introduced a framework for incorporating teacher advice into reinforcement learning called ask-for-help RL. Querying the teacher is an action that can be taken alongside the environment actions in a given state. Querying incurs a fixed cost, but returns the optimal action to the agent. In this framework, the challenge becomes how to balance exploration, exploitation and teacher-querying. We addressed this problem by proposing a simple action selection strategy called VOE/Q that is based on the value-of-information principle and applicable with Bayesian Q-learning. We show how the strategy can be implemented in deep Q-learning, building off of Bootstrapped DQN [50]. Uncertainty is estimated by an ensemble, with each member trained on different subsets

of the data that is stored in a shared replay buffer. The Q-ensemble can be trained using both reinforcement and supervised learning updates.

We empirically test this approach in Chapter 4 using two domains: a maze navigation problem, and the Atari game Freeway. Ignoring the teacher-query action, we find that VOE is a surprisingly good exploration strategy that outperforms other approaches like ϵ-greedy, Thompson sampling, and UCB. When we introduce teacher-querying, we find VOE/Q action selection again does better than all baselines that were considered. We observe how the algorithm adjusts to changes in query cost, and how querying is highly concentrated in critical states at the beginning of learning, when teacher advice is most useful. We also find that the algorithm can remain effective even if the teacher frequently provides suboptimal advice.

6.1 Future work

There are many avenues for future work. It would be interesting to study the performance of VOE/Q action selection when combined with other approaches to uncertainty-aware Q-learning. We might also look for at entirely new algorithms for ask-for-help RL. For example, other promising approaches might arise in model-based RL.

Ask-for-help RL uses an idealized version of a teacher, but we would ultimately like the framework to be useful in a broad range of applications. Therefore, we might consider relaxing any of the following assumptions about the teacher:

- **Teacher optimality.** The teacher might be suboptimal. In this case, the agent might be provided information about the states in which the teacher provides the best advice, or the agent might learn this through experience. Agent decisions should take into account how good the advice is expected to be in current and future states.
- **Teacher availability.** We assume that the teacher is available to be queried in every state. Instead, teacher availability might be given to the agent or learned through experience. Agent decisions should take into account whether the teacher is currently available, or will be in future states.
- **Fixed query costs.** Cost of querying the teacher might vary depending on the state, and may be initially known or unknown. This cost might reflect the difficulty or inconvenience of obtaining advice in particular states. Analogous work has been done in active learning to consider variable labelling costs [36, 65, 81].

- **Single-action advice.** Instead, it might only be possible for the teacher to provide advice over longer courses of action. The options framework might be useful for such a formulation [74].

- **Single-teacher advice.** There might be a set of teachers from which advice can be obtained, as explored in some recent work [41, 87, 43]. These teachers might each be suboptimal, but have expertise in particular states. In addition, these teachers might each have different querying costs.

Another direction of research would be study the role of human teachers with VOE/Q. Important questions include how humans are best suited to give advice to artificial agents, how the cost of human advice can be quantified, and how to promote trust between human teachers and agents that ask for help. Further study of algorithms that can reason about the costs and benefits of teacher advice opens the door to more practical RL agents that can safely integrate into real-world environments.

References

[1] Shipra Agrawal and Navin Goyal. Further optimal regret bounds for thompson sampling. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2013, Scottsdale, AZ, USA, April 29 - May 1, 2013*, pages 99–107, 2013.

[2] Ofra Amir, Ece Kamar, Andrey Kolobov, and Barbara J. Grosz. Interactive teaching strategies for agent training. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 804–811, 2016.

[3] Oron Anschel, Nir Baram, and Nahum Shimkin. Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 176–185, 2017.

[4] Brenna D. Argall, Sonia Chernova, Manuela M. Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics Auton. Syst.*, 57(5):469–483, 2009.

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

[6] Kamyar Azizzadenesheli, Emma Brunskill, and Animashree Anandkumar. Efficient exploration through bayesian deep q-networks. *CoRR*, abs/1802.04412, 2018.

[7] Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 449–458, 2017.

[8] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *CoRR*, abs/1505.05424, 2015.

[9] Victor Uc Cetina. Autonomous agent learning using an actor-critic algorithm and behavior models. In *7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008), Estoril, Portugal, May 12-16, 2008, Volume 3*, pages 1353–1356, 2008.

[10] Richard Y Chen, Szymon Sidor, Pieter Abbeel, and John Schulman. Ucb exploration via q-ensembles. *arXiv preprint arXiv:1706.01502*, 2017.

[11] Sonia Chernova and Manuela M. Veloso. Interactive policy learning through confidence-based autonomy. *J. Artif. Intell. Res.*, 34:1–25, 2009.

[12] Jeffery A Clouse and Paul E Utgoff. A teaching method for reinforcement learning. In *Machine Learning Proceedings 1992*, pages 92–101. Elsevier, 1992.

[13] Jeffery Allen Clouse. *On integrating apprentice learning and reinforcement learning.* University of Massachusetts Amherst, 1996.

[14] Felipe Leno Da Silva, Pablo Hernandez-Leal, Bilal Kartal, and Matthew E Taylor. Uncertainty-aware action advising for deep reinforcement learning agents. *34th AAAI Conference on Artificial Intelligence*, 2020.

[15] Richard Dearden, Nir Friedman, and David Andre. Model based bayesian exploration. In *UAI '99: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden, July 30 - August 1, 1999*, pages 150–159, 1999.

[16] Richard Dearden, Nir Friedman, and Stuart J. Russell. Bayesian q-learning. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference, AAAI 98, IAAI 98, July 26-30, 1998, Madison, Wisconsin, USA*, pages 761–768, 1998.

[17] Finale Doshi-Velez, David Wingate, Nicholas Roy, and Joshua B. Tenenbaum. Nonparametric bayesian policy priors for reinforcement learning. In *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada*, pages 532–540, 2010.

[18] Kurt Driessens and Sašo Džeroski. Integrating guidance into relational reinforcement learning. *Machine Learning*, 57(3):271–304, 2004.

[19] Michael O'Gordon Duff and Andrew Barto. *Optimal Learning: Computational procedures for Bayes-adaptive Markov decision processes.* PhD thesis, University of Massachusetts at Amherst, 2002.

[20] Yaakov Engel, Shie Mannor, and Ron Meir. Reinforcement learning with gaussian processes. In *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005*, pages 201–208, 2005.

[21] Fernando Fernández-Rebollo, Javier García, and Manuela M. Veloso. Probabilistic policy reuse for inter-task transfer learning. *Robotics Auton. Syst.*, 58(7):866–871, 2010.

[22] Fernando Fernández-Rebollo and Manuela M. Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan, May 8-12, 2006*, pages 720–727, 2006.

[23] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 1050–1059, 2016.

[24] Yang Gao, Huazhe Xu, Ji Lin, Fisher Yu, Sergey Levine, and Trevor Darrell. Reinforcement learning from imperfect demonstrations. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*, 2018.

[25] Javier Garcia and Fernando Fernández. Safe exploration of state and action spaces in reinforcement learning. *Journal of Artificial Intelligence Research*, 45:515–564, 2012.

[26] Francisco Javier García-Polo and Fernando Fernández-Rebollo. Safe reinforcement learning in high-risk tasks through policy improvement. In *2011 IEEE Symposium on Adaptive Dynamic Programming And Reinforcement Learning, ADPRL 2011, Paris, France, April 12-14, 2011*, pages 76–83, 2011.

[27] Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, Aviv Tamar, et al. Bayesian reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 8(5-6):359–483, 2015.

[28] José Antonio Martín H. and Javier de Lope Asiaín. Learning autonomous helicopter flight with evolutionary reinforcement learning. In *Computer Aided Systems Theory - EUROCAST 2009, 12th International Conference, Las Palmas de Gran Canaria, Spain, February 15-20, 2009, Revised Selected Papers*, pages 75–82, 2009.

[29] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 1026–1034, 2015.

[30] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 3215–3222, 2018.

[31] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[32] Ionel-Alexandru Hosu and Traian Rebedea. Playing atari games with deep reinforcement learning and human checkpoint replay. *CoRR*, abs/1607.05077, 2016.

[33] Ronald A Howard. Dynamic programming and markov processes. 1960.

[34] Ronald A. Howard. Information value theory. *IEEE Trans. Systems Science and Cybernetics*, 2(1):22–26, 1966.

[35] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Learning rhythmic movements by demonstration using nonlinear oscillators. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, Lausanne, Switzerland, September 30 - October 4, 2002*, pages 958–963, 2002.

[36] Ashish Kapoor, Eric Horvitz, and Sumit Basu. Selective supervision: Guiding supervised learning with decision-theoretic active learning. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 877–882, 2007.

[37] Emilie Kaufmann, Olivier Cappé, and Aurélien Garivier. On bayesian upper confidence bounds for bandit problems. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2012, La Palma, Canary Islands, Spain, April 21-23, 2012*, pages 592–600, 2012.

[38] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[39] W Bradley Knox and Peter Stone. Interactively shaping agents via human reinforcement: The tamer framework. In *Proceedings of the fifth international conference on Knowledge capture*, pages 9–16. ACM, 2009.

[40] Igor Kononenko. Bayesian neural networks. *Biological Cybernetics*, 61(5):361–370, 1989.

[41] Andrey Kurenkov, Ajay Mandlekar, Roberto Martin Martin, Silvio Savarese, and Animesh Garg. Ac-teach: A bayesian actor-critic method for policy learning with an ensemble of suboptimal teachers. *CoRR*, abs/1909.04121, 2019.

[42] Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

[43] Siyuan Li and Chongjie Zhang. An optimal online method of selecting source policies for reinforcement learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 3562–3570, 2018.

[44] Zachary C. Lipton, Xiujun Li, Jianfeng Gao, Lihong Li, Faisal Ahmed, and Li Deng. Bbq-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5237–5244, 2018.

[45] David J. C. MacKay. A practical bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472, 1992.

[46] Frederic Maire and Vadim Bulitko. Apprenticeship learning for initial value functions in reinforcement learning. *Planning and Learning in A Priori Unknown or Dynamic Domains*, page 23, 2005.

[47] 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, 2015.

[48] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*, pages 6292–6299, 2018.

[49] Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999), Bled, Slovenia, June 27 - 30, 1999*, pages 278–287, 1999.

[50] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped DQN. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4026–4034, 2016.

[51] Ian Osband, Daniel Russo, and Benjamin Van Roy. (more) efficient reinforcement learning via posterior sampling. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3003–3011, 2013.

[52] Art B Owen, Dean Eckles, et al. Bootstrapping data arrays of arbitrary order. *The Annals of Applied Statistics*, 6(3):895–927, 2012.

[53] Tim Pearce, Nicolas Anastassacos, Mohamed Zaki, and Andy Neely. Bayesian inference with anchored ensembles of neural networks, and application to reinforcement learning. *CoRR*, abs/1805.11324, 2018.

[54] Bilal Piot, Matthieu Geist, and Olivier Pietquin. Boosted bellman residual minimization handling expert demonstrations. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 549–564. Springer, 2014.

[55] Dean Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3(1):88–97, 1991.

[56] VN Pradyot Korupolu and Balaraman Ravindran. Beyond rewards: Learning from richer supervision. 2011.

[57] Pablo Quintía Vidal, Roberto Iglesias Rodríguez, Miguel Ángel Rodríguez González, and Carlos Vázquez Regueiro. Learning on real robots from experience and simple user feedback. *Journal of Physical Agents*, 2013.

[58] Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, pages 627–635, 2011.

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

[60] Ahmad El Sallab, Mohammed Abdou, Etienne Perot, and Senthil Kumar Yogamani. Deep reinforcement learning framework for autonomous driving. *CoRR*, abs/1704.02532, 2017.

[61] Claude Sammut, Scott Hurst, Dana Kedzier, and Donald Michie. Learning to fly. In *Proceedings of the Ninth International Workshop on Machine Learning (ML 1992), Aberdeen, Scotland, UK, July 1-3, 1992*, pages 385–393, 1992.

[62] Joe Saunders, Chrystopher L. Nehaniv, and Kerstin Dautenhahn. Teaching robots by moulding behavior and scaffolding the environment. In *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction, HRI 2006, Salt Lake City, Utah, USA, March 2-3, 2006*, pages 118–125, 2006.

[63] Stefan Schaal. Learning from demonstration. In *Advances in Neural Information Processing Systems 9, NIPS, Denver, CO, USA, December 2-5, 1996*, pages 1040–1046, 1996.

[64] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[65] Burr Settles, Mark Craven, and Lewis Friedland. Active learning with real annotation costs. In *Proceedings of the NIPS workshop on cost-sensitive learning*, pages 1–10. Vancouver, CA:, 2008.

[66] Nils T. Siebel and Gerald Sommer. Evolutionary reinforcement learning of artificial neural networks. *Int. J. Hybrid Intell. Syst.*, 4(3):171–183, 2007.

[67] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[68] William D. Smart and Leslie Pack Kaelbling. Practical reinforcement learning in continuous spaces. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*, pages 903–910, 2000.

[69] William D. Smart and Leslie Pack Kaelbling. Effective reinforcement learning for mobile robots. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation, ICRA 2002, May 11-15, 2002, Washington, DC, USA*, pages 3404–3410, 2002.

[70] Malcolm J. A. Strens. A bayesian framework for reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*, pages 943–950, 2000.

[71] Halit Bener Suay, Tim Brys, Matthew E. Taylor, and Sonia Chernova. Learning from demonstration for shaping through inverse reinforcement learning. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, Singapore, May 9-13, 2016*, pages 429–437, 2016.

[72] Halit Bener Suay and Sonia Chernova. Effect of human guidance and state space size on interactive reinforcement learning. In *20th IEEE International Symposium on Robot and Human Interactive Communication, RO-MAN 2011, Atlanta, Georgia, USA, July 31 - August 3, 2011*, pages 1–6, 2011.

[73] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning - an introduction*. Adaptive computation and machine learning. MIT Press, 1998.

[74] Richard S. Sutton, Doina Precup, and Satinder P. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.*, 112(1-2):181–211, 1999.

[75] Matthew Edmund Taylor, Halit Bener Suay, and Sonia Chernova. Using human demonstrations to improve reinforcement learning. In *Help Me Help You: Bridging the Gaps in Human-Agent Collaboration, Papers from the 2011 AAAI Spring Symposium, Technical Report SS-11-05, Stanford, California, USA, March 21-23, 2011*, 2011.

[76] Andrea Lockerd Thomaz and Cynthia Breazeal. Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In *Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA*, pages 1000–1006, 2006.

[77] 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.

[78] Lisa Torrey and Matthew E. Taylor. Teaching on a budget: agents advising agents in reinforcement learning. In *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '13, Saint Paul, MN, USA, May 6-10, 2013*, pages 1053–1060, 2013.

[79] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 2094–2100, 2016.

[80] Matej Vecerík, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin A. Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *CoRR*, abs/1707.08817, 2017.

[81] Sudheendra Vijayanarasimhan and Kristen Grauman. What's it going to cost you?: Predicting effort vs. informativeness for multi-label image annotations. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 2262–2269, 2009.

[82] Thomas J. Walsh, Daniel Hewlett, and Clayton T. Morrison. Blending autonomous exploration and apprenticeship learning. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*, pages 2258–2266, 2011.

[83] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 1995–2003, 2016.

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

[85] Jason D. Williams and Geoffrey Zweig. End-to-end lstm-based dialog control optimized with supervised and reinforcement learning. *CoRR*, abs/1606.01269, 2016.

[86] Jeremy Wyatt. *Exploration and inference in learning from reinforcement*. PhD thesis, University of Edinburgh, UK, 1998.

[87] Yusen Zhan, Haitham Bou-Ammar, and Matthew E. Taylor. Theoretically-grounded policy advice from multiple teachers in reinforcement learning settings with applications to negative transfer. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 2315–2321, 2016.

[88] Jiakai Zhang and Kyunghyun Cho. Query-efficient imitation learning for end-to-end autonomous driving. *arXiv preprint arXiv:1605.06450*, 2016.

APPENDICES

Appendix A

Proofs

Proof of Theorem 3: *If $C = 0$, VOE/Q will immediately follow an optimal policy of querying in every state.*

Proof. Let a^e be some environment action.

$$\begin{aligned}
EVOQ(s) - ECOQ(s) &= \mathbb{E}[\max_a q^*(s,a)] - \max_a \mathbb{E}[q^*(s,a)] - (VE^* - (C + \mathbb{E}[\max_a q^*(s,a)])) \\
&= 2\mathbb{E}[\max_a q^*(s,a)] - \max_a \mathbb{E}[q^*(s,a)] - VE^* \\
&\geq \mathbb{E}[\max_a q^*(s,a)] - \max_a \mathbb{E}[q^*(s,a)] + \mathbb{E}[q^*(s,a^e)] - VE^* \\
&= \mathbb{E}[\max_a q^*(s,a)] - \max_a \mathbb{E}[q^*(s,a)] - ECOE(s,a^e) \\
&= \mathbb{E}[\max_a q^*(s,a) - \max_a \mathbb{E}[q^*(s,a)]] - ECOE(s,a^e) \\
&\geq \mathbb{E}[\max\{q^*(s,a^e) - \max_a \mathbb{E}[q^*(s,a)], 0\}] - ECOE(s,a^e) \\
&\geq EVOE(s,a^e) - ECOE(s,a^e)
\end{aligned}$$

Hence, the query action has the highest utility and will always be selected (assuming ties are broken by querying). □

Proof of Theorem 4: *If $C > 0$ and the Q^* posterior converges to the optimal action-value function, VOE/Q will follow an optimal policy.*

Proof. Suppose a^{1st} is an environment action with the highest value, $a^{1st} = \text{argmax}_a\, q^*(s,a)$, and a^e is some suboptimal environment action $q^*(s,a^e) < q^*(s,a^{1st})$. At convergence, we assume the posterior probability concentrates on the optimal action-value function. Therefore,

$$EVOE(s,a^e) = \mathbb{E}[\max\{q^*(s,a^e) - \mathbb{E}[q^*(s,a^{1st}),\ 0\}] = max\{q^*(s,a^e) - q^*(s,a^{1st}),\ 0\} = 0$$
$$EVOQ(s) = \mathbb{E}[\max_a q^*(s,a)] - \max_a \mathbb{E}[q^*(s,a)] = \max_a q^*(s,a) - \max_a q^*(s,a) = 0$$

Since both EVOE and EVOQ go to 0, the action with the lowest cost will be selected. .

$$ECOE(s,a^{1st}) = \mathbb{E}[q^*(s,a^{1st})] - \mathbb{E}[q^*(s,a^{1st})] = 0$$
$$ECOE(s,a^e) = \mathbb{E}[q^*(s,a^{1st})] - \mathbb{E}[q^*(s,a^e)] > 0$$
$$ECOQ(s) = \max_a \mathbb{E}[q^*(s,a)] - (\mathbb{E}[\max_a q^*(s,a)] - C) = \max_a q^*(s,a) - \max_a q^*(s,a) + C = C > 0$$

Hence, the greedy action a^{1st} has the lowest cost and will always be selected. □

Appendix B

Cumulative Reward Tables

Method	**Maze**	**Freeway**
VOE	**21908.0**	**104439.0**
EG	-1447.0	51047.0
UCB	21813.0	100313.0
TS	21276.5	53871.0
ETS	20125.0	55029.0
VOI	5834.5	79747.0

Table B.1: Median cumulative reward over 10 repetitions using various exploration strategies.

Method	**Maze**
VOE/Q	**25941.36 (-143.14)**
VOE (never query)	21908.0 (0)
always query	9380.77 (-19701.23)
ϵ-greedy-VOE/Q	25613.28 (-712.22)
VOE/Q-ϵ-greedy	8629.98 (-443.02)
π-query-VOE	23640.10 (-142.90)

Table B.2: Median cumulative reward (including query costs) over 10 repetitions using various action selection strategies and $C = 0.02$. The contribution of query costs is specified in brackets.

Query Cost	**Maze**
0 (always query)	29411.0 (0)
0.00125	28138.64 (-198.36)
0.0025	27367.90 (-429.60)
0.005	26467.22 (-790.78)
0.01	26140.39 (-317.61)
0.02	25941.36 (-143.14)
0.04	25498.66 (-53.84)
0.08	25231.02 (-14.48)
VOE (never query)	21908.0 (0)

Table B.3: Median cumulative reward (including query costs) over 10 repetitions using VOE/Q action selection, varying the query cost. The contribution of query costs is specified in brackets.

Probability Optimal	**Maze**
0%	786.67 (-81.83)
12.5%	13150.85 (-73.15)
25%	20388.04 (-80.96)
37.5%	24199.64 (-97.36)
50%	23615.28 (-121.22)
62.5%	22230.70 (-162.80)
75%	23072.85 (-148.65)
87.5%	23964.27 (-132.23)
100%	25941.36 (-143.14)
VOE (never query)	21908.0 (0)

Table B.4: Median cumulative reward (including query costs) over 10 repetitions using VOE/Q action selection and $C = 0.02$, varying the probability of optimal advice from the teacher. The contribution of query costs is specified in brackets.

Update	**Maze**
reinforcement + supervised	27271.38 (-13.62)
reinforcement	25941.36 (-143.14)

Table B.5: Median cumulative reward (including query costs) over 10 repetitions using VOE/Q action selection and $C = 0.02$, with and without the supervised update. The contribution of query costs is specified in brackets.