Assumed Density Filtering Q-learning

Abstract

While off-policy temporal difference (TD) methods have widely been used in reinforcement learning due to their efficiency and simple implementation, their Bayesian counterparts have not been utilized as frequently. One reason is that the non-linear max operation in the Bellman optimality equation makes it difficult to define conjugate distributions over the value functions. In this paper, we introduce a novel Bayesian approach to off-policy TD methods, called as ADFQ, which updates beliefs on state-action values, Q, through an online Bayesian inference method known as *Assumed Density Filtering*. In order to formulate a closed-form update, we approximately estimate analytic parameters of the posterior of the Q-beliefs. Uncertainty measures in the beliefs not only are used in exploration but also provide a natural regularization for learning. We show that ADFQ converges to Q-learning as the uncertainty measures of the Q-beliefs decrease. ADFQ improves common drawbacks of other Bayesian RL algorithms such as computational complexity. We also extend ADFQ with a neural network. Our empirical results demonstrate that the proposed ADFQ algorithm outperforms comparable algorithms on various domains including continuous state domains and games from the Arcade Learning Environment.

Introduction

Bayesian reinforcement learning (BRL) is a classic reinforcement learning (RL) technique that utilizes Bayesian inference to integrate new experiences with prior information about the problem in a probabilistic distribution. It explicitly quantifies the uncertainty of the learning parameters unlike standard RL approaches which do not properly account for uncertainty in the parameters. Explicit quantification of the uncertainty can help guide policies that consider the exploration-exploitation trade-off by exploring actions with higher uncertainty more often. Moreover, it can also regularize posterior updates by properly accounting for uncertainty.

Motivated by these potential advantages, a number of algorithms have been proposed in both model-based BRL (Dearden, Friedman, and Andre 1999; Strens 2000; Duff 2002; Guez, Silver, and Dayan 2012; Poupart et al. 2006) and model-free BRL (Dearden, Friedman, and Russell 1998; Engel, Mannor, and Meir 2003; 2005; Geist and Pietquin 2010; Chowdhary et al. 2014; Ghavamzadeh and Engel 2006). However, Bayesian approaches to *off-policy temporal difference (TD) learning* has been less well-studied compared to other methods due to difficulty in handling the max non-linearity in the Bellman optimality equation. Yet off-policy TD methods have been widely used in standard RL, including extensions integrating neural network function approximations such as Deep Q-Networks (DQN) (Mnih et al. 2013; 2015). One recent influential algorithm for Bayesian off-policy TD learning is KTD-Q, an extension of Kalman Temporal Difference (KTD) (Geist and Pietquin 2010). KTD approximates the value function using the Kalman filtering scheme, and handles the non-linearity in the Bellman optimality equation by applying the Unscented Transform. Although the KTD framework is able to integrate some important features in RL, it requires numerous hyperparameters and is difficult to extend to function approximation methods with large numbers of parameters due to its high computational complexity.

Another field of probabilistic approaches to RL is Distributional RL which learns a value distribution or a return density function from the distributional Bellman equation. Recent work (Bellemare, Dabney, and Munos) proposed a gradient-based categorical algorithm using a distributional perspective and showed the state-of-the art performance in several games from the Arcade Learning Environment (ALE). The probabilistic approach to the value function is similar to our approach, however, the value distribution in their work represents the distribution of the random *return* that a learning agent receives, while the Q-belief defined in ADFQ is a belief distribution of a learning agent on a certain state-action pair. As we show in the experiments, we are able to utilize the uncertainty measures in the exploration, while only ϵ-greedy is used in their experiments.

In this paper, we introduce a novel approximate Bayesian off-policy TD learning algorithm, which we denote as ADFQ, that updates beliefs for Q (action value function) and approximates their posteriors using an online Bayesian inference algorithm known as assumed density filtering (ADF). We handle the difficulty in finding a conjugate prior for the Bellman equation using ADF. In order to reduce the computational burden of estimating parameters of the ap-

proximated posterior, we propose a method to analytically estimate the parameters. Unlike Q-learning, the ADFQ update rule considers all possible actions for the next state and returns a soft-max behavior and regularization using the uncertainty measures of the Q-beliefs. This can alleviate the instability of the greedy update discussed by (Harutyunyan et al. 2016; Tsitsiklis 2002). We prove the convergence of ADFQ to the optimal Q-values by showing that ADFQ becomes identical to Q-learning as all state and action pairs are visited infinitely often. In addition, ADFQ has better computational complexity compared with other BRL algorithms.

We first implement the ADFQ algorithm in a small discrete domain and then show how it can be extended to continuous or large discrete state environments using a neural network. There are previous works that implement Bayesian approaches to RL by using uncertainty in the neural network weights(Azizzadenesheli, Brunskill, and Anandkumar 2018; O'Donoghue et al. 2017). Our method differs in that it explicitly computes the variances of the Q-values and can use the them in the ADFQ update rule. In our experiments, ADFQ outperforms not only Q-learning and KTD-Q in tabular settings, but also DQN, and Double DQN ((Hasselt 2010)) in large or continuous domains. Particularly it showed dramatic improvements in a stochastic domain and domains with a large action set.

Background

Assumed Density Filtering

ADF is a general technique for approximating the true posterior with a tractable parametric distribution in Bayesian networks. It has been independently rediscovered for a number of applications and is also known as *moment matching*, *online Bayesian learning*, and *weak marginalization* (Opper 1999; Boyen and Koller 1998; Maybeck 1982). Suppose that a hidden variable $\mathbf{x}$ follows a tractable parametric distribution $p(\mathbf{x}|\theta_t)$ where θ_t is a set of parameters at time t. In the Bayesian framework, the distribution can be updated after observing some new data (D_t) using Bayes' rule, $\hat{p}(\mathbf{x}|\theta_t, D_t) \propto p(D_t|\mathbf{x}, \theta_t)p(\mathbf{x}|\theta_t)$. In online settings, a Bayesian update is typically performed after a new data point is observed, and the updated posterior is then used as a prior for the following iteration.

When the posterior computed by Bayes' rule does not belong to the original parametric family, it can be approximated by a distribution belonging to the parametric family. In ADF, the posterior is projected onto the closest distribution in the family chosen by minimizing the reverse *Kullback-Leibler* divergence denoted as $KL(\hat{p}||p)$ where $\hat{p}$ is the original posterior distribution and p is a distribution in a parametric family of interest. Thus, for online Bayesian filtering, the parameters for the ADF estimate is given by $\theta_{t+1} = \operatorname{argmin}_\theta KL(\hat{p}(\cdot|\theta_t, D_t)||p(\cdot|\theta))$.

Q-learning

RL problems can be formulated in terms of a Markov Decision Process (MDP) described by the tuple, $M = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, respectively, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to I\!R$ is the state transition probability kernel, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to I\!R$ is a reward function, and $\gamma \in [0, 1]$ is a discount factor. The value function is defined as $V^\pi(s) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t(s_t, a_t)|s_0 = s]$ for all $s \in \mathcal{S}$, the expected value of cumulative future rewards starting at a state s and following a policy π thereafter. The state-action value (Q) function is defined as the value for a state-action pair, $Q^\pi(s, a) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t(s_t, a_t)|s_0 = s, a_0 = a]$ for all $s \in \mathcal{S}, a \in \mathcal{A}$. The objective of a learning agent in RL is to find an optimal policy $\pi^* = \operatorname{argmax}_\pi V^\pi$. Finding the optimal values, $V^*(\cdot)$ and $Q^*(\cdot, \cdot)$, requires solving the Bellman optimality equation:

$$Q^*(s, a) = \mathbb{E}_{s' \sim P(\cdot|s,a)}[R(s, a) + \gamma \max_{a' \in \mathcal{A}} Q^*(s', a')] \quad (1)$$

and $V^*(s) = \max_{a \in \mathcal{A}(s)} Q^*(s, a) \quad \forall s \in \mathcal{S}$ where s' is the subsequent state after executing the action a at the state s.

Q-learning is the most popular off-policy TD learning technique due to its relatively easy implementation and guarantee of convergence to an optimal policy (Watkins and Dayan 1992; Kaelbling, Littman, and Moore 1996). Q-learning updates a Q-value of the current state s and action a after observing a reward $R(s, a)$ and the next state s' (one-step TD learning). The update is based on *TD error*—a difference between the *TD target*, $R(s, a) + \gamma \max_b Q(s', b)$, and the current $Q(s, a)$ with a learning rate $\alpha \in [0, 1]$ as shown below:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left(R(s, a) + \gamma \max_b Q(s', b) - Q(s, a) \right)$$

Bayesian Q-learning with ADF

Belief Updates on Q-values

We define $Q_{s,a}$ as a Gaussian random variable with mean $\mu_{s,a}$ and variance $\sigma^2_{s,a}$ corresponding to the action value function $Q(s, a)$ for $s \in \mathcal{S}$ and $a \in \mathcal{A}$. We assume that the random variables for different states and actions are independent and have different means and variances, $Q_{s,a} \sim \mathcal{N}(\mu_{s,a}, \sigma^2_{s,a})$ where $\mu_{s,a} \neq \mu_{s',a'}$ if $s \neq s'$ or $a \neq a'$ $\forall s \in \mathcal{S}, \forall a \in \mathcal{A}$.

According to the Bellman optimality equation in Eq.1, we can define a random variable for $V(s)$ as $V_s = \max_a Q_{s,a}$. In general, the probability density function for the maximum of Gaussian random variables, $M = \max_{1 \leq k \leq N} X_k$ where $X_k \sim \mathcal{N}(\mu_k, \sigma^2_k)$ and $1 \leq k \leq N$, is no longer Gaussian:

$$p\left(M = x\right) = \sum_{i=1}^{N} \frac{1}{\sigma_i} \phi\left(\frac{x - \mu_i}{\sigma_i}\right) \prod_{j \neq i}^{N} \Phi\left(\frac{x - \mu_j}{\sigma_j}\right) \quad (2)$$

where $\phi(\cdot)$ is the standard Gaussian probability density function (PDF) and $\Phi(\cdot)$ is the standard Gaussian cumulative distribution function (CDF) (derivation details are provided in Appendix A).

For one-step Bayesian TD learning, the beliefs on $\mathbf{Q} = \{Q_{s,a}\}_{\forall s \in \mathcal{S}, \forall a \in \mathcal{A}}$ can be updated at time t after observing a reward r_t and the next state s_{t+1} using Bayes rule. In order to reduce notation, we drop the dependency on t denoting $s_t = s$, $a_t = a$, $s_{t+1} = s'$, $r_t = r$, yielding the causally related 4-tuple $\tau =< s, a, r, s' >$. We use the one-step TD

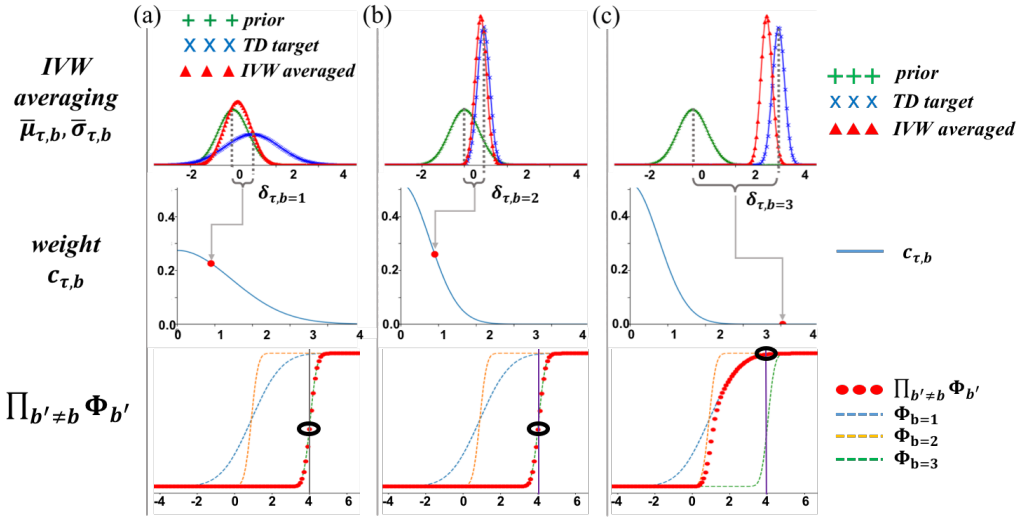

Figure 1: An example of the belief update in Eq.3 when $|\mathcal{A}| = 3, r = 0.0, \gamma = 0.9$ and prior (+ green) has $\mu_{s,a} = 0.0, \sigma^2_{s,a} = 0.5$. Each column corresponds to a subsequent state and action pair, **(a)** $b = 1$: $\mu_{s',b} = 1.0, \sigma^2_b = 2.0$, **(b)** $b = 2$: $\mu_{s',b} = 1.0, \sigma^2_b = 0.1$, **(c)** $b = 3$: $\mu_{s',b} = 4.5, \sigma^2_b = 0.1$. The first row presents how $\bar{\mu}_{\tau,b}$ and $\bar{\sigma}_{\tau,b}$ are determined from prior and a part of the likelihood. The second row shows how $c_{\tau,b}$ (y-axis of the dot) is assigned by TD error ($\delta_{\tau,b}$, x-axis of the dot and uncertainty measures $\sigma^2_{s,a} + \gamma^2 \sigma^2_{s',b}$. The third row shows a softmax-like behavior of the product of CDFs.

target, $r + \gamma V_{s'}$ to give the likelihood, $p(r + \gamma V_{s'}|q, \theta) = p_{V_{s'}}((q - r)/\gamma|s', q, \theta)$ where q is a value corresponding to $Q_{s,a}$ and θ is a set of mean and variance of $\mathbf{Q}$. From the independence assumptions on $\mathbf{Q}$, the posterior update can be reduced to an update for the belief on $Q_{s,a}$:

$$\hat{p}_{Q_{s,a}}(q|\theta, r, s') \propto p_{V_{s'}}\left(\frac{q - r}{\gamma}\middle| q, s', \theta\right) p_{Q_{s,a}}(q|\theta)$$

From the Bellman optimality in Eq.1, $V_{s'}$ follows the distribution presented in Eq.2. The resulting posterior distribution is given as follows (derivation details in Appendix B):

$$\hat{p}_{Q_{s,a}}(q|\theta, r, s') = \frac{1}{Z} \sum_{b \in \mathcal{A}} \frac{c_{\tau,b}}{\bar{\sigma}_{\tau,b}} \phi\left(\frac{q - \bar{\mu}_{\tau,b}}{\bar{\sigma}_{\tau,b}}\right) \prod_{\substack{b' \in \mathcal{A} \\ b' \neq b}} \Phi\left(\frac{q - (r + \gamma \mu_{s',b'})}{\gamma \sigma_{s',b'}}\right) \quad (3)$$

where Z is a normalization constant and

$$c_{\tau,b} = \frac{1}{\sqrt{\sigma^2_{s,a} + \gamma^2 \sigma^2_{s',b}}} \phi\left(\frac{(r + \gamma \mu_{s',b}) - \mu_{s,a}}{\sqrt{\sigma^2_{s,a} + \gamma^2 \sigma^2_{s',b}}}\right) \quad (4)$$

$$\bar{\mu}_{\tau,b} = \bar{\sigma}^2_{\tau,b}\left(\frac{\mu_{s,a}}{\sigma^2_{s,a}} + \frac{r + \gamma \mu_{s',b}}{\gamma^2 \sigma^2_{s',b}}\right) \quad (5)$$

$$\frac{1}{\bar{\sigma}^2_{\tau,b}} = \frac{1}{\sigma^2_{s,a}} + \frac{1}{\gamma^2 \sigma^2_{s',b}} \quad (6)$$

Note that all next actions are considered in Eq.3 unlike the conventional Q-learning update which only considers the subsequent action resulting in the maximum Q-value at the next step ($\max_b Q(s', b)$). This can lead to a more stable update rule as updating with only the maximum Q value has inherent instability (Harutyunyan et al. 2016; Tsitsiklis 2002). The Bayesian update considers the scenario where the true maximum Q-value may not be the one with the highest estimated mean and weights each subsequence Q-value accordingly. For a stochastic MDP, we can add a small noise to the likelihood, $\gamma \sigma_{s',b} + \epsilon$.

Each term for action b inside the summation in Eq.3 has three important features. First of all, $\bar{\mu}_{\tau,b}$ is an inverse-variance weighted (IVW) average of the prior mean and the TD target mean. Therefore, the Gaussian PDF part becomes closer to the TD target distribution if it has a lower uncertainty than the prior, and vice versa as compared in the first row (a) and (b) of Fig.1. Next, the TD error, $\delta_{\tau,b} = (r + \gamma \mu_{s',b}) - \mu_{s,a}$, is naturally incorporated in the posterior distribution with the form of a Gaussian PDF in the weight $c_{\tau,b}$. Thus, a subsequent action which results in a smaller TD error contributes more to the update. The sensitivity of a weight value is determined by the prior and target uncertainties. An example case is described in the second row of Fig.1 where $\delta_{\tau,1} = \delta_{\tau,2} > \delta_{\tau,3}$ and $\sigma_{s',1} > \sigma_{s',2} = \sigma_{s',3}$. Finally, the product of Gaussian CDFs provides for a softmax operation. The red curve with dots in the third row of Fig.1 represents $\prod_{b' \neq b} \Phi(q|r + \gamma \mu_{\tau,b'}, \gamma \sigma_{\tau,b'})$ for each b. For a certain q value (x-axis), the term returns a larger value for a larger $\mu_{s',b}$ as seen in the black circles.

Assumed Density Filtering with Q-Beliefs

The posterior in Eq.3, however, is no longer Gaussian. In order to continue the Bayesian updates, we approximate the posterior with a Gaussian distribution using ADF. We minimize $KL(\hat{p}_{Q_{s,a}}||p)$ with respect to the mean μ and variance σ^2 where $p = \mathcal{N}(\cdot|\mu, \sigma^2)$. When the parametric family is spherical Gaussian, it is shown that $\mu^* = \mathbb{E}_{\mathbf{q} \sim \hat{p}_{Q_{s,a}}(\cdot)}[\mathbf{q}]$ and

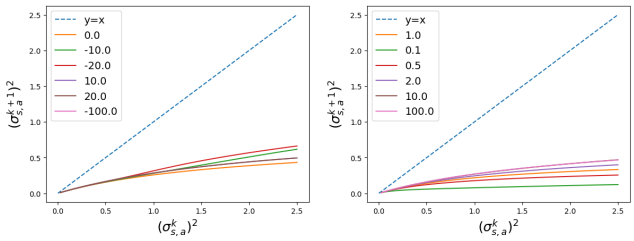

Figure 2: Relationship between $\sigma^2_{s,a}$ and its updated value. Each solid curve represents a different set of parameters. Left: Differing values of $\mu_{s',2} - \mu_{s',1}$. Right: Differing values of $\sigma^2_{s',1}/\sigma^2_{s',2}$

$\sigma^{*2} = \mathrm{Var}_{q\sim\hat{p}_{Q_{s,a}}(\cdot)}[q]$. Therefore, the approximate posterior distribution, $p_{Q_{s,a}}$, will be a Gaussian distribution having the mean and the variance of the true posterior as its mean and variance, respectively.

It is fairly easy to analytically derive the mean and the variance of the true posterior (Eq.3) when $|\mathcal{A}| = 2$. The derivation and the solutions are presented in Appendix C. However, to our knowledge, when $|\mathcal{A}| > 2$, there is no closed-form solution for the ADF parameters. In the next sections, we prove the convergence of the ADF parameters to the optimal Q-values for the case $|\mathcal{A}| = 2$ where we can find the exact mean and variance solutions. Then, we show how to derive an analytic approximation for the mean and variance of the ADF parameters that becomes exact in the small variance limit.

Convergence to Optimal Q-values

The convergence theorem of the Q-learning algorithm has previously been proven (Watkins and Dayan 1992). We, therefore, show that the online Bayesian update using ADF with the posterior in Eq.3 converges to Q-learning for $|A| = 2$. We apply the approximation from Lemma 1 in order to prove Theorem 1. Proofs for Lemma 1 and Theorem 1 are presented in Appendix.

Lemma 1. *Let X be a random variable following a normal distribution, $\mathcal{N}(\mu, \sigma^2)$. Then we have:*

$$\lim_{\sigma\to 0}\left[\Phi\left(\frac{x-\mu}{\sigma}\right) - \exp\left\{-\frac{1}{2}\left[-\frac{x-\mu}{\sigma}\right]_+^2\right\}\right] = 0 \quad (7)$$

where $[x]_+ = \max(0, x)$ is the ReLU *nonlinearity.*

Theorem 1. *Suppose that the mean and the variance of $Q_{s,a}$ $\forall s \in \mathcal{S}, \forall a \in \mathcal{A}$ are iteratively updated by the mean and the variance of $\hat{p}_{Q_{s,a}}$ in Eq.3 after observing r and s' at every step. When $|\mathcal{A}| = 2$, the update rule of the mean $\mu_{s,a}$ is equivalent to the Q-learning update if the variances approach to 0. In other words, at the kth update:*

$$\lim_{\substack{k\to\infty \\ \{\sigma\}\to 0}} \mu^{(k+1)}_{s,a} = (1-\alpha^{(k)}_\tau)\mu^{(k)}_{s,a} + \alpha^{(k)}_\tau(r + \gamma \max_{b\in\mathcal{A}} \mu^{(k)}_{s',b})$$

where $\alpha_{\tau^{(k)}} = {\sigma^{(k)}_{s,a}}^2 / \left({\sigma^{(k)}_{s,a}}^2 + \gamma^2 {\sigma^{(k)}_{s',b^+}}^2\right)$ and $b^+ = \mathrm{argmax}_{b\in\mathcal{A}}\mu_{s',b}$.

Interestingly, α_τ approaches 1 when $\sigma_{s,a}/\sigma_{s',b^*} \to \infty$ and 0 when $\sigma_{s,a}/\sigma_{s',b^*} \to 0$ for a deterministic MDP ($\epsilon = 0$). For a stochastic case, such behavior remains but α_τ eventually approaches 0 as the number of visits to (s, a) grows. This provides a natural learning rate - the smaller the variance of the next state (the higher the confidence), the more $Q_{s,a}$ is updated from the target information rather than the current belief.

Fig. 2 shows empirical evidence that the assumption on variance for Theorem 1 holds. The updated variance is less than the current variance for a large range of different parameters. In addition, it is easily shown that 0 is the fixed point of Eq.7 in Appendix D.

Analytic ADF Parameter Estimates

Numerical computation of $\mu^* = \mathbb{E}_{\mathbf{q}\sim\hat{p}_{Q_{s,a}}(\cdot)}[\mathbf{q}]$ and $\sigma^{*2} = \mathrm{Var}_{q\sim\hat{p}_{Q_{s,a}}(\cdot)}[q]$ using samples becomes unwieldy due to the large number of samples needed for accurate estimates. We may expect that this becomes especially problematic in accurately estimating small variances as the number of visits to corresponding state-action pairs grows. Therefore, in this section, we show how to accurately estimate the ADF parameters using an analytic approximation. This estimate becomes exact for small variances.

Approximation

Using Lemma 1, the true posterior in Eq.3 is approximated as the following distribution:

$$\tilde{p}_{Q_{s,a}}(q) = \frac{1}{Z}\sum_{b\in\mathcal{A}} \frac{c_{\tau,b}}{\sqrt{2\pi}\bar{\sigma}_{\tau,b}} \times \exp\left\{-\frac{(q-\bar{\mu}_{\tau,b})^2}{2\bar{\sigma}^2_{\tau,b}} - \sum_{b'\neq b}\frac{[r+\gamma\mu_{s',b'}-q]_+^2}{2\gamma^2\sigma^2_{s',b'}}\right\} \quad (8)$$

Each term for $b \in \mathcal{A}$ inside the summation can then be approximated by a Gaussian PDF. Similar to Laplace's method, we approximate each term as a Gaussian distribution by matching the maximum values as well as the curvature at the peak of the distribution. In other words, the maximum of the distribution is modeled locally near its peak by the quadratic concave function:

$$-\frac{(q-\bar{\mu}_{\tau,b})^2}{2\bar{\sigma}^2_{\tau,b}} - \sum_{b'\neq b}\frac{[r+\gamma\mu_{s',b}-q]_+^2}{2\gamma^2\sigma^2_{s',b}} \approx -\frac{(q-\mu^*_b)^2}{2{\sigma^*_b}^2} \quad (9)$$

We find μ^*_b and σ^*_b by matching the first and the second derivatives, respectively (the coefficient of the quadratic term gives the local curvature):

$$\frac{\mu^*_b - \bar{\mu}_{\tau,b}}{\bar{\sigma}^2_{\tau,b}} = \sum_{b'\neq b}\frac{[r+\gamma\mu_{s',b'}-\mu^*_b]_+}{\gamma^2\sigma^2_{s',b'}} \quad (10)$$

$$\frac{1}{{\sigma^*_b}^2} = \frac{1}{\bar{\sigma}^2_{\tau,b}} + \sum_{b'\neq b}\frac{H\left(r+\gamma\mu_{s',b'}-\mu^*_b\right)}{\gamma^2\sigma^2_{s',b'}} \quad (11)$$

Table 1: ADFQ algorithm

Algorithm 1: ADFQ

Initialize $\mu_{s,a}$, $\sigma_{s,a}$ $\forall s \in \mathcal{S}$ and $\forall a \in \mathcal{A}$
for each time step t **do**
  $a_t \sim \pi^{action}(s_t; \theta_t)$
  Perform the action and observe r_t and s_{t+1}
  for each $b \in \mathcal{A}$
    Compute μ_b^*, σ_b^*, k_b^* using Eq.11-13
  Update μ_{s_t,a_t} and σ_{s_t,a_t} using Eq.15 and Eq.16

where $H(\cdot)$ is a Heaviside step function. The self-consistent piece-wise linear equation for μ_b^* can be rewritten as Eq.12.

$$\mu_b^* = \left(\frac{1}{\bar{\sigma}^2_{\tau,b}} + \sum_{b' \neq b} \frac{H(r + \gamma\mu_{s',b'} - \mu_b^*)}{\gamma^2 \sigma^2_{s',b'}} \right)^{-1} \times \left(\frac{\bar{\mu}_{\tau,b}}{\bar{\sigma}^2_{\tau,b}} + \sum_{b' \neq b} \frac{(r + \gamma\mu_{s',b'})}{\gamma^2 \sigma^2_{s',b'}} H(r + \gamma\mu_{s',b'} - \mu_b^*) \right) \quad (12)$$

This is an IVW average mean of the prior, the TD target distribution of b, and other TD target distributions whose means are larger than μ_b^*. The height of the peak is computed for $q = \mu_b^*$,

$$k_b^* = \frac{c_{\tau,b}\sigma_b^*}{\bar{\sigma}_{\tau,b}} \exp\left\{ -\frac{(\mu_b^* - \bar{\mu}_{\tau,b})^2}{2\bar{\sigma}^2_{\tau,b}} - \sum_{b' \neq b} \frac{[r + \gamma\mu_{s',b'} - \mu_b^*]_+^2}{2\gamma^2 \sigma^2_{s',b'}} \right\} \quad (13)$$

The final approximated distribution is a Gaussian mixture model with $\mu_b^*, \sigma_b^*, k_b^*$ for all $b \in \mathcal{A}$:

$$\tilde{p}_{Q_{s,a}} = \frac{1}{Z} \sum_{b \in \mathcal{A}} \frac{k_b^*}{\sigma_b^*} \phi\left(\frac{q - \mu_b^*}{\sigma_b^*} \right) \quad (14)$$

Finally, we can update the belief distribution over $Q_{s,a}$ with the mean and variance of the mixture model:

$$\mathbb{E}_{q \sim \tilde{p}(\cdot)}[q] = \frac{\sum_{b \in \mathcal{A}} k_b^* \mu_b^*}{\sum_{b \in \mathcal{A}} k_b^*} \quad (15)$$

$$\mathrm{Var}_{q \sim \tilde{p}(\cdot)}[q] = \frac{\sum_{b \in \mathcal{A}} k_b^* (\mu_b^{*2} + \sigma_b^{*2})}{\sum_{b \in \mathcal{A}} k_b^*} - \left(\mathbb{E}_{q \sim \tilde{p}(\cdot)}[q] \right)^2 \quad (16)$$

k_b^* can be seen as a weight. As shown in Eq.13, it has the TD error penalizing term, $c_{\tau,b}$, but also penalizes how far $\bar{\mu}_{\tau,b}$ is shifted towards larger TD target distributions. Moreover, the remaining terms provides a softened maximum property over b. The final algorithm is summarized in Table.1. The computational complexity of each update of the algorithm is $O(|\mathcal{A}|^2)$. If we numerically compute the mean and variance of the true posterior from samples, the computational complexity becomes $O(m|\mathcal{A}|)$ where m is the number of samples. The space complexity of the algorithm is $O(|\mathcal{S}||\mathcal{A}|)$. As a result, ADFQ is more efficient than KTD-Q where the computational complexity is $O(|\mathcal{S}|^2|\mathcal{A}|^3)$ and the space complexity is $O(|\mathcal{S}|^2|\mathcal{A}|^2)$ in finite state and action spaces.

Convergence of ADFQ

Theorem 1 extends to the ADFQ algorithm. The contraction behavior of the variances in the case of Theorem 1 is also empirically observed in ADFQ (Proof in Appendix D).

Theorem 2. *The ADFQ update on the mean $\mu_{s,a}$ $\forall s \in \mathcal{S}$, $\forall a \in \mathcal{A}$ for $|\mathcal{A}| = 2$ is equivalent to the Q-learning update if the variances approach 0 and if all state-action pairs are visited infinitely often. In other words, we have :*

$$\lim_{\substack{k \to \infty \\ \{\sigma\} \to 0}} \mu_{s,a}^{(k+1)} = (1 - \alpha_\tau^{(k)})\mu_{s,a}^{(k)} + \alpha_\tau^{(k)}(r + \gamma \max_{b \in \mathcal{A}} \mu_{s',b}^{(k)})$$

where $\alpha_{\tau^{(k)}} = {\sigma_{s,a}^{(k)}}^2 / \left({\sigma_{s,a}^{(k)}}^2 + \gamma^2 {\sigma_{s',b^+}^{(k)}}^2 \right)$ *and* $b^+ = \mathrm{argmax}_{b \in \mathcal{A}} \mu_{s',b}$.

As we have observed the behavior of α_τ in Theorem 1, the learning rate α_τ again provides a natural learning rate with the ADFQ update. We can therefore think of Q-learning as a special case of ADFQ.

Experiments in a Discrete MDP

Algorithms and Domain

In addition to ADFQ, we evaluate a numerical approximation of the mean and variance of Eq.3 (denoted as ADFQ-Numeric). For both ADFQ and ADFQ-Numeric, we use two action policies: *Bayesian Sampling (BS)* selects $a_t = \mathrm{argmax}_a\, q_{s_t,a}$ where $q_{s_t,a} \sim p_{Q_{s_t,a}}(\cdot|\theta_t)$, and *$\epsilon$-greedy* selects a random action with ϵ probability and selects the action with the highest mean otherwise. In implementation, we fixed the initial variance to 100.0 and the variances are bounded by a small value $\in [10^{-5}, 10^{-20}]$ since variance dramatically drops and it eventually exceeds the precision range of computers. For a stochastic case, we used a noise ($\epsilon = 0.001$) and experience replay (Lin 1993) with a batch size of 30. For comparison, we test Q-learning with ϵ-greedy and Boltzmann action policies. The learning rate decreases as the number of visits to a state-action pair increases starting from 0.5 (Lagoudakis and Parr 2003). KTD-Q with ϵ-greedy and its active learning scheme are also examined. The same hyperparameter values as the ones in the original paper are used if presented. All other hyperparameters are selected through cross-validation (presented in Appendix E).

We test our algorithms in Maze ($\gamma = 0.95$, Figure 3) from (Dearden, Friedman, and Russell 1998) with/without stochasticity in finite learning steps ($T_H = 30000$). Since the KTD-Q algorithm was not able to handle a large discrete state space in reasonable time due to its high computational complexity, we reduced the state space to $|\mathcal{S}| = 112$. The agent's goal is to collect the flags "F" and escape the maze through the goal position "G" starting from "S". It receives

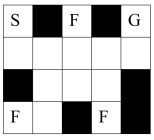

Figure 3: Maze Domain

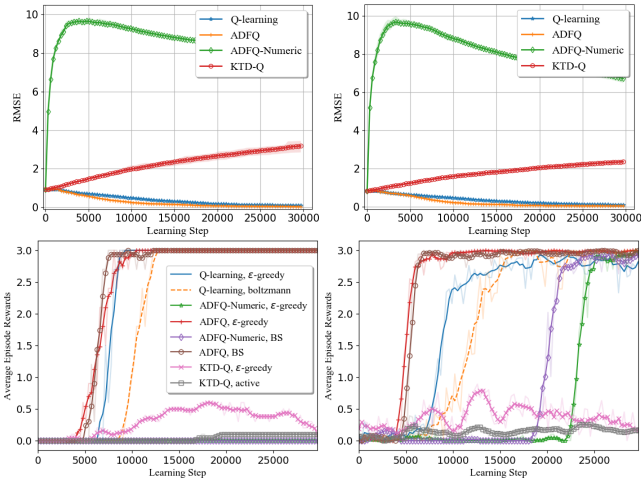

Figure 4: Top: Root Mean Square Error (RMSE) of Q or μ from the optimal Q-values in Maze (left: deterministic, right: stochastic). Bottom: Greedy evaluation during learning smoothed by a moving average with window 4 (left: deterministic, right: stochastic)

a reward equivalent to the number of flags it has collected at "G". The agent remains at the current state if it performs an action toward a wall (black block). For a stochastic case, the agent slips with a probability 0.1 and moves to the right perpendicular direction.

Results

We first examined the convergence to the optimal Q-values using randomly generated fixed trajectories $< s_0, a_0, r_0, s_1, \cdots >$ for all algorithms in order to evaluate only the update part of each algorithm. During learning, we computed the root mean square error (RMSE) between the estimated Q-values (or means) and the true optimal Q-values, and plotted the averaged results over 10 trials in Fig .4. Next, we evaluated the performance of each algorithm with different action policies during learning. At every 300 steps, the current policy was greedily evaluated where the maximum number of steps was bounded by 1.5 times of the optimal path length or it was terminated when the goal was reached. The entire experiment was repeated 10 times and the results were averaged.

As shown in Fig.4, ADFQ converged to the optimal Q-values quicker than all other algorithms including Q-learning. In addition, ADFQ with ϵ-greedy and ADFQ with *BS* showed similar results and converged to the optimal performance (3.0) faster than the comparing algorithms in both deterministic and stochastic cases. Q-learning with ϵ-greedy learned an optimal policy almost as fast as ADFQ in the deterministic case, but the performance of ADFQ was improved dramatically in the stochastic case. KTD-Q diverged and performed poorly since its derivative-free approximation nature does not scale well with the number of parameters. In Appendix F, we show good performance of KTD-Q and its converging behavior in a small domain.

ADFQ-Numeric initially resulted in a large jump in RMSE and learned very little in the deterministic domain. This can be explained by the fact that the mean of the maximum of Gaussian random variables is equal to or larger than the maximum of means of Gaussian random variables (i.e. $\mathbb{E}[M = \max_{i=1\cdots N} X_i] \geq \max_{i=1\cdots N} \mathbb{E}[X_i]$). While this can speed up learning in a certain type of domain, it impedes learning in the Maze domain. When the agent performs an upward action, the agent receives no reward and remains at the current position. In this step, ADFQ-Numeric increases the Q-value while Q-learning and ADFQ decreases it. ADFQ reduces this amount through the small-variance approximation. In the stochastic case, ADFQ-numeric reaches the optimal performance after a large number of time steps. This shows the validity of the ADF update, though performance is hindered by the aforementioned problem.

ADFQ with Neural Networks

In this section, we extend our algorithm to a continuous or large state space environment with neural networks similar to Deep Q-Networks (DQN) proposed in (Mnih et al. 2013; 2015). In the Deep ADFQ model with network parameters θ, the output of the network is mean $\mu(s, a; \theta)$ and variance $\sigma^2(s, a; \theta)$ of each action for a given state s as shown in Fig.5. In practice, we use $-\log(\sigma_{s,a})$ instead of $\sigma^2_{s,a}$ for the output in order to ensure positive values for the variance. As in DQN, we have a train network(θ) and a target network(θ'). Mean and variance values for s and s' from the target network are used as inputs into the ADFQ algorithm to compute the desired mean, μ^{ADFQ}, and standard deviation, σ^{ADFQ} for the train network. We used experience replay (prioritized (Schaul et al. 2015) for Atari games) and a combined Huber loss functions of mean and variance.

In order to demonstrate the effectiveness of our algorithm, we tested on continuous state domains, CartPole and Acrobot, and on Atari games, Breakout($|\mathcal{A}| = 4$), Pong($|\mathcal{A}| = 6$), Asterix($|\mathcal{A}| = 9$), Enduro($|\mathcal{A}| = 9$) from the OpenAI gym simulator (Brockman et al. 2016). For baselines, we used DQN and Double DQN (DDQN) with experience replay (prioritized for Atari games) implemented in OpenAI baselines (Dhariwal et al. 2017) with their default hyperparameters except for setting $\gamma = 0.99$ for all tasks. We used ϵ-greedy action policy with ϵ annealed from 1.0 to 0.01 (0.02 for CartPole and Acrobot) for the baselines as well as ADFQ. Additionally, we used *Bayesian Sampling (BS)* for ADFQ action policy. Further details on the network archi-

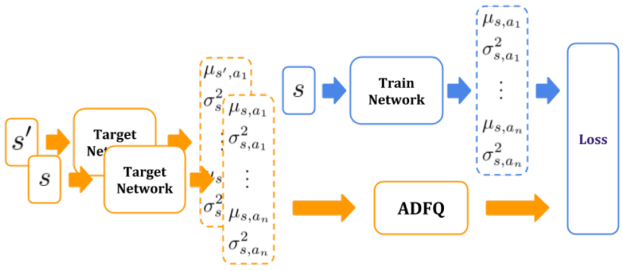

Figure 5: A neural network model for ADFQ

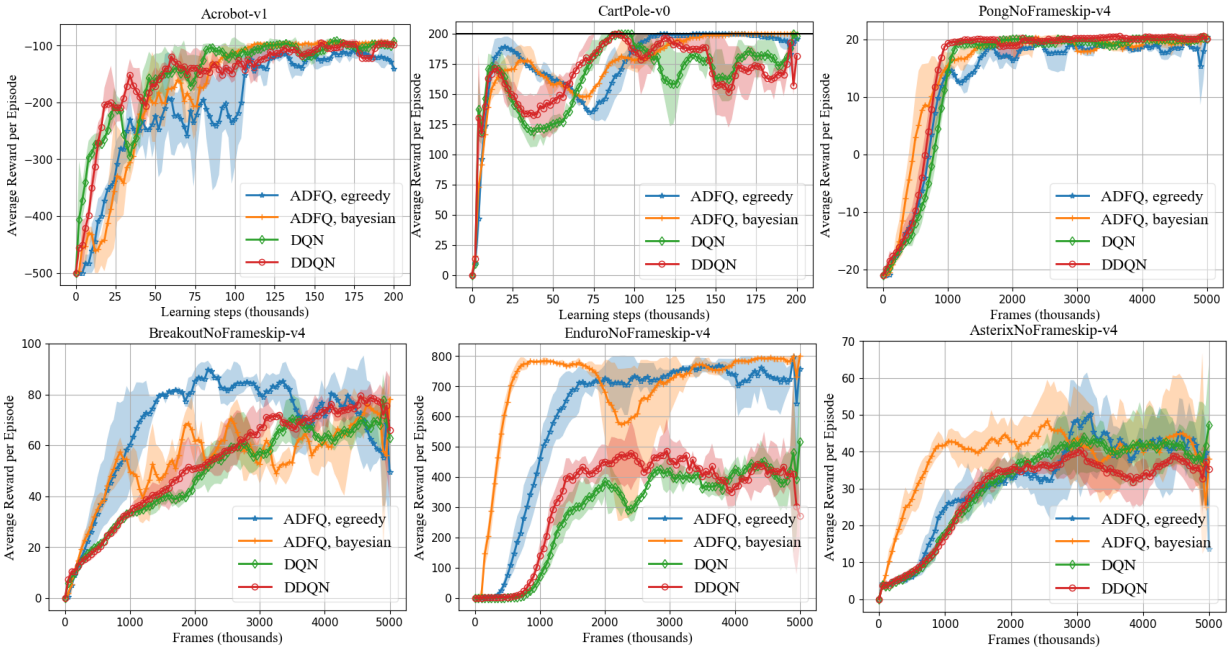

Figure 6: Performance of ADFQ, DQN, and Double DQN (DDQN) during learning smoothed by a moving average with window 4. Shaded areas denote inter-quartile range. From the top left to the bottom right : Cartpole, Acrobot, Breakout, Pong, Asterix, Enduro

tecture and hyperparameters are provided in Appendix G.

The algorithms were evaluated for 200K training steps in the continuous domains and for 5M frames (1.25M training steps) in the Atari games. Similar to the previous evaluation, each learning was greedily evaluated at every epoch for 5 times bounded by 10K steps in each trial, and their averaged results are presented in Fig.6. The entire experiment was repeated for 5 and 3 random seeds for the continuous domains and the Atari games, respectively. Rewards were normalized to $(-1, 0, 1)$ and different from raw scores of the games.

In all Atari games, ADFQ with BS showed dramatic increases in its performance at the beginning. It also notably surpassed the other algorithms in the domains where accounting for uncertainty in exploration is more advantageous (the large number of actions). Particularly, in Enduro, ADFQ with Bayesian sampling achieved the near optimal performance within 1M frames with a raw score of up to 7,181 (a video is attached)! This is very impressive compared to the raw scores of the other state-of-the art results after 200M training frames (Categorical DQN: 3,454, Prioritized Dueling Architecture: 2,306.4 (Bellemare, Dabney, and Munos)). In addition, the variance estimates may help in the initial learning stages as it can trust certain state action pairs heavily and update aggressively towards them.

Discussion

We proposed an approach to Bayesian off-policy TD method called ADFQ. ADFQ surpassed the performance of Q-learning and KTD-Q in a small finite domain, and outperformed DQN and Double DQN in various continuous and large discrete domains. The presented ADFQ algorithm demonstrates several intriguing results.

Non-greedy Update. Unlike the conventional Q-learning algorithm, ADFQ incorporates the information of all possible actions for the subsequent state in the update with weights depending on TD errors and uncertainty measures.

Regularization with uncertainty. ADFQ provides an intuitive update - a state-action pair with higher uncertainty in its Q belief has a smaller weight contributing less to the update. Therefore, we make use of our uncertainty measures not only in exploration but also in the value update with natural regularization based on the current beliefs.

Convergence to Q-learning. We prove that ADFQ converges to Q-learning as the variances decrease and can be seen as a more general form of Q-learning.

Improved drawbacks of BRL. One of the major drawbacks of BRL approaches is their higher computational complexity than standard RL algorithms (Ghavamzadeh et al. 2015). ADFQ is computationally more efficient than KTD-Q and requires only two hyperparameters to be chosen.

Scalability ADFQ is extended to Deep ADFQ with a neural network, and with the Bayesian sampling, it demonstrates that it makes use of the uncertainty information in exploration especially when the number of available actions is large and reasonable exploration is required.

We would like to highlight the fact that ADFQ is the Bayesian counterpart of Q-learning and is orthogonal to most other advancements made in Deep RL. ADFQ merely changes the loss function and we compare with basic architectures here to provide insight as to how it may improve the performance. ADFQ can be used in conjunction with other extensions and techniques such as Double DQN, multistep returns, and Dueling Architecture (Wang et al. 2015).

References

Azizzadenesheli, K.; Brunskill, E.; and Anandkumar, A. 2018. Efficient exploration through bayesian deep q-networks. *arXiv preprint arXiv:1802.04412*.

Bellemare, M. G.; Dabney, W.; and Munos, R. A distributional perspective on reinforcement learning.

Boyen, X., and Koller, D. 1998. Tractable inference for complex stochastic processes. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. Openai gym.

Chowdhary, G.; Liu, M.; Grande, R.; Walsh, T.; How, J.; and Carin, L. 2014. Off-policy reinforcement learning with gaussian process. *IEEE/CAA Journal of Automatica Sinica* 1(3):227–238.

Dearden, R.; Friedman, N.; and Andre, D. 1999. Model based bayesian exploration. In *Proceedings of the 15th conference on Uncertainty in artificial intelligence*, 150–159. Morgan Kaufmann Publishers Inc.

Dearden, R.; Friedman, N.; and Russell, S. 1998. Bayesian q-learning. In *AAAI/IAAI*, 761–768.

Dhariwal, P.; Hesse, C.; Klimov, O.; Nichol, A.; Plappert, M.; Radford, A.; Schulman, J.; Sidor, S.; and Wu, Y. 2017. Openai baselines. `https://github.com/openai/baselines`.

Duff, M. 2002. Optimal learning: Computational procedures for bayes-adaptive markov decision processes. *PhD thesis, University of Massachusetts, Amherst*.

Engel, Y.; Mannor, S.; and Meir, R. 2003. Bayes meets bellman: The gaussian process approach to temporal difference learning. In *Proceedings of the 20th International Conference on Machine Learning*, volume 20.

Engel, Y.; Mannor, S.; and Meir, R. 2005. Reinforcement learning with gaussian processes. In *Proceedings of the 22nd International Conference on Machine Learning*, 201–208.

Geist, M., and Pietquin, O. 2010. Kalman temporal differences. *Journal of artificial intelligence research* 39:483–532.

Ghavamzadeh, M., and Engel, Y. 2006. Bayesian policy gradient algorithm. In *Advances in Neural Information Processing Systems (NIPS)*, 457–464.

Ghavamzadeh, M.; Mannor, S.; Pineau, J.; and Tamar, A. 2015. Bayesian reinforcement learning: A survey. *Foundation and Trends in Machine Learning* 8(5-6):359–483.

Guez, A.; Silver, D.; and Dayan, P. 2012. Efficient bayes-adaptive reinforcement learning using sample-based search. In *Advances in Neural Information Processing Systems (NIPS)*, 1071–1079.

Harutyunyan, A.; Bellemare, M. G.; Stepleton, T.; and Munos, R. 2016. Q (λ) with off-policy corrections. In *International Conference on Algorithmic Learning Theory*, 305–320. Springer.

Hasselt, H. V. 2010. Double q-learning. In *Advances in Neural Information Processing Systems (NIPS)*.

Kaelbling, L. P.; Littman, M. L.; and Moore, A. W. 1996. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research* 4:237–285.

Lagoudakis, M. G., and Parr, R. 2003. Least-squares policy iteration. *Journal of machine learning research* 4(Dec):1107–1149.

Lin, L.-J. 1993. Reinforcement learning for robots using neural networks. Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science.

Maybeck, P. S. 1982. Stochastic models, estimation and control. *Academic Press* chapter 12.7.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS) Deep Learning Workshop*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7549):529–533.

O'Donoghue, B.; Osband, I.; Munos, R.; and Mnih, V. 2017. The uncertainty bellman equation and exploration. *arXiv preprint arXiv:1709.05380*.

Opper, M. 1999. A bayesian approach to online learning. *On-Line Learning in Neural Networks*.

Poupart, P.; Vlassis, N.; Hoey, J.; and Regan, K. 2006. An analytic solution to discrete bayesian reinforcement learning. In *Proceedings of the 23rd International Conference on Machine Learning*, volume 20, 697–704.

Schaul, T.; Quan, J.; Antonoglou, I.; and Silver, D. 2015. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

Strens, M. 2000. A bayesian framework for reinforcement learning. In *Proceedings of the 17th International Conference on Machine Learning*, 943–950.

Tsitsiklis, J. N. 2002. On the convergence of optimistic policy iteration. *Journal of Machine Learning Research* 3(Jul):59–72.

Wang, Z.; Schaul, T.; Hessel, M.; Van Hasselt, H.; Lanctot, M.; and De Freitas, N. 2015. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*.

Watkins, C. J., and Dayan, P. 1992. Q-learning. In *Machine Learning*, 279–292.