Journal of Machine Learning Research 22 (2021) 1-51 Submitted 6/20; Revised 10/21; Published 10/21

Gaussian Approximation for Bias Reduction in Q-Learning

Carlo D'Eramo[1*] carlo.deramo@tu-darmstadt.de
Andrea Cini[2,3*] andrea.cini@usi.ch
Alessandro Nuara[4] alessandro.nuara@polimi.it
Matteo Pirotta[5] pirotta@fb.com
Cesare Alippi[2,3,4] cesare.alippi@usi.ch
Jan Peters[1] mail@jan-peters.net
Marcello Restelli[4] marcello.restelli@polimi.it
[1] *TU Darmstadt, Darmstadt, Germany*
[2] *The Swiss AI Lab IDSIA, Lugano, Switzerland*
[3] *Università della Svizzera italiana, Lugano, Switzerland*
[4] *Politecnico di Milano, Milano, Italy*
[5] *Facebook AI Research*
$*$ *Indicates equal contribution.*

Editor: Laurent Orseau

Abstract

Temporal-Difference off-policy algorithms are among the building blocks of reinforcement learning (RL). Within this family, Q-Learning is arguably the most famous one, which has been widely studied and extended. The update rule of Q-learning involves the use of the maximum operator to estimate the maximum expected value of the return. However, this estimate is positively biased, and may hinder the learning process, especially in stochastic environments and when function approximation is used. We introduce the *Weighted Estimator* as an effective solution to mitigate the negative effects of overestimation in Q-Learning. The Weighted Estimator estimates the maximum expected value as a weighted sum of the action values, with the weights being the probabilities that each action value is the maximum. In this work, we study the problem from the statistical perspective of estimating the maximum expected value of a set of random variables and provide bounds to the bias and the variance of the Weighted Estimator, showing its advantages over other estimators present in literature. Then, we derive algorithms to enable the use of the Weighted Estimator, in place of the Maximum Estimator, in online and batch RL, and we introduce a novel algorithm for deep RL. Finally, we empirically evaluate our algorithms in a large set of heterogeneous problems, encompassing discrete and continuous, low and high dimensional, deterministic and stochastic environments. Experimental results show the effectiveness of the Weighted Estimator in controlling the bias of the estimate, resulting in better performance than representative baselines and robust learning w.r.t. a large set of diverse environments.

Keywords: Reinforcement Learning, Q-Learning, Bias reduction

1. Introduction

Reinforcement Learning (RL) aims at learning how to make optimal decisions in unknown environments by solving credit assignment problems that extend over time. In order to be sample-efficient learners, agents are required to constantly update their own beliefs about the environment and which actions are good and which are not. Temporal difference (TD) (Sutton and Barto, 1998) and off-policy learning are the core elements of this kind of behavior, for which Q-Learning (Watkins, 1989) is one of the most representative and well-studied algorithms. Q-Learning-based algorithms have been successful in a wide variety of problems and, in recent years, they are driving RL research towards solving complex problems, achieving super-human performance on many of them, e.g., deep Q-Learning (Mnih et al., 2015; Badia et al., 2020). Nonetheless, the Maximum Estimator (ME) used in Q-Learning to estimate the maximum expected value (MEV) of the action-values, is positively biased (Van Hasselt, 2010). The resulting overoptimism can be particularly harmful in stochastic environments and when using function approximation (Thrun and Schwartz, 1993), especially in the case where the approximators are deep neural networks (Van Hasselt et al., 2016). Systematic overestimation of the action-values, coupled with the inherently high variance of stochastic problems, can lead to incremental accumulations of errors which slow down the convergence of the learning algorithm or, more critically, cause the algorithm to diverge when function approximation is used.

The Double Q-Learning algorithm (Van Hasselt, 2010) and its extension to deep RL, Double DQN (Van Hasselt et al., 2016), tackle the overestimation problem by replacing ME with the Double Estimator (DE) (Van Hasselt, 2013), an estimator that disentangles the choice of the target action and its evaluation. This estimate is negatively biased (Van Hasselt, 2013), resulting in an underestimation that can lead to better performance in some environments due to its robustness to stochastic returns and errors introduced by function approximation, compared to the standard Q-Learning. More recently, Lan et al. (2020) introduced the Maxmin Estimator (MME) as a novel solution to mitigate the overestimation bias, which uses an ensemble of function approximators to estimate the MEV. Overoptimism, in general, is not uniform over the state space and may induce to overestimate the value of arbitrarily bad actions. The same applies, symmetrically, to overly pessimistic estimates that might undervalue a good course of action. Ideally, we would like to exploit the uncertainty about the optimality of each action to make informed estimations of expected returns. This is exactly what we achieve in this work.

More specifically, we propose the *Weighted Estimator* (WE), which consists of a weighted average of the action values, where the weights are obtained by estimating the probability that each action value is the largest one (D'Eramo et al., 2016). Computing such probabilities requires knowledge of the distribution of the action values, notably unknown. By relying on the central limit theorem, we approximate the distributions of the action values with Gaussian distributions parameterized by the sample mean and the sample variance. We study the theoretical properties of WE showing that its bias can be either positive or negative, but it is always between the bias of ME, DE, and MME. Then, we derive a variant of Q-Learning, and its offline variant Fitted Q-Iteration (FQI) (Ernst et al., 2005), replacing ME with WE, leading respectively to the *Weighted Q-Learning* (D'Eramo et al., 2016) and *Weighted Fitted Q-Iteration* (D'Eramo et al., 2017) algorithms. Furthermore, in this work,

we propose a novel method enabling the use of WE in deep RL. In particular, we exploit recent developments in Bayesian deep learning to model the uncertainty of deep RL agents using neural networks trained with dropout variational inference (Kingma et al., 2015; Gal and Ghahramani, 2016; Gal et al., 2017). We combine, in a novel way, the dropout uncertainty estimates with the Weighted Q-Learning algorithm, extending it to the deep RL setting. The proposed *Deep Weighted Q-Learning* algorithm, or *Weighted DQN* (WDQN), leverages on an approximated posterior distribution on Q-networks to reduce the bias of deep Q-Learning. WDQN requires only minor modifications to the baseline algorithm and its computational overhead is negligible on modern GPUs. WDQN extends the approach to the large and expressive family of value functions parameterized with deep neural networks, able to cope with the complexity and high dimensionality of real-world problems. Overall, the framework for bias reduction in Q-Learning proposed in (D'Eramo et al., 2016, 2017) is theoretically grounded and general, and can be straightforwardly used in classic RL algorithms. WDQN completes the picture adding scalability. Moreover, the use of WE is orthogonal to a wide range of (deep) Q-Learning extensions. Empirically, we show that WE reduces the bias of Q-Learning and improves performance on a wide variety of tasks from simple grid worlds to Atari games.

The rest of the paper is organized as follows. In the next section, we formally describe the notions and mathematical notation of the RL setting we consider: we adopt an estimation theory point of view to study the maximum action value estimation problem. In Section 3, we introduce WE for both discrete and continuous action spaces. Section 4 studies the theoretical properties of WE in terms of bias and variance, comparing them with the ME, DE, and MME ones, and assesses its computational complexity. In Section 5, we introduce WE-based algorithms for online, offline, and deep RL. Section 6 is dedicated to the extensive evaluation of our RL methodologies, where we show and discuss the empirical evidence of the benefit of WE w.r.t. ME, DE, and MME. Section 7 provides additional references to the state of the art and Section 8 draws the conclusions of this work.

2. Preliminaries

2.1 Markov Decision Processes

Reinforcement Learning (RL) aims at solving decision problems formalized as Markov Decision Processes (MDPs) and defined as a 5-tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ is a Markovian transition model with $\mathcal{P}(s'|s,a)$ being the probability density of reaching state s' when taking action a in state s, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function and $\gamma \in [0,1)$ is the discount factor of future rewards. A policy π defines a distribution on the action space $\pi : \mathcal{S} \times \mathcal{A} \to [0,1]$ for each state. Following a policy π, the Q-value of an action a in a state s is the expected discounted cumulative reward that is obtained performing action a in state s and following the policy π thereafter. The Q-function maps each state-action pair to the respective Q-value and can be written as $Q^\pi(s,a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s, a_0 = a \right]$, where r_{t+1} is the reward received after the t-th transition. An *optimal* policy π^* is one that maximizes the expected discounted cumulative reward. Optimal policies can be found by learning the optimal action-value function Q^*, i.e., the Q-function of the optimal policy. The optimal action-value function satisfies the Bellman optimality equation (Bellman, 1954):

$$Q^*(s,a) = \mathbb{E}\left[r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1},a') \middle| s_t = s, a_t = a, s_{t+1} = s'\right]$$
$$= \int_{\mathcal{S}} \mathcal{P}(s'|s,a)\left(\mathcal{R}(s,a,s') + \gamma \max_{a'} Q^*\left(s',a'\right)\right) \mathrm{d}s'. \quad (1)$$

For the sake of clarity, in our notation, we will omit the timestep index t when it is obvious from context. The estimation of the *max* operator in Equation 1 has a critical role in RL, and in particular in the well-known Q-Learning algorithm (Watkins, 1989), which is the object of the study conducted in this paper. In the following, we approach the maximum expected value estimation problem from a statistical point of view, and then we focus on the specific case of Q-Learning.

2.2 Estimating the maximum action value

Given a finite action space $\mathcal{A}$ of $M \geq 2$ actions, we define a set of M independent random variables $R^s = \{R^s_{a_1}, \ldots, R^s_{a_M}\} = \{R^s_1, \ldots, R^s_M\}$, each referring to the stochastic return obtained by executing a particular action a_i in state s, and following policy π thereafter, such that $Q^\pi(s,a) = \mathbb{E}_\pi\left[R^s_a | s, a\right]$. Consequently, the optimal action-value function (1) can be rewritten as

$$Q^*(s,a) = \mathbb{E}\left[r + \gamma \max_{a'} \mathbb{E}_{\pi^*}\left[R^{s'}_{a'}\right] | s, a, s'\right]. \quad (2)$$

For ease of notation, in the following we omit the state s in superscripts when the information about the state is not relevant and we use interchangeably as subscripts actions a_i and their indexes i. For each variable R_a we denote with $f_a : \mathbb{R} \to \mathbb{R}_{\geq 0}$ its probability density function (PDF), with $F_a : \mathbb{R} \to [0,1]$ its cumulative density function (CDF), with q_a its mean, and with σ^2_a its variance. We adopt a statistical of point view to examine the crucial problem of estimating the maximum action value in Equation (2), by framing it as the maximum expected value (MEV) estimation problem, defined as

$$q_*(R) = \max_a q_a = \max_a \int_{-\infty}^{+\infty} x f_a(x)\, \mathrm{d}x. \quad (3)$$

Unfortunately, $q_*(R)$ cannot be found analytically if PDFs are unknown. However, $q_*(R)$ can be estimated by using a set samples $\Gamma = \{\Gamma_1, ..., \Gamma_M\}$ from the unknown distributions of each R_a to build an estimator $\hat{q}_*(\Gamma)$. Given the sample means $\hat{q}_1(\Gamma), ..., \hat{q}_M(\Gamma)$, unbiased estimators of the true means $q_1, ..., q_M$, we denote the PDF and CDF of $\hat{q}_a(\Gamma)$ by $\hat{f}^\Gamma_a$ and $\hat{F}^\Gamma_a$. In general, we indicate with the superscript Γ that the superscripted quantity should be evaluated over the sampling of Γ (i.e., Γ itself should be considered as a random variable). Finally, it should be noted that action value estimates are generally not independent in sequential problems, nonetheless the independence assumption is useful to analyze the properties of estimators.

2.2.1 Maximum Estimator

Several methods to estimate the MEV have been proposed in the literature. The most straightforward one is the *Maximum Estimator* (ME) which approximates the MEV with

the maximum of the sample means

$$\hat{q}_*^{ME}(\Gamma) = \max_a \hat{q}_a(\Gamma) \approx q_*(R), \tag{4}$$

where the symbol $\approx$ means that $\hat{q}_*^{ME}(\Gamma)$ should be intended as an estimate of $q_*(R)$. However, the expected value of the ME is different from the MEV in (3). Consider the CDF $\hat{F}_{\max}(x)$ of ME, corresponding to the probability that ME is less than or equal to x. This probability is equal to the probability that all other estimates are less than or equal to x:

$$\hat{F}_{\max}(x) = P(\max_a \hat{q}_a \leq x) = \prod_{i=1}^{M} P(\hat{q}_i \leq x) = \prod_{i=1}^{M} \hat{F}_i(x).$$

Considering the PDF $\hat{f}_{\max}$, the expected value of the ME is $\mathbb{E}\left[\hat{q}_*^{ME}\right] = \mathbb{E}[\max_a \hat{q}_a] = \int_{-\infty}^{\infty} x\hat{f}_{\max}(x)dx$. This is equal to

$$\mathbb{E}\left[\hat{q}_*^{ME}\right] = \int_{-\infty}^{\infty} x \frac{d}{dx} \prod_{j=1}^{M} \hat{F}_j(x)\ \mathrm{d}x = \sum_{i}^{M} \int_{-\infty}^{\infty} x\hat{f}_i(x) \prod_{i\neq j}^{M} \hat{F}_j(x)\ \mathrm{d}x.$$

The presence of x in the integral correlates with the monotonically increasing product $\prod_{i\neq j}^{M} \hat{F}_j(x)$, making the estimator positively biased (Van Hasselt, 2010; Smith and Winkler, 2006).

2.2.2 Double Estimator

To solve the overestimation problem in RL, a method called *Double Estimator* (DE) was proposed in (Van Hasselt, 2010). DE uses a sample set Γ drawn from the true unknown distribution as in ME, and splits it into two disjoint subsets $\Gamma^A = \{\Gamma_1^A, ..., \Gamma_M^A\}$ and $\Gamma^B = \{\Gamma_1^B, ..., \Gamma_M^B\}$. If sets are populated by uniformly drawing samples from Γ without replacement the sample means $\hat{q}_a(\Gamma^A)$ and $\hat{q}_a(\Gamma^B)$ are unbiased estimates of the corresponding true mean q_a. One of the two sets is used as a validation set to select a candidate a^*, such that $\hat{q}_{a^*}(\Gamma^A) = \max_a \hat{q}_a(\Gamma^A)$, and the other set is used to estimate the corresponding expected value, i.e., the DE estimate of the MEV:

$$\hat{q}_*^{DE}(\Gamma) = \hat{q}_{a^*}(\Gamma^B) \approx q_*(R). \tag{5}$$

Obviously, this can be done all the way around by flipping the two datasets, obtaining a second estimate, hence obtaining a second estimate. The 2-fold *Cross Validation* (CV) estimator averages of the two estimates (Van Hasselt, 2013)[1]. Like ME, the expected value of DE can be found as

$$\mathbb{E}\left[\hat{q}_*^{DE}\right] = \sum_{i=1}^{M} P\left(a_i = a^*\right) \mathbb{E}\left[\hat{q}_i(\Gamma^B)\right] = \sum_{i=0}^{M} \mathbb{E}\left[\hat{q}_i(\Gamma^B)\right] \int_{-\infty}^{\infty} \hat{f}_i^A(x) \prod_{j\neq i}^{M} \hat{F}_j^A(x)\ \mathrm{d}x. \tag{6}$$

1. To keep the sets Γ^A and Γ^B disjoint, the CV estimator cannot be used in the Q-learning update rule, but only in the policy (Van Hasselt, 2010). To achieve a fair comparison, however, we use the CV estimator in place of DE whenever possible (e.g., bandits and single-step problems).

Note that if $|\Gamma^A| = |\Gamma^B| = |\Gamma|/2$, the expected value of the 2-fold CV estimator and DE (and consequently their bias) are the same (while DE has double the variance of the CV estimator). This formula can be seen as a weighted sum of the expected values of random variables in which the weights are the probabilities of each variable being the maximum given the set of observed samples. Since these probabilities sum up to one, the approximation given by DE results in a value less than or equal to the MEV.

2.2.3 Maxmin Estimator

Lan et al. (2020) introduced the Maxmin Q-Learning as a flexible algorithm to mitigate the overestimation bias using a new estimator, here referred to as Maxmin Estimator (MME). MME randomly splits the sample set Γ in N disjoint and equally sized subsets $\Gamma^1, \dots, \Gamma^N$, such that each of the N estimators $\hat{q}_a(\Gamma^j)$ is an unbiased estimator of q_a. Then, MME selects for each sample mean the minimum among the N estimates, and finally estimates the MEV by taking the maximum over actions:

$$\hat{q}_*^{MME} = \max_a \min_j \hat{q}_a(\Gamma^j) \approx q_*(R). \tag{7}$$

The intuitive idea behind the effectiveness of MME is that the negative bias introduced by the use of the min operator balances the positively polarized bias introduced by the max. Unfortunately, striking a balance depends on the number of actions M, the number of estimators N, and the actual distribution of the random variables in each state. Lan et al. (2020) analyze the bias of MME under the assumption that the estimators (the sample means) are uniformly distributed. The expected value of MME can be written as

$$\mathbb{E}\left[\hat{q}_*^{MME}\right] = \sum_{i=1}^{M} \sum_{j=1}^{N} P\left(i = \arg\max_i \min_j \hat{q}_i(\Gamma^j)\right) \mathbb{E}\left[\hat{q}_i(\Gamma^j)\right]. \tag{8}$$

2.3 Q-Learning and the bias of action-value estimates

A classic algorithm to solve finite MDPs is Q-Learning (Watkins, 1989), an off-policy algorithm that updates the action-value function Q at each step as

$$Q_{t+1}(s_t, a_t) \leftarrow Q_t(s_t, a_t) + \alpha_t(s_t, a_t)\left(r_t + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t)\right), \tag{9}$$

where $\alpha_t(s_t, a_t)$ is a learning rate, γ is a discount factor and r_t is the immediate reward obtained by taking action a_t in state s_t. It is possible to show that since Q-learning is a stochastic approximation algorithm, assuming that each state-action pair is visited infinitely often and the step sequence satisfies the Robbins-Monro conditions, Q_k converges to Q^* (Watkins, 1989). However, under particular conditions, such as wrong tuning of hyperparameters and highly stochastic environments, the convergence speed of the Q-function estimates to Q^* can be impractically slow. One of the main reasons is that Q-learning uses the ME to estimate the maximum Q-value of the next state s_{t+1}. Since this estimator is positively biased and the error is propagated at each step, wrong estimates of the Q-function may cause the algorithm to never converge in practice.

In recent years, different approaches have been proposed trying to overcome this issue (Lee et al., 2013; Bellemare et al., 2016; Zhang et al., 2017; Lan et al., 2020). In

particular, the most commonly used one is the Double Q-Learning algorithm (Van Hasselt, 2010) which replaces the ME used in Q-Learning with DE. The underestimation of the Q-function performed by Double Q-Learning allows learning a good policy in environments where Q-Learning fails (e.g., highly stochastic environments with a long or infinite time horizon or large action spaces). On the other hand, as shown by D'Eramo et al. (2016) and Lan et al. (2020), the overestimation bias of Q-Learning is not always harmful and it may even be convenient when the action values are significantly different (e.g., deterministic environments with a short time horizon or with small action spaces). In fact, depending on the problem, both algorithms have properties that can be detrimental to learning and it is not always clear which estimator should be preferred. Maxmin Q-Learning, conversely, allows the designer to control the direction of the bias by tuning the number of estimators. However the distribution of the action values might drastically change over the state-space, making the selection of the optimal configuration an hard task.

Given the above, we argue that the study of an estimator whose bias is not always polarized in the same direction across the state-space, regardless of the hyperparameters, is in order.

3. Weighted Estimator

We propose the *Weighted Estimator* (WE), to estimate the maximum expected value (MEV) $q_*(R)$ computing a weighted mean of all the sample averages

$$\hat{q}_*^{WE}(\Gamma) = \sum_{a \in \mathcal{A}} \hat{q}_a(\Gamma) w_a(\Gamma) \approx q_*(R), \tag{10}$$

where each weight $w_a(\Gamma)$ is an estimate of the probability of $\hat{q}_a(\Gamma)$ being larger than all other samples means:

$$w_a(\Gamma) \approx P\left(\hat{q}_a(\Gamma) = \max_{a'} \hat{q}_{a'}(\Gamma)\right).$$

Assuming the PDFs $\hat{f}_a^\Gamma$ for each $\hat{q}_a(\Gamma)$ are known, we can compute the *Distribution-aware Weighted Estimator* (DWE):

$$\hat{q}_*^{DWE}(\Gamma) = \sum_{i=1}^{M} P\left(\hat{q}_i(\Gamma) = \max_a \hat{q}_a(\Gamma)\right) \mathbb{E}\left[\hat{q}_i(\Gamma)\right] = \sum_{i=1}^{M} \mathbb{E}[\hat{q}_i(\Gamma)] \int_{-\infty}^{+\infty} \hat{f}_i^\Gamma(x) \prod_{j \neq i}^{M} \hat{F}_j^\Gamma(x) \, \mathrm{d}x. \tag{11}$$

The sample mean $\hat{q}_a(\Gamma)$ is a random variable whose expected value is q_a and whose variance is $\frac{\sigma_a^2}{|\Gamma_a|}$. Unfortunately, its PDF $\hat{f}_a^\Gamma$ depends on the PDF f_a of the random variable R_a that is assumed to be unknown. In particular, if R_a is normally distributed, then, independently of the sample size, the sampling distribution of its sample mean is normal too: $\hat{q}_a(\Gamma) \sim \mathcal{N}\left(q_a, \frac{\sigma_a^2}{|\Gamma_a|}\right)$. On the other hand, by the central limit theorem (CLT), the sampling distribution $\hat{f}_a^\Gamma$ of the sample mean $\hat{q}_a(\Gamma)$ approaches the normal distribution as the number of samples $|\Gamma_a|$ increases, independently of the distribution of R_a. Leveraging on these considerations, we propose to approximate the distribution of the sample mean $\hat{q}_a(\Gamma)$ with a normal distribution, where we replace the (unknown) population mean and

variance of variable R_a with their (unbiased) sample estimates $\hat{\mu}_a(\Gamma)$ and $\hat{\sigma}_a(\Gamma)$:

$$\hat{f}_a^\Gamma \approx \tilde{f}_a^\Gamma = \mathcal{N}\left(\hat{q}_a(\Gamma), \frac{\hat{\sigma}_a^2(\Gamma)}{|\Gamma_a|}\right).$$

Finally, the WE can be computed as:

$$\hat{q}_*^{WE}(\Gamma) = \sum_{a\in\mathcal{A}} \hat{q}_a(\Gamma) w_a(\Gamma) = \sum_{i=1}^{M} \hat{q}_i(\Gamma) \int_{-\infty}^{+\infty} \tilde{f}_i^\Gamma(x) \prod_{j\neq i}^{M} \tilde{F}_j^\Gamma(x)\, \mathrm{d}x. \tag{12}$$

Note that under the assumption that the variance of each R_a is finite, WE is consistent with $q_*(R)$: as the number of samples grows to infinity, each sample mean $\hat{q}_a$ converges to the related population mean q_a, and the variance of the normal distribution $\tilde{f}_a$ tends to zero. Consequently, the weights of the variables with expected value less than $q_*(R)$ go to zero, so that $\hat{q}_*^{WE} \to q_*(R)$.

3.1 Generalization to infinite random variables

We want to generalize WE to handle an infinite number of continuous random variables, i.e., a continuous action space. Let us consider a continuous space of random variables $\mathcal{A}$ equipped with some metrics (e.g., a Polish space) and assume that the variables in $\mathcal{A}$ have some spatial correlation and that a maximum expected value exists. We choose $\mathcal{A}$ to be a closed interval in $\mathbb{R}$, and we indicate each random variable (return) using the corresponding real-valued index (action). We assume that each variable indexed by $\mathfrak{a} \in \mathcal{A}$ has unknown bounded mean $q_\mathfrak{a}$ and bounded variance $\sigma_\mathfrak{a}^2$. Given a set of samples Γ, we can compute an estimate $\hat{q}_\mathfrak{a}(\Gamma)$ of the expected value $q_\mathfrak{a}$ for any $\mathfrak{a} \in \mathcal{A}$. The weighted sum of equation (10) generalizes to an integral over the space $\mathcal{A}$:

$$\hat{q}_*^{\mathrm{WE}}(\Gamma) = \int_{\mathcal{A}} \hat{q}_\mathfrak{a}(\Gamma)\, \mathfrak{w}_\mathfrak{a}(\Gamma) \mathrm{d}\mathfrak{a}, \tag{13}$$

where $\mathfrak{w}_\mathfrak{a}(\Gamma)$ is the approximate probability density for $\mathfrak{a}$ *corresponding to the variable with the largest mean* (i.e., the largest expected return), which plays the same role of the weights used in (10). Given the distribution $\hat{f}_\mathfrak{a}^\Gamma$ of $\hat{q}_\mathfrak{a}(\Gamma)$, the computation of the exact density is similar to what is done in (11) for the computation of the DWE, with the major difference that in the continuous case we have to (ideally) consider a product of infinite cumulative distributions. We provide a tractable formulation of such density function as

$$\mathfrak{w}^*_{\mathfrak{a}}(\Gamma) = f\left(\hat{q}_{\mathfrak{a}}(\Gamma) = \sup_{\boldsymbol{y}\in\mathcal{A}} \hat{q}_{\boldsymbol{y}}(\Gamma)\right) \tag{14}$$

$$= \int_{-\infty}^{\infty} f(\hat{q}_{\mathfrak{a}}(\Gamma) = \boldsymbol{x})\; P\Big(\hat{q}_{\boldsymbol{y}}(\Gamma) \leq \boldsymbol{x},\; \forall \boldsymbol{y} \in \mathcal{A}\setminus\{\mathfrak{a}\}\Big)\mathrm{d}\boldsymbol{x} \tag{15}$$

$$= \int_{-\infty}^{\infty} \hat{f}^{\Gamma}_{\mathfrak{a}}(\boldsymbol{x})\; P\left(\bigwedge_{\boldsymbol{y}\in\mathcal{A}\setminus\{\mathfrak{a}\}} \hat{q}_{\boldsymbol{y}}(\Gamma) \leq \boldsymbol{x}\right)\mathrm{d}\boldsymbol{x} \tag{16}$$

$$= \int_{-\infty}^{\infty} \hat{f}^{\Gamma}_{\mathfrak{a}}(\boldsymbol{x})\; \frac{P\left(\bigwedge_{\boldsymbol{y}\in\mathcal{A}} \hat{q}_{\boldsymbol{y}}(\Gamma) \leq \boldsymbol{x}\right)}{P\left(\hat{q}_{\mathfrak{a}}(\Gamma) \leq \boldsymbol{x}\right)}\mathrm{d}\boldsymbol{x} \tag{17}$$

$$= \int_{-\infty}^{\infty} \hat{f}^{\Gamma}_{\mathfrak{a}}(\boldsymbol{x})\; \frac{\prod_{\mathcal{A}} \hat{F}^{\Gamma}_{\boldsymbol{y}}(\boldsymbol{x})^{d\boldsymbol{y}}}{\hat{F}^{\Gamma}_{\mathfrak{a}}(\boldsymbol{x})}\mathrm{d}\boldsymbol{x} \tag{18}$$

where (16)-(17) follow from the independence assumption. The numerator term in (18) is the product integral in geometric calculus (i.e., the generalization of the product operator to continuous supports) and can be related to the classical calculus through the following relation: $\prod_{\mathcal{A}} F^{\Gamma}_{\boldsymbol{y}}(\boldsymbol{x})^{d\boldsymbol{y}} = \exp\left(\int_{\mathcal{A}} \ln F^{\Gamma}_{\boldsymbol{y}}(x) d\boldsymbol{y}\right)$ (Grossman and Katz, 1972, Sec. 2.6).

3.1.1 Spatially correlated variables

We have assumed the random variables to be spatially correlated to be able to use any regression technique to approximate the empirical means and generalize over poorly or unobserved regions. Now, we need to restrict the regression class to methods for which the uncertainty of the outcome can be evaluated. Let g be a generic regressor whose predictions are the mean of a variable indicated by $\mathfrak{a}$ and the variance (confidence) of the predicted mean $\Big(\text{i.e., } (\hat{q}_{\mathfrak{a}}, \hat{\sigma}^2_{\hat{q}_{\mathfrak{a}}}) \leftarrow g(\mathfrak{a})\Big)$. Analogously to the the discrete case, we exploit the CLT to approximate the distribution of the sample mean $\hat{f}^{\Gamma}_{\mathfrak{a}}$ with a normal distribution $\tilde{f}^{\Gamma}_{\mathfrak{a}} = \mathcal{N}\left(\hat{q}_{\mathfrak{a}}, \hat{\sigma}^2_{\hat{q}_{\mathfrak{a}}}\right)$.

As a result, the WE for infinite random variables is

$$\hat{q}^{\mathrm{WE}}_*(\Gamma) = \int_{\mathcal{A}} \hat{q}_{\mathfrak{a}}(\Gamma)\, \mathfrak{w}_{\mathfrak{a}}(\Gamma)\mathrm{d}\mathfrak{a} = \int_{\mathcal{A}} \hat{q}_{\mathfrak{a}}(\Gamma) \int_{-\infty}^{\infty} \tilde{f}^{\Gamma}_{\mathfrak{a}}(\boldsymbol{x}) \frac{e^{\int_{\mathcal{A}} \ln \tilde{F}^{\Gamma}_{\boldsymbol{y}}(\boldsymbol{x})\mathrm{d}\boldsymbol{y}}}{\tilde{F}^{\Gamma}_{\mathfrak{a}}(\boldsymbol{x})}\mathrm{d}\boldsymbol{x}\mathrm{d}\mathfrak{a}. \tag{19}$$

Since in the general case no closed-form solution exists for the above integrals, as in the finite case, the WE can be computed through numerical integration (e.g., trapezoidal rule or Romberg integration).

Gaussian Process regression. We propose to use Gaussian Process (GP) regression (Rasmussen and Williams, 2005), since it provides both an estimate of the process mean, and its variance. Consider having a GP trained on a data set of N samples $\mathcal{D} = \{a_i, q_i\}_{i=1}^N$, where q_i is a noisy sample drawn from the distribution of the random variable in a_i. Given a new input a_*, our objective is to predict the expected value of q_* such that $q_* = f(a_*) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2_\eta)$ is a noise random variable and f is the unknown function generating the data. Given a kernel function k used to measure the covariance between two points

(a_i, a_j) and an estimate of the noise variance σ_η^2, the GP approximation for a certain a_* is $q_* \sim \mathcal{N}\left(\hat{q}_*, \hat{\sigma}^2_{\hat{q}_*} + \sigma_\eta^2 I\right)$ where:

$$\begin{aligned} \hat{q}_* &= \mathbf{k}_*^T \left(K + \sigma_\eta^2 I\right)^{-1} \mathbf{q}, \\ \hat{\sigma}^2_{\hat{q}_{z_*}} &= \mathrm{Cov}\left(q_*\right) = k(z_*, z_*) - \mathbf{k}_*^\intercal \left(K + \sigma_\eta^2 I\right)^{-1} \mathbf{k}_*, \end{aligned} \tag{20}$$

and $\mathbf{k}_*$ is the column vector of covariances between a_* and all the input points in $\mathcal{D}$ ($\mathbf{k}_*^{(i)} = K(a_i, a_*)$), K is the covariance matrix computed over the training inputs ($K^{(ij)} = k(a_i, a_j)$), and $\mathbf{q}$ is the vector of training targets. Given the mean estimate in (20), the application of ME and DE is straightforward, while using WE requires to estimate also the variance of the mean estimates. The variance of the GP target q_* is composed by the variance of the mean ($\hat{\sigma}^2_{\hat{q}_*}$) and the variance of the noise (σ_η^2) (Rasmussen and Williams, 2005). As a result, by only considering the mean contribute, we approximate the distribution of the sample mean by $\hat{f}_{\mathfrak{a}}^{\mathcal{D}} \approx \tilde{f}_{\mathfrak{a}}^{\mathcal{D}} = \mathcal{N}\left(\hat{q}_{\mathfrak{a}}, \hat{\sigma}^2_{\hat{q}_{\mathfrak{a}}}\right)$ as defined in Equations (20).

4. Theoretical and empirical analysis

We theoretically analyze the estimation error of WE in terms of bias and variance, comparing it with the results available for ME and DE. We also discuss some of the properties of MME and how it compares with the other estimators. Although DWE cannot be used in practice, we include it in the following analysis as it provides an upper limit to the accuracy of WE. Note that except for the DWE case, we do not assume the distribution of the random variables to be known.

4.1 Bias

We start by summarizing the main results on the bias of ME and DE reported in (D'Eramo et al., 2016; Van Hasselt, 2013). For what concerns the direction of the bias, ME is positively biased, while DE is negatively biased. However, if we look at the absolute bias, there is no clear winner. An easy way to understand why ME is positively biased is to evaluate it in a limit case. Suppose we have two identically distributed random variables and that we want to estimate the MEV of these two variables. Using ME, given two sample means, we would estimate the MEV as the maximum of the two sample means. However, as we said, the two random variables have the same true mean but using this approach we would overestimate the true mean by choosing a sample mean that, due to randomness, is greater than the other and probably higher than the true average. Conversely, DE is unbiased in this setting. When the maximum expected value is sufficiently larger than the expected values of the other variables, the absolute bias of ME can be significantly smaller than the one of DE. A wide analysis provided in (Van Hasselt, 2013) shows that the bias of ME is bounded by:

$$\mathrm{Bias}\left(\hat{q}_*^{ME}\right) \leq \sqrt{\frac{M-1}{M} \sum_{i=1}^{M} \frac{\sigma_i^2}{|\Gamma_i|}}. \tag{21}$$

For the bias of DE, Van Hasselt (2013) *conjectures* the following bound (which is proved for two variables):

$$\text{Bias}\left(\hat{q}_*^{DE}\right) \geq -\frac{1}{2}\left(\sqrt{\sum_{i=1}^{M}\frac{\sigma_i^2}{|\Gamma_i^A|}}+\sqrt{\sum_{i=1}^{M}\frac{\sigma_i^2}{|\Gamma_i^B|}}\right).$$

The bias of WE can be positive or negative. By estimating the probabilities and the means from the same dataset, WE introduces a positive bias due to the positive correlation of these measures. On the other hand, the bias is reduced by computing a weighted average of the sample means. In the next theorem, we provide a relationship between the bias of WE and the one of ME.

Theorem 1 *For any given set R of M random variables:*

$$\text{Bias}(\hat{q}_*^{WE}) \leq \text{Bias}(\hat{q}_*^{ME}) \leq \sqrt{\frac{M-1}{M}\sum_{i=1}^{M}\frac{\sigma_i^2}{|\Gamma_i|}}.$$

The proof of this theorem follows directly from observing that $\hat{q}_*^{WE}$ is always smaller than $\hat{q}_*^{ME}$. In fact, ME can be seen as a WE giving probability one to the variable associated with the largest sample mean $\hat{\mu}_i$, so that any other weighting cannot produce a larger value.

As shown empirically in Section 6, this does not mean that the absolute bias of WE is necessarily smaller than the one of ME, since the bias of WE can be also negative. The bias of WE, indeed, is lower bounded by the bias of DE as shown in (D'Eramo et al., 2016).

Theorem 2 *For any given set R of M random variables:*

$$\text{Bias}(\hat{q}_*^{WE}) \geq \text{Bias}(\hat{q}_*^{DE}) \geq -\frac{1}{2}\left(\sqrt{\sum_{i=1}^{M}\frac{\sigma_i^2}{|\Gamma_i^A|}}+\sqrt{\sum_{i=1}^{M}\frac{\sigma_i^2}{|\Gamma_i^B|}}\right).$$

Proof Similarly to Theorem 1, we consider Equation (6) and Equation (12). Both are weighted averages of samples means, where weights are estimates of the probability for each sample mean of being the maximum. However, as noted before, the weights of WE are computed from the same dataset, causing a positive correlation between them and sample means. In contrast, the weights of DE are computed w.r.t. two independent set of samples, thus not being correlated to the sample means. Hence, the bias of WE is greater or equal than the bias of DE. ■

The bias of MME depends on both the number of random variables M and the number of estimators N, which can represent an advantage, allowing the designer to calibrate the estimate for a specific problem (Lan et al., 2020). On the other hand, choosing the right number of estimators is not an easy task since in RL the distribution of the expected returns is unknown and varies across the state space. MME is trivially upper-bounded by the bias of ME, as MME with $N = 1$ is ME and the bias monotonically decreases with N. Differently, *for all* $N \geq 2$, the bias of MME can be lower than the bias of DE. Let us consider the case

where there exists a^* such that $P\left(a^* = argmax_a \hat{q}_a(\Gamma^j)\right) \approx 1$, in this case the MME is equal to $\min_j \hat{q}_{a^*}(\Gamma^j)$, which is a negatively biased estimate of the true mean q_{a^*}. A lower bound can be found by noting that $\min_j \hat{q}_{a^*}(\Gamma^j) = -\max_j -\hat{q}_{a^*}(\Gamma^j)$ and using Equation (21). Note that in this setting, which corresponds to the case where one action dominates the others, all the other estimators are unbiased. While this might not be always a problem in practice, it shows that MME can be sensitive to the actual distribution of the action-values.

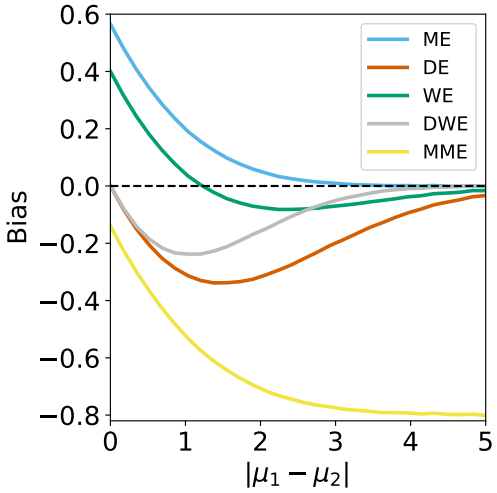


Figure 1: Comparison of the bias of the estimators varying the difference of the means

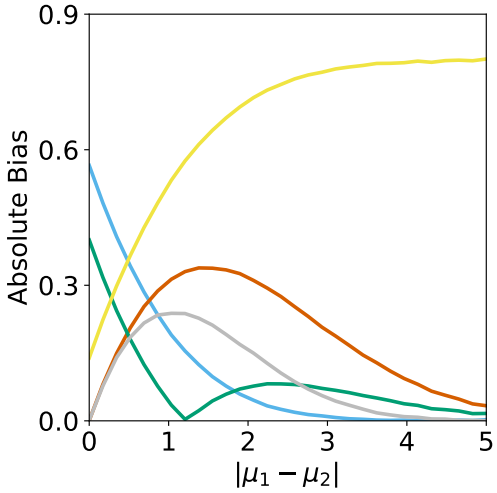


Figure 2: Comparison of the absolute bias of the estimators varying the difference of the means.

Example 1. We analyse the bias of the different MEV estimators in a setting with two normally distributed random variables ($R_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $R_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$) as a function of the difference of their expected values. Both variables have standard deviation equal to 10 ($\sigma_1 = \sigma_2 = 10$) and we assume to have 100 samples for each variable ($|\Gamma_1| = |\Gamma_2| = 100$). For MME we use $N = 2$ and WE is computed using a Monte Carlo approach (see Section 4.3) drawing 100 samples from each approximate PDF. Figure 1 shows that the bias of ME is always positive, while the biases of DWE and DE are always negative, with the latter always worse than the former. The bias of WE can be either positive or negative, but it always falls in the range between the biases of ME and DE. MME behavior is consistent with our previous observations and shows a large negative bias when the difference between the means is large. By looking at the absolute biases shown in Figure 2, we can notice that there is no clear winner. These results confirm the previous theoretical analysis. As mentioned above, when the variables have the same mean, both DE and DWE are unbiased, while it represents a worst case for the bias of ME and WE. It follows that, when the difference between the two means is small (less than 0.5 in the example), DE suffers less absolute bias than ME and WE. For moderate differences between the means (between 0.5 and 1.8 in the example), WE has the minimum absolute bias, while ME is preferable for larger differences.

In this settings MME is outperformed by DE, while it shows lower bias than ME and WE when the difference between the means is very small.

4.2 Variance

Van Hasselt (2013) proved that both the variance of ME and the one of DE[2] can be upper bounded with the sum of the variances of the sample means: $\mathrm{Var}\left(\hat{q}_*^{ME}\right) \leq \sum_{i=1}^{M} \frac{\sigma_i^2}{|\Gamma_i|}$, $\mathrm{Var}\left(\hat{q}_*^{DE}\right) \leq \sum_{i=1}^{M} \frac{\sigma_i^2}{|\Gamma_i|}$. The next theorem shows that the same upper bound holds also for the variance of WE.

Theorem 3 *The variance of WE is upper bounded by*

$$\mathrm{Var}\left(\hat{q}_*^{WE}\right) \leq \sum_{i=1}^{M} \frac{\sigma_i^2}{|\Gamma_i|}.$$

Proof We can derive the bound to the variance as

$$\mathrm{Var}\left(\hat{q}_*^{WE}\right) = \mathrm{Var}\left(\sum_{i=1}^{M} \hat{q}_i(\Gamma) w_i^{\Gamma}\right) \leq \mathrm{Var}\left(\sum_{i=1}^{M} \hat{q}_i(\Gamma)\right) = \sum_{i=1}^{M} \mathrm{Var}\left(\hat{q}_i(\Gamma)\right),$$

where the inequality is a consequence of the maximization of each weight w_i^{Γ} with one and the last equality comes from the independence of the sample means. ■

The bound in Theorem 3 is overly pessimistic: even if each weight w_i^{Γ} is correlated to the other weights and to the sample mean $\hat{q}_i(\Gamma)$, their sum is equal to one. The following upper bound for the variance of DWE shows that the variance becomes smaller as the accuracy of the Gaussian approximation used by WE improves.

Theorem 4 *The variance of DWE is upper bounded by*

$$\mathrm{Var}\left(\hat{q}_*^{DWE}\right) \leq \max_{i \in 1,\ldots,M} \frac{\sigma_i^2}{|\Gamma_i|}.$$

Proof Since the weights w_i computed by DWE are not random variables, it follows

$$\mathrm{Var}\left(\hat{q}_*^{DWE}\right) = \mathrm{Var}\left(\sum_{i=1}^{M} \hat{q}_i(\Gamma) w_i\right) = \sum_{i=1}^{M} w_i^2 \mathrm{Var}\left(\hat{q}_i(\Gamma)\right) \leq \max_{i \in 1,\ldots,M} \frac{\sigma_i^2}{|\Gamma_i|},$$

where the inequality is motivated by $w_i^2 \leq 1$, $\forall i$. ■

The variance of MME depends on the number of estimators N. Lan et al. (2020) show, under some assumption on the distribution of the random variables, that the variance of

2. As already mentioned, in this section what we refer to as DE is actually the 2-fold CV estimator, which has half the variance of DE. We believe this to be fair, since CV is in fact used in Double Q-Learning (Van Hasselt, 2010) to take greedy actions. However the reader should keep in mind that the variance of the action-value function updates when using DE is actually double the one reported here.

MME can be higher than ME if N is small, while it decreases as N increases, at the expense of a potentially higher absolute bias.

We study the variance of the different estimators under the same settings described in Example 1. Figure 3 shows that as the difference of the means of the two variables grows, the variance of all the estimators converges to the variance of the sample mean of the variable with MEV. DE is the estimator with the largest variance because of the use of half the number of samples w.r.t. the other estimators. WE exhibits a variance slightly larger than ME, while, as expected, the variance of DWE is always the smallest. Figure 4 shows the MSE (variance + bias^2) of the different estimators. When the difference between the two means is less than one, WE suffers from a lower MSE than the other two estimators. On the other hand, ME is preferable when there is a variable with an expected value which is significantly larger than the other ones.

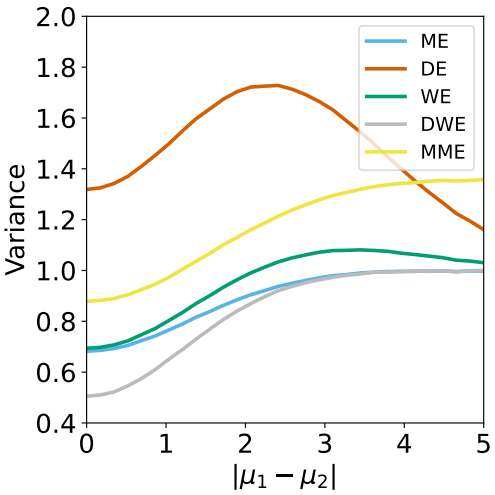


Figure 3: Comparison of the variance of the different estimators varying the difference of the means.

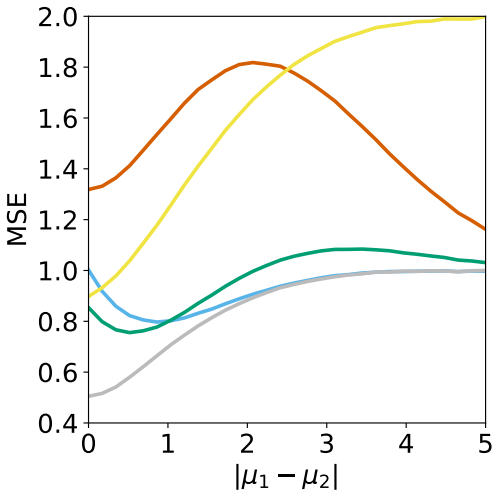


Figure 4: Comparison of the MSE of the different estimators varying the difference of the means.

4.3 Monte Carlo approximation

The integral calculation of Equation (12) makes the WE computation more time-consuming than the ones of the other estimators. In order to mitigate this issue, we propose a Monte-Carlo (MC) approximation of WE, consisting of drawing T samples from each PDF $\tilde{f}_a^{\Gamma}(x)$, and calculating the weights w_a using the collected samples. More precisely, the weights are computed as

$$w_i = \frac{1}{T} \sum_{j}^{T} \left[\!\!\left[\Gamma_{i,j} = \max_k \Gamma_{k,j} \right]\!\!\right], \tag{22}$$

where $\Gamma_{k,j}$ is the j-th sample drawn from the k-th random variable and $[\![\ldots]\!]$ are the Iverson brackets ($[\![P]\!]$ is 1 if P is true, 0 otherwise). It is worth noting that while computing the integral in Equation (12) with numerical integration methods, e.g., trapezoidal rule, would require an asymptotic time complexity of $O(M^2K)$, where K is the number of trapezoids, the aforementioned approach provides a time complexity of $O(MT)$. While the number of trapezoids requires to accurately compute the integral depends on the characteristic of the integrated function, we consistently observe satisfactory performance using only a small number of MC samples. The adoption of the MC approach leads to a significant reduction in the computational cost of running the experiments presented in Section 6, with no relevant decrease in performance. A relatively small number of samples (e.g., 100), in fact, shows to be sufficient to account for the variance introduced by the MC approximation in all the considered settings. Furthermore, from a very practical standpoint, we argue that the constant part of the computational cost of numerical integration methods can be much larger than taking a few tens of samples from a Gaussian distribution (see Appendix C).

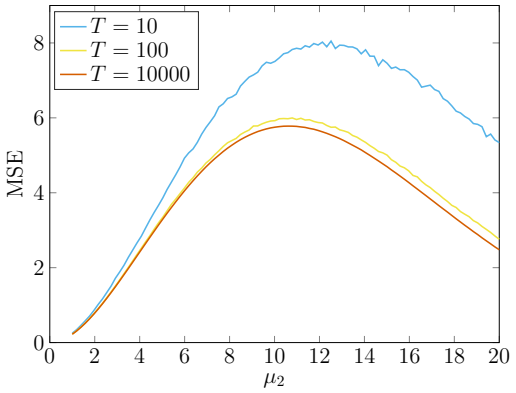


Figure 5: MSE of WE computed with the Monte-Carlo approximation method for different values of T varying the difference of the means of two random variables (results are averaged over $10,000$ seeds).

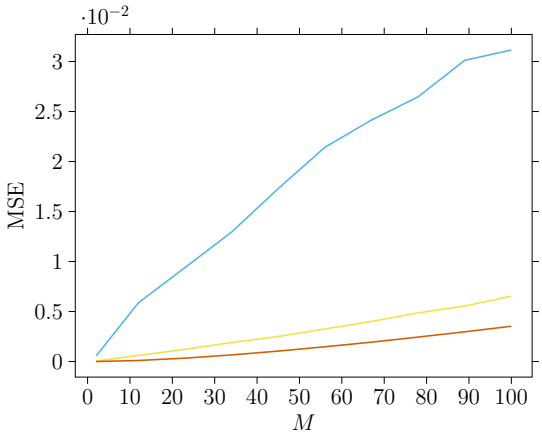


Figure 6: MSE of WE computed with the Monte-Carlo approximation method for different values of T varying the number of random variables M (results are averaged over $10,000$ seeds).

In Figure 5, we show, in a setting with $M = 2$ two random variables sharing the same variance $R_1 \sim \mathcal{N}(0, \sigma^2)$ and $R_2 \sim \mathcal{N}(\mu_2, \sigma^2)$, how the MSE, which accounts for both variance and bias, varies as the number of samples T and the expected value of μ_2 increase. We notice that while $T = 10$ samples are not sufficient to provide an accurate approximation, for values of $\mu_2 < 10$, the MSE value obtained with $T = 100$ approaches the one we obtain with $T = 10,000$. These results suggest us that adopting low number of samples ($T \approx 100$) provides accurate estimates of the integral. In Figure 6, we show

how the MSE varies as the number of samples T and the number of random variables M increase. We define a setting with $M-1$ random variables $\{R_1, \dots, R_{M-1}\}$ that follow the same distribution $\mathcal{N}(0,1)$ and one random variable $R_M \sim \mathcal{N}(5,1)$. Note that the MSE increases as the value of M increases and the value of T decreases.

5. Maximum expected value in Reinforcement Learning

In this section, we show applications of WE in the RL scenario. At first, we introduce the basic notions about Markov Decision Processes (MDPs) and how MDPs can be solved with the model-free value-based approaches. Then, we discuss online and batch algorithms based on ME and DE and present their counterparts based on WE. Eventually, we extend the focus to the deep RL framework where a novel algorithm based on WE is proposed. It is worth noting, that all maximum value estimators analyzed in the previous sections rely on the assumption that data are i.i.d.; this is clearly not the case in online sequential RL where the policy changes overtime. However, we show, supported by empirical evidence, that WE is nonetheless effective and leads to better estimation and performance in several settings. In the following sections, we further comment on this aspect and motivate our algorithm design choices.

5.1 Weighted Estimator in Reinforcement Learning algorithms

Applying WE in the RL settings requires to model uncertainty over action-value estimates. By assuming a Gaussian approximation, we consider a distribution $\widetilde{Q}(s,a)$ over action-value functions such that:

$$\widetilde{Q}(s,a) \approx \mathcal{N}\left(Q(s,a), \sigma^2(s,a)\right), \tag{23}$$

where $Q(s,a)$ and $\sigma^2(s,a)$ are the mean and the variance of the estimator w.r.t. the state-action pair (s,a), respectively.

Two types of uncertainty. It is important to stress the difference between our approach and distributional RL (Morimura et al., 2010; Bellemare et al., 2017), a difference that has its roots in closely related, but distinct, sources of statistical *uncertainty*. Standard value-based methods are interested in learning the expected return associated with a policy. Distributional approaches, conversely, model the full distribution of the stochastic return of the decision-making process. In other words, distributional methods account for the *aleatoric uncertainty* that is intrinsic in the MDP, a concept related to *risk* and of paramount importance for AI safety (Morimura et al., 2010). With WE, conversely, we are concerned with modeling uncertainty over the estimator, that is, *epistemic uncertainty*. Going back to RL, this is the agent's uncertainty about its predictions. With this view, our approach is more related to Bayesian RL methods such as Bayesian Q-Learning (Dearden et al., 1998) and Randomized Value Functions (Osband et al., 2019).

Among the value-based methods, we consider online and offline algorithms that approximate optimal Q-values without the need for a model of the environment. We mostly consider MDPs with discrete action spaces, except for the batch algorithm based on WE which can also be extended to MDPs with continuous action spaces.

5.1.1 Online RL: Weighted Q-Learning

Recently, the replacement of ME with WE has been proposed in the *Weighted Q-Learning* algorithm (D'Eramo et al., 2016). Weighted Q-Learning (WQL) maintains an estimate of the mean value of the Q-function and its variance in order to compute the weights of WE (Algorithm 1). In tabular Q-learning, the expected return of the optimal policy $Q(s,a)$ is learned incrementally as an exponentially weighted sample mean of the Q-Learning target values $y_t^{QL} = r_t + \gamma \max_a Q_t(s_{t+1}, a)$. The variance of the return can be estimated similarly as the weighted sample variance $\hat{\sigma}^2(s,a)$. Following the Gaussian approximation, we consider the expected return to be normally distributed with a mean corresponding to the aforementioned sample mean and a variance corresponding to the sample variance divided by the effective sample size, i.e., the number of updates adjusted to reflect the effect of the weighting:

$$\widetilde{Q}_t(s,a) \approx \mathcal{N}\left(Q_t(s,a), \frac{\hat{\sigma}_t^2(s,a)}{ESS_t(s,a)}\right). \tag{24}$$

The effective samples size $ESS_t(s,a)$ can be estimated using the Kish's effective sample size (Kish, 1965; Martino et al., 2017), that can be computed incrementally as:

$$\omega_t'(s,a) \leftarrow (1-\alpha_t(s,a))\omega_{t-1}'(s,a) + \alpha_t(s,a),$$

$$\omega_t''(s,a) \leftarrow (1-\alpha_t(s,a))^2\omega_{t-1}''(s,a) + \alpha_t(s,a)^2,$$

$$ESS_t(s,a) = \frac{(\omega_t'(s,a))^2}{\omega_t''(s,a)}.$$

We refer to D'Eramo et al. (2019) for examples of iterative methods to compute online the weighted sample variance. Note that this weighting partially accounts for nonstationarities caused by changes in the policy.

Weighted Q-Learning exploits the approximated distribution and adjusts the Q-Learning target value using the WE. The resulting WQL target value can be computed as

$$y_t^{WQL} = r_t + \gamma \sum_{a \in \mathcal{A}} w_a^{s_{t+1}} Q_t(s_{t+1}, a), \tag{25}$$

where $w_a^{s_{t+1}}$ are the weights of the WE and correspond to the probability of each action-value being the maximum:

$$w_a^s = P\left(a = \arg\max_{a'} Q_t(s,a')\right). \tag{26}$$

Note that the approach used in Algorithm 1 to evaluate the variance of the mean estimator conflates an aleatoric component. This component can be completely removed by adopting a Bayesian approach, e.g., Bayesian Q-Learning. In this tabular setting, we use a frequentist approach to study WE as a drop-in replacement to other maximum value estimators in the standard Q-Learning algorithm. In the following we consider the Bayesian perspective in batch RL and deep RL algorithms.

Algorithm 1 Weighted Q-learning

Inputs: $Q(s,a)$, $\hat{\sigma}(s,a)$, $ESS(s,a)$ initial state s
repeat
  Select an action a according to some policy $\pi(\cdot|s)$ (e.g., ε-greedy)
  Execute action a and observe reward r and next state s'
  For each action $a \in \mathcal{A}$, compute density $\tilde{f}_a$ and cumulative $\tilde{F}_a$ distributions as a normal distribution $\mathcal{N}(Q(s,a), \frac{\hat{\sigma}^2(s,a)}{ESS(s,a)})$
  For each action $a \in \mathcal{A}$, compute weight $w_a^{s'}$ (integral in Eq. (12) or MC approximation)
  $y^{WQL} = r + \gamma \sum_{a \in \mathcal{A}} w_a^{s'} Q(s', a)$
  $Q(s,a) \leftarrow Q(s,a) + \alpha(s,a)(y^{WQL} - Q(s,a))$
  Update $\hat{\sigma}(s,a)$ and $ESS(s,a)$ using tuple $\langle s, a, r \rangle$
  $s \leftarrow s'$
until stopping condition

5.1.2 Batch RL: Weighted Fitted Q-Iterations

A well-known batch variant of Q-Learning is the Fitted Q-Iteration (FQI) algorithm (Ernst et al., 2005). FQI reformulates the RL problem as a sequence of supervised learning problems. Given a set of samples $\mathcal{D} = \{\langle s_i, a_i, s_i', r_i \rangle\}_{1 \leq i \leq N}$ previously collected by the agent according to a given sampling strategy, at each iteration t, FQI builds an approximation of the optimal Q-function by fitting a regression model on a bootstrapped sample set:

$$\mathcal{D}_t = \left\{ \langle (s_i, a_i), r_i + \gamma \max_{a'} Q_{t-1}\left(s_i', a'\right) \rangle \right\}_{1 \leq i \leq N}. \tag{27}$$

The FQI update, similarly to the Q-Learning update, exploits the computation of ME which causes the same overestimation problem as Q-Learning. Intuitively, replacing ME with DE or WE can help solve this issue in FQI as well. The FQI variant that replaces ME with DE is called *Double FQI*, and the variant that uses WE is called *Weighted FQI* (D'Eramo et al., 2017). Weighted FQI uses Gaussian Process (GP) regression to compute the mean Q-value and its variance in continuous state spaces (Algorithm 2). The interesting aspect of Weighted FQI is that it can also handle infinite action spaces, as explained in Section 3 and shown in Algorithm 3. Finally, note that in the batch setting, being offline, there are no issues w.r.t. nonstationarity of the policy learning process.

5.2 Weighted Estimator in deep Reinforcement Learning

The popular Deep Q-Network algorithm (DQN) (Mnih et al., 2015) is a variant of Q-Learning designed to stabilize off-policy learning with deep neural networks in highly dimensional state spaces. The two most relevant changes introduced by DQN to regular Q-Learning are the adoption of a replay memory to learn from past experience and the use of a target network to reduce the correlation between the current model estimate and the bootstrapped target value.

In practice, DQN learns the Q-values online, using a neural network with parameters $\boldsymbol{\theta}$, sampling from the replay memory, and with a target network whose parameters $\boldsymbol{\theta}^-$ are updated to match those of the online network every C steps. The model is trained to

Algorithm 2 Weighted FQI (finite actions)

Inputs: dataset $\mathcal{D} = \{s_i, a_i, r_i, s'_i\}_{i=1}^K$, GP regressor $\widehat{Q}$, horizon $T \in \mathbb{N}$, discrete action space $\mathcal{A} = \{a_1, \ldots, a_M\}$
Train $\widehat{Q}_0$ on $\mathcal{T}_0 = \{\langle s_i, a_i, r_i \rangle\}$
for t=1 **to** T **do**
  for j=1 **to** K **do**
    for m=1 **to** M **do**
      $\hat{q}_m, \sigma^2_{\hat{q}_m} \leftarrow \widehat{Q}_{t-1}(s'_j, a_m)$ (evaluate GP)
      $\hat{f}_{a_m} \leftarrow \mathcal{N}(\hat{q}_m, \sigma^2_{\hat{q}_m})$
      For each action $a_m \in \mathcal{A}$, compute weight w_m (integral in Eq. 12)
    end for
    $y_j \leftarrow r_j + \gamma \sum_{a_m \in \mathcal{A}} w_{a_m} \hat{q}_{a_m}$
    $\mathcal{T}_t \leftarrow \mathcal{T}_t \cup \{\langle s_j, a_j, y_j \rangle\}$
  end for
  Train $\widehat{Q}_t$ on $\mathcal{T}_t$
end for

Algorithm 3 Weighted FQI$_\infty$ (continuous actions)

Inputs: dataset $\mathcal{D} = \{s_i, a_i, r_i, s'_i\}_{i=1}^K$, GP regressor $\widehat{Q}$, horizon $T \in \mathbb{N}$
Train $\widehat{Q}_0$ on $\mathcal{T}_0 = \{\langle s_i, a_i, r_i \rangle\}$
for t=1 **to** T **do**
  for i=1 **to** K **do**
    Given that $\hat{q}_a, \sigma^2_{\hat{q}_a} := \widehat{Q}_{t-1}(s'_i, a)$ and $\hat{f}_a := \mathcal{N}(\hat{q}_a, \sigma^2_{\hat{q}_a}) \quad \forall a \in \mathcal{A}$
    Compute $\hat{q}^{\mathrm{WE}}_*$ (Eq. 19)
    $y_i \leftarrow r_i + \gamma \hat{q}^{\mathrm{WE}}_*$
    $\mathcal{T}_t \leftarrow \mathcal{T}_t \cup \{\langle s_i, a_i, y_i \rangle\}$
  end for
  Train $\widehat{Q}_t$ on $\mathcal{T}_t$
end for

minimize the loss

$$L(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{\langle s_i, a_i, r_i, s'_i \rangle \sim m} \left[\left(y_i^{DQN} - Q(s_i, a_i; \boldsymbol{\theta}) \right)^2 \right], \tag{28}$$

where m is a uniform distribution over the transitions stored in the replay buffer and y^{DQN} is defined as

$$y_i^{DQN} = r_i + \gamma \max_a Q(s'_i, a; \boldsymbol{\theta}^-). \tag{29}$$

DoubleDQN. Among the many studied improvements and extensions of the baseline DQN algorithm (Wang et al., 2016; Schaul et al., 2016; Bellemare et al., 2017; Hessel et al., 2018; Badia et al., 2020), DoubleDQN (DDQN) (Van Hasselt et al., 2016) reduces the overestimation bias of DQN with a simple modification of the update rule. In particular, DDQN uses the target network to decouple selection and evaluation of the action, and estimates the target value as

$$y_i^{DDQN} = r_i + \gamma Q(s'_i, \arg\max_a Q\left(s'_i, a; \boldsymbol{\theta}\right); \boldsymbol{\theta}^-). \tag{30}$$

DDQN improves on DQN converging to a more accurate approximation of the value function while maintaining the same model complexity and adding minimal computational overhead.

AveragedDQN. AveragedDQN (AvgDQN) (Anschel et al., 2017), on the other hand, tackles the problem of stabilizing the learning procedure of DQN by averaging the output of the last K checkpoints of the target network to reduce the variance of the target value. i.e.,

$$y_i^{AvgDQN} = r_i + \gamma \max_a \frac{1}{K} \sum_{k=1}^{K} Q(s'_i, a; \boldsymbol{\theta}_k^-). \tag{31}$$

MaxminDQN. Maxmin Q-Learning is easily extended to the deep RL using N different neural networks to approximate the action value function. The inconvenient of training multiple approximators is a possible increase in computational time or sample complexity, depending on the rate at which the Q-networks are updated. The MaxminDQN target can be computed as:

$$y_i^{MaxminDQN} = r_i + \gamma \max_a \min_j Q(s'_i, a; \boldsymbol{\theta}_j^-). \tag{32}$$

5.2.1 Deep Weighted Q-Learning

A natural way to extend the WQL algorithm to deep RL settings is to consider uncertainty about the model parameters using a Bayesian approach. Among the possible solutions to estimate uncertainty, bootstrapping has had the greatest success in RL problems, with BootstrappedDQN (BDQN) (Osband et al., 2016, 2018) which has achieved impressive results in environments where exploration is critical. On the other hand, bootstrapping necessitates significant modifications to the baseline DQN architecture and requires to train a model for each sample of the approximate posterior distribution. This limits the number of samples available considerably and is a major drawback in using BDQN to approximate the WE weights. Conversely, dropout (Srivastava et al., 2014) does not affect the model complexity and allows to compute the weights of the WE by sampling an arbitrary number of dropout masks.

In the following, we first introduce how neural networks trained with dropout can be used for approximate Bayesian inference and discuss how this approach has been used with success in RL problems. Then, we propose a novel approach to exploit the uncertainty over the model parameters for action evaluation, adapting the WE to the deep RL settings. Finally, we analyze a possible shortcoming of the proposed method and identify a solution from the literature to address it.

Bayesian inference with dropout. Dropout (Srivastava et al., 2014) is a regularization technique used to train large neural networks by randomly dropping units during learning. In recent years, the dropout technique has been analyzed from a Bayesian perspective (Kingma et al., 2015; Gal and Ghahramani, 2016) and interpreted as a variational approximation of a posterior distribution over the neural network parameters. In particular, Gal and Ghahramani (2016) show how a neural network trained with dropout and weight decay can bee seen as an approximation of a deep Gaussian process (Damianou and Lawrence, 2013). The result is a theoretically grounded interpretation of dropout and a class of Bayesian neural networks (BNNs) that are cheap to train and can be queried to obtain uncertainty estimates. In fact, a single stochastic forward pass through the BNN can

be interpreted as taking a sample from the model's predictive distribution, while the predictive mean can be computed as the average of multiple samples. This inference technique is known as *Monte Carlo dropout* and can be efficiently parallelized in modern GPUs.

A straightforward application of Bayesian models to RL is Thompson Sampling (TS) (Thompson, 1933). TS is an exploration technique that aims at improving the sample complexity of RL algorithms by selecting actions according to their probability of being optimal given the current agent's beliefs. A practical way to use TS in deep RL is to take a single sample from a Q-network trained with dropout and select the action that corresponds to the maximum sampled action value (Gal and Ghahramani, 2016). TS based on dropout achieves superior data efficiency compared against naïve exploration strategies, such as ε-greedy, in both sequential decision-making problems (Gal and Ghahramani, 2016; Stadie et al., 2015) and contextual bandits (Riquelme et al., 2018; Collier and Llorens, 2018). Furthermore, the dropout technique has been successfully used in model-based RL, to estimate the agent's uncertainty about the environment dynamics (Gal et al., 2016; Kahn et al., 2017; Malik et al., 2019).

Here we focus on the problem of action evaluation. We show how to use approximate Bayesian inference to evaluate the WE by introducing a novel approach to exploit uncertainty estimates in deep RL. Our method empirically reduces Q-Learning bias, is grounded in theory and simple to implement.

5.2.2 Weighted DQN

Let $Q(\,\cdot\,,\,\cdot\,;\boldsymbol{\theta},\boldsymbol{\omega})$ be a BNN with weights $\boldsymbol{\theta}$ trained with a Gaussian prior and dropout variational inference to learn the optimal action-value function of a certain MDP. We indicate with $\boldsymbol{\omega}$ the set of random variables that represents the dropout mask, with $\boldsymbol{\omega}_i$ the i-th realization of the random variables and with $\boldsymbol{\Omega}$ their joint distribution:

$$\boldsymbol{\omega} = \{\omega_{lk} : l = 1, \dots, L, k = 1, \dots, K_l\}, \tag{33}$$

$$\omega_{lk} \sim \text{Bernoulli}(p_l), \quad \boldsymbol{\omega}_i \sim \boldsymbol{\Omega}(p_1, \dots, p_L), \tag{34}$$

where L is the number of weight layers of the network and K_l is the number of units in layer l.

Consider a sample r_a^s of the MDP return, obtained taking action a in s and following the optimal policy afterwards. Following the GP interpretation of dropout by Gal and Ghahramani (2016), we approximate the likelihood of this *observation* as a Gaussian distribution such that

$$r_a^s \sim \mathcal{N}\left(Q(s, a; \boldsymbol{\theta}, \boldsymbol{\omega}), \tau^{-1}\right), \tag{35}$$

where τ is the precision hyperparameter.

We can approximate the predictive mean of the process, and the expectation over the posterior distribution of the Q-value estimates, as the average of T stochastic forward passes through the network:

$$\mathbb{E}\left[Q(s, a; \boldsymbol{\theta}, \boldsymbol{\omega})\right] \approx \widehat{Q}_T(s, a; \boldsymbol{\theta}) = \frac{1}{T} \sum_{t=1}^{T} Q(s, a; \boldsymbol{\theta}, \boldsymbol{\omega}_t). \tag{36}$$

$\widehat{Q}_T(s, a; \boldsymbol{\theta})$ is an MC estimate of the BNN prediction of the action-values associated to state s and action a. A similar, computationally more efficient approximation can be obtained

Algorithm 4 Weighted DQN

Input: Q-network parameters $\boldsymbol{\theta}$, dropout rates $p_1, \dots, p_L$, a policy π, replay memory $\mathcal{D}$
$\boldsymbol{\theta}^- \leftarrow \boldsymbol{\theta}$
Initialize memory $\mathcal{D}$.
for step $t = 0, \dots$ **do**
  Select an action a_t according to some policy π given the distribution over action-value functions $Q(s, \cdot\,; \boldsymbol{\theta}, \boldsymbol{\omega})$
  Execute a_t and add $\langle s_t, a_t, r_t, s_{t+1} \rangle$ to $\mathcal{D}$
  Sample a mini-batch of transitions $\{\langle s_i, a_i, r_i, s_i' \rangle, i = 1, \dots, M\}$ from $\mathcal{D}$
  for $i = 1, \dots, M$ **do** {can be done in parallel}
    Take T samples from $Q(s_i, \cdot\,; \boldsymbol{\theta}^-, \boldsymbol{\omega})$ by performing T stochastic forward passes
    Use the samples to compute the WDQN weights (Eq.38) and targets (Eq.39)
  end for
  Perform a SGD step on the WDQN loss (Eq. 40)
  Eventually update $\boldsymbol{\theta}^-$
end for

through *weight averaging*, which consists in scaling the output of the neurons in layer l by $1 - p_l$ during training and leaving them unchanged at inference time. We indicate this estimate as $\overline{Q}(s, a; \boldsymbol{\theta})$ and use it for selecting actions during training.

The epistemic uncertainty of the model, i.e., the uncertainty about the model parameters, can be measured similarly as the sample variance across T realizations of the dropout random variables:

$$Var\left[Q(s, a; \boldsymbol{\theta}, \boldsymbol{\omega})\right] \approx \frac{1}{T} \sum_{t=1}^{T} Q(s, a; \boldsymbol{\theta}, \boldsymbol{\omega}_t)^2 - \left(\frac{1}{T} \sum_{t=1}^{T} Q(s, a; \boldsymbol{\theta}, \boldsymbol{\omega}_t) \right)^2 . \tag{37}$$

As shown by Gal and Ghahramani (2016), the predictive variance can be approximated with the variance of the estimator in Eq. (37) plus the model inverse precision τ^{-1}. As in the batch case, we are only interested in the first term.

We can estimate the probability needed to calculate the WE in a similar way. Given T realizations of the dropout variables, the probability that an action a corresponds to the maximum expected action value can be approximated as the ratio between the number of times the sampled action-value of a is the maximum and the number of samples T:

$$w_a^s(\boldsymbol{\theta}) = P\left(a = \arg\max_{a'} Q(s, a'; \boldsymbol{\theta}, \boldsymbol{\omega}) \right) \approx \frac{1}{T} \sum_{t=1}^{T} \left[\!\!\left[a = \arg\max_{a'} Q(s, a'; \boldsymbol{\theta}, \boldsymbol{\omega}_t) \right]\!\!\right] . \tag{38}$$

The weights can be efficiently inferred in parallel without any impact on the computational time. We can define the WE given the Bayesian target Q-network estimates using the obtained weights as:

$$y_i^{WDQN} = r_i + \gamma \sum_{a \in \mathcal{A}} w_a^{s_i'}(\boldsymbol{\theta}^-) \widehat{Q}_T(s_i', a; \boldsymbol{\theta}^-). \tag{39}$$

Finally, we report for completeness the loss minimized by WDQN, where the parameter updates are backpropagated using the dropout masks:

$$L(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{\langle s_i,a_i,r_i,s_i'\rangle\sim m,\boldsymbol{\omega}_i\sim\boldsymbol{\Omega}}\left[\left(y_i^{WDQN} - Q(s_i,a_i;\boldsymbol{\theta},\boldsymbol{\omega_i})\right)^2\right] + \lambda\sum_{l=1}^{L}\|\boldsymbol{\theta}_l\|_2^2\,, \tag{40}$$

where $\boldsymbol{\theta}_l$ are the weights of layer l and λ is the weight decay coefficient. Using weight decay is necessary for the the variational approximation being valid. The complete WDQN algorithm is reported in Algorithm 4.

Concrete dropout. The dropout probabilities are variational parameters and influence the quality of the approximation. Ideally, they should be tuned to maximize the log-likelihood of observations using a validation method. This is clearly not possible in RL, where, as already highlighted, the available samples and the underlying distribution change as the policy improves. In fact, dropout with a fixed probability might lead to poor uncertainty estimates (Osband et al., 2016, 2018; Gal et al., 2017). Concrete Dropout (Gal et al., 2017) mitigates this problem by using a differentiable continuous relaxation of the Bernoulli distribution and learning the dropout rate from data.

In practice, this means that the distribution of the dropout random variables ω_{lk} becomes

$$\omega_{lk} = \sigma\left(\beta\left(\log p_l - \log(1-p_l) + \log u - \log(1-p_l)\right)\right), \tag{41}$$

where β is a temperature parameter (fixed at $\beta = 10$), u is a uniform random variable $u \sim \mathcal{U}(0,1)$ and $\sigma(\cdot)$ is the *sigmoid* function. With this formulation the sampling procedure becomes differentiable and the loss in Eq. (40) can be rewritten as:

$$L(\boldsymbol{\theta},\boldsymbol{p}) = \mathop{\mathbb{E}}_{\langle s_i,a_i,r_i,s_i'\rangle\sim m,\boldsymbol{\omega}_i\sim\boldsymbol{\Omega}}\left[\left(y_i^{WDQN} - Q(s_i,a_i;\boldsymbol{\theta},\boldsymbol{\omega_i})\right)^2\right] + \sum_{l=1}^{L}\left(\lambda\|\boldsymbol{\theta}_l\|_2^2 - \zeta K_l\mathcal{H}(p_l)\right), \tag{42}$$

where ζ is a dropout regularization coefficient, $\mathcal{H}(p)$ is the entropy of a Bernoulli distribution with parameter p and K_l is the number of neurons in layer l.

6. Experiments

We evaluate the performance of ME, DE, MME and WE, in Multi-Armed Bandit (MAB) and RL problems. We start from considering discrete state and action spaces, then we move to continuous ones and to deep RL problems.

For MME in the tabular and batch settings, differently from Lan et al. (2020), we do not use a replay memory for a more direct comparison. Consequently, we fix the number of estimators to $N = 2$ to limit the impact on sample complexity. In the deep RL experiments, instead, we run MaxminDQN with both $N = 2$ and $N = 3$, and we select the best configuration in each setting. Since using only $N = 2$ proved to be the best configuration in most of the considered settings, we did not further increase the number of approximators.

6.1 Discrete states and action spaces

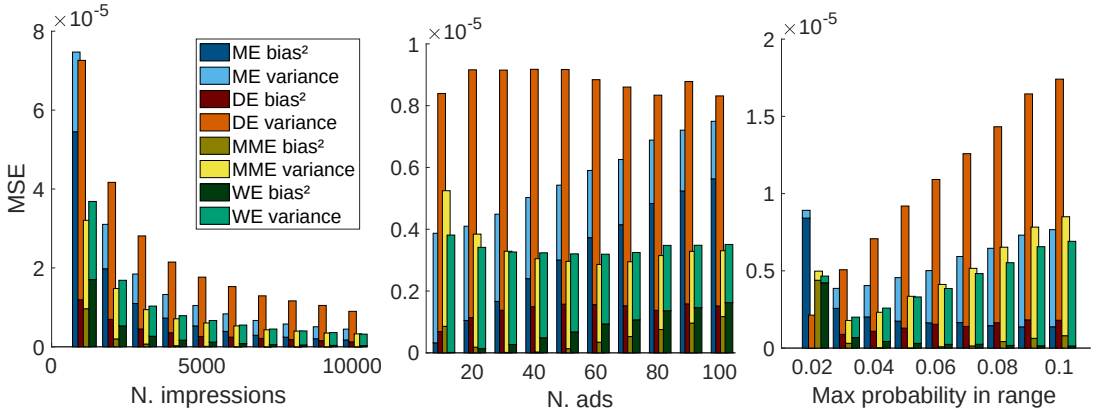
6.1.1 Internet ads

We consider an online advertising problem as formulated in (Van Hasselt, 2013), where the goal is to select the most convenient ad to show on a website from a set of M possible ads, each one with an unknown expected return per visitor. Ads have the same return per click, thus the best ad is the one with the highest clickthrough rate (CTR). Since CTRs are unknown, they have to be estimated from the data. We assume that, given N visitors, each ad is shown the same number of times, so that we have N/M samples to compute the sample CTR. Obtaining a quick and accurate estimate of the maximum CTR is typically a critical task in order to effectively determine future investment strategies. We compare the results of ME, DE, MME and WE, in different settings. We consider two default configurations: a first one (first row in Figure 7) where we have $M = 30$ ads and mean CTR uniformly sampled from the interval $[0.02, 0.05]$ and a second one where we have $M = 10$ ads and mean CTR sampled from $[0.02, 0.2]$. In Figure 7(a) and Figure 7(d), we vary the number of visitors so that the number of impressions per ad ranges from $1,000$ to $10,000$. In Figure 7(b) and Figure 7(e), we vary the number of ads $M = \{10, 20, ..., 90, 100\}$ and the number of visitors is set to $N = 10,000M$. In Figure 7(c) and Figure 7(f), we modify the interval of the mean CTR by changing the value of the upper limit with values in $\{0.02, 0.03, ..., 0.09, 0.1\}$, with the lower fixed at 0.02. We show the $MSE = bias^2 + variance$ for all the settings comparing the results obtained by each estimator.

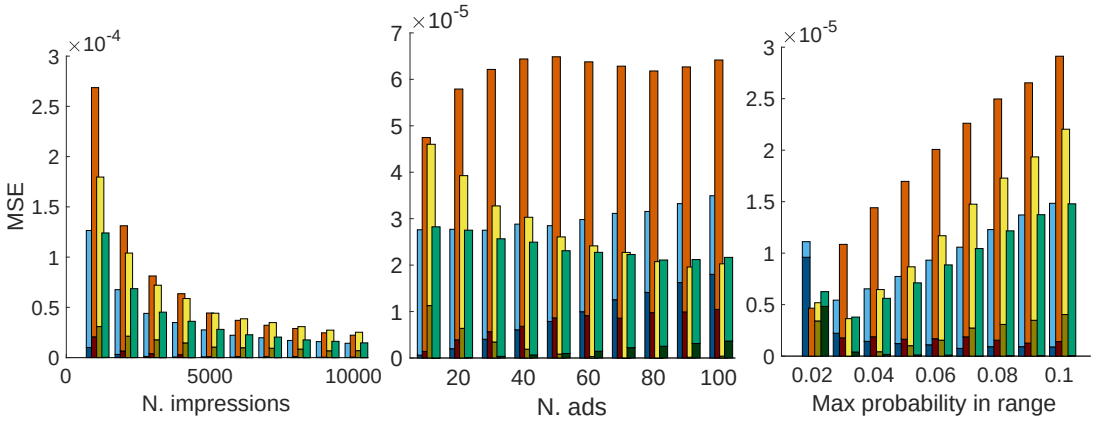
WE is the most robust estimator, performing reasonably well in all settings. DE shows lower bias than ME in many instances, e.g., when the ads have a similar CTR (left most part of Figure 7(c) and 7(f)). On the other hand DE suffers from high variance. MME performs well when ME shows high bias, but its performance degrades when ME is less biased.

6.1.2 Sponsored search auctions

We consider a sponsored search auction problem as described in Xu et al. (2013), where a search engine runs an auction to select the best ad to show from a pool of candidates with the goal of maximizing over a value that depends on each advertiser's bid and click probability. When an ad is clicked, the advertiser is charged from the search engine of a fee that depends on the bids b of the advertisers and the ad's CTRs ρ, namely $\frac{b_2\rho_2}{\rho_1}$, where the subscript i indicates the i-th biggest value. CTRs are generally unknown, therefore the search engine should use data to estimate which is the best ad (i.e., the one that maximizes $b \cdot \rho$) and the payment in case of click; reasonably, wrong estimations may significantly harm revenue. On the other hand, advertisers have to decide the value of their bid b_i according to the true values v_i of a click. A desirable property of an auction mechanism, called *incentive compatibility*, requires advertisers to maximize their utility by truthfully bidding $b_i = v_i$. Incentive compatibility may not occur when the estimate of the click probabilities is not accurate. We want to evaluate how the estimators favor the incentive compatibility. We measure the utility gain, defined as $\frac{utility(b)}{utility(v)} - 1$, of advertiser 1, whose true per click value is $v_1 = 1$, for different bid b_1 values and in competition with four other advertisers whose bids are $b_{-1} = \{0.9, 1, 2, 1\}$. The CTRs are: $\rho = \{0.15, 0.11, 0.1, 0.05, 0.01\}$. CTRs are estimated from data collected using the UCB1 algorithm (Auer et al., 2002) in a learning



(a) $p \in [0.02, 0.05]$, n. ads $= 30$. (b) $p \in [0.02, 0.05]$, visits $= 10k$. (c) n. ads $= 30$, visits $= 300k$.



(d) $p \in [0.02, 0.2]$, n. ads $= 10$. (e) $p \in [0.02, 0.2]$, visits $= 10k$. (f) n. ads $= 10$, visits $= 300k$.

Figure 7: MSE for each setting. Results are averaged over $2,000$ experiments.

phase consisting of $10,000$ rounds of exploration (i.e. impressions), using the same settings of Xu et al. (2013).

Figure 8 shows the utility gain of advertiser 1 when using ME, DE, MME, and WE.The true bid price is highlighted with a black vertical bar. ME is the only one which is not able to achieve incentive compatibility, since the utility has positive values before the true bid price. On the contrary with DE, MME, and WE, the advertiser has no incentive to underbid, but there is an incentive to overbid using DE. Therefore MME and WE are the only estimators that succeed to achieve incentive compatibility.

6.1.3 Grid world

This simple MDP consists of a 3×3 grid world in which the start state is the lower-left cell and the goal state is the upper-right cell (Van Hasselt, 2010). We test the performance of Q-Learning using ME, DE, MME, and WE; furthermore, we also test Weighted Q-

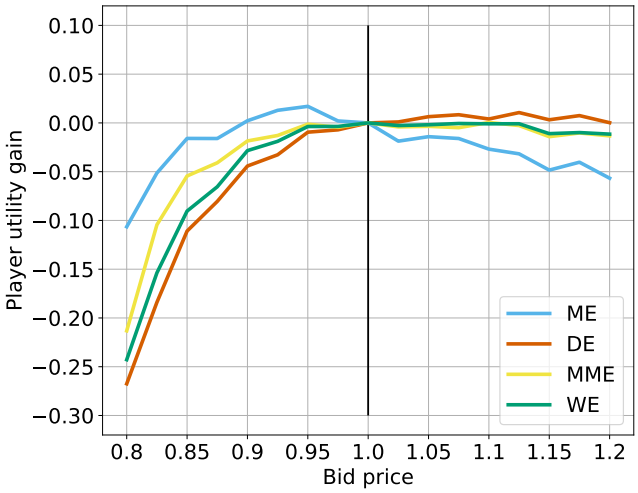


Figure 8: Relative player 1 utility gain for different value of the bid. Results are averaged over $2,000$ experiments.

Learning using a different policy, named *weighted policy*[3], which samples the action to be performed in a state from the probability distribution of the weights of WE. We use the same hyperparameters of Van Hasselt (2010). In particular, we use an ε-greedy policy with $\varepsilon = \frac{1}{\sqrt{n(s)}}$ where $n(s)$ is the number of times the state s has been visited. Learning rate is $\alpha_t(s,a) = \frac{1}{n_t(s,a)^{0.8}}$ where $n_t(s,a)$ is the current number of updates of that action value and the discount factor is $\gamma = 0.95$. In Double Q-Learning and Maxmin Q-Learning we use two learning rates $\alpha_t^A(s,a) = \frac{1}{n_t^A(s,a)^{0.8}}$ and $\alpha_t^B(s,a) = \frac{1}{n_t^B(s,a)^{0.8}}$ where $n_t^A(s,a)$ and $n_t^B(s,a)$ are respectively the number of times when table A and table B are updated. The reward function is considered in three different settings: Bernoulli, -12 or 10 randomly at each step, Gaussian with mean $\mu = -1$ and standard deviation $\sigma = 5$, Gaussian with mean $\mu = -1$ and standard deviation $\sigma = 1$. Once in the goal state, each action ends the episode and returns a reward of 5. The optimal policy ends the episode in five actions, therefore the optimal average reward per step is 0.2. Furthermore, the optimal value of the action maximizing the Q-value is $5\gamma^4 - \sum_{k=0}^{3}\gamma^k \approx 0.36$. In Figure 9, the top plots show the average reward per step obtained by each algorithm and the plots at the bottom show the estimate of the maximum state-action value at the starting state for each algorithm. Figures 9 show that, regardless of the bad approximation of the Q-function, the underestimation of Double Q-Learning and Maxmin Q-Learning might allow learning the best policy faster than other algorithms in these noisy settings. Weighted Q-Learning shows much less bias than other estimators in all settings; moreover, the use of the weighted policy generally reduces the bias of the estimation and achieves the best performance in the case with $\sigma = 1$. These good results are explained considering that the weighted policy is able to reduce the exploration faster than ε-greedy due to the exploitation of the good approximation of the Q-function

3. Equivalent to Thompson Sampling (Thompson, 1933). Note that Weighted Q-Learning with weighted policy can be seen as an instance of Expected SARSA (Van Seijen et al., 2009).

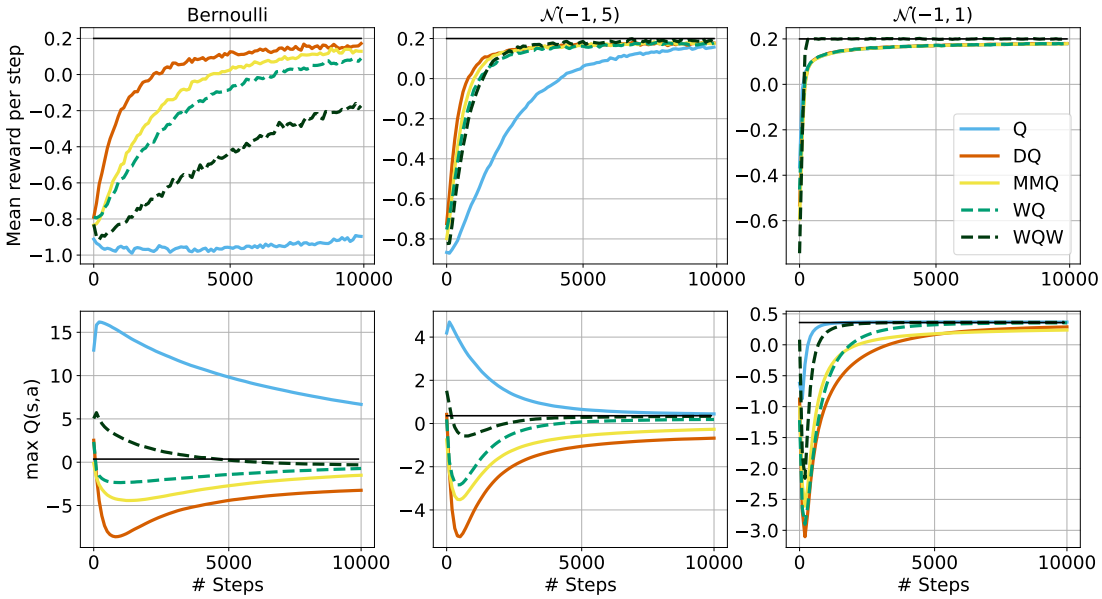


Figure 9: Grid world results with the three reward functions averaged over 10,000 experiments. Optimal policy is the black line.

computed by Weighted Q-Learning. It is worth noting that Weighted Q-Learning works well for both Gaussian and Bernoullian rewards, showing that WE is effective also with non-Gaussian distributions.

6.2 Continuous state spaces

6.2.1 Pricing problem

This problem consists in estimating the MEV of the gross profit in a pricing problem. In this MAB problem we validate the WE with infinite random variables (WE_∞) and we compare its performance against ME, DE, and MME, whose support (actions) has been discretized. It is crucial to estimate the value of the gross profit accurately in order to evaluate, for example, an investment decision or to analyze the profitability of products. The support (action) space is bounded but continuous and represents the price p to be shown to the user ($p \in [0, 10]$). The reserve price τ , which is the highest price a buyer is willing to pay, is modeled as a mixture of 3 Gaussian distributions with mean $\mu = \{2, 4, 8\}$, covariances $\sigma^2 = \{0.01, 0.01, 0.09\}$ and weights $w = \{0.6, 0.1, 0.3\}$. The revenue function $r_\tau(p)$ is p when $\tau \geq p$ and 0 otherwise. The maximum revenue is about 2.17. In each test the algorithms are fed with a set of samples $\mathcal{T} = \{\langle p_i, r_i \rangle\}_{i=1}^{n_s}$. Each sample is obtained by sampling a reserve price τ_i from the Gaussian mixture, a price p_i from a uniform distribution over the price range, and by evaluating the revenue function ($r_i = r_{\tau_i}(p_i)$). Clearly, the reserve price is unknown to the algorithm. Results are averaged over 50 runs and confidence intervals at 95% are shown. WE exploits a Gaussian process with a squared exponential kernel to generalize over the continuous price (GPs are fitted on $\mathcal{T}$), while ME, DE, and MME,

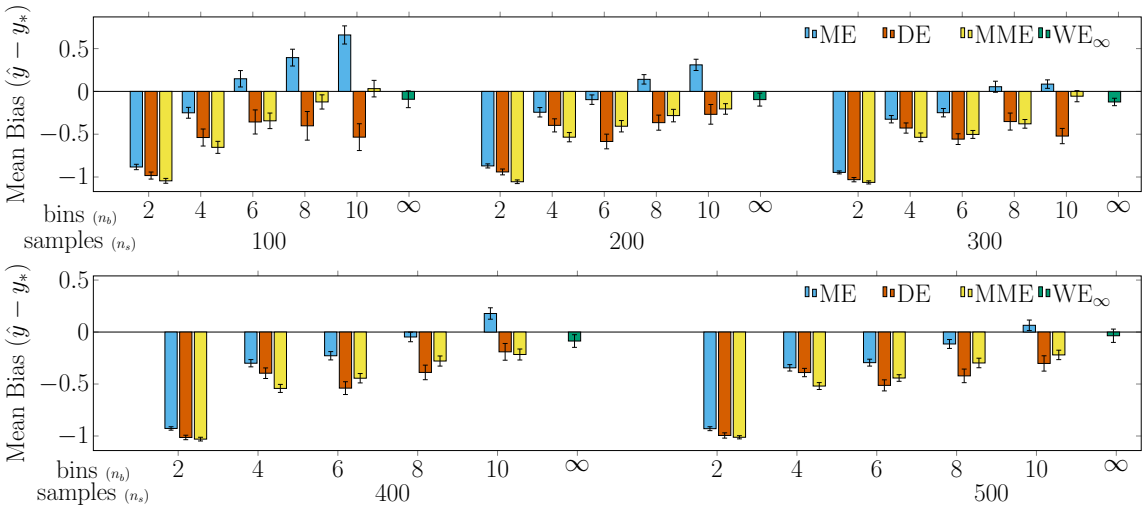


Figure 10: Mean bias obtained by ME, DE, MMME, and WE_∞, with different sample sizes and bins (only for ME and DE).

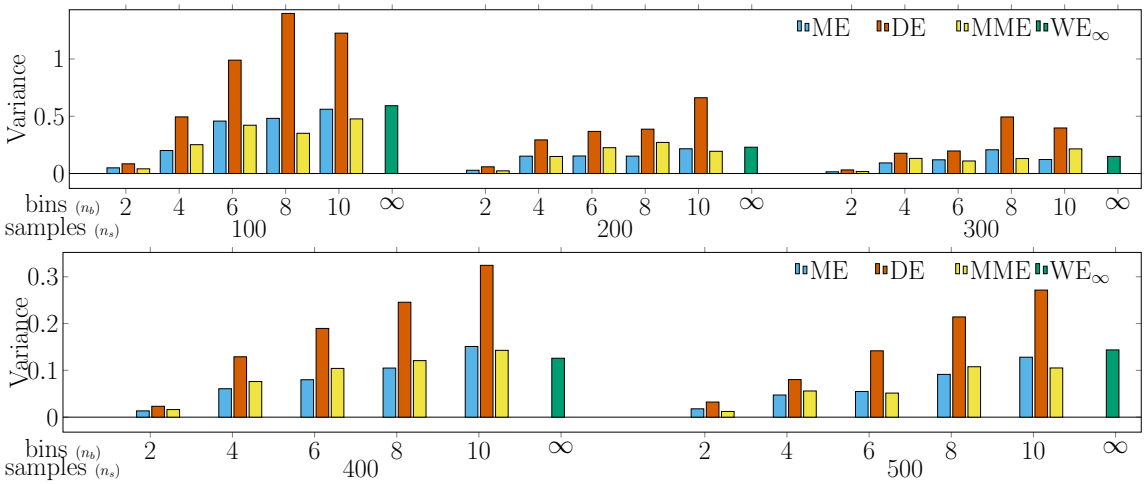
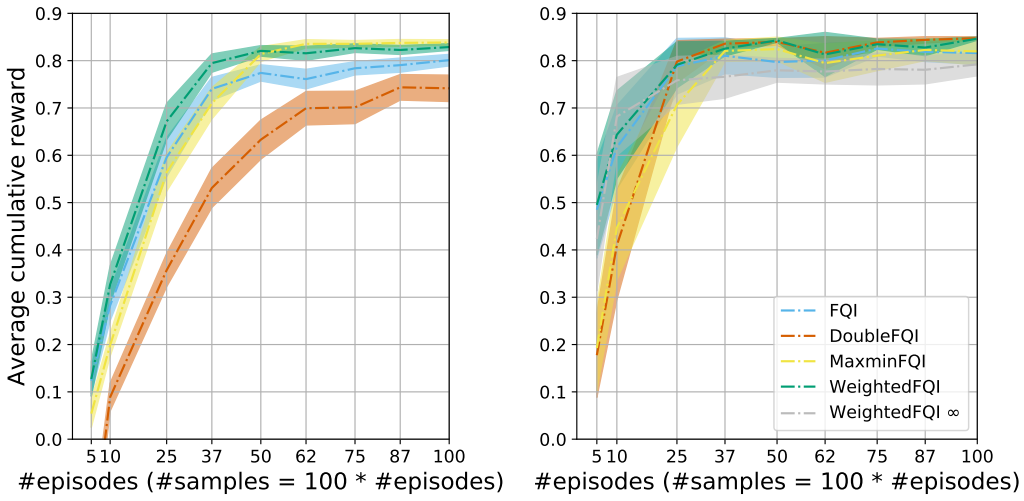


Figure 11: Variance of the bias obtained by ME, DE and WE_∞, with different sample sizes and bins.

discretize the price space into n_b uniformly spaced bins. As shown in Figure 10, the number n_b of optimal bins varies with the number n_s of available samples. This means that, once the samples have been collected, ME, DE, and MME, need an optimization phase for selecting the appropriate number of bins (not required by WE). WE is able to achieve the lowest or a comparable level of bias with every batch dimension even though it exploits a sensibly wider action space (infinite). In fact, as shown by the experiments, the performance of ME and DE may degrade as the number of bins increases, i.e., the action space increases. The variance of WE, as shown in Figure 11, is always comparable to the one of the best



(a) Discrete actions.

(b) Continuous actions.

Figure 12: Pendulum experiment.

configuration of the other estimators. From Figure 10 we can see that ME is prone to overestimation, while WE bias is almost always smaller or stays between ME and DE. [4] This discrete approximation introduces an additional (here negative) term to the bias.

6.2.2 Swing-Up Pendulum

A more complex scenario is represented by the continuous control problem analyzed in this section: the Swing-Up Pendulum with limited torque (Doya, 2000). The aim of these experiments is to compare FQI variants in continuous state space with both discrete and continuous actions. The peculiarity of this environment lies in the fact that the control with a limited torque ($u \in [-5, 5]$) makes the policy learning non-trivial. The continuous state space is $x = (\theta, \omega)$, where θ is the angle and ω is the angular velocity. An episode starts with $x_0 = (\theta_0, 0)$ where $\theta_0 \sim \mathcal{U}(-\pi, \pi)$, evolves according to the the dynamic system $\dot{\theta} = \omega$ and $ml^2\dot{\omega} = -\mu\omega + mgl\sin(\theta) + u$, and terminates after 100 steps. The physical parameters are mass $m = 1$, friction $\mu = 0.01$, length $l = 1$, $g = 9.8$, step time $\tau_0 = 0.01$. The reward depends on the height of the pendulum: $r(x) = \cos(\theta)$. The problem is discounted with $\gamma = 0.9$. The GP uses a squared exponential kernel with an independent length scale for each input dimension (ARD SE). The hyperparameters are tuned on the samples and the input values are normalized between $[-1, 1]$. We collected training sets of different sizes using a random policy. The FQI horizon is 10 iterations. The final performance of the algorithm is the *average reward*, calculated starting from 36 different initial angles $\theta_0 = \{\frac{2\pi k}{36} | k = \{0, 1, \ldots, 35\}\}$. We consider two settings for this problem: one with a discrete set of 11 evenly-spaced torque values in $[-5, 5]$ and another with a continuous action space. In the former setting we use a different GP for each action. The results in Figure 12(a) show

4. The reason why ME bias is not always positive, as stated by its theoretical properties, is due to the use of binning for the discretization of the continuous MAB.

that in the discrete case Weighted FQI reaches the highest average reward if the number of episodes is less or equal to 50 (with statistical confidence of 95% obtained over 100 runs), while for an higher number of episodes it is matched by the Maxmin variant of FQI. In the continuous case, on the other hand, Figure 12(b) shows that there is not statistical evidence of one algorithm outperforming the others (95% confidence bounds obtained over 25 runs). In this setting, the only algorithm that is able to directly handle the continuous space is the Weighted FQI defined in Algorithm 3 and indicated in the plot with the symbol ∞. The other algorithms use a GP and 100 actions to approximate the maximum. Interestingly, the continuous version of WE, does not improve performance. Our guess is that, in this setting, the variance introduced by considering an infinite number of actions outweighs the benefits of a lower bias.

6.3 Deep Reinforcement Learning scenario

In this section we compare the deep RL variants of the different algorithms. We measure the effects of the different estimators on the quality of the learned value functions and policies across all the experiments. First we run a proof of concept experiment on the Lunar Lander environment (Brockman et al., 2016). Then we test WDQN in three environments of the Minatar benchmark (Young and Tian, 2019). WDQN achieves a more accurate estimation of the expected return which often results in better policies. Furthermore, the experiments show the limits of DDQN, which is only an approximation of the DE and is still prone to overestimation. Finally we perform a set of experiments on two Atari games from the Arcade Learning Environment (ALE) (Bellemare et al., 2013) where DQN is known for overestimating the action-values (Van Hasselt et al., 2016).

All the agents are evaluated after each training epoch using the greedy policy, for Asterix we follow the evaluation protocol of Mnih et al. (2015). AvgDQN uses 3 approximators in all experiments. For each experiment we report both the average cumulative reward at each evaluation step and the prediction of the expected return, compared to the real discounted return obtained by the agents. Even if Thompson Sampling (TS) (Thompson, 1933) is the natural choice for WDQN, all the algorithms, unless explicitly stated, use an ε-greedy policy to guarantee fair comparison. For WDQN the greedy action is selected during training as the action corresponding to the maximum Q-value estimated with weight averaging, while during evaluation we take the action with the highest probability of being estimated as optimal as in Eq. (38). We found WDQN to be robust to the number of dropout samples used to compute the WE (e.g., $T \geq 30$). We used low values for the weight decay term, as in Farebrother et al. (2018), and tuned the dropout regularization coefficient for the problem at hand. The algorithms and the experimental setup have been developed using the open-source RL libraries MushroomRL[5] (D'Eramo et al., 2021) and OpenAI Gym (Brockman et al., 2016). Full description of the complete experimental setup and further results can be found in the appendix.

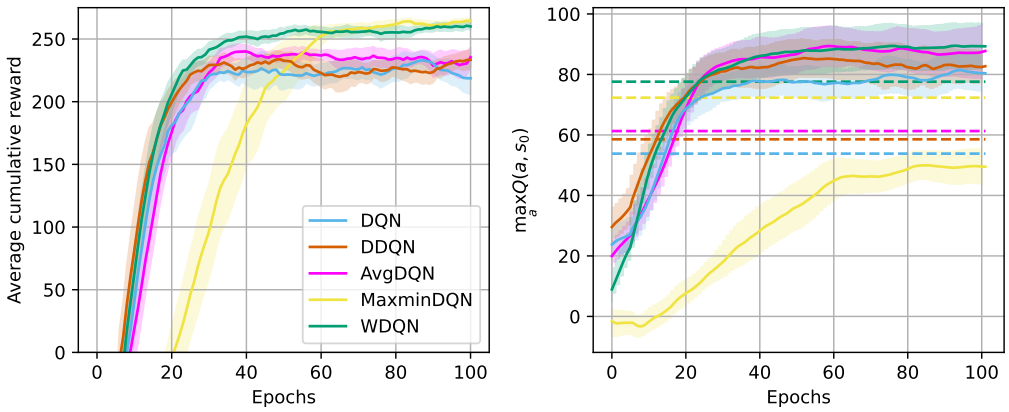


Figure 13: Learning curves on **Lunar Lander**. On the **left** the evaluation scores of each epoch (20k steps). On the **right** the maximum action-value estimated by each agent at the initial state of the MDP; the dashed lines are the real obtained discounted return. The results shown are the average of 20 independent runs, shaded areas are 95% confidence intervals. The curves are smoothed using a moving average of 10 epochs.

6.4 Lunar Lander

Lunar Lander is an MDP from the Gym collection. To solve the environment, the agent has to control the thrusters of a spacecraft to safely land in a specific location. In order to make prediction and control more challenging, we increased the stochasticity of the environment adding a 10% chance of repeating the last executed action, instead of the one selected by the agent. The three agents use the same exact network architecture, two hidden layers with 100 units and *relu* activation, the Adam optimizer with a learning rate of 3e−4, a target updated frequency of 300, a replay buffer of $10k$ transitions and ε-greedy exploration with ε linearly decreasing from 1 to 0.01 in the first $1,000$ steps. WDQN uses Concrete Dropout in each hidden layer. Figure 13 shows the learning curves of the three agents. WDQN and MaxminDQN achieve the highest average reward, but WDQN is more sample efficient and shows better prediction accuracy w.r.t. the other estimators.

6.5 Minatar

Minatar (Young and Tian, 2019) is an RL testbed with environments mimicking the dynamics of games from ALE, but with a simplified state representation. Minatar implements also sticky actions and difficulty ramping (Machado et al., 2018). The results of the experiment are shown in Figure 14. We use the same convolutional neural network and hyperparameters of Young and Tian (2019), but replace RMSProp with the Adam optimizer [6] and learning rate 1e−4. For WDQN we choose the dropout regularization coefficient with random search

5. Open source code at `https://github.com/MushroomRL/mushroom-rl`.
6. We found that Adam provides significantly more stable learning across all the agents.

and use the same value in the three games. MaxminDQN uses $N = 2$ approximators, as using $N = 3$ provided only a moderate performance increase in Seaquest and markedly worse performance on Breakout (see appendix).

On Breakout, WDQN achieves the highest average reward, and is more sample efficient in Freeway. MaxminDQN underperforms on Breakout, but it is able to widely outperform the other algorithms on Seaquest. DQN, DDQN and AvgDQN obtain similar average reward in all the considered environments. For what concerns the estimation of the expected return, WDQN shows lower bias in the first epochs of the learning procedure and converges to fairly accurate estimates. MaxminDQN is negatively biased in all the settings, while DQN has the largest positive bias. Since the number of actions and the regularization coefficient of WDQN are the same among the three environments, it is clear that the entropy of the WE weights depends heavily on the dynamics and the input space of the problem.

6.6 Atari games

The purpose of this experiment is to asses whether WDQN can be used to tackle the learning instability of vanilla DQN in the Atari Learning Environment (Bellemare et al., 2013) observed by Van Hasselt (2010). We consider two Atari games, Asterix and Wizard of Wor, in which Van Hasselt et al. (2016) showed that the overestimation bias causes vanilla *DQN* to fail. We use the same neural network and hyperparameters of Mnih et al. (2015), replicating the same experimental setting. For WDQN we use Concrete Dropout only in the fully connected layer after the convolutional block. Figure 15 shows the result of the comparison in terms of average reward and prediction accuracy of WDQN against the DQN and DDQN. DQN is unstable and widely overestimates the discounted return. This instability causes the performance to collapse in both scenarios. DDQN manages to stabilize the learning performance on Asterix, but, in our settings, is not enough to yield satisfactory performance on Wizard of Wor differently from what observed by Van Hasselt et al. (2016), probably due to minor implementation details (Henderson et al. (2017)). However, it is worth mentioning the the performance of DDQN would most likely improve in this scenario by reducing the target update frequency. WDQN, on the other hand, manages to control the bias and the learning instability in both cases, achieving a comparable performance and prediction accuracy w.r.t. DDQN in Asterix and outperforming the other two baselines on Wizard of Wor. Finally, in Figure 16 we show the comparison of WDQN against methods using multiple Q-networks, namely MaxminDQN and AvgDQN. On Asterix, MaxminDQN is able to achieve a much higher cumulative reward, while showing a large negative bias. On Wizard of Wor, WDQN achieves the best scores, but due to the high-variance of the experiments, the performance difference is not statistically significant. MaxminDQN, on the other hand, is accurate in estimating the maximum action-value of the initial state. It is interesting to note that increasing the number of approximators from 2 to 3, MaxminDQN exhibited poor performance and learning instability, showing the importance of properly calibrating N. Results for MaxminDQN with 3 approximators are reported in the appendix. AvgDQN achieves good results, behaving similarly to DDQN on Asterix, but showing a large variance in performance on Wizard of Wor. The number of approximators of AvgDQN was chosen as to use a similar time budget w.r.t. MaxminDQN and WDQN.

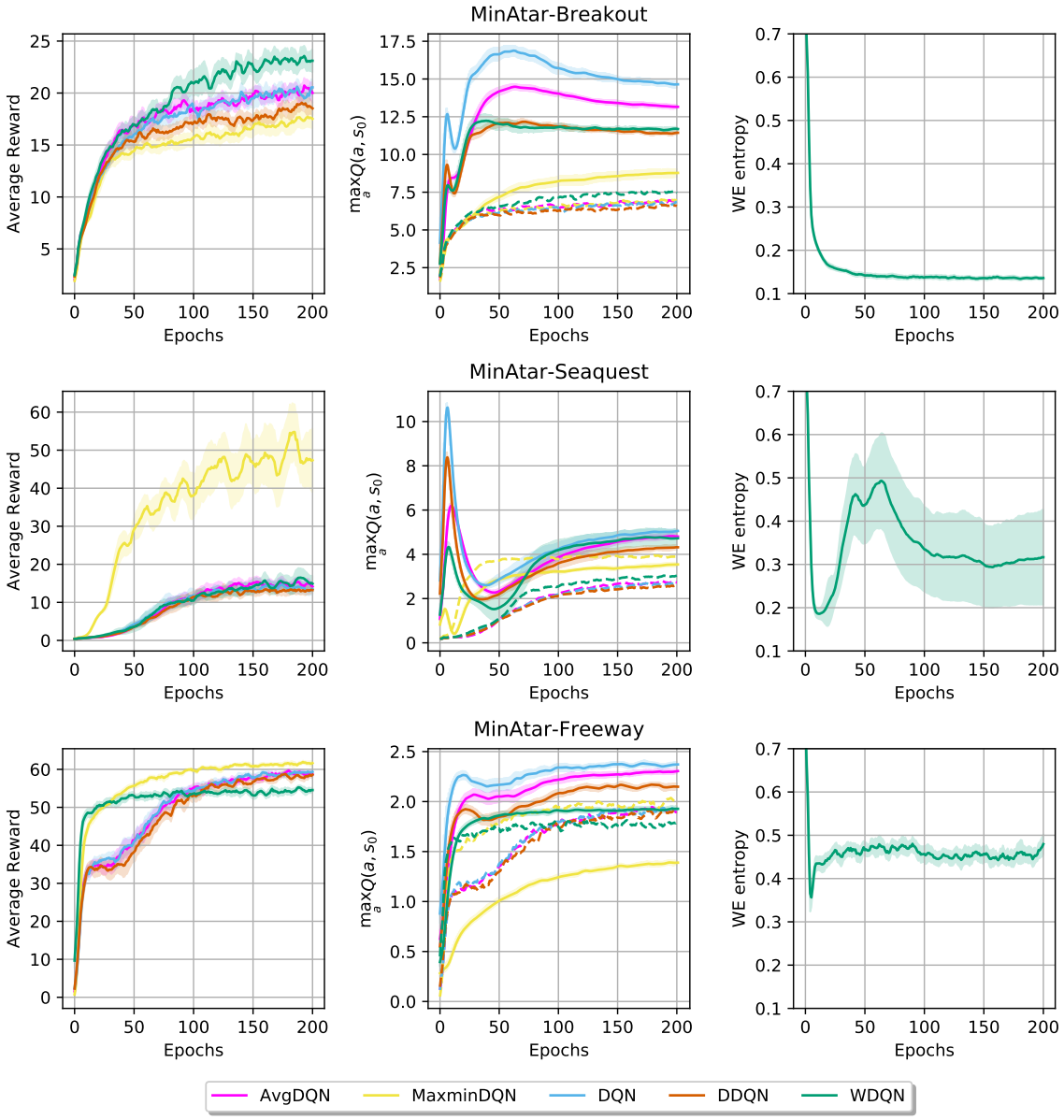


Figure 14: Learning curves of the analyzed DQN variants on three **Minatar** games. In the **leftmost** column, we show the evaluation scores after each of the 200 training epochs, where 1 epoch corresponds to 25k steps. In the **middle** column we report for each game and agent the estimate of the expected return w.r.t. the initial state of the environment; the dashed lines indicate the real discounted return obtained by the agents. In the **rightmost** column we show the moving average of the entropy of the WDQN weights. The results shown here are averaged over 20 independent runs and the shaded areas represent 95% confidence intervals. The curves are smoothed using a moving average of 10 epochs.

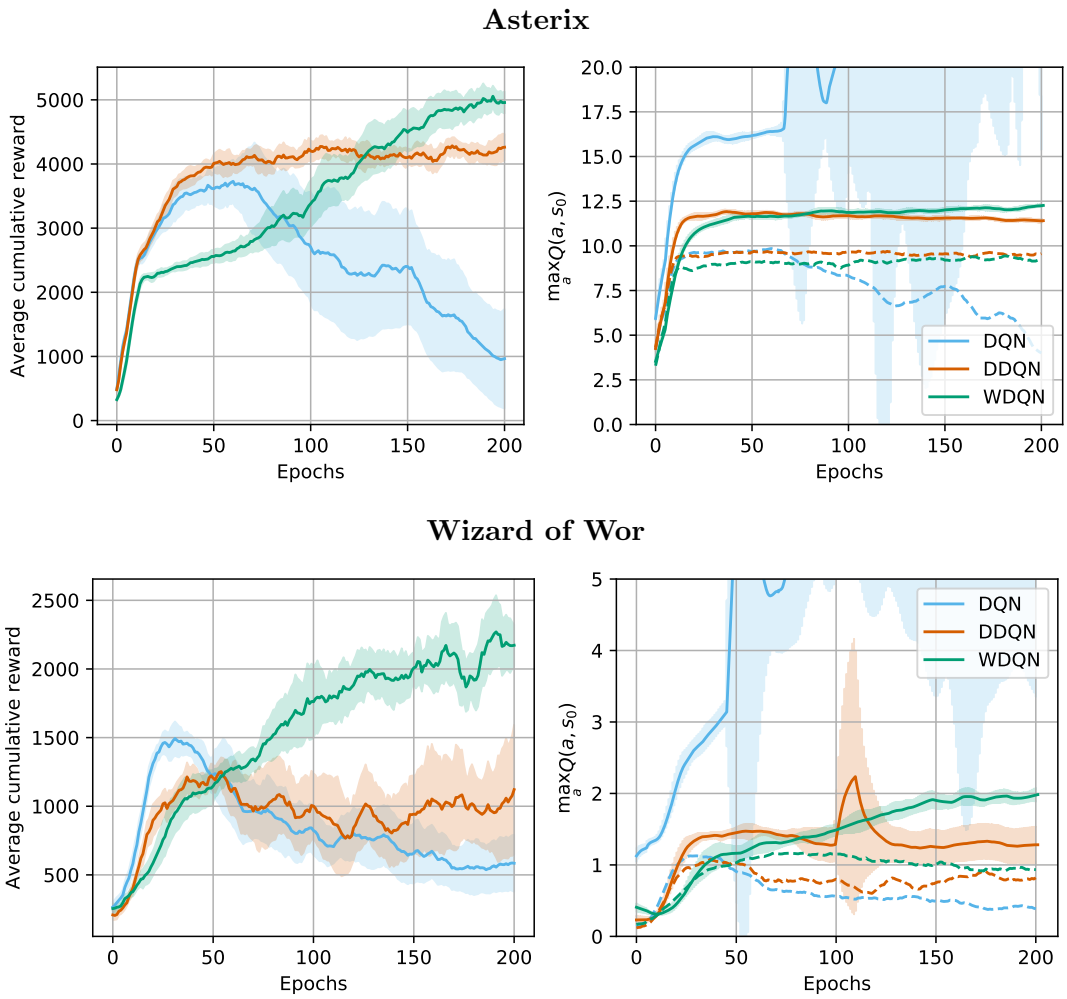


Figure 15: The curves on the **left** show the evaluation scores. The figure in the **right** reports, for each agent, the estimate of the expected return w.r.t. the starting screen of the game; the dashed curves indicate the real discounted return for each agent. Each epoch corresponds to 1M frames. The results are averaged over 10 independent runs and the shaded areas represent 95% confidence intervals. The curves are smoothed using a moving average of 10 epochs to improve readability.

It is worth mentioning that empirically we noticed that the optimizer has a large impact on the performance. In particular, we observed that using Adam (Kingma and Ba, 2015), following modern best practices (Hessel et al., 2018; Dabney et al., 2018), was enough to stabilize learning for DQN, even if the overestimation bias remained large. These observations are confirmed by the recent work of Gogianu et al. (2021), that thoroughly analyze the impact of different optimization techniques in the learning dynamics of deep RL algorithms. However, we remark again the objective of this was to asses the robustness of WDQN in the

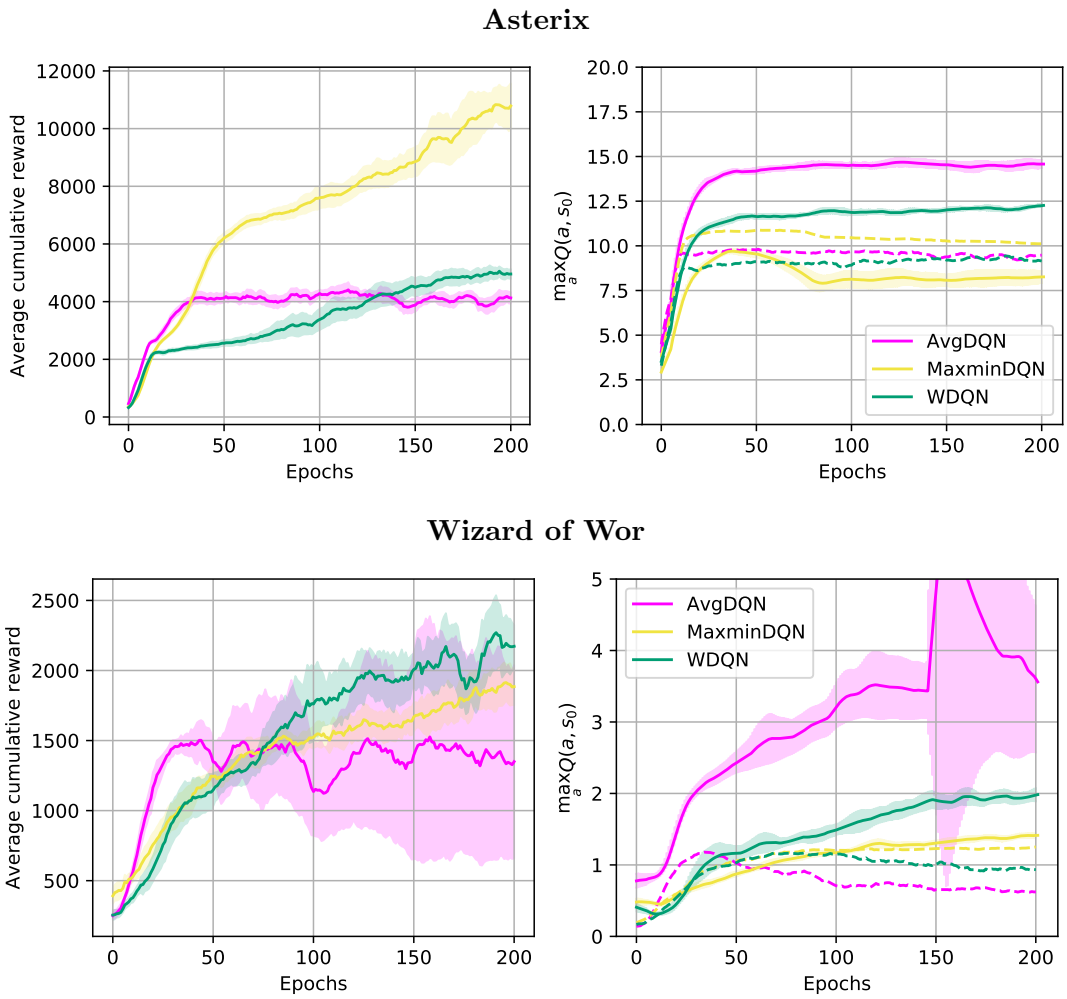


Figure 16: Comparison of methods using multiple approximators against WDQN on Atari games. The results shown here are the average of 10 independent runs and the shaded areas represent 95% confidence intervals. The curves are smoothed using a moving average of 10 epochs to improve readability.

same settings where ME fails to learn and not to claim state-of-the-art results that would require computationally expensive hyperparameter tuning.

7. Additional references

We already discussed several approaches to tackle the problem of estimation biases in Q-Learning. In this section, we make a brief summary of this line of research, providing an overview on how the same problem has been addressed in different RL settings. Thrun and Schwartz (1993) discussed the issues of overestimation bias in Q-Learning with function approximation, in particular they provided a theoretical analysis of the phenomenon showing

how noisy predictions can cause systematic overestimation of values. Interestingly, Thrun and Schwartz (1993) show examples in which Q-Learning is *expected* to fail. Van Hasselt (2010) introduced Double Q-Learning and showed that overestimation can also hinder the performance of tabular Q-Learning by making the algorithm converge at an impracticably slow rate. Notably, they identified the root cause of the overestimation in the positive bias of ME by analyzing the problem from a statistical perspective, which was further expanded in (Van Hasselt, 2013). Estimating the MEV is, in fact, a known problem in statistics (Smith and Winkler, 2006; Tibshirani and Tibshirani, 2009) and for which an unbiased estimator, in general, does not exist (Blumenthal and Cohen, 1968; Bhaeiyal Ishwaei et al., 1985).

Weighted Q-Learning has to be considered, then, in the context of the existing algorithms that attempt to address the bias of Q-Learning. Among the others, Bias-corrected Q-Learning (Lee et al., 2013) improves the learning stability of Q-Learning by using a correction term that depends on the variance of the rewards. The Weighted Double Q-Learning algorithm (Zhang et al., 2017) uses instead a combination of ME and DE to balance between the overestimation and underestimation of the two estimators. The already discussed Maxmin Q-Learning (Lan et al., 2020) introduced the Maxmin Estimator and showed how it can be used to effectively control the error in the setting studied by Thrun and Schwartz (1993). Overestimation of the action-values can be even more problematic in the DQN algorithm (Mnih et al., 2015), due to the high variance typical of deep RL approaches. The Double DQN algorithm (Van Hasselt et al., 2016) introduced the use of DE in DQN, which shows better estimate of action-values and superior performance w.r.t. vanilla DQN. Another way to limit the overestimation is by reducing the variance of the estimate, as in the case of Averaged DQN (Anschel et al., 2017). The Weighted Estimator, being a convex combination of action-values, is also related to softmax operators typically used in RL (Sutton and Barto, 1998; Asadi and Littman, 2017).

The problem of biased value estimates has also been studied in maximum-entropy RL (Haarnoja et al., 2017), where recent works (Fox et al., 2016; Fox, 2019) studied how to learn unbiased estimates by schedules of the temperature of the Soft Q-Learning updates. Finally, overestimation is also detrimental in deep actor-critic algorithms, such as Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2015). Notably, the Twin Delayed DDPG algorithm (TD3) (Fujimoto et al., 2018) uses a clipped version of the Double Q-Learning update that significantly improves the approximation of the value function and the overall performance of DDPG.

8. Conclusions

Off-policy learning offers a sample-efficient solution to the problem of optimal decision making under uncertainty and it is an integral part of modern approaches for designing autonomous intelligent agents. Accurate estimation of action-value functions is a constitutional element of algorithms based on these ideas. However, state-of-the-art methods show to be unreliable when applied in stochastic environments, a typical characteristic of real-world problems.

In this paper we presented a set of principled tools and techniques, grounded in statistics, to address this problem. We showed that the introduced Weighted Estimator can be effectively applied in different settings and that it is orthogonal to the underlying learn-

ing procedure. Empirical results suggest, in agreement with previous works (Van Hasselt, 2010; D'Eramo et al., 2016; Lan et al., 2020), that none of the estimators outperform the others in general, but that final performance are tightly related to specific characteristics of the problem at hand. Furthermore, due the peculiarities of the shape of some reward functions, estimation biases may also turn out to be beneficial. However, while it is possible to construct toy problems to show and partially understand this phenomenon (see for example Lan et al. (2020)), in practice there is no good heuristic to decide whether a positive or negative bias might hurt performance or not. This is particularly true when no prior knowledge on the environment is available and/or different regions of the state-action space have radically different characteristics. In this context, the Weighted Estimator strikes as a strong contender due to its robustness and the unpolarized nature of its bias. In fact, our method successfully overcomes the limits of well-established alternatives, generally without compromising computational time and ease of implementation.

The main outcome of this work is a unifying view advocating for the adoption of the Weighted Estimator in RL, and substantiating this claim by proposing an extension to the deep RL setting. There are multiple possible directions for future work. From the theoretical perspective, future work should analyze the different estimators taking into account the sequential nature of the decision making process. On the same note, dropout uncertainty estimates can be poorly calibrated (Osband et al., 2018; Lakshminarayanan et al., 2017) and do not propagate uncertainty through the Bellman equation (Osband et al., 2018). Future work should combine ideas that solve this issue (e.g., Metelli et al. (2019); Osband et al. (2018)) with methods that allow to compute the weights of the Weighted Estimator with acceptable precision. Finally, as already mentioned, the combination of the Weighted Estimator with state-of-the-art value-based RL algorithms should be further investigated.

Appendix

The appendix is organized as follows. In Appendix A, we provide additional empirical results and further details on the experimental settings used to evaluate the agents. In Appendix B, we compare WE against softmax operators frequently used in RL. Then, in Appendix C, we provide a comparison on the grid world environment for the Monte Carlo approximation of WE against the approximation obtained with the trapezoidal rule.

Appendix A. Experiments details and additional results

In this section we provide additional empirical results and details on the experimental setup used for the empirical evaluation of the proposed methods.

For the implementation of the algorithms and the simulation environments we rely on the following open-source libraries:

- numpy (Harris et al., 2020);
- MushroomRL (D'Eramo et al., 2021);
- Gym (Brockman et al., 2016);
- ALE (Bellemare et al., 2013);
- MinAtar (Young and Tian, 2019);
- PyTorch (Paszke et al., 2019).

The experiments were run on a server with an Intel Xeon Silver 4116 CPU with NVIDIA Titan V GPUs. Most of the hyperparameters were selected based on published results, as we mention for each experiment in Section 6.

A.1 Lunar Lander

For Lunar Lander (Brockman et al., 2016) we limit the length of an episode (during evaluation and training) at 1,000 steps. To increase the complexity of the problem, we make the environment stochastic using sticky actions (see previous subsection) with a $p_{\text{repeat}} = 10\%$. We use a small neural network with only two hidden layers. For WDQN, we use Concrete Dropout on each hidden layer. Table 1 reports the hyperparameters used to train the agents. Hyperparameters were tuned on vanilla DQN and kept fixed for the competitors. Hyperparameters specific to WDQN were tuned with a small random search. For MaxminDQN we ran the experiments with both 2 and 3 approximators, since 3 approximators led to clearly worse performance (as shown in Figure 17) we did not increase the number of estimators further.

A.2 MinAtar

MinAtar (Young and Tian, 2019) offers a collection of environments resembling games from the Atari Learning Environment (Bellemare et al., 2013). The state representation of MinAtar environments is a matrix with multiple channels, where each channel gives specific

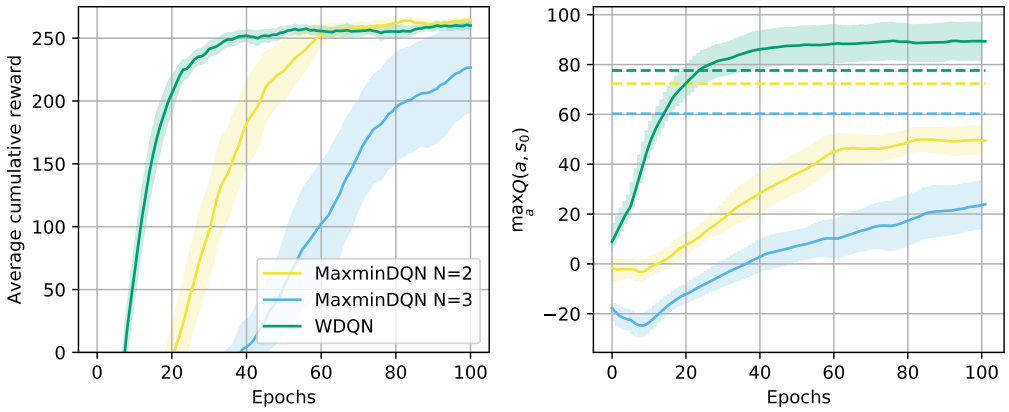


Figure 17: Learning curves on **Lunar Lander** for MaxminDQN with different number of approximators. The results averaged over 20 independent runs.

information about some aspects of the environment (e.g., position and speed of a moving object). MinAtar implements the ALE modifications suggested by Machado et al. (2018), i.e., sticky actions and difficulty ramping.

We use the same hyperparameters and network of Young and Tian (2019), but we use Adam for training, which yields good results across all the considered algorithms. The learning rate used for Adam is the default used by several popular DQN implementations. For WDQN, we select the additional hyperparameters by choosing the value providing better performance with a small search and keeping it fixed across the three games. Table 2 shows the relevant hyperparameters.

For WDQN, we run an additional experiment to asses the impact of the dropout regularization coefficient. We set the initial dropout rate at $p = 0.5$ (which corresponds to the maximum entropy) and test the agents using regularization coefficients of different magnitude. The results, reported in Figure 18, show how higher levels of regularization generally correspond to higher entropy of the weights used to compute the WE. Finally, in Figure 19, we show results for MaxminDQN with $N = 2$, and $N = 3$ approximators; we also show WDQN learning curves as reference.

A.3 Atari games

For the experiment on the Atari games we use the settings of Mnih et al. (2015). The additional hyperparameters introduced by WDQN are tuned with a - very small - random search. Concrete Dropout is used only on the neurons of the last (dense) hidden layer, as usually done in convolutional neural networks; which means that the stochastic forward passes needed to compute the Weighted Estimator can be efficiently performed through the last layer alone. Agents are evaluated during training every 1M frames, with the 30 no-op starting condition. During evaluation the episode length is capped at 30 minutes. Table 3 reports a list of additional relevant hyperparameters.

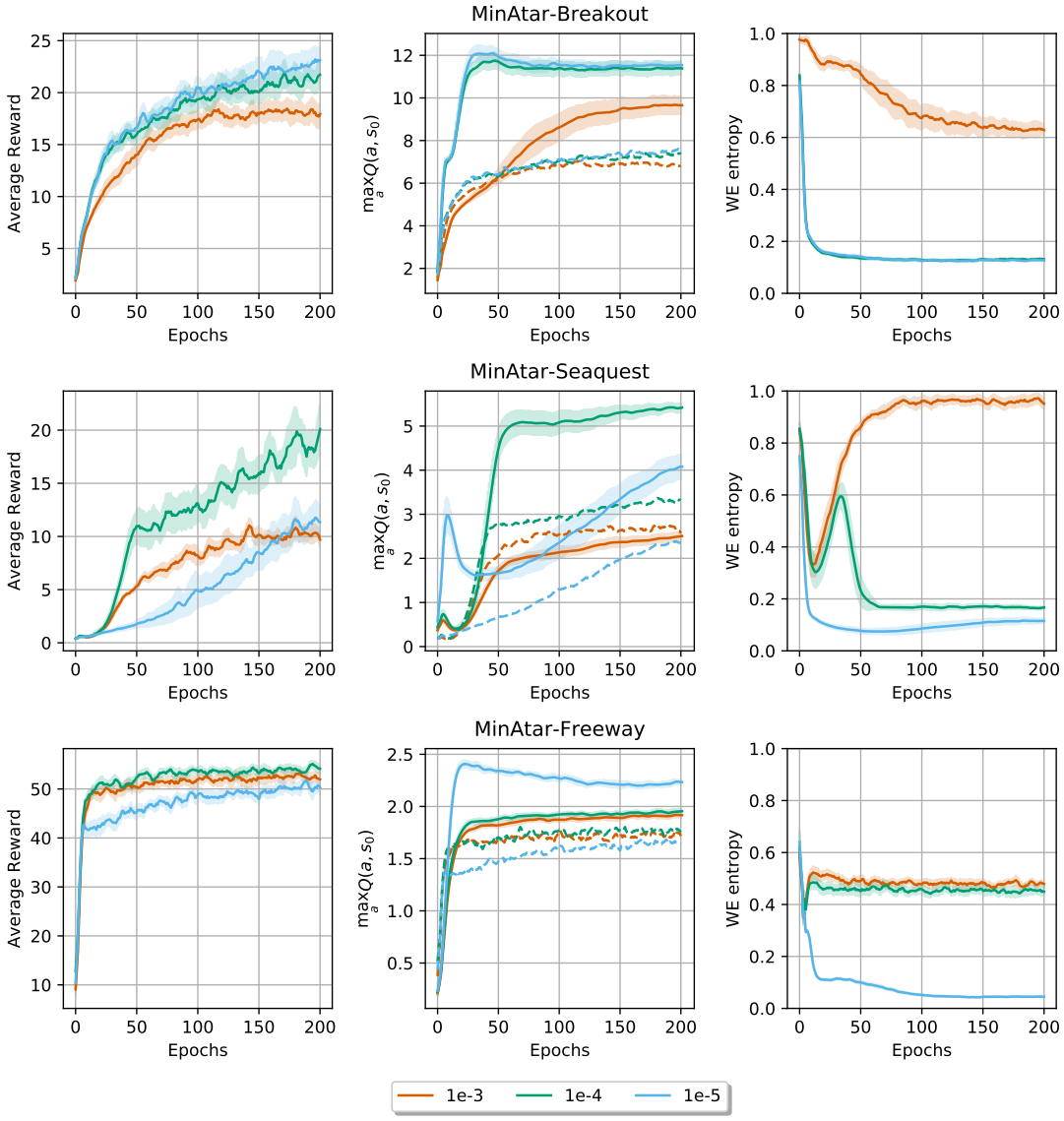


Figure 18: Learning curves on three **Minatar** games for WDQN with 3 different dropout regularization values. The results are averaged over 20 independent runs.

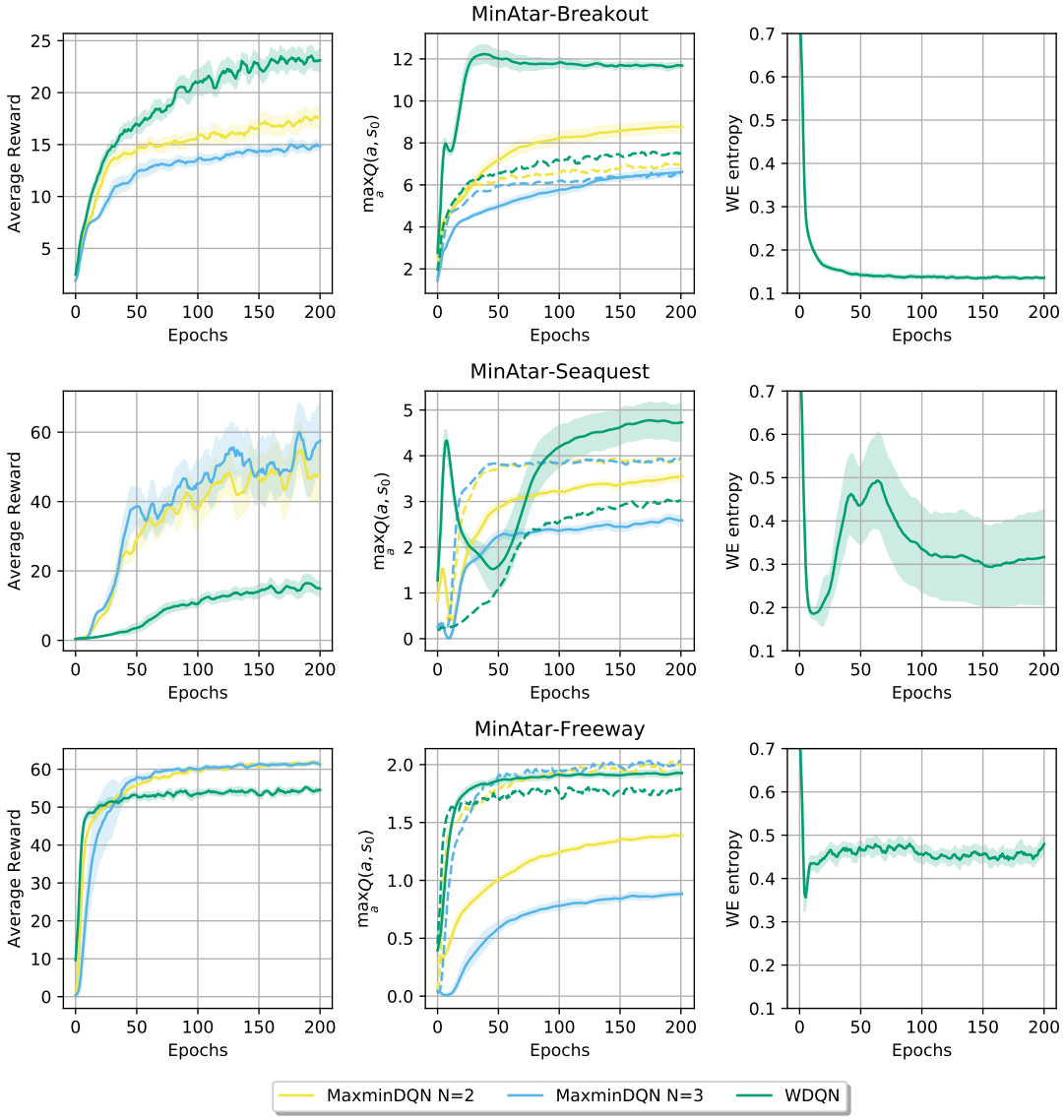


Figure 19: Learning curves on three **Minatar** games for MaxminDQN with different numbers of approximators. The results are averaged over 20 independent runs.

Table 1: DQN hyperparameters on Lunar Lander. Hyperparameters marked with a $*$ are used only for WDQN.

Hyperparameter	**Value**
Units per layer	$[100, 100]$
Activation	$relu$
Optimizer	Adam
Learning rate	3e−4
Batch size	32
Loss function	MSE
Training frequency	1
Target network update frequency	300
Min. memory size	250
Max. memory size	$10,000$
Discount factor (γ)	0.99
Initial exploration rate (ε_{start})	1.0
Final exploration rate (ε_{end})	0.01
Exploration steps	$1,000$
Evaluation exploration rate (ε_{test})	0.0
MC dropout samples*	50
Weight decay coefficient (λ)*	1e−6
Dropout regularization coefficient (ζ)*	2.5e−3
Initial dropout rate (p)*	0.2
N. approximators – AvgDQN (n_{avg})	3
N. approximators – MaxminDQN (n_{mm})	2

Figure 20 shows results for MaxminDQN when increasing the number of approximators from $N = 2$ to $N = 3$. Besides to the clear impact in terms of learning speed due to the reduced number of samples to train each model, the negative bias introduced by MME heavily affects performance in Asterix, where the learning algorithm completely fails.

Appendix B. Comparison against softmax operators

Weighted Q-Learning uses a convex combination of action values, resulting from the use of the WE, to compute the target of the update. We resort to the grid world problem to provide a comparison of Weighted Q-Learning against other methods based on a convex combination of action values. We consider the standard softmax operator (Sutton and Barto, 1998), and the *mellowmax* operator (Asadi and Littman, 2017). To directly compare against WE, we use a temperature value of 1 for both operators. Figure 21 shows that the standard softmax performs considerably worse than other methods, obtaining strongly suboptimal performance and inaccurate value estimation. Mellowmax behaves better than softmax in terms of performance, but tends to largely underestimate the action-value.

Table 2: DQN hyperparameters on Minatar environments. Hyperparameters marked with a $*$ are used only for WDQN.

Hyperparameter	**Value**
Optimizer	Adam
Learning rate	1e−4
Batch size	32
Loss function	Huber
Training frequency	1
Target network update frequency	$1,000$
Min. memory size	$5,000$
Max. memory size	$100,000$
Discount factor (γ)	0.99
Initial exploration rate (ε_{start})	1.0
Final exploration rate (ε_{end})	0.1
Exploration steps	$100,000$
Evaluation exploration rate (ε_{test})	0.0
MC dropout samples*	100
Weight decay coefficient (λ)*	1e−6
Dropout regularization coefficient (ζ)*	1e−4
Initial dropout rate (p)*	0.1
N. approximators – AvgDQN (n_{avg})	3
N. approximators – MaxminDQN (n_{mm})	2

Table 3: DQN hyperparameters on Atari. Hyperparameters marked with a $*$ are used only for WDQN.

Hyperparameter	**Value**
MC dropout samples – WDQN	100
Weight decay coefficient – WDQN (λ)	1e−6
Dropout regularization coefficient – WDQN (ζ)	5e−2
Initial dropout rate (p)	0.5
N. approximators – AvgDQN (n_{avg})	3
N. approximators – MaxminDQN (n_{mm})	2

Appendix C. Computational analysis for Weighted Estimator

We provide an analysis of the difference between the integration and the sampling methods for computing the weights of WE. In Figure 22, we show the maximum action-value in the starting state of the grid world problem described in Section 6, using the setting with reward distributed as $\mathcal{N}(5,1)$. The experiment was performed using standard scientific computing libraries for both numerical integration and random sampling. We show that the maximum action-value estimates obtained with the integration method with $1,000$ and

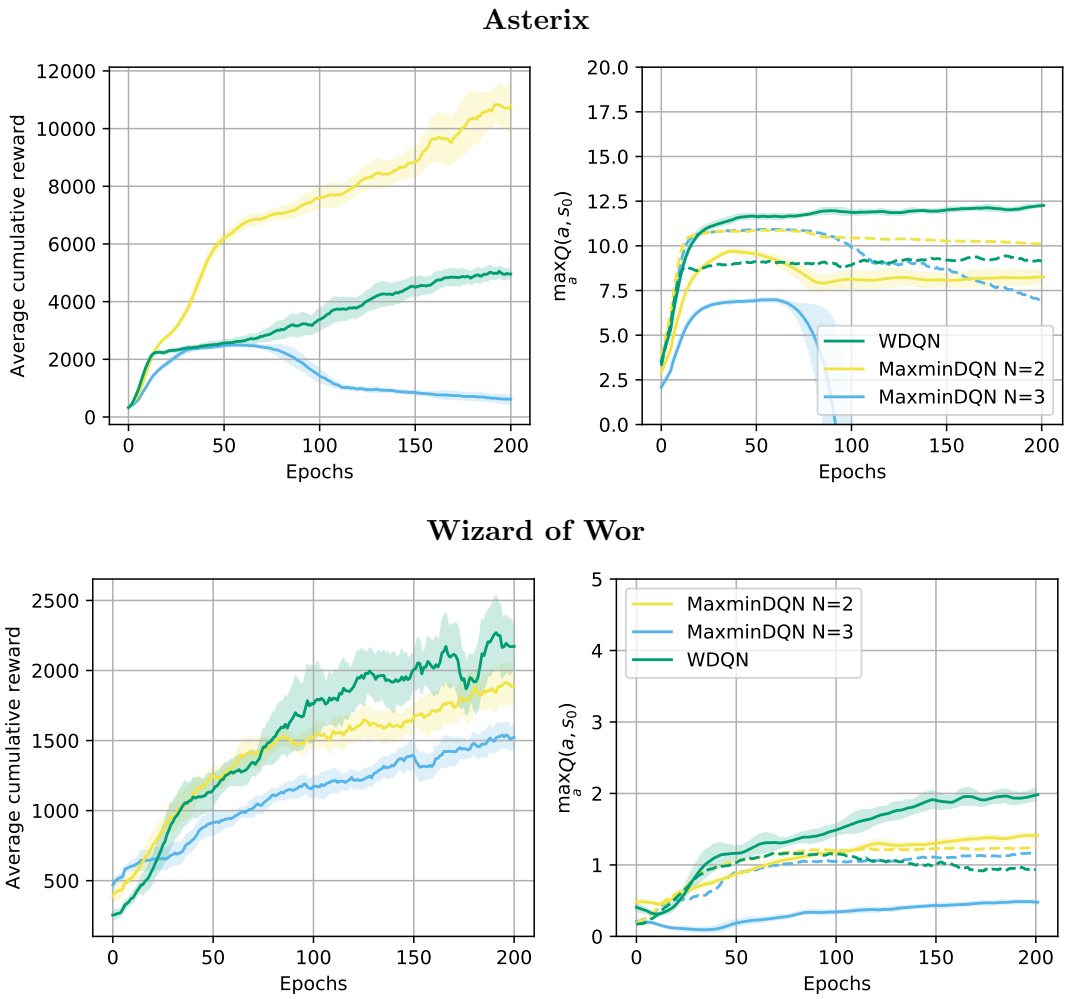


Figure 20: Learning curves on **Atari games** for MaxminDQN with different numbers of approximators. Results are averaged over 10 independent runs.

25 trapezoids, and with the sampling methods with 100 samples, do not have substantial differences. Moreover, the average computation times to compute the weights of the WE are 0.619ms and 0.204ms, respectively for the integration method with $1,000$ and 25 trapezoids, and 0.0325ms for the sampling method with 100 samples. This result shows that the sampling method can provide an accurate estimate of the weights of the WE while being computationally more advantageous than the integration method.

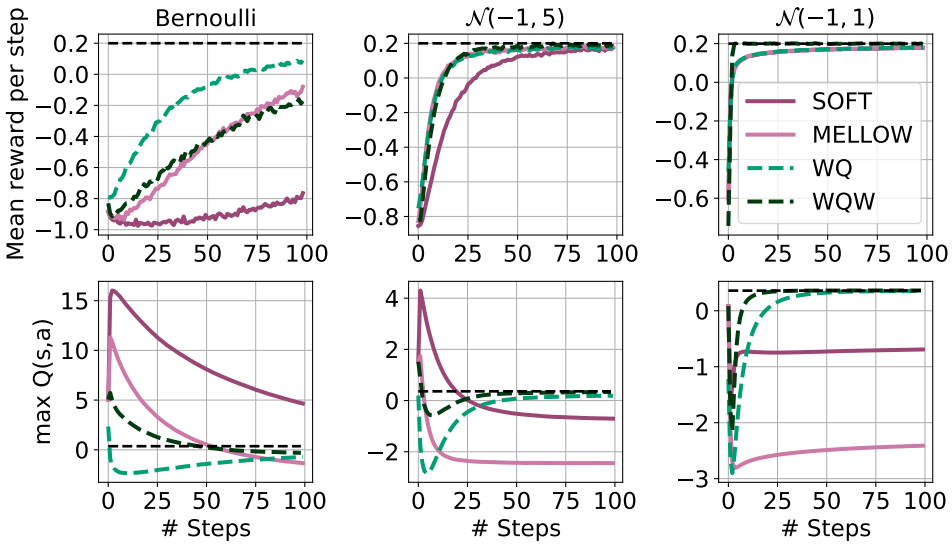


Figure 21: Comparison of different softmax operators against WE. Results averaged over 10,000 experiments. Optimal policy is the black line.

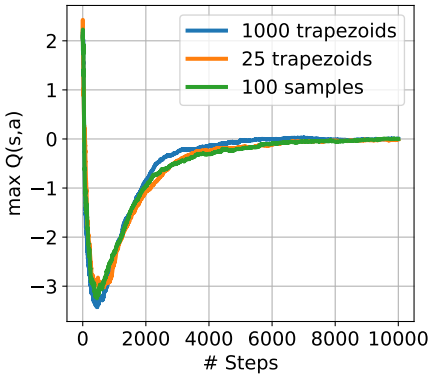


Figure 22: Estimates of the maximum action-value in the starting state of the grid world using integral and sampling methods for computing the Weighed Estimator.

.

References

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-Volume 70*, pages 176–185. JMLR. org, 2017.

Kavosh Asadi and Michael L Littman. An alternative softmax operator for reinforcement learning. In *International Conference on Machine Learning*, pages 243–252. PMLR, 2017.

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

Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark, 2020.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47: 253–279, jun 2013.

Marc G. Bellemare, Georg Ostrovski, Arthur Guez, Philip S. Thomas, and Rémi Munos. Increasing the action gap: New operators for reinforcement learning. In *Proceedings of the thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, page 449–458. JMLR.org, 2017.

Richard Bellman. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–515, 1954.

D. Bhaeiyal Ishwaei, Shabma Divakar, and Krishnamoorthy K. Non-existence of unbiased estimators of ordered parameters. *Statistics*, 16(1):89–95, 1985. doi: 10.1080/02331888508801827.

Saul Blumenthal and Arthur Cohen. Estimation of the larger of two normal means. *Journal of the American Statistical Association*, 63(323):861–876, 1968.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Mark Collier and Hector Urdiales Llorens. Deep contextual multi-armed bandits. *CoRR*, abs/1807.09809, 2018.

Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *AAAI Conference on Artificial Intelligence*, 2018.

Andreas Damianou and Neil Lawrence. Deep gaussian processes. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, volume 31, pages 207–215, 29 Apr–01 May 2013.

Richard Dearden, Nir Friedman, and Stuart Russell. Bayesian Q-learning. In *Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, pages 761–768. American Association for Artificial Intelligence, July 1998. ISBN 978-0-262-51098-1.

Carlo D'Eramo, Marcello Restelli, and Alessandro Nuara. Estimating maximum expected value through gaussian approximation. In *International Conference on Machine Learning*, pages 1032–1040, 2016.

Carlo D'Eramo, Alessandro Nuara, Matteo Pirotta, and Marcello Restelli. Estimating the maximum expected value in continuous reinforcement learning problems. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Carlo D'Eramo, Andrea Cini, and Marcello Restelli. Exploiting Action-Value Uncertainty to Drive Exploration in Reinforcement Learning. July 2019. doi: 10.1109/IJCNN.2019.8852326. ISSN: 2161-4407.

Kenji Doya. Reinforcement learning in continuous time and space. *Neural computation*, 12 (1):219–245, 2000.

Carlo D'Eramo, Davide Tateo, Andrea Bonarini, Marcello Restelli, and Jan Peters. Mushroomrl: Simplifying reinforcement learning research. *Journal of Machine Learning Research*, 22(131):1–5, 2021.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(Apr):503–556, 2005.

Jesse Farebrother, Marlos C. Machado, and Michael Bowling. Generalization and regularization in dqn, 2018.

Roy Fox. Toward provably unbiased temporal-difference value estimation. In *Optimization Foundations for Reinforcement Learning workshop (OPTRL @ NeurIPS)*, 2019.

Roy Fox, Ari Pakman, and Naftali Tishby. Taming the noise in reinforcement learning via soft updates. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, page 202–211. AUAI Press, 2016.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596, 2018.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48, pages 1050–1059, 20–22 Jun 2016.

Yarin Gal, Rowan Thomas McAllister, and Carl Edward Rasmussen. Improving pilco with bayesian neural network dynamics models. In *Data-Efficient Machine Learning workshop, ICML 2016*, 2016.

Yarin Gal, Jiri Hron, and Alex Kendall. Concrete dropout. In *Advances in Neural Information Processing Systems 30*, pages 3581–3590. Curran Associates, Inc., 2017.

Florin Gogianu, Tudor Berariu, Mihaela Rosca, Claudia Clopath, Lucian Busoniu, and Razvan Pascanu. Spectral normalisation for deep reinforcement learning: an optimisation perspective, 2021.

Michael Grossman and Robert Katz. *Non-Newtonian Calculus: A Self-contained, Elementary Exposition of the Authors' Investigations...* Non-Newtonian Calculus, 1972.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. volume 70 of *Proceedings of Machine Learning Research*, pages 1352–1361, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

Charles R Harris, K Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep Reinforcement Learning that Matters. *arXiv preprint arXiv:1709.06560*, 2017. URL `https://arxiv.org/pdf/1709.06560.pdf`.

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 *The Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Gregory Kahn, Adam Villaflor, Vitchyr Pong, Pieter Abbeel, and Sergey Levine. Uncertainty-aware reinforcement learning for collision avoidance. *CoRR*, abs/1702.01182, 2017.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems 28*, pages 2575–2583. Curran Associates, Inc., 2015.

Leslie Kish. *Survey sampling*. J. Wiley, 1965. Google-Books-ID: xiZmAAAAIAAJ.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 6402–6413. Curran Associates, Inc., 2017.

Qingfeng Lan, Yangchen Pan, Alona Fyshe, and Martha White. Maxmin q-learning: Controlling the estimation bias of q-learning. In *International Conference on Learning Representations*, 2020.

Donghun Lee, Boris Defourny, and Warren B Powell. Bias-corrected q-learning to control max-operator bias in q-learning. In *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pages 93–99. IEEE, 2013.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2015.

Marlos C. Machado, Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew J. Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61: 523–562, 2018.

Ali Malik, Volodymyr Kuleshov, Jiaming Song, Danny Nemer, Harlan Seymour, and Stefano Ermon. Calibrated model-based deep reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 4314–4323. PMLR, 09–15 Jun 2019.

Luca Martino, Víctor Elvira, and Francisco Louzada. Effective sample size for importance sampling based on discrepancy measures. *Signal Processing*, 131:386–401, February 2017. doi: 10.1016/j.sigpro.2016.08.025.

Alberto Maria Metelli, Amarildo Likmeta, and Marcello Restelli. Propagating uncertainty in reinforcement learning via wasserstein barycenters. In *Advances in Neural Information Processing Systems 32*, pages 4335–4347, 2019.

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.

Tetsuro Morimura, Masashi Sugiyama, Hisashi Kashima, Hirotaka Hachiya, and Toshiyuki Tanaka. Parametric return density estimation for reinforcement learning. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, pages 368–375, Catalina Island, CA, July 2010. AUAI Press. ISBN 978-0-9749039-6-5.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In *Advances in Neural Information Processing Systems*, pages 4026–4034, 2016.

Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. In *Advances in Neural Information Processing Systems 31*, pages 8617–8629. Curran Associates, Inc., 2018.

Ian Osband, Benjamin Van Roy, Daniel J. Russo, and Zheng Wen. Deep Exploration via Randomized Value Functions. *Journal of Machine Learning Research*, 20(124):1–62, 2019. ISSN 1533-7928.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN 026218253X.

Carlos Riquelme, George Tucker, and Jasper Snoek. Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. In *International Conference on Learning Representations*, 2018.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *International Conference on Learning Representations*, Puerto Rico, 2016.

James E Smith and Robert L Winkler. The optimizer's curse: Skepticism and postdecision surprise in decision analysis. *Management Science*, 52(3):311–322, 2006.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

Bradly C. Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *CoRR*, abs/1507.00814, 2015.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

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.

Sebastian Thrun and Anton Schwartz. Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 Connectionist Models Summer School*, pages 255–263. Lawrence Erlbaum, 1993.

Ryan J. Tibshirani and Robert Tibshirani. A bias correction for the minimum error rate in cross-validation. *Ann. Appl. Stat.*, 3(2):822–829, 06 2009. doi: 10.1214/08-AOAS224.

Hado Van Hasselt. Double q-learning. In *Advances in Neural Information Processing Systems*, 2010.

Hado Van Hasselt. Estimating the maximum expected value: an analysis of (nested) cross-validation and the maximum sample average. *arXiv preprint arXiv:1302.7175*, 2013.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI*, volume 16, pages 2094–2100, 2016.

Harm Van Seijen, Hado Van Hasselt, Shimon Whiteson, and Marco Wiering. A theoretical and empirical analysis of expected sarsa. In *2009 ieee symposium on adaptive dynamic programming and reinforcement learning*, pages 177–184. IEEE, 2009.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48, pages 1995–2003. PMLR, 20–22 Jun 2016.

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

Min Xu, Tao Qin, and Tie yan Liu. Estimation bias in multi-armed bandit algorithms for search advertising. In *Advances in Neural Information Processing Systems 26*, pages 2400–2408, 2013.

Kenny Young and Tian Tian. Minatar: An atari-inspired testbed for thorough and reproducible reinforcement learning experiments. *arXiv preprint arXiv:1903.03176*, 2019.

Zongzhang Zhang, Zhiyuan Pan, and Mykel J. Kochenderfer. Weighted double q-learning. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 3455–3461, 2017. doi: 10.24963/ijcai.2017/483.