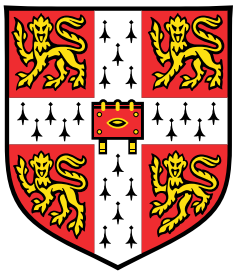
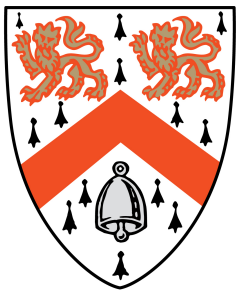


Exploration vs Exploitation in Reinforcement Learning



Igor Jerzy Adamski

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Master of Philosophy

Wolfson College

August 2020

Declaration

I, Igor Adamski of Wolfson College, being a candidate for the MPhil in Machine Learning and Machine Intelligence, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

I further declare the software that was used for this thesis. All computing experiments were carried out in Python. All tabular experiments relied on standard libraries such as Numpy and Scipy. All plots were made using the Matplotlib library. None of the above software was modified and no other third-party software was used. Several diagrams in this thesis were made using the draw.io website.

The word count of this thesis is 14,837 (including captions and appendices).

Igor Jerzy Adamski
August 2020

Acknowledgements

Most importantly, I would like to thank Prof. Carl Rasmussen for his invaluable support during the course of this project. Our discussions have taught me a great deal about probabilistic modelling and made me aware of many important intuitions one needs to have to tackle a truly challenging machine learning problem. I also want to thank him for constantly pushing me to go forwards, which, to my great satisfaction, has allowed me to cover the topic quite extensively.

I would also want to thank all the people from the MLMI 2020 cohort, for making this such a fantastic year. Our formals, discussions and common struggles with courseworks, will remain in my memory as important highlights of my year in Cambridge.

Further I want to thank the Department of Engineering, for supporting me financially, which has allowed me to pursue this great course without having to fall into more debt.

Last, but not least, I would like to thank my family for putting up with my extended presence during the uncertain times of the pandemic and for creating a wonderful environment for me to focus on this research.

Abstract

This work explores the exploration vs exploitation trade-off in Reinforcement Learning (RL). We tackle the problem from a Bayesian perspective, by building uncertainty-aware models of the environment and leveraging the Bellman relations to propagate that uncertainty through the recursively defined value functions. We argue that the posterior variance of the said value functions is critical to perform directed exploration and to naturally deviate into exploitation once the world is sufficiently inspected.

We analyze ways of obtaining the posterior uncertainty of the value functions, utilizing the Bayesian world models. We discuss the standard Monte Carlo approach and the Uncertainty Bellman equation (UBE) from recent RL literature. We proceed by introducing two novel estimates, which we derive directly from the recursive Bellman relations and show that they represent far more accurate alternatives to UBE.

The second part of this work, is concerned with analyzing uncertainty-aware action selection schemes. We investigate Upper Confidence Bound (UCB) methods, Thompson sampling and Posterior Sampling for Reinforcement Learning (PSRL). We give an overview of the Knowledge Gradient (KG), a provably-efficient method in the Multi Armed Bandit space and offer its natural generalization to more complex RL environments. We also introduce Monte Carlo sampling versions of the KG policy and show that they significantly outperform the state-of-the-art PSRL on tabular experiments.

Table of contents

List of figures

List of tables

Symbols

Other Symbols

$\mathcal{W}$ The RL environment (MDP)

Acronyms / Abbreviations

ML Machine learning

RL Reinforcement learning

BE Bellman equation

BO Bayesian Optimisation

BOE Bellman optimality equation

Corridor MAB Corridor Multi Armed Bandit

DP Dynamic Programming

EUB Epistemic Uncertainty Bound

EUDV Epistemic Uncertainty of Decorrelated Value

EWP Exploration will Pay off Corridor MAB

GKG Generalised Knowledge gradient

GLIE Greedy in the Limit of Infinite Exploration

KG Knowledge Gradient

LET Light at the End of Tunnel Corridor MAB

MAB Multi Armed Bandit

MC Monte Carlo

MDP Markov Decision Process

NG Normal-Gamma (distribution)

OFU Optimism in the Face of Uncertainty

PBT Perfect Binary Tree

PE Policy evaluation

PI Policy iteration

PSRL Posterior Sampling for Reinforcement Learning

TD Temporal Difference

UBE Uncertainty Bellman Equation

UCB Upper Confidence Bound

VI Value iteration

Chapter 1

Introduction

1.1 Motivations

> What we want is a machine that can learn from experience and the possibility of letting the machine alter its own instructions provides the mechanism for this.
>
> *Alan Turing*

The beginning of the second decade of the current millennium saw a rapid increase in the availability of large amounts of computing power. This lead tireless researchers from around the world to progressively move the boundary of what humans thought was achievable by machines. Machine learning (ML) has stood and is still standing at the forefront of this technological advancement, providing the building blocks for automation of hard tasks such as face identification (*supervised learning*) or clustering of high-dimensional data (*unsupervised learning*).

One truly spectacular and hair raising application of ML, has come from Reinforcement Learning (RL), when in March of 2016 a computer beat the 18-time world champion Lee Sedol in the game of Go (Silver et al. (2017)). In addition to game playing, RL has also delivered major advancements in fields of robotic control (Levine et al. (2016), Gu et al. (2017)), autonomous driving (Filos et al. (2020)), personalised medical treatment (Atan et al. (2018)) and dialogue systems (Singh et al. (2000b), Ferreira and Lefèvre (2015)). These complex applications differ from classical ML problems, where having labelled or unlabelled examples is enough for a model to successfully predict quantities of interest (f.eg. face

detection). One common difference, is that these applications necessitate a successful model to be able to *make decisions* and collect data by itself.

In many aspects, RL algorithms resemble human children, who at the beginning know nothing about the world they are born to and learn about it by *interacting* with it and *observing* the outcomes of their *actions*. In the long run, the RL algorithm must optimise for the total reward it obtains from the environment. Since we require it to constantly take actions, at any given point since its 'birth' (or *initialization*) to the environment, the RL agent has an idea of what actions are profitable. The agent can choose to *exploit* this knowledge and act so that it achieves highest rewards in accordance to what it previously observed. However, if the world is not sufficiently *explored*, the exploitation behaviour might actually turn out to be globally sub-optimal.

One of the most important open questions of RL, concerns finding the optimal moment when the agent should stop exploring and focus on exploiting what it already knows. Clearly, without enough exploration, it is impossible to discover optimal strategies in any given world. Many successful RL algorithms (Mnih et al. (2013), Mnih et al. (2015), Mnih et al. (2016)) handle this problem by telling the agent to exploit its knowledge most of the time and occasionally do something completely random. This approach, albeit proven to work, can be extremely data inefficient (Osband and Van Roy (2017)), since it might take a lot of these random steps to eventually find rewarding states and actions. It is therefore important for the exploration to be *directed* (Thrun (1992)) rather than incidental.

Successful directed exploration, should leverage not just the value estimates of certain behaviours, but also their *uncertainties*. To make sure that the agent turns to exploitation once the world is sufficiently explored, it should make use of value uncertainties to consider *expected improvements* that certain behaviours bring about.

1.2 Outline and contributions

In this work, we tackle the problem of the exploration-exploitation trade-off from a Bayesian perspective. We begin by introducing the RL problem and establishing consistent mathematical notation in chapter 2. In the same chapter, we also give an overview of the most fundamental RL results and methods and we finish off by providing an example of why classical random exploration is bound to fail in certain environments.

In chapter 3, we introduce Bayesian Reinforcement learning (BRL) and re-establish results and quantities from chapter 2 under the Bayesian regime. We proceed with demonstrating a difference between aleatoric and epistemic uncertainty and argue that only the latter should be used for directed exploration. We then give four ways of obtaining the said uncertainty: one

standard sampling method (Monte Carlo), one method from the literature (UBE)(O'Donoghue et al. (2017)) and two original methods (EUDV, EUB) leveraging the Bellman consistency.

Chapter 4 gives an overview of uncertainty sensitive decision making algorithms. We present two standard methods (UCB, Thompson sampling) as well as two algorithms from the recent literature (PSRL (Osband et al. (2013)), Knowledge gradient (KG) (Gupta and Miescke (1996))). We derive a novel generalisation of the KG idea to a wide range of discrete environments and discuss optimisation techniques for speeding up computations. Lastly we introduce novel KG-inspired algorithms based on Monte Carlo sampling.

In chapter 5 we introduce experimental methods on which we test the above algorithms. We present results and offer a discussion in chapter 6. In chapter 7 we conclude this works findings and contemplate on the future direction of this research.

Chapter 2

Reinforcement learning background

2.1 The Reinforcement learning problem

The field of Reinforcement learning (RL) is a class of solution methods dealing with the problem of optimal behaviour in an environment. An RL agents sole purpose is to find behaviours (or *actions*) in certain situations (or *states*) that lead to the accumulation of the highest possible numerical reward from the environment. We can call the RL environment a *world* in which an agent, without any supervision, is tasked to find actions or sequences of actions associated with high reward signals. Such a formulation of the problem is interesting from a philosophical standpoint - we might liken the RL agent with a human baby getting scolded by its mother or a human learning to play a computer game for the first time.

In RL, the interaction between the agent and the environment is sequential: an **action** taken by the agent makes the environment trigger a **reward** and evolve to the next **state**. This process is summarised in Figure 2.1.

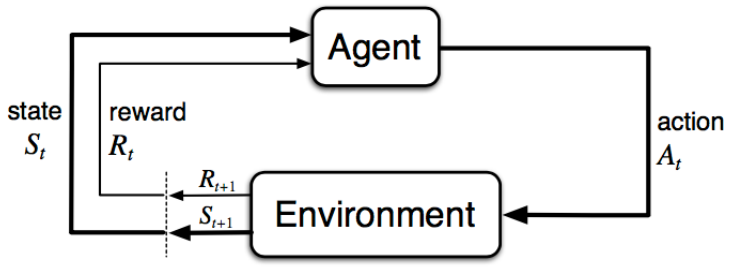


Fig. 2.1 Schematic interaction between the agent and the environment in RL (Sutton and Barto (2018)).

More formally, the interaction of the agent with the environment occurs at discrete time-steps $t \in \mathbb{Z}^+$ and can be formulated as a Markov Decision Process (MDP)(Silver (2015)).

Definition 1 (Markov Decision Process)**.** A Markov Decision Process (MDP) $\mathcal{W}$ is a tuple $\mathcal{W} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma, t_{\text{max}} \rangle$ where:

- $\mathcal{S}$ is the set of states.
- $\mathcal{A}$ is the set of actions.
- $\mathcal{T}$ is the per state-action state transition probability function, such that

$$\mathcal{T}^{a_t}_{s_t s_{t+1}} = P(S_{t+1} = s_{t+1} | S_t = s_t, A_t = a_t)$$

  where S_t and A_t are the state and action taken at the t^{th} step and $s_t \in \mathcal{S}$, $a_t \in \mathcal{A} \quad \forall t$.

- $\mathcal{R}$ is a per state-action reward random variable collection, such that

$$\mathcal{R}^{a_t}_{s_t} = R_{t+1} | S_t = s_t, A_t = a_t$$

  such that its first two moments $\mathbb{E}_{\mathcal{W}}[\mathcal{R}^{a_t}_{s_t}]$, $\mathbb{E}_{\mathcal{W}}[(\mathcal{R}^{a_t}_{s_t})^2]$ exist and are finite and bounded. $R_t \in \mathbb{R}$ is the reward obtained at the t^{th} step and $s_t \in \mathcal{S}$, $a_t \in \mathcal{A} \quad \forall t$.

- Next step transitions and mean rewards possess the Markov property:

$$P(S_{t+1} = s_{t+1} | \mathcal{H}_t) = P(S_{t+1} = s_{t+1} | S_t = s_t, A_t = a_t)$$
$$\mathbb{E}_{\mathcal{W}}[R_{t+1} | \mathcal{H}_t] = \mathbb{E}_{\mathcal{W}}[R_{t+1} | S_t = s_t, A_t = a_t]$$

  where $\mathcal{H}_t = \{(S_n = s_n, A_n = a_n, R_n = r_n)\}_{n=0}^{t}$ is the set containing the entire history up until time t.

- $\gamma \in [0, 1)$ is a discount factor.
- $t_{\text{max}} \in \mathbb{Z}^+$ is the maximal number of time steps the agent can interact with the environment for in a single episode.

We will represent the MDP with $\mathcal{W}$, since it contains every bit of information necessary to describe the entire *world* the agent lives in. In the case when t_{max} is finite, we call the MDP *episodic*, whereas when $t_{\text{max}} \to \infty$ we call it *continuing*. It is useful to note that γ is allowed to be 1 only in the case when the MDP is episodic. In this work we focus mostly on continuing tasks, where strictly $\gamma < 1$.

The interaction of the agent with the environment is carried out following deterministic or stochastic set of rules, which we call the policy.

Definition 2 (Policy)**.** The policy $\pi = \pi(a|s)$ of the agent is:

$$\pi(a|s) = P(A_t = a_t|S_t = s_t)$$

The goal of the RL agent is to find a policy π^* that maximizes the cumulative reward signal. More formally, we define the return function from time t following a policy π by:

$$G_{\pi,t} = \sum_{k=t+1}^{t_{\text{max}}} \gamma^{k-t-1} R_k|\pi$$

where the $R_k|\pi$ represent rewards obtained at time-step k following policy π. The cumulative sum of rewards is discounted by the factor γ to ensure that the sum stays finite for continuing tasks, when $t_{\text{max}} \to \infty$. Besides having a desirable effect of keeping $G_{\pi,t}$ finite, discounting future rewards makes intuitive sense as it incorporates the concept of future events having more uncertainty (Sutton and Barto (2018)). The low discount rate has the effect of favouring immediate rewards over future ones and as $\gamma \to 1$ the agent becomes more far-sighted.

2.2 Value functions

Most RL algorithms considered in this work involve estimating *value functions*. These inform the agent about the *expected* return of starting in a certain state and following a certain policy. Having that information, the agent is able to distinguish good and bad states and in turn directly compare behaviours in an environment in terms of their profitability.

Definition 3 (State value function)**.** The state value function $V^{\pi,\mathcal{W}}$ of the MDP $\mathcal{W}$, starting at state $s \in \mathcal{S}$ and following policy π on-wards is:

$$V^{\pi,\mathcal{W}}(s) = \mathbb{E}_{\pi,\mathcal{W}}[G_{\pi,t}|S_t = s]$$

Definition 4 (State-action value function)**.** The state-action value function $Q^{\pi,\mathcal{W}}$ of the MDP $\mathcal{W}$, starting at state $s \in \mathcal{S}$, performing action $a \in \mathcal{A}$ and following policy π on-wards is:

$$Q^{\pi,\mathcal{W}}(s,a) = \mathbb{E}_{\pi,\mathcal{W}}[G_{\pi,t}|S_t = s, A_t = a]$$

In both definitions, the expectation is taken over all possible trajectories generated from $\mathcal{W}$ following a policy π, weighted by their respective probabilities under $\mathcal{T}$ and $\mathcal{R}$ (Szepesvari

(2010)). The state value and state-action value functions are related to each other with the following relations:

$$V^{\pi,\mathcal{W}}(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}[Q^{\pi,\mathcal{W}}(s,a)] \tag{2.1}$$

$$Q^{\pi,\mathcal{W}}(s,a) = \mathbb{E}_{\mathcal{W}}[\mathcal{R}_s^a] + \gamma\,\mathbb{E}_{s'\sim\mathcal{W}}[V^{\pi,\mathcal{W}}(s')] \tag{2.2}$$

where we use $s' \sim \mathcal{W}$ as a shorthand for drawing s' from $\mathcal{T}_{s,\cdot}^a$.

As previously mentioned, the task of the agent is to find a policy, which on average achieves the highest possible return. Naturally, the state value and state-action value functions define a convenient partial ordering over policies (Silver (2015)), as a 'better' policy will always produce higher expected returns.

Definition 5 (Ordering of policies)**.** $\pi \geq \pi' \iff V^{\pi,\mathcal{W}}(s) \geq V^{\pi',\mathcal{W}}(s) \quad \forall s \in \mathcal{S}$

Definition 6 (Optimal policy)**.** A policy π^* is optimal $\iff \pi^* \geq \pi \quad \forall \pi$

It is useful to note that the optimal policy need not be unique, as many different behaviours might lead to identical expected outcomes in an MDP. However, value functions evaluated for different optimal policies necessarily have to be the same (Sutton and Barto (2018)). This ensures, that even though there might exist several optimal policies, the optimal value functions are unique for a given MDP $\mathcal{W}$.

Definition 7 (Optimal state value function)**.** The optimal state value function $V^{*,\mathcal{W}}$ satisfies:

$$V^{*,\mathcal{W}}(s) = \max_{\pi} V^{\pi,\mathcal{W}}(s) \quad \forall s \in \mathcal{S}$$

Definition 8 (Optimal state-action value function)**.** The optimal state-action value function $Q^{*,\mathcal{W}}$ satisfies:

$$Q^{*,\mathcal{W}}(s,a) = \max_{\pi} Q^{\pi,\mathcal{W}}(s,a) \quad \forall s \in \mathcal{S}, a \in \mathcal{A}$$

In both cases in Definitions 7 and 8, the max is taken over all possible policies π in $\mathcal{W}$.

2.3 Bellman equations

The returns $G_{\pi,t}$ at successive time-steps have a recursive nature and are related to each other by the relation:

$$\begin{aligned} G_{\pi,t} &= R_{t+1}|\pi + \gamma \sum_{k=t+2}^{t_{\max}} \gamma^{k-t-2} R_k|\pi \\ &= R_{t+1}|\pi + \gamma G_{\pi,t+1} \end{aligned} \tag{2.3}$$

with $G_{\pi,t_{\max}} = 0$ in the episodic case. This relation can be used to derive Bellman equations (BEs)(Bellman (1957)):

$$V^{\pi,\mathcal{W}} = \mathcal{B}^{\pi}_{\mathcal{V}} V^{\pi,\mathcal{W}} \tag{2.4}$$

$$Q^{\pi,\mathcal{W}} = \mathcal{B}^{\pi}_{\mathcal{Q}} Q^{\pi,\mathcal{W}} \tag{2.5}$$

where $\mathcal{B}^{\pi}_{\mathcal{V}}, \mathcal{B}^{\pi}_{\mathcal{Q}}$ are Bellman operators that satisfy:

$$\mathcal{B}^{\pi}_{\mathcal{V}} V^{\pi,\mathcal{W}}(s_t) = \mathbb{E}_{a_t \sim \pi(\cdot|s), s_{t+1} \sim \mathcal{W}} \left[\mathbb{E}_{\mathcal{W}}[\mathcal{R}^{a_t}_{s_t}] + \gamma V^{\pi,\mathcal{W}}(s_{t+1}) \right] \tag{2.6}$$

$$\mathcal{B}^{\pi}_{\mathcal{Q}} Q^{\pi,\mathcal{W}}(s_t, a_t) = \mathbb{E}_{a_{t+1} \sim \pi(\cdot|s_{t+1}), s_{t+1} \sim \mathcal{W}} \left[\mathbb{E}_{\mathcal{W}}[\mathcal{R}^{a_t}_{s_t}] + \gamma Q^{\pi,\mathcal{W}}(s_{t+1}, a_{t+1}) \right] \tag{2.7}$$

with $V^{\pi,\mathcal{W}}(s_{t_{\max}}) = 0$ and $Q^{\pi,\mathcal{W}}(s_{t_{\max}}, a_{t_{\max}}) = 0 \ \forall s_{t_{\max}} \in \mathcal{S}, a_{t_{\max}} \in \mathcal{A}$. Note that if we consider the continuing case of $t_{\max} \to \infty$, then we no longer need to have these restrictions. In the above equations we purposefully add time subscripts to the state and actions to emphasize the difference between the BEs for Q and V. In 2.6, the expectation is taken over all possible *immediate* actions a_t and states s_{t+1} following (s_t, a_t), whereas in 2.7 the expectation is over all possible next states s_{t+1} following (s_t, a_t) and actions a_{t+1} subsequent to that s_{t+1}. Again, we use the shorthand $s_{t+1} \sim \mathcal{W}$ to represent sampling from the transitions $\mathcal{T}$.

The BEs express the values of states (or state-actions) in terms of the values of their successor states (or state-actions). They serve as a form of stability constraint, relating the expected discounted return to the immediate expected reward and the average expected discounted return from all possible states we transition to (Szepesvari (2010)).

Optimal policies also yield recursive BEs of their own, called Bellman optimality equations (BOEs)(Bellman (1957)):

$$V^{*,\mathcal{W}} = \mathcal{B}^{*}_{\mathcal{V}} V^{*,\mathcal{W}} \tag{2.8}$$

$$Q^{*,\mathcal{W}} = \mathcal{B}^{*}_{\mathcal{Q}} Q^{*,\mathcal{W}} \tag{2.9}$$

where $\mathcal{B}^*_{\mathcal{V}}, \mathcal{B}^*_{\mathcal{Q}}$ are Bellman optimality operators that satisfy:

$$\mathcal{B}^*_{\mathcal{V}} V^{*,\mathcal{W}}(s_t) = \max_{a_t \in \mathcal{A}} \mathbb{E}_{s_{t+1} \sim \mathcal{W}} \left[\mathbb{E}_{\mathcal{W}}[\mathcal{R}^{a_t}_{s_t}] + \gamma V^{*,\mathcal{W}}(s_{t+1}) \right] \tag{2.10}$$

$$\mathcal{B}^*_{\mathcal{Q}} Q^{*,\mathcal{W}}(s_t, a_t) = \mathbb{E}_{s_{t+1} \sim \mathcal{W}} \left[\mathbb{E}_{\mathcal{W}}[\mathcal{R}^{a_t}_{s_t}] + \gamma \max_{a_{t+1} \in \mathcal{A}} Q^{*,\mathcal{W}}(s_{t+1}, a_{t+1}) \right] \tag{2.11}$$

A very important result states that all of the above mentioned Bellman operators are $\gamma-$contractions, meaning that:

$$\|\mathcal{B}^{\pi_1}_V V^{\pi_1,\mathcal{W}}_1 - \mathcal{B}^{\pi_2}_V V^{\pi_2,\mathcal{W}}_2\|_\infty \leq \gamma \|V^{\pi_1,\mathcal{W}}_1 - V^{\pi_2,\mathcal{W}}_2\|_\infty \tag{2.12}$$

$$\|\mathcal{B}^*_V V^{\pi_1,\mathcal{W}}_1 - \mathcal{B}^*_V V^{\pi_2,\mathcal{W}}_2\|_\infty \leq \gamma \|V^{\pi_1,\mathcal{W}}_1 - V^{\pi_2,\mathcal{W}}_2\|_\infty \tag{2.13}$$

for arbitrary $V^{\pi_1,\mathcal{W}}_1, V^{\pi_2,\mathcal{W}}_2 \in \mathcal{V}$ where $\mathcal{V}$ is the vector space over all possible state value functions in $\mathcal{W}$. The same result holds for $\mathcal{B}^\pi_Q, \mathcal{B}^*_Q$ in the vector space of all possible state-action value functions. By Banach's fixed point theorem (Banach (1922)), this implies that repeatedly applying $\mathcal{B}^\pi_V, \mathcal{B}^*_V$ to an initial guess $V^{\pi_1,\mathcal{W}}_1$ converges to fixed points $V^{\pi,\mathcal{W}}, V^{*,\mathcal{W}}$ respectively. This is a huge result, as it means that we can obtain globally optimal value functions from any initial guess, on the condition that we know how to evaluate the Bellman operator. As we will see in the later sections, the Bellman operator is easy to evaluate in the case when $|\mathcal{S}|$ and $|\mathcal{A}|$ are finite, as the expectations in the BE simply turn into sums over possible states and actions.

2.4 Fundamental RL methods

In this section, we walk through fundamental approaches to calculating the optimal policies in RL. We can separate algorithms into two classes:

1. **Model-based** - algorithms that approximate value functions by solving BEs on learnt models of the MDP $\mathcal{W}$.

2. **Model-free** - algorithms that approximate value functions directly from agents experience, without explicitly modelling $\mathcal{W}$.

For simplicity of exposition and because we perform all experiments on continuing MDPs, from now on we focus on the subset of MDPs for which $t_{\text{max}} \to \infty$ and hence $\gamma < 1$.

2.4.1 Model-based algorithms

In model-based RL, we assume that we have access to the transitions $\mathcal{T}$ and reward process $\mathcal{R}$. In practical applications, where the underlying MDP dynamics are not known, it is still possible to benefit from model-based approaches by using learnt models of $\mathcal{T}$ and $\mathcal{R}$ instead (Rasmussen and Kuss (2004), Deisenroth (2009)). Knowing the MDP dynamics allows us to utilize the recursive BEs and directly solve them using *dynamic programming* (DP) (Bertsekas et al. (1995)).

Policy evaluation

Algorithm 1 Policy evaluation by DP

Input MDP $\mathcal{W}$, policy π and tolerance $\Delta \in \mathbb{R}$
procedure PolicyEvaluation($\mathcal{W}, \pi, \Delta$)
    $Q_{\text{old}}(s,a) \leftarrow 0, \forall s \in \mathcal{S}, a \in \mathcal{A}$
    while $\|Q_{\text{old}}(s,a) - Q_{\text{new}}(s,a)\|_\infty > \Delta$ **do**
        $Q_{\text{new}}(s,a) \leftarrow \mathcal{B}^\pi_Q Q_{\text{old}}(s,a)$ using eq. 2.7
        $Q_{\text{old}}(s,a) \leftarrow Q_{\text{new}}(s,a)$
    end while
    return $Q_{\text{new}}(s,a)$

Assuming that the agent uses some policy π to make decisions in $\mathcal{W}$, it is fairly straightforward to compute $V^{\pi,\mathcal{W}}$ and $Q^{\pi,\mathcal{W}}$ using Bellman equations 2.6 and 2.7 respectively. The fact that both $\mathcal{B}^\pi_\mathcal{V}$ and $\mathcal{B}^\pi_\mathcal{Q}$ are contractions guarantees that repeatedly applying them to an initial guess, will reach a fixed point at the optimal $V^{\pi,\mathcal{W}}$ and $Q^{\pi,\mathcal{W}}$ respectively, a process given in the Policy evaluation (PE) Algorithm 1 (Silver (2015)).

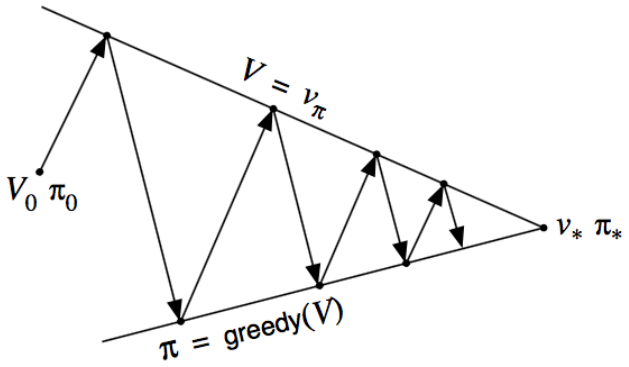


Fig. 2.2 Illustration of the policy iteration process (Sutton and Barto (2018)).

Policy iteration

One benefit of having a policy evaluation routine, is that we can try out different policies and see which ones produce greater value functions. However, it is important to be able to easily augment a considered policy π such that its augmented version π' satisfies $\pi \leq \pi'$.

Policy iteration (PI) is an algorithm that relies on the policy improvement theorem (Sutton and Barto (2018)). The theorem states that if we construct π' to be greedy with respect to $Q^{\pi,\mathcal{W}}$ then necessarily $\pi \leq \pi'$.

Definition 9 (Greedy policy)**.** A policy π' is greedy w.r.t. $Q^{\pi,\mathcal{W}}$ if

$$\pi'(a|s) = \begin{cases} 1 & \text{for } a = \operatorname{argmax}_{a'} Q^{\pi,\mathcal{W}}(s,a') \\ 0 & \text{otherwise} \end{cases}$$

Therefore, policy improvement theorem allows us to start with any arbitrary policy π, improve it by making it greedy with respect to $Q^{\pi,\mathcal{W}}$, recompute the new value and iterate the process to obtain optimal π^* and $Q^{*,\mathcal{W}}$, as shown in Figure 2.2 and summarized in Algorithm 2 (Silver (2015)).

Algorithm 2 Policy iteration

Input MDP $\mathcal{W}$ and tolerance $\Delta \in \mathbb{R}$
procedure PolicyIteration($\mathcal{W}, \Delta$)
    Initialize random π_{old}
    while $\pi_{\text{new}} \neq \pi_{\text{old}} \forall s \in \mathcal{S}, a \in \mathcal{A}$ **do**
        $Q^{*,\mathcal{W}} \leftarrow$ PolicyEvaluation($\mathcal{W}, \pi_{\text{old}}, \Delta$) using Algorithm 1
        $\pi_{\text{new}} \leftarrow$ greedy w.r.t. $Q^{*,\mathcal{W}}$ and set $\pi_{\text{old}} \leftarrow \pi_{\text{new}}$
    end while
    return $Q^{*,\mathcal{W}}$

Value iteration

The value iteration (VI) algorithm utilizes the BOEs directly, by iterating from an initial guess, as shown in Algorithm 3. In practice, value iteration tends to take a lot more iterations to converge to the optimal value in comparison with policy iteration, but each iteration it makes is much less expensive, making it suitable to certain environments (Silver (2015)).

Algorithm 3 Value iteration

Input MDP $\mathcal{W}$ and tolerance $\Delta \in \mathbb{R}$
procedure VALUEITERATION($\mathcal{W}, \Delta$)
    $Q^*_{\text{old}}(s,a) \leftarrow 0, \forall s \in \mathcal{S}, a \in \mathcal{A}$
    while $\|Q^*_{\text{old}}(s,a) - Q^*_{\text{new}}(s,a)\|_\infty > \Delta$ **do**
        $Q^*_{\text{new}} \leftarrow \mathcal{B}^*_{\mathcal{Q}} Q^*_{\text{old}}$ using eq. 2.11 and set $Q^*_{\text{old}} \leftarrow Q^*_{\text{new}}$
    end while
    return Q^*_{new}

2.4.2 Model-free algorithms

In model-free RL methods, the true dynamics of $\mathcal{W}$ are unknown and the agent does not make any efforts to model them. Instead, they are characterized by the agent maintaining a initial estimate of Q and refining it through interaction with the world, using the rewards and transitions sampled from $\mathcal{W}$ directly.

Fundamental algorithms can be divided into Monte Carlo (MC) and Temporal Difference (TD) methods. MC methods estimate the value functions as empirical averages of encountered rewards, directly using the equations from definitions 3 and 4. TD algorithms utilize the recursive nature of $G_{\pi,t}$ and try to update the estimated values *towards* their relations defined by the BEs.

MC methods are guaranteed to reach optimal values by the law of large numbers but can be painfully slow in doing so (Silver (2015)), rendering them unpractical. TD methods naturally divide into *on-policy* and *off-policy* algorithms. The update rule for SARSA, a fundamental on-policy method, bears striking resemblance to the BE in equation 2.7, with the difference being that the expectations are substituted for empirical samples from π, $\mathcal{T}$ and $\mathcal{R}$:

$$Q^{\pi,\mathcal{W}}(s_t,a_t) \leftarrow Q^{\pi,\mathcal{W}}(s_t,a_t) + \eta_t(\underbrace{R_{t+1} + \gamma Q(s_{t+1},a_{t+1}) - Q^{\pi,\mathcal{W}}(s_t,a_t)}_{\text{SARSA TD term}}) \tag{2.14}$$

where η_t is teh step-size, s_{t+1} is the observed state following s_t and action a_t and a_{t+1} is sampled from policy $\pi(\cdot|s_{t+1})$.

On-policy methods can be associated with value iteration, in that they update the Q values towards a recursion defined by the BOEs. One variant of on-policy control is Q-learning (Watkins and Dayan (1992)) and its update rule for the value function sets about directly

computing $Q^{*,\mathcal{W}}$:

$$Q^{*,\mathcal{W}}(s_t, a_t) \leftarrow Q^{*,\mathcal{W}}(s_t, a_t) + \eta_t \underbrace{(R_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q^{\pi,\mathcal{W}}(s_t, a_t))}_{\text{Q-learning TD term}} \tag{2.15}$$

In practice, both SARSA and Q-learning are implemented so that the policy π is frequently improved with respect to the most current value estimation, a process which very much resembles policy iteration. However, it is not simply sufficient to make the policy greedy w.r.t. the latest value estimation - in fact, this policy improvement scheme is bound to fail in methods where the explicit dynamics of $\mathcal{W}$ are unknown. Instead, policy improvement under uncertain estimates of $Q^{*,\mathcal{W}}$ have to be performed *softly*, which we discuss later in section 2.4.3.

The full process of Q-learning is given in Algorithm 4.

Algorithm 4 Q-learning

Input step-size vector η_t and t_{max}.
procedure QLEARNING(η_t)
    Initialize $Q(s, a) \leftarrow 0, \forall s \in \mathcal{S}, a \in \mathcal{A}$ and π
    for t = 1,2,3,...,t_{max} **do**
        Observe s_t
        Act $a_t \sim \pi(\cdot|s_{t+1})$ and observe s_{t+1}, R_{t+1}
        Update $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \eta_t(R_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$
        $\pi \leftarrow \epsilon$-greedy w.r.t. Q
    end for
    return Q

2.4.3 Policy improvement under unknown dynamics of $\mathcal{W}$

When the dynamics of the environment $\mathcal{W}$ are unknown, we require more elaborate policy improvement schemes than the simple greedy policy. This is because the accuracy of the agents estimation of the value function depends directly on the number of interactions it has performed in the environment. This makes the obtained values of $Q^{*,\mathcal{W}}$ only *estimates* which can in fact be very different from the actual optimal values in $\mathcal{W}$. Thus, if the agent was to act greedily after exploring only a small subset of states and actions, it might get stuck executing a sub-optimal strategy over and over again.

The agent needs to *explore* a sufficiently large portion of the available states and actions in the environment before it can feasibly *exploit* its knowledge of it and act greedily with

respect to its estimation of $Q^{*,\mathcal{W}}$. Q-learning's and SARSA's value estimations are guaranteed to converge to the optimal $Q^{*,\mathcal{W}}$ provided that each state-action is visited infinitely often in the limit of interaction and that the action selection policy is *Greedy in the Limit of Infinite Exploration* (GLIE), meaning that the policy must approach greedy in the limit (Singh et al. (2000a)). The first requirement is satisfied by *soft* policies, so ones that have an element of stochasticity to them.

Two most commonly used policies that satisfy both requirements are $\varepsilon-$greedy and Boltzman.

Definition 10 ($\varepsilon-$greedy policy)**.** A policy π' is $\varepsilon-$greedy w.r.t. $Q^{\pi,\mathcal{W}}$ if

$$\pi'(a|s) = \begin{cases} \pi \text{ greedy w.r.t. } Q^{\pi,\mathcal{W}} & \text{with probability } 1-\varepsilon \\ |A_s|^{-1} & \text{with probability } \varepsilon \end{cases}$$

where A_s is the set of available actions from state s and $\varepsilon \in [0,1]$.

Definition 11 (Boltzman policy)**.** A policy π' is Boltzman w.r.t. $Q^{\pi,\mathcal{W}}$ if

$$\pi'(a|s) = \frac{e^{Q^{\pi,\mathcal{W}}(s,a)/B}}{\sum_{a'} e^{Q^{\pi,\mathcal{W}}(s,a')/B}} \tag{2.16}$$

for $B \in \mathbb{R}^+$.

Both of these policies are GLIE, provided that $\varepsilon \to 0$ and $B \to 0$ as $t \to \infty$. They also satisfy the requirement of visiting every state-action by the inherent randomness in the action selection.

2.4.4 Failure modes of $\varepsilon-$greedy and Boltzman

The random nature of $\varepsilon-$greedy and Boltzman policies, makes them unfit for hard exploration tasks (Osband et al. (2017)), where the optimal strategy requires performing a sequence of exploratory actions, leading to negative rewards in the short term, in order to find a state giving the overall highest reward.

A simple example of an environment where these two policies fail miserably is the binary tree MDP, shown in figure 2.3 (Janz et al. (2018)). In this environment there are $2L+1$ states and two actions, 'up' and 'down'. The agent starts at state s_0. Going 'down' at any even state leads to the agent being teleported back to s_0 whereas 'up' always transitions to the next even state. The best state, s_{2L}, is at the end of the chain and it takes L uninterrupted 'up' actions to get there. Once the agent is in s_{2L}, it stays there and keeps getting a reward of 1. The rewards for going 'up' are always $-1/L$ whereas for going 'down' they are 0.

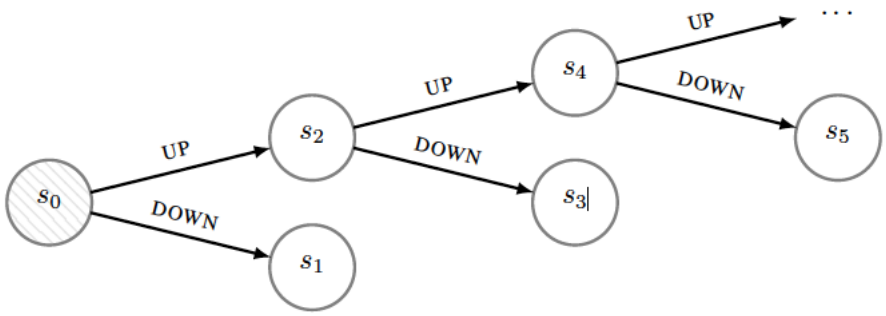


Fig. 2.3 The binary tree MDP (Janz et al. (2018))

This environment represents a truly hard exploratory problem, because the agent needs to learn to perform a long sequence of uninterrupted actions that seem sub-optimal in the short term. It is only after the s_{2L} is reached that the agent can understand that it is the best state.

Whenever the agent goes up in state s_n for $n < L$, it will update the value estimate of that action towards the immediate reward, which is negative. Before the rewarding state s_{2L} is first reached, every time the agent comes back to s_n, its estimate for going 'up' will be much smaller than for going 'down'. Policies like $\varepsilon-$greedy are bound to fail in such a scenario, because their exploitation (with probability $1 - \varepsilon$) is based on false principles prior to their random exploration (with probability ε) uncovers the optimal state. In the binary tree MDP, until the agent reaches s_{2L}, $Q(s, \text{'up'}) < Q(s, \text{'down'})$ for all s and hence the probability of reaching the last state is $(\frac{\varepsilon}{2})^L$, which for large L tends to 0 very quickly.

Optimistic initialization

One simple way of mitigating this failure is adhering to the *optimism in the face of uncertainty* (OFU) principle. It tells us that the agent should value every unknown state highly a-priori and reduce the values to realistic levels once it actually visits the states. In practice, this can be achieved by initializing $Q(s, a)$ to high numbers (instead of 0 as in algorithm 4). This is a neat idea and works well in practice. However, the OFU principle is harder to implement when dealing with large, continuous state spaces, where state generalisation wipes out the effect completely (Bellemare et al. (2017)).

Chapter 3

Bayesian Reinforcement Learning

In this chapter we introduce Bayesian RL (BRL), a framework which leverages Bayesian inference to integrate information (of lack thereof) into standard RL. In BRL, the agent has no knowledge of the underlying dynamics of the MDP $\mathcal{W}$ and applies model-based techniques on learnt dynamics models to calculate value functions and their uncertainties. A Bayesian approach to modelling allows the agent to capture the full state of its knowledge in relation to a specified prior, providing a principled way to tackle the exploration-exploitation trade-off.

3.1 Modelling the environment

In order for the agent to be able to use model-based methods from section 2.4.1, it needs to maintain the MDP $\mathcal{W}$ dynamics models. The two functions that it needs to learn are the transition dynamics $\mathcal{T}$ and reward dynamics $\mathcal{R}$, both described in Definition 1. In BRL, a Bayesian modelling approach is employed, meaning that we parametrize specified $\mathcal{T}, \mathcal{R} \in \mathcal{W}$ distributions with parameters $\{\boldsymbol{\theta}_{\mathcal{T}}, \boldsymbol{\theta}_{\mathcal{R}}\} = \boldsymbol{\theta}$, that we also treat as distributions rather than point estimates. We refine our belief of the distribution of $\boldsymbol{\theta}$ once we observe data $\mathcal{D}_t$, by computing the posterior using Bayes rule:

$$p(\boldsymbol{\theta} \,|\mathcal{D}_t) \propto p(\mathcal{D}_t|\, \boldsymbol{\theta}) p(\boldsymbol{\theta}) \tag{3.1}$$

where $\mathcal{D}_t = \{(s_n, a_n, r_{n+1}, s_{n+1})\}_{n=1}^{t-1}$ represents all the data gathered by the agent up until time t. The Bayesian framework requires us to specify prior distributions on the parameters $p(\boldsymbol{\theta})$. This addition is appealing in the context of RL trying to emulate human decision making, as humans always approach a problem with some degree of expectations.

The next two sections will introduce candidate Bayesian models for $\mathcal{W}$ and present results that will be useful later in the work.

3.1.1 Reward model

A natural model for the reward distribution functions $\mathcal{R}$, is a Gaussian distribution. Since both the mean and variance of this Gaussian are unknown, a suitable conjugate prior [1] is the normal-gamma (NG) distribution. The full reward model is then specified as:

$$R_{t+1}|s_t, a_t, \boldsymbol{\theta} \sim \mathcal{N}(\mu_{s_t,a_t}, \tau^{-1}_{s_t,a_t}) \tag{3.2}$$

$$\mu_{s_t,a_t}, \tau_{s_t,a_t} \sim NG(m_0, \lambda_0, \alpha_0, \beta_0) \tag{3.3}$$

Since the NG distribution is a conjugate prior to the Gaussian, the posterior distribution of the parameters $\mu_{s_t,a_t}, \tau_{s_t,a_t} \in \boldsymbol{\theta}_{\mathcal{R}} \subset \boldsymbol{\theta}$ is also NG. More specifically:

$$\mu_{s_t,a_t}, \tau_{s_t,a_t}|\mathcal{D}_t \sim NG(m_{s_t,a_t}, \lambda_{s_t,a_t}, \alpha_{s_t,a_t}, \beta_{s_t,a_t}) \tag{3.4}$$

where

$$m_{s_t,a_t} = \frac{\lambda_0 m_0 + n_{s_t,a_t}\overline{x}_{s_t,a_t}}{\lambda_0 + n_{s_t,a_t}} \tag{3.5}$$

$$\lambda_{s_t,a_t} = \lambda_0 + n_{s_t,a_t} \tag{3.6}$$

$$\alpha_{s_t,a_t} = \alpha_0 + \frac{n_{s_t,a_t}}{2} \tag{3.7}$$

$$\beta_{s_t,a_t} = \beta_0 + \frac{(n_{s_t,a_t} - 1)S_{s_t,a_t}}{2} + \frac{\lambda_0 n_{s_t,a_t}(\overline{x}_{s_t,a_t} - \mu_0)^2}{2(\lambda_0 + n_{s_t,a_t})} \tag{3.8}$$

where n_{s_t,a_t} represents the total number of rewards seen until time t from action $a_t \in \mathcal{A}$ in state $s_t \in \mathcal{S}$, $\overline{x}_{s_t,a_t}$ is the sample mean of these rewards and S_{s_t,a_t} their sample variance (Bishop (2006)).

Next we present results, that are going to prove very useful in the later sections. The full derivations can be found in Appendix A.1.1. These results are:

$$\mathbb{E}_{\boldsymbol{\theta}}\left[\mathbb{E}_{\pi,\mathcal{W}|\boldsymbol{\theta}}[R_{t+1}|s_t, a_t, \boldsymbol{\theta}]\right] = m_{s_t,a_t} \tag{3.9}$$

$$\mathbb{E}_{\boldsymbol{\theta}}\left[\mathbb{E}_{\pi,\mathcal{W}|\boldsymbol{\theta}}[R^2_{t+1}|s_t, a_t, \boldsymbol{\theta}]\right] = m^2_{s_t,a_t} + \frac{\beta_{s_t,a_t}}{\alpha_{s_t,a_t} - 1}\left(1 + \frac{1}{\lambda_{s_t,a_t}}\right) \tag{3.10}$$

$$\mathrm{Var}_{\boldsymbol{\theta}}\left[\mathbb{E}_{\pi,\mathcal{W}|\boldsymbol{\theta}}[R_{t+1}|s_t, a_t, \boldsymbol{\theta}]\right] = \frac{\beta_{s_t,a_t}}{\lambda_{s_t,a_t}(\alpha_{s_t,a_t} - 1)} \tag{3.11}$$

and they are the predictive first and second moment and the predictive variance of the reward respectively. The inner expectation is taken with respect to the MDP $\mathcal{W}$ parametrized by the

[1] Conjugate priors are priors that induce posteriors in the same probability distribution family for the given likelihood function (Murphy (2007)).

posterior $\boldsymbol{\theta}\,|\mathcal{D}_t$. This expectation is independent of π but we keep the subscript π there for ease of notation. The outer expectations and variances are taken with respect to the posterior parameters $\boldsymbol{\theta}\,|\mathcal{D}_t$ and they represent the operation of marginalizing them out.

In the above, as well as in the further parts of this work, whenever we refer to $\boldsymbol{\theta}$ we actually mean the posterior $\boldsymbol{\theta}\,|\mathcal{D}_t$, unless otherwise specified.

3.1.2 Transition model

A natural model for the transition function $\mathcal{T}$ in discrete state-space environments, is a Categorical distribution. The suitable conjugate prior to the categorical is Dirichlet (Bishop (2006)), which implies the following transition function model:

$$s_{t+1}|s_t, a_t, \boldsymbol{\theta} \sim \text{Cat}(\boldsymbol{\kappa}_{s_t,a_t}) \tag{3.12}$$

$$\boldsymbol{\kappa}_{s_t,a_t} \sim \text{Dir}(\boldsymbol{a}_{s_t,a_t}) \tag{3.13}$$

where again we use the shorthand $\boldsymbol{\kappa}_{s_t,a_t} \in \boldsymbol{\theta}_{\mathcal{T}} \subset \boldsymbol{\theta}$ and $\boldsymbol{a}_{s_t,a_t} \in \mathbb{R}^{|\mathcal{S}|}, \forall s_t \in \mathcal{S}, a_t \in \mathcal{A}$. By construction, the posterior distribution of the parameters is also Dirichlet:

$$\boldsymbol{\kappa}_{s_t,a_t}|\mathcal{D}_t \sim \text{Dir}(\boldsymbol{a}_{s_t,a_t} + \boldsymbol{c}_{s_t,a_t}) \tag{3.14}$$

where $\boldsymbol{c}_{s_t,a_t} \in \mathbb{R}^{|\mathcal{S}|}$ is a count vector, such that its i^{th} element is the number of times the agent transitioned to state i having made action $a_t \in \mathcal{A}$ in state $s_t \in \mathcal{S}$, up until time t Bishop (2006).

We again present useful results that will come in handy in the coming sections:

$$\mathbb{E}_{\boldsymbol{\theta}}[p(s'|s,a,\boldsymbol{\theta})] = \frac{[\boldsymbol{a}_{s,a} + \boldsymbol{c}_{s,a}]_{s'}}{\mathbf{1}^T(\boldsymbol{a}_{s,a} + \boldsymbol{c}_{s,a})} \tag{3.15}$$

$$\text{Var}_{\boldsymbol{\theta}}[p(s'|s,a,\boldsymbol{\theta})] = \frac{\frac{[\boldsymbol{a}_{s,a} + \boldsymbol{c}_{s,a}]_{s'}}{\mathbf{1}^T(\boldsymbol{a}_{s,a} + \boldsymbol{c}_{s,a})}\Big(1 - \frac{[\boldsymbol{a}_{s,a} + \boldsymbol{c}_{s,a}]_{s'}}{\mathbf{1}^T(\boldsymbol{a}_{s,a} + \boldsymbol{c}_{s,a})}\Big)}{1 + \mathbf{1}^T(\boldsymbol{a}_{s,a} + \boldsymbol{c}_{s,a})} \tag{3.16}$$

$$\text{Cov}_{\boldsymbol{\theta}}[p(s'|s,a,\boldsymbol{\theta}), p(s''|s,a,\boldsymbol{\theta})] = \frac{-[\boldsymbol{a}_{s,a} + \boldsymbol{c}_{s,a}]_{s'}[\boldsymbol{a}_{s,a} + \boldsymbol{c}_{s,a}]_{s''}}{\big(\mathbf{1}^T(\boldsymbol{a}_{s,a} + \boldsymbol{c}_{s,a})\big)^2\big(\mathbf{1}^T(\boldsymbol{a}_{s,a} + \boldsymbol{c}_{s,a}) + 1\big)} \tag{3.17}$$

where $\mathbf{1} \in \mathbb{R}^{|\mathcal{S}|}$ is a vector of ones, $[\cdot]_i$ represents the i^{th} element of a vector and in 3.17 $s' \neq s'$. The derivations of the above results are given in Appendix A.1.2.

3.2 Value functions under modelling uncertainty

The introduction of Bayesian models of the environment $\mathcal{W}$, allows us to make use of the uncertainty about the world dynamics. We have seen in section 2 that iterative approaches using DP are guaranteed to converge to the optimal value functions, given the full specifications of the MDP $\mathcal{W}$. These guarantees still hold for parametrized environments $\mathcal{W} \mid \boldsymbol{\theta}$, however prior to exploring the world sufficiently and shrinking the posteriors, the modelled environment is likely to be very different from the actual one. Because of that, the obtained value functions will likely be inaccurate and it would be useful to the agent to know the extent of this inaccuracy (Dabney et al. (2018)).

The Bayesian picture allows the agent to obtain predictive distributions of the dynamics of $\mathcal{W}$, which we access through marginalizing over all possible model posterior parameters $\boldsymbol{\theta}$. Marginalizing over the posterior parameters provides a fully probabilistic view (Jaynes (2003)) of the possible environment dynamics - this in turn allows the agent to imagine what the most probable $\mathcal{W}$ is and what is the uncertainty surrounding that estimation. Similarly, we would like to incorporate that information when calculating value functions, to obtain the most probable optimal $Q^{*,\mathcal{W}}$ and the uncertainty around it (Osband et al. (2017)).

The law of iterated expectations (Weiss et al. (2006)) gives us a way to obtain the most probable (in expectation) value function for any policy π:

$$\begin{aligned} Q^{\pi,\mathcal{W}}(s_t, a_t) = \mathbb{E}_{\pi,\mathcal{W}}[G_{\pi,t} | S_t = s_t, A_t = a_t] &= \mathbb{E}_{\boldsymbol{\theta}}\left[\mathbb{E}_{\pi,\mathcal{W}\mid\boldsymbol{\theta}}[G_{\pi,t} | S_t = s_t, A_t = a_t, \boldsymbol{\theta}]\right] \\ &= \mathbb{E}_{\boldsymbol{\theta}}[Q^{\pi,\mathcal{W}\mid\boldsymbol{\theta}}(s_t, a_t)] \end{aligned} \tag{3.18}$$

and we can see that it involves marginalizing possible Q-values over the posterior parameters $\boldsymbol{\theta}$. Since the BEs give the Q-value relations in terms of reward and transition dynamics, the marginalization in 3.18 is equivalent to using standard PE with the predictive distributions of the rewards and transitions (Janz et al. (2018)), yielding the following recursive BE:

$$\begin{aligned} \mathbb{E}_{\boldsymbol{\theta}}[Q^{\pi,\mathcal{W}\mid\boldsymbol{\theta}}(s_t, a_t)] = \mathbb{E}_{\boldsymbol{\theta}}&\left[\mathbb{E}_{\pi,\mathcal{W}\mid\boldsymbol{\theta}}[R_{t+1} | s_t, a_t, \boldsymbol{\theta}]\right] \\ &+ \gamma \sum_{s_{t+1}, a_{t+1}} \pi(a_{t+1} | s_{t+1})\, \mathbb{E}_{\boldsymbol{\theta}}[p(s_{t+1} | s_t, a_t, \boldsymbol{\theta})]\, \mathbb{E}_{\boldsymbol{\theta}}[Q^{\pi,\mathcal{W}\mid\boldsymbol{\theta}}(s_{t+1}, a_{t+1})] \end{aligned} \tag{3.19}$$

where we made the following association in comparison to the original BE from 2.7:

$$\begin{aligned} \mathbb{E}_{\mathcal{W}}[\mathcal{R}^{a_t}_{s_t}] &\longleftrightarrow \mathbb{E}_{\boldsymbol{\theta}}\left[\mathbb{E}_{\pi,\mathcal{W}\mid\boldsymbol{\theta}}[R_{t+1} | s_t, a_t, \boldsymbol{\theta}]\right] \\ \mathcal{T}^{a_t}_{s_t, s_{t+1}} &\longleftrightarrow \mathbb{E}_{\boldsymbol{\theta}}[p(s_{t+1} | s_t, a_t, \boldsymbol{\theta})] \end{aligned}$$

In the above recursive equation, the sums are performed over all possible states and actions and can be exchanged for integrals in continuous MDPs (Rasmussen and Kuss (2004)). The equation informs us that substituting the marginalized dynamics models in the BE, yields the marginalized value function, which can be later used in PI to obtain the expected marginalized optimal value function $\mathbb{E}_{\boldsymbol{\theta}}[Q^{*,\mathcal{W}|\boldsymbol{\theta}}(s_t, a_t)]$.

3.2.1 Epistemic vs aleatoric uncertainty

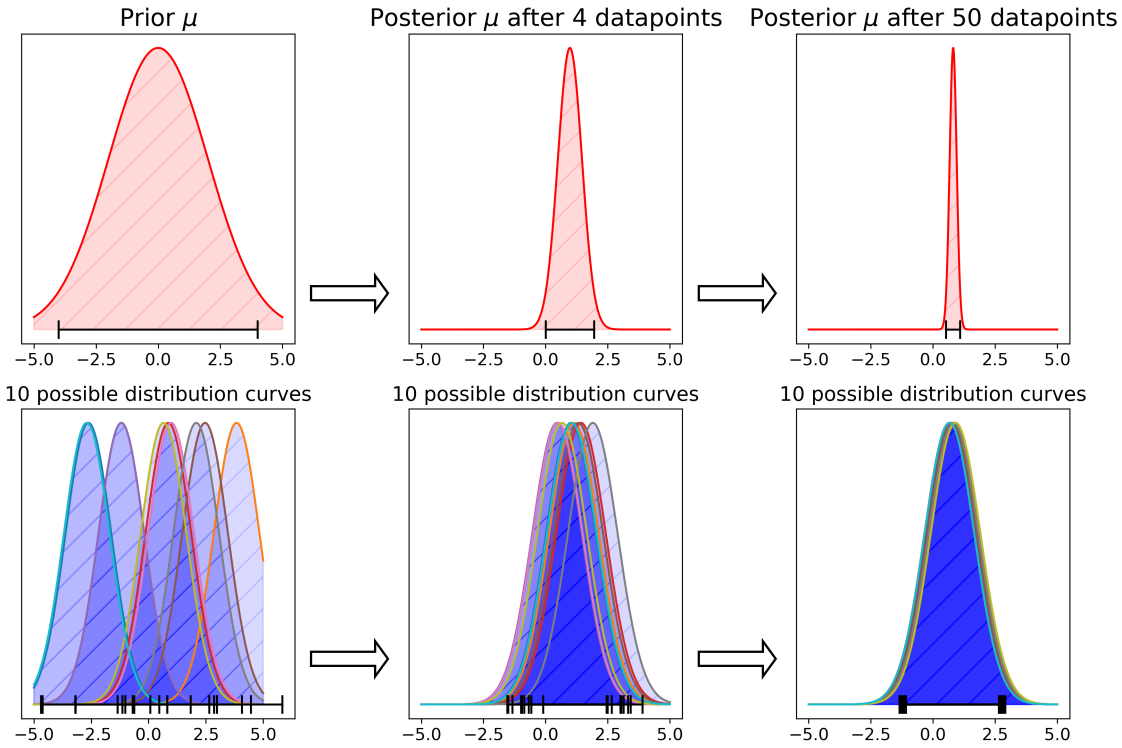


Fig. 3.1 Illustration of differences in epistemic (top row) and aleatoric (bottom row) uncertainty. Top row shows posterior distributions of the Gaussian mean, while the bottom shows possible data curves that posterior induces. Black lines show distribution variances. Data comes from $\mathcal{N}(1, 1)$, prior on mean is $\mathcal{N}(0, 4)$.

As argued before, it is likely that in the beginning stages of training, the estimates $Q^{*,\mathcal{W}|\boldsymbol{\theta}}$ will deviate significantly from the true value $Q^{*,\mathcal{W}}$. Knowing the posterior distribution of $Q^{*,\mathcal{W}|\boldsymbol{\theta}}$ can greatly aid the agent in the search for optimal actions, as it allows it to understand that some states are more explored than others. A recently emerging field called Distributional RL (Bellemare et al. (2017), Dabney et al. (2018)) deals with exactly approximating the posterior value distributions but it has been shown experimentally that simply knowing the posterior variance is enough to deliver huge efficiency improvements (Mavrin et al. (2019)).

Whereas we have ready access to the posterior variances of the reward and transition models, we would like to have a similar concept for the posterior variance of $Q^{*,\mathcal{W}\,|\,\boldsymbol{\theta}}$. Unlike the posterior dynamics variances, we also require the posterior value variance to be propagated across all state-action pairs (Janz et al. (2018)) - meaning that even though dynamics variances might shrink for certain states and actions, their value variances should still remain high if other successor states still possess a lot of uncertainty.

Using the law of iterated variances (Weiss et al. (2006)) on the variance of return, we can see two types of uncertainty present under the model parametrization:

$$\mathrm{Var}_{\pi,\mathcal{W}}[G_{\pi,t}|s_t,a_t] = \underbrace{\mathbb{E}_{\boldsymbol{\theta}}\left[\,\mathrm{Var}_{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}}[G_{\pi,t}|s_t,a_t,\boldsymbol{\theta}]\right]}_{\text{aleatoric uncertainty}} + \underbrace{\mathrm{Var}_{\boldsymbol{\theta}}\left[\,\mathbb{E}_{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}}[G_{\pi,t}|s_t,a_t,\boldsymbol{\theta}]\right]}_{\text{epistemic uncertainty}} \tag{3.20}$$

Aleatoric uncertainty is the expected irreducible variance of return, that arises due to the inherent stochasticity of $\mathcal{W}$. The *epistemic* uncertainty, is the variance of the value function that arises due to our uncertain posteriors and which vanishes once we observe more and more data. The difference between the two is visualised in Figure 3.1, where we can see that as we see more data-points, the posterior variance (epistemic) reduces, while the predictive variance (aleatoric) stays the same (in this model specification the predictive distribution is known to have unit variance).

The epistemic uncertainty $\mathrm{Var}_{\boldsymbol{\theta}}\left[\,\mathbb{E}_{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}}[G_t^{\pi}|s_t,a_t,\boldsymbol{\theta}]\right] = \mathrm{Var}_{\boldsymbol{\theta}}[Q^{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}}(s_t,a_t)]$ is the generalisation of the dynamics posterior variance onto the long term expected return. It represents the variance of the value function estimate purely due to the uncertain representation of the world models. Following the argument above, it makes sense to consider it in the action selection process as it gives the true measure of the agents skepticism about the world it interacts with, with respect to prior beliefs. In the next section, we will see several ways in which we can calculate this quantity.

3.3 Estimates of the epistemic uncertainty of return

As we saw from equation 3.20, the uncertainty of the return $G_{\pi,t}$ separates into an epistemic and aleatoric part. In this section, we will look at different ways of approximating the epistemic component of this variance, which also happens to be the marginalized variance of the value function $Q^{\pi,\mathcal{W}|\boldsymbol{\theta}}$.

3.3.1 Monte Carlo estimate

One possible way of calculating the epistemic uncertainty of return, is to sample a set of model parameters from the parameter posteriors, calculate the optimal value $Q^{*,\mathcal{W}|\boldsymbol{\theta}}$ for each member of the set and look at the variance of the sample. This approach is formally called Monte Carlo sampling.

In detail, to obtain the estimate, we sample a set of N model parameters from the posterior, so $\{\boldsymbol{\theta}_k\}_{k=1}^N$ where $\boldsymbol{\theta}_k \sim \boldsymbol{\theta}\,|\mathcal{D}_t$. For each $\boldsymbol{\theta}_k$, we calculate $Q^{*,\mathcal{W}|\boldsymbol{\theta}_k}$ using methods from section 2, like policy or value iteration. We then compute the variance of the sample as our estimate of the epistemic uncertainty:

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{\theta}}\left[Q^{*,\mathcal{W}|\boldsymbol{\theta}}(s_t,a_t)\right] &\approx \frac{1}{N}\sum_{k=1}^{N} Q^{*,\mathcal{W}|\boldsymbol{\theta}_k}(s_t,a_t) := M(s_t,a_t) \\
\mathrm{Var}_{\boldsymbol{\theta}}\left[Q^{*,\mathcal{W}|\boldsymbol{\theta}}(s_t,a_t)\right] &\approx \frac{1}{N-1}\sum_{k=1}^{N}(Q^{*,\mathcal{W}|\boldsymbol{\theta}_k}(s_t,a_t) - M(s_t,a_t))^2
\end{aligned} \tag{3.21}$$

A desirable property of the Monte Carlo estimate is that as $N \to \infty$, equation 3.21 tends to the true $\mathrm{Var}_{\boldsymbol{\theta}}\left[Q^{*,\mathcal{W}|\boldsymbol{\theta}}(s_t,a_t)\right]$ (Weiss et al. (2006)). One disadvantage of that approach, is that it is very computationally heavy, since we have to go through the entire policy iteration process N times.

3.3.2 Uncertainty Bellman Equation

Another possible way of estimating the epistemic uncertainty, that avoids heavy computation like Monte Carlo, is to try and derive its bound. The Uncertainty Bellman Equation (UBE) (O'Donoghue et al. (2017)), leverages the BEs to connect epistemic uncertainties across time steps. The authors show that the UBE has a unique fixed point at the upper bound of the variance of the posterior distribution of $Q^{\pi,\mathcal{W}|\boldsymbol{\theta}}$.

In the derivation of the upper bound, the authors make use of two assumptions.

Assumption 1. *The MDP is a directed acyclic graph (DAG).*

The first assumption implies that the agent cannot revisit the same state s twice, during a course of an episode. This is a common assumption in RL literature (O'Donoghue (2018), Osband et al. (2017)) and is frequently violated in any experimental setup, which is also the case in the UBE paper. It is employed purely because of mathematical convenience, in order to separate the reward and dynamics terms:

$$\begin{aligned}&\text{Var}_{\boldsymbol{\theta}}\left[\mathbb{E}_{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}}[R_{t+1}|s,a,\boldsymbol{\theta}] + p(s'|s,a,\boldsymbol{\theta})Q^{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}}(s',a')\right]\\ =\,&\text{Var}_{\boldsymbol{\theta}}\left[\mathbb{E}_{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}}[R_{t+1}|s,a,\boldsymbol{\theta}]\right] + \text{Var}_{\boldsymbol{\theta}}\left[p(s'|s,a,\boldsymbol{\theta})Q^{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}}(s',a')\right]\end{aligned}$$

which without the DAG assumption would not be possible. The authors point out that any episodic task MDP $\mathcal{W}$ that does not satisfy the DAG criterion can be turned into one by a process of unrolling, so creating t_{max} copies of $\mathcal{W}$ per every time-step.

Assumption 2. *Mean rewards are bounded and in a known interval:*

$$\mathbb{E}_{\mathcal{W}}[\mathcal{R}_s^a] \in [R_{min}, R_{max}], \forall s \in \mathcal{S}, a \in \mathcal{A}$$

This assumption further implies that the the Q-values can be bounded, so that $|Q^{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}}(s,a)| \leq Q_{\text{max}}, \forall s \in \mathcal{S}, a \in \mathcal{A}$ for any policy π. Making use of these two assumptions, the authors derive the upper bound for the epistemic uncertainty:

$$\begin{aligned}\text{Var}_{\boldsymbol{\theta}}\left[Q^{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}}(s_t,a_t)\right] \leq\,&\nu^{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}}(s_t,a_t)\\ &+\gamma^2\sum_{s',a'}\pi(a'|s')\,\mathbb{E}_{\boldsymbol{\theta}}[p(s'|s_t,a_t,\boldsymbol{\theta})]\,\text{Var}_{\boldsymbol{\theta}}\left[Q^{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}}(s',a')\right]\end{aligned} \tag{3.22}$$

where $\nu^{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}}(s_t,a_t)$ is called the *local uncertainty* and is given by:

$$\nu^{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}}(s_t,a_t) = \text{Var}_{\boldsymbol{\theta}}\left[\mathbb{E}_{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}}[R_{t+1}|s_t,a_t,\boldsymbol{\theta}]\right] + Q_{\text{max}}^2\sum_{s'}\frac{\text{Var}_{\boldsymbol{\theta}}[p(s'|s_t,a_t,\boldsymbol{\theta})]}{\mathbb{E}_{\boldsymbol{\theta}}[p(s'|s_t,a_t,\boldsymbol{\theta})]} \tag{3.23}$$

In the original UBE paper, the authors give equation 3.22 without including the γ^2 term, because in their derivation they focus on the undiscounted episodic tasks. However, looking at their implementation and generalization to neural networks, one can see that they introduce the discount γ when testing on continuing tasks.

Equation 3.22 gives a recursive equation for the epistemic variance for any policy π, which means that it also holds for the optimal policy π^*. It can be solved by DP using standard

methods like PI. The authors prove that the operation is a contraction, which in the continuing case is readily seen directly from equation 3.22, since it has the same form as the BE, with an augmented reward signal. Because the BE is proven to be a contraction for any MDP, the UBE is also a contraction by contruction.

3.3.3 Direct derivation using Bellman consistency

Similarly to the UBE, we propose utilizing the Bellman consistency equation for the Q-value, in order to exactly derive the equation for the epistemic uncertainty of return. We introduce two estimates, one bound and one approximation, which we call the Epistemic Uncertainty Bound (EUB) and Epistemic Uncertainty of Decorrelated Value (EUDV) respectively. We start by reminding ourselves that the value function under the parametrized MDP $\mathcal{W} \mid \boldsymbol{\theta}$, obeys the Bellman equation given in 2.7:

$$Q^{\pi,\mathcal{W}\mid\boldsymbol{\theta}}(s_t,a_t) = \mathbb{E}_{\pi,\mathcal{W}\mid\boldsymbol{\theta}}[R_{t+1}|s_t,a_t,\boldsymbol{\theta}] + \sum_{s',a'} \pi(a'|s')p(s'|s_t,a_t,\boldsymbol{\theta})Q^{\pi,\mathcal{W}\mid\boldsymbol{\theta}}(s',a') \quad (3.24)$$

with the sum naturally exchangeable for an integral in continuous state-action spaces. The ensuing derivations for both proposed estimates require the following assumption:

Assumption 1. *The MDP is a directed acyclic graph (DAG).*

Our analysis shares this common supposition with the derivation of UBE and many other applications in the RL literature. Taking the variance with respect to model parameters $\boldsymbol{\theta}$ on both sides of equation 3.24 and using the above DAG assumption to simplify, we get:

$$\begin{aligned}
\mathrm{Var}_{\boldsymbol{\theta}}[Q^{\pi,\mathcal{W}\mid\boldsymbol{\theta}}(s_t,a_t)] = \mathrm{Var}_{\boldsymbol{\theta}}&\left[\mathbb{E}_{\pi,\mathcal{W}\mid\boldsymbol{\theta}}[R_{t+1}|s_t,a_t,\boldsymbol{\theta}]\right] \\
&+ \gamma^2 \sum_{a',s'} \pi(a'|s')^2 \Big[\mathbb{E}_{\boldsymbol{\theta}}\left[Q^{\pi,\mathcal{W}\mid\boldsymbol{\theta}}(s',a')\right]^2 \mathrm{Var}_{\boldsymbol{\theta}}[p(s'|s_t,a_t,\boldsymbol{\theta})] \\
&\quad + \mathrm{Var}_{\boldsymbol{\theta}}[Q^{\pi,\mathcal{W}\mid\boldsymbol{\theta}}(s',a')]\left[\mathbb{E}_{\boldsymbol{\theta}}[p(s'|s_t,a_t,\boldsymbol{\theta})]^2 + \mathrm{Var}_{\boldsymbol{\theta}}[p(s'|s_t,a_t,\boldsymbol{\theta})]\right]\Big] \\
&+ 2\gamma^2 \sum_{\substack{a',s' \\ a''\neq a', s''\neq s'}} \pi(a'|s')^2 C(s',a',s'',a''|s_t,a_t,\boldsymbol{\theta}) \qquad (3.25)
\end{aligned}$$

where

$$\begin{aligned}
C(s',a',s'',a''|s_t,a_t,\boldsymbol{\theta}) = \mathrm{Cov}_{\boldsymbol{\theta}}\big[&p(s'|s_t,a_t,\boldsymbol{\theta})Q^{\pi,\mathcal{W}\mid\boldsymbol{\theta}}(s',a') \\
&p(s''|s_t,a_t,\boldsymbol{\theta})Q^{\pi,\mathcal{W}\mid\boldsymbol{\theta}}(s'',a'')\big] \qquad (3.26)
\end{aligned}$$

Details of this derivation can be found in Appendix A.2.1. From equation 3.25 we can compute both of our proposed epistemic uncertainty estimates.

Epistemic Uncertainty of Decorrelated Value (EUDV)

We obtain the EUDV estimate by assuming that:

Assumption 2. *The value function evaluated at different state-actions and weighed by the probability of arrival, is decorrelated:*

$$C(s', a', s'', a''|s_t, a_t, \boldsymbol{\theta}) = 0 \quad \forall s_t, s', s'' \in \mathcal{S} \text{ s.t. } s'' \neq s' \text{ and } \forall a_t, a', a'' \in \mathcal{A} \text{ s.t. } a' \neq a''$$

The assumption postulates, that marginalizing over all possible world parameters, there is no correlation between the value functions of state-action pairs weighed by the probability of arrival from any initial s_t, a_t. In general, states that are reachable from s_t, a_t will likely have similar Q-values - this is because if a high valued state is reachable from a nearby state, the nearby state will also have high value as it is possible for the agent to quickly transfer there. With that in mind, this assumption is likely to be violated in many MDPs, similarly to the DAG supposition.

The benefit of neglecting the covariance of value term $C(s', a', s'', a''|s_t, a_t, \boldsymbol{\theta})$, is that it greatly simplifies the epistemic uncertainty formula given in 3.25. It is clear, that as the agent gathers more data, the EUDV estimate will tend to 0. This is because:

$$\begin{aligned} \text{Var}_{\boldsymbol{\theta}}\left[\mathbb{E}_{\pi, \mathcal{W}|\boldsymbol{\theta}}[R_{t+1}|s_t, a_t, \boldsymbol{\theta}]\right] &\to 0 \\ \text{Var}_{\boldsymbol{\theta}}[p(s'|s_t, a_t, \boldsymbol{\theta})] &\to 0 \end{aligned}$$

as $|\mathcal{D}_t| \to \infty$. This is a desirable property, referring to Figure 3.1 and the previous discussion about types of uncertainties. A interesting addition to the equation of the EUDV estimate, is that while the epistemic variance of the transitions are relatively big, it depends on the accumulation of the squared marginalized value functions. This is an advantageous property, as it enables the agent to scale the epistemic uncertainty directly to the level of the Q-value, rather than using a fixed Q_{max} as in UBE.

We can see that the EUDV estimate operator is also a contraction and hence has a unique fixed point. This follows a similar argument as in UBE, since we can consider an augmented

version of the MDP, having a reward signal:

$$r(s_t, a_t) = \text{Var}_{\boldsymbol{\theta}}\left[\mathbb{E}_{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}}[R_{t+1}|s_t, a_t, \boldsymbol{\theta}]\right] + \gamma^2 \sum_{a',s'} \pi(a'|s')^2\, \mathbb{E}_{\boldsymbol{\theta}}\left[Q^{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}}(s', a')\right]^2 \text{Var}_{\boldsymbol{\theta}}[p(s'|s_t, a_t, \boldsymbol{\theta})] \tag{3.27}$$

and transition probabilities:

$$t(s'|s_t, a_t) = \mathbb{E}_{\boldsymbol{\theta}}[p(s'|s_t, a_t, \boldsymbol{\theta})]^2 + \text{Var}_{\boldsymbol{\theta}}[p(s'|s_t, a_t, \boldsymbol{\theta})]$$

In the discrete Dirichlet transition model, $0 \leq t(s'|s_t, a_t) \leq 1$ so that it is a proper definition of probability. Then, the EUDV estimate obeys a familiar Bellman equation, with $r(s_t, a_t)$ and $t(s'|s_t, a_t)$ as properly defined dynamics. Therefore, the EUDV has a unique fixed point.

Epistemic Uncertainty Bound

In addition to the above estimate, we also propose a bound on the epistemic uncertainty of return, which does not assume state-action values to be decorrelated. The derivation of the bound is possible thanks to the following assumption:

Assumption 3. *The value function is independent of the transition probabilities:*

$$Q^{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}}(s', a') \perp p(s''|s_t, a_t, \boldsymbol{\theta}) \quad \forall s_t, s', s'' \in \mathcal{S}, s' \neq s'' \tag{3.28}$$

$$\forall a_t, a' \in \mathcal{A} \tag{3.29}$$

It means that the state-action values are independent (with respect to parameters $\boldsymbol{\theta}$) of the transition probabilities. This, again, is a strong assumption as we know from equation 3.24 that Q-values depend directly on these probabilities. It is, however, a much lighter assumption than in the EUDV derivation and allows us to formulate a neat bound on the value-correlation coefficient:

$$\begin{aligned} C(s', a', s'', a''|s_t, a_t, \boldsymbol{\theta}) \leq \sqrt{u^{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}}(s', a') u^{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}}(s'', a'')}\Big(& \\ \left|\text{Cov}_{\boldsymbol{\theta}}[p(s'|s_t, a_t, \boldsymbol{\theta}), p(s''|s_t, a_t, \boldsymbol{\theta})]\right| + & \\ \mathbb{E}_{\boldsymbol{\theta}}[p(s'|s_t, a_t, \boldsymbol{\theta})]\, \mathbb{E}_{\boldsymbol{\theta}}[p(s''|s_t, a_t, \boldsymbol{\theta})]\Big) & \\ + \left|\, \mathbb{E}_{\boldsymbol{\theta}}[Q^{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}}(s', a')]\, \mathbb{E}_{\boldsymbol{\theta}}[Q^{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}}(s', a')] \text{Cov}_{\boldsymbol{\theta}}[p(s'|s_t, a_t, \boldsymbol{\theta}), p(s''|s_t)]\right| & \end{aligned} \tag{3.30}$$

where we used $u^{\pi,\mathcal{W}|\boldsymbol{\theta}}(s,a) = \text{Var}_{\boldsymbol{\theta}}[Q^{\pi,\mathcal{W}|\boldsymbol{\theta}}(s,a)]$ as a shorthand for the epistemic uncertainty. See Appendix A.2.2 for a full derivation.

It is tempting to think, that since the bound on the value correlation coefficient is given in terms of the epistemic uncertainties, it can be directly used in DP to recursively update an arbitrary initial guess. However, we were not able to prove that the resulting consistency equation 3.25 has a unique fixed point and preliminary experiments on MDPs defined in chapter 5 showed that the DP approach actually diverges.

To mitigate this failure, we propose excluding the $C(s', a', s'', a''|s_t, a_t, \boldsymbol{\theta})$ term from the DP update equation and adding it back to the uncertainty estimate in the aftermath. In that case, the EUB estimate becomes the EUDV estimate with an added correlation term given in equation 3.30.

Due to the absolute values present in the additional term, we have $C(s', a', s'', a''|s_t, a_t, \boldsymbol{\theta}) \geq 0$, which has an effect of making the epistemic uncertainty quite loose. The disadvantage of that, is that some value correlations can actually be negative in various MDPs, which the EUB term fails to capture.

Chapter 4

Action selection under uncertainty

In the previous section, we presented a few methods of estimating the epistemic uncertainty of return, value representing the agents uncertainty over the state-action values with respect to its world parametrization. In this section, we will discuss ways of the agent leveraging that quantity in order to efficiently make decisions in the environment.

Having an action selection heuristic coupled with ways of calculating the posterior mean and variance of Q-values, will give us a general BRL algorithm for efficient interaction with the environment $\mathcal{W}$. Such an algorithm is given in Algorithm 5.

Algorithm 5 General BRL algorithm for agent-world interaction

Input epistemic uncertainty estimate procedure `EpVar`, action selection under uncertainty routine `ActSel` and priors on $\boldsymbol{\theta}$.
procedure GENERALBRL(`EpVar`, `ActSel`, $\boldsymbol{\theta}$)
    Initialize $\mathcal{R}$ and $\mathcal{T}$ with respect to the prior on $\boldsymbol{\theta}$.
    Initialize $\mathcal{D}_t = []$ as empty list.
    for t = 1,2,3,...,t_{max} **do**
        Observe s_t.
        Compute $\mathbb{E}_{\boldsymbol{\theta}}[Q^{*,\mathcal{W}\,|\,\boldsymbol{\theta}}]$ using equation 3.19 and PI.
        Compute $\text{Var}_{\boldsymbol{\theta}}[Q^{*,\mathcal{W}\,|\,\boldsymbol{\theta}}]$ using `EpVar`.
        Compute a_t from `ActSel`$(\mathbb{E}_{\boldsymbol{\theta}}[Q^{*,\mathcal{W}\,|\,\boldsymbol{\theta}}], \text{Var}_{\boldsymbol{\theta}}[Q^{*,\mathcal{W}\,|\,\boldsymbol{\theta}}])$.
        Observe s_{t+1} and R_{t+1}, append the tuple $(s_t, a_t, R_{t+1}, s_{t+1})$ to $\mathcal{D}_t$.
        Compute posteriors $\boldsymbol{\theta}\,|\mathcal{D}_t$
    end for
    return $Q^{*,\mathcal{W}\,|\,\boldsymbol{\theta}}$

4.1 Upper Confidence Bound

The simplest and quite powerful method of incorporating the epistemic uncertainty estimate into action selection, is the Upper Confidence Bound (UCB) method (Lai and Robbins (1985)). In UCB, the agent picks actions according to their mean value estimates and the potential of them actually being optimal. This means that the action taken at time t in state s_t becomes:

$$a_t = \underset{a'}{\operatorname{argmax}} \left[\mathbb{E}_{\boldsymbol{\theta}}[Q^{*,\mathcal{W}|\boldsymbol{\theta}}(s_t, a')] + \beta \operatorname{Var}_{\boldsymbol{\theta}}[Q^{*,\mathcal{W}|\boldsymbol{\theta}}(s_t, a')]^{1/2} \right]$$

where $\beta \in \mathbb{R}$ is a hyperparameter describing how much the agent values the potential of certain states to be optimal. A high value of β will cause the agent to over-explore, as even small improvements to the highest optimal $Q^{*,\mathcal{W}|\boldsymbol{\theta}}$ will seem worthwhile for the agent to review. A too small β can in turn cause under-exploration, which can lead to the agent focusing its attention on a sub-optimal part of the environment.

UCB action selection is GLIE, because as $|\mathcal{D}_t| \to \infty$ the epistemic variance vanishes and the agent starts picking the action with the highest optmial value function. This is a conceptual benefit, which with the right setting of the priors and β guarantees that the agent will find the true optimal $Q^{*,\mathcal{W}}$ for any MDP $\mathcal{W}$. The biggest disadvantage of UCB is that the parameter β needs to be tuned so that learning is as efficient (no over-exploration) and effective (no under-exploration) as possible.

4.2 Thompson sampling

One of the more famous methods for action selection in the presence of uncertainty is Thompson sampling (Thompson (1933)), a heuristic historically developed to address the exploration-exploitation dilemma in Multi Armed Bandits (MAB), a simple environment we will consider later in this chapter. Its simplicity and optimality properties have been often praised by RL researchers, yet the difficulty of obtaining a Q-value posterior under approximations in complex environments has made it a promising but often unavailable option (Ghavamzadeh et al. (2016)).

Thompson sampling is based on posterior sampling. It necessitates having a posterior distribution over optmial Q-values, from which samples can be drawn. The action with the highest Q-value sample is then picked. One problem, is that in our setting, we only have access to the posterior mean and variance, rather than the whole posterior distribution. A sensible way of mitigating this problem, is to estimate the posterior Q distribution by a Gaussian (Mavrin et al. (2019), O'Donoghue et al. (2017)). Specifically, in state s_t, for each possible

action $a \in \mathcal{A}$ the agent samples:

$$q_a \sim \mathcal{N}(\mathbb{E}_{\boldsymbol{\theta}}[Q^{*,\mathcal{W}\,|\,\boldsymbol{\theta}}(s_t, a)], \beta^2 \operatorname{Var}_{\boldsymbol{\theta}}[Q^{*,\mathcal{W}\,|\,\boldsymbol{\theta}}(s_t, a)])$$

for $\beta \in \mathbb{R}$ a analogous hyperparameter to the β in UCB. Action is then picked that satisfies:

$$a_t = \underset{a'}{\operatorname{argmax}} \left[q_{a'}\right]$$

Since the epistemic uncertainty vanishes with agents experience, Thompson sampling is also considered GLIE.

One obvious disadvantage of Thompson sampling is its random nature. This often makes it unstable and unreliable, leading to the agent discovering optimal Q's in some but not all seeds (Mingxi Li (2017)). Another consequence of that is that once the agent actually finds the optimal Q and is quite sure of it, due to sampling, the algorithm can still from time to time return sub-optimal actions, so that the GLIE property might take a very long time to come into effect.

4.3 Posterior Sampling for Reinforcement Learning

An idea very closely related to Thompson sampling, called Posterior Sampling for Reinforcement Learning (PSRL) (Osband et al. (2013)), avoids the necessity of having a tractable posterior over Q-values, or even knowing its mean and epistemic uncertainty. This simple but provably efficient algorithm, is able to leverage the modelling uncertainty by directly sampling a possible world $\mathcal{W} \,|\, \boldsymbol{\theta}$ at each iteration and acting greedily with respect to that sample. Because the parameters $\boldsymbol{\theta}$ already contain the uncertainty about the true dynamics of $\mathcal{W}$, this approach achieves a similar goal to Thompson sampling, without the need of having a routine to compute $\operatorname{Var}_{\boldsymbol{\theta}}[Q^{*,\mathcal{W}\,|\,\boldsymbol{\theta}}]$. Specifically, at each step and state s_t, the agent samples a MDP $\mathcal{W}_t$ from the posterior over parameters:

$$\boldsymbol{\theta}_t \sim \boldsymbol{\theta}\,|\mathcal{D}_t$$
$$\mathcal{W}_t = \mathcal{W}\,|\,\boldsymbol{\theta}_t$$

and picks an action which is optimal in this sampled world:

$$a_t = \underset{a'}{\operatorname{argmax}} \left[Q^{*,\mathcal{W}_t}(s_t, a')\right]$$

with $Q^{*,\mathcal{W}_t}$ easily computable either by PI or VI as explained in section 2.4.1. Practically, the computed optimal policy at time t with respect to the sampled world $\mathcal{W} \mid \boldsymbol{\theta}_t$ is followed over the course of T time-steps. This helps with regards to the computational complexity and has minor effects on the efficiency of the algorithm in finding the optimal policy.

By definition, PSRL is naturally GLIE, as it acts greedily with respect to a sampled MDP $\mathcal{W}_t$ at each timestep. Since the posteriors over the parameters are guaranteed to shrink with agents experience, PSRL is sure to find and follow π^* at some point during the learning process. The authors of the paper give a bound on the expected regret (see full definition in chapter 5), a quantity representing the loss of reward that the PSRL agent is expected to achieve in comparison to a optimally behaving agent and show, that it is significantly lower than the regret bounds for count-based algorithms relying purely on the OFU principle.

4.4 Knowledge Gradient

The Knowledge Gradient (KG) (Gupta and Miescke (1996)) is an action selection heuristic, which has been extensively studied in the Multi Armed Bandit (MAB) setting, but which remains relatively unknown in more general RL research. In this section, we present KG as it is known in the MAB environments and will generalize the idea to other, stateful, MDPs.

4.4.1 Knowledge Gradient in Multi Armed Bandits

A Multi Armed Bandit (MAB) MDP, is a very simple environment in which there is only one state. The agent remains in this state forever, regardless of its choice of actions. Each action is mapped to a *bandit*, which spits out a reward according to some stationary distribution. The task of the agent is to find the *best bandit*, so the one whose reward distribution has the highest mean, in the least amount of steps, and continue playing it until the end of the episode.

The KG heuristic is based on a simple idea, that at any given decision point, the agent has a last chance for exploration, before he turns to behaving fully greedily. This means, that the agent at time t imagines that a_t will be the last exploratory action it makes, after which a_k for $k > t$ are chosen greedily. This in turn forces the agent to pick a_t so that it has the highest probability of uncovering an action which is better compared to its previous belief about the best action (Powell and Ryzhov (2012)). The theory behind KG is derived in the Bayesian setting, which makes it easily transferable onto our notation.

The authors of the original paper proposing KG, introduce a variable called the *state of knowledge* after t steps in the environment (where one step is considered to be one pull of the arm), represented as $S^t = (\boldsymbol{\vartheta}^t, \boldsymbol{\sigma}^t)$, where $\boldsymbol{\vartheta}^t \in \mathbb{R}^{|\mathcal{A}|}$ is the vector of predictive mean rewards

of pulling each arm and $\boldsymbol{\sigma}^t \in \mathbb{R}^{|\mathcal{A}|}$ is the vector of predictive variances of these rewards. In the notation we defined in section 3, we can make the following association:

$$\vartheta_x^t \longleftrightarrow \mathbb{E}_{\boldsymbol{\theta}} \left[\mathbb{E}_{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}}[R_{t+1}|s_t, a_t = x, \boldsymbol{\theta}] \right]$$
$$\sigma_x^t \longleftrightarrow \text{Var}_{\boldsymbol{\theta}} \left[\mathbb{E}_{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}}[R_{t+1}|s_t, a_t = x, \boldsymbol{\theta}] \right]$$

The value of being in the information state S^t is given by the highest mean reward that we believe we can achieve:

$$V(S^t) = \max_x \vartheta_x^t$$

Given that we choose to pull the arm x', so that $a_t = x'$, the value of the next step knowledge state will be:

$$V(S^{t+1}|a_t = x') = \max_x \vartheta_x^{t+1}$$

The entire idea behind KG is to choose x' so that we maximize the expected value of $V(S^{t+1})$ (Frazier (2009)). This is equivalent to choosing x' to maximize the incremental value improvement (Powell and Ryzhov (2012)):

$$\nu_x^{KG,t} = \mathbb{E}[V(S^{t+1}|a_t = x) - V(S^t)]$$

a quantity we call the Knowledge Gradient. The right hand side of the equation resembles the gradient of $V(S^t)$ with respect to x' and hence the KG name.

In the following analysis, we will assume that the MAB arms are independent and normally distributed. A well known result (DeGroot (1970)) states that:

$$\vartheta_x^{t+1} \sim \mathcal{N}(\vartheta_x^t, \tilde{\sigma}_x^{2,t}) \tag{4.1}$$

where $\tilde{\sigma}_x^{2,n}$ is the expected reduction in the uncertainty about the reward after we play arm x. The reduction in uncertainty is given by:

$$\tilde{\sigma}_x^{2,t} = \sigma_x^{2,t} - \mathbb{E}[\sigma_x^{2,t+1}]$$

where $\mathbb{E}[\sigma_x^{2,t+1}]$ represents the expected variance of reward of arm x, after we play arm x at time-step t. The expectation is taken over all possible rewards $R_{t+1}|s_t, a_t = x, \boldsymbol{\theta}$. In the Bayesian setting, as we can see in formulas given in section 3.1, the mean and variance of the reward from a given arm depends on the number of samples seen n, the mean of these

samples $\overline{x}$ and their sample variance S. Since the only predictions about the future we can make are based on our posterior beliefs about the reward distribution up to time t, we can compute $\mathbb{E}[\sigma_x^{2,t+1}]$ by adding ϑ_x^t as a new 'fake' observation to the samples, recomputing the posterior and observing the new variance that it induces. This approach is equivalent to integrating over all possible values the reward can take, weighed by their probability under the posterior $\boldsymbol{\theta}\,|\mathcal{D}_t$ (Wang and Powell (2016)).

To compute the Knowledge gradient heuristic, the authors draw ideas from Bayesian optimisation (BO), specifically the concept of *expected improvement*. Expected improvement is an acquisition function used in BO, that calculates the increase, in expectation, that pulling an arm causes to the maximal mean reward among all arms. In the original paper, the authors consider the expected improvement over the next best action. In other words, the expected improvement of pulling arm x measures the improvement $I_x = \vartheta_x^t - \max_{x' \neq x} \vartheta_{x'}^{t+1}$ weighed by the probability of each individual value of ϑ_x^{t+1}, which we get from equation 4.1. That gives us the equation for the Knowledge gradient of pulling arm x:

$$\nu_x^{KG,t} = -|\vartheta_x^t - \max_{x' \neq x} \vartheta_{x'}^t|\Phi(-\Big|\frac{\vartheta_x^t - \max_{x' \neq x} \vartheta_{x'}^t}{\tilde{\sigma}_x^t}\Big|) + \tilde{\sigma}_x^t \phi(-\Big|\frac{\vartheta_x^t - \max_{x' \neq x} \vartheta_{x'}^t}{\tilde{\sigma}_x^t}\Big|) \quad (4.2)$$

where $\Phi(\cdot)$ is the cumulative standard normal distribution and $\phi(\cdot)$ is the standard normal density. For the full derivation of the KG heuristic, we refer the reader to the original paper (Gupta and Miescke (1996)).

4.4.2 Generalization of Knowledge Gradient to stateful MDPs

In the previous section, we derived the KG heuristic for the MAB environment, using the assumption that the actions are independent and normally distributed. In the following, we will generalize that metric to stateful MDPs - environments where there is more than one state.

In order to incorporate the state and transition dynamics into the equation, we make use of a intuitive generalisation of the knowledge state:

$$S^t = (\mathbb{E}_{\boldsymbol{\theta}_t}[Q^{*,\mathcal{W}\,|\,\boldsymbol{\theta}_t}], \mathrm{Var}_{\boldsymbol{\theta}_t}[Q^{*,\mathcal{W}\,|\,\boldsymbol{\theta}_t}]) \quad (4.3)$$

where $\boldsymbol{\theta}_t$ represents the posterior over world model parameters after seeing t steps, so:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}\,|\mathcal{D}_t$$

In that formulation, at any given state s_t, the agent will try to pick an action that will most improve upon its estimation of the currently optimal Q-value, in expectation.

We can imagine a stateful MDP to be a series of states that each individually resemble a separate MAB problem. Just like in the MAB, the *mean reward* ϑ_x^t is the quantity that represents the overall worthiness of picking $a_t = x$, in a stateful MDP at state s_t, the *mean Q-value* $\mathbb{E}_{\boldsymbol{\theta}_t}[Q^{*,\mathcal{W}\,|\,\boldsymbol{\theta}_t}(s_t, x)]$ is the analogous quantity that captures the overall worth of $a_t = x$, taking into account transition dynamics and both immediate and future rewards. We can therefore make the following association with the previous sections results, that at state s_t:

$$\vartheta_x^t \longleftrightarrow \mathbb{E}_{\boldsymbol{\theta}_t}[Q^{*,\mathcal{W}\,|\,\boldsymbol{\theta}_t}(s_t, x)]$$
$$\sigma_x^t \longleftrightarrow \text{Var}_{\boldsymbol{\theta}_t}[Q^{*,\mathcal{W}\,|\,\boldsymbol{\theta}_t}(s_t, x)]$$

and we can treat action selection across different states as an independent optimization problem on Q-values, since they already encompass information about state transitions.

In order to be able to imitate the analysis from section 4.4.1, we firstly need to assume that in any s_t, the Q-values of all available actions are independent and normally distributed. Note that we make the exact same assumption while describing Thompson sampling in section 4.2. Secondly, we need to derive analogous expressions for the reduction in uncertainty in state s_t, $\tilde{\sigma}_{s_t,x}^t$:

$$\tilde{\sigma}_{s_t,x}^t = \text{Var}_{\boldsymbol{\theta}_t}[Q^{*,\mathcal{W}\,|\,\boldsymbol{\theta}_t}(s_t, x)] - \mathbb{E}\left[\,\text{Var}_{\boldsymbol{\theta}_{t+1}}[Q^{*,\mathcal{W}\,|\,\boldsymbol{\theta}_{t+1}}(s_t, x)]\right] \tag{4.4}$$

where now the expectation needs to be computed over all possible values of $R_{t+1}|s_t, a_t = x, \boldsymbol{\theta}_t$ and $s_{t+1}|s_t, a_t = x, \boldsymbol{\theta}_t$ weighed by their respective probabilities under the posterior $\boldsymbol{\theta}_t$. In the following, we present methods of calculating $\tilde{\sigma}_{s_t,x}^t$ for every epistemic uncertainty estimate that we introduced in section 3.2.

Monte Carlo estimate

Calculating the reduction in uncertainty for the Monte Carlo estimate is very computationally heavy. Analogously to the MAB setting, for each action x, we need to add 'fake' observations to the reward and transition history. After that we need to recompute the Monte Carlo epistemic uncertainty estimate, in the same process as described in section 3.2.

Adding a 'fake' mean observation to the history of rewards is straight-forward. In state s_t and for action x, we simply add $\mathbb{E}_{\boldsymbol{\theta}_t}\left[\,\mathbb{E}_{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}_t}[R_{t+1}|s_t, a_t = x, \boldsymbol{\theta}_t]\right]$ as the simulated next step observation, since as argued before, that is equivalent to integrating over all possible reward values from the posterior.

Simulating a 'fake' mean transition is more problematic, because here the discrete nature of the distribution renders the notion of an average transition undefined. In that case, in order to truly consider the effect of observing a mean transition, we need to sum over the effects of observing each possible transition, weighed by their probability of occurring under the posterior. For the Monte Carlo estimate, recalculating the epistemic variance for each action and for each possible transition requires enormous amounts of computation, in the order $O(N \times |A| \times |S|)$ of policy iterations at every time-step, where N is the number of posterior samples we average for the Monte Carlo estimate. Because of that, we deem this approach unpractical.

Dynamic Programming estimates

It is much easier to calculate the reduction in uncertainty for the epistemic uncertainty estimates that rely on DP. Here, we can leverage the recursive Bellman-like equations that define the epistemic uncertainty to directly find relations for the reduction in uncertainty. Moreover, since unlike in the Monte Carlo estimate, calculation of the epistemic uncertainty does not require sampling from the posteriors, it is much more computationally efficient to compute $\mathbb{E}\left[\operatorname{Var}_{\boldsymbol{\theta}_{t+1}}[Q^{*,\mathcal{W}\,|\,\boldsymbol{\theta}_{t+1}}(s_t,x)]\right]$ naively - an approach we describe next.

As described in their respective sections in Chapter 3, the DP epistemic uncertainty estimates can be calculated using PI (Algorithm 1). A naive approach of calculating the average next-step epistemic uncertainty, involves recalculating the epistemic uncertainty for each possible next transition and mean reward. This means, that at state s_t and for action x:

$$\mathbb{E}\left[\operatorname{Var}_{\boldsymbol{\theta}_{t+1}}[Q^{*,\mathcal{W}\,|\,\boldsymbol{\theta}_{t+1}}(s_t,x)]\right] = \sum_{s'} p(s'|s_t, a_t = x, \boldsymbol{\theta}_t)\operatorname{Var}_{\boldsymbol{\theta}_{t+1}(x,s')}[Q^{*,\mathcal{W}\,|\,\boldsymbol{\theta}_{t+1}(x,s')}] \quad (4.5)$$

where

$$\boldsymbol{\theta}_{t+1}(x,s') = \boldsymbol{\theta}\,|\left[\mathcal{D}_t \cup \{(s_t,x,s',\overline{r})\}\right]$$

is the posterior distribution recalculated with the simulated observation s' and mean reward $\overline{r} = \mathbb{E}_{\boldsymbol{\theta}_t}\left[\mathbb{E}_{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}_t}[R_{t+1}|s_t, a_t = x, \boldsymbol{\theta}_t]\right]$ added to the observation history. This requires running the PI routine $|S| \times |A|$ times at every time-step, to compute the next-step epistemic variance for each considered action and for each possible next state.

However, it is also possible to calculate the reduction in uncertainty without having to resort to DP at all, by finding an explicit formula for $\tilde{\sigma}^t_{s_t,x}$ from the Bellman-like recursive formulas for the estimates. We will present such a formula here for the EUDV estimate. We note that a similar equation can be easily derived for the UBE estimate, which we give in

Appendix. We were not able to obtain a similar equation for the EUB estimate and hence have to resort to the slower naive approach descibed above in all experiments.

For the EUDV estimate, the reduction in uncertainty obeys the following formula:

$$\tilde{\sigma}^t_{s_t,x} = \frac{A_{s_t,x} + \gamma^2 B_{s_t,x}}{1 - \gamma^2 \pi^*(x|s_t)(C_{s_t,x} + D_{s_t,x})} \tag{4.6}$$

where

$$A_{s_t,x} = \underbrace{\mathrm{Var}_{\boldsymbol{\theta}_t}\left[\mathbb{E}_{\pi^*,\mathcal{W}\,|\,\boldsymbol{\theta}_t}[R_{t+1}|s_t,x,\boldsymbol{\theta}_t]\right] - \mathbb{E}\left[\mathrm{Var}_{\boldsymbol{\theta}_{t+1}}\left[\mathbb{E}_{\pi^*,\mathcal{W}\,|\,\boldsymbol{\theta}_{t+1}}[R_{t+1}|s_t,x,\boldsymbol{\theta}_{t+1}]\right]\right]}_{\text{Reduction in reward variance = \textbf{(1)}}}$$

$$B_{s_t,x} = \sum_{a',s'} \pi(a'|s')\,\mathbb{E}_{\boldsymbol{\theta}_t}[Q^{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}_t}(s',a')]^2 \Big[\underbrace{\mathrm{Var}_{\boldsymbol{\theta}_t}[p(s'|s_t,x,\boldsymbol{\theta}_t)] - \mathbb{E}\left[\mathrm{Var}_{\boldsymbol{\theta}_{t+1}}[p(s'|s_t,x,\boldsymbol{\theta}_{t+1})]\right]}_{\text{Reduction in dynamics variance = \textbf{(2)}}} \Big]$$

and

$$C_{s_t,x} = \underbrace{\mathrm{Var}_{\boldsymbol{\theta}_t}[p(s_t|s_t,x,\boldsymbol{\theta}_t)] - \mathbb{E}\left[\mathrm{Var}_{\boldsymbol{\theta}_{t+1}}[p(s_t|s_t,x,\boldsymbol{\theta}_{t+1})]\right]}_{\text{Reduction in dynamics variance = \textbf{(2)}}}$$

$$D_{s_t,x} = \underbrace{\mathbb{E}_{\boldsymbol{\theta}_t}[p(s_t|s_t,x,\boldsymbol{\theta}_t)]^2 - \mathbb{E}\left[\mathbb{E}_{\boldsymbol{\theta}_{t+1}}[p(s_t|s_t,x,\boldsymbol{\theta}_{t+1})]^2\right]}_{\text{Increase in mean squared probability of next state = \textbf{(3)}}}$$

The full derivation of the above result is given in Appendix A.3.1.

Using the above formula for the reduction in uncertainty instead of the naive approach described earlier, significantly reduces the computational complexity of calculating KG, since the number of policy iteration calls reduces from $|S| \times |A|$ to zero! Since the formula for the Knowledge gradient (given in equation 4.2) does not involve the epistemic variance, but simply its reduction, being able to use formula 4.6 greatly improves the computational complexity of the algorithm.

Calculating the individual components of equation 4.6 is straight-forward and can easily be vectorized over possible actions x. The reduction in reward variance **(1)**, in state s_t for action x, can be obtained using formulas given in section 3.2 and:

$$\mathbb{E}\left[\mathrm{Var}_{\boldsymbol{\theta}_{t+1}}\left[\mathbb{E}_{\pi^*,\mathcal{W}\,|\,\boldsymbol{\theta}_{t+1}}[R_{t+1}|s_t,x,\boldsymbol{\theta}_{t+1}]\right]\right] = \mathrm{Var}_{\boldsymbol{\theta}_t(\overline{r})}\left[\mathbb{E}_{\pi^*,\mathcal{W}\,|\,\boldsymbol{\theta}_t(\overline{r})}[R_{t+1}|s_t,x,\boldsymbol{\theta}_t(\overline{r})]\right]$$

where

$$\boldsymbol{\theta}_t(\overline{r}) = \boldsymbol{\theta}_t\,|\left[\mathcal{D}_t \cup \{(s_t,x,\cdot,\overline{r})\}\right]$$

represents the posterior after adding $\overline{r} = \mathbb{E}_{\boldsymbol{\theta}_t}\left[\mathbb{E}_{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}_t}[R_{t+1}|s_t, a_t = x, \boldsymbol{\theta}_t]\right]$ to the observed rewards from state s_t and action x.

The reduction of dynamics variance **(2)** can similarly be obtained using equations from 3.1 and noting that:

$$\mathbb{E}\left[\,\mathrm{Var}_{\boldsymbol{\theta}_{t+1}}[p(s'|s_t, x, \boldsymbol{\theta}_{t+1})]\right] = \sum_{s''} p(s''|s_t, a_t = x, \boldsymbol{\theta}_t)\,\mathrm{Var}_{\boldsymbol{\theta}_t(s'')}[p(s'|s_t, x, \boldsymbol{\theta}_t(s''))]$$

and similarly **(3)**:

$$\mathbb{E}\left[\,\mathbb{E}_{\boldsymbol{\theta}_{t+1}}[p(s'|s_t, x, \boldsymbol{\theta}_{t+1})]^2\right] = \sum_{s''} p(s''|s_t, a_t = x, \boldsymbol{\theta}_t)\,\mathbb{E}_{\boldsymbol{\theta}_t(s'')}[p(s'|s_t, x, \boldsymbol{\theta}_t(s''))]^2$$

where

$$\boldsymbol{\theta}_t(s'') = \boldsymbol{\theta}_t\,|\mathcal{D}_t \cup \{(s_t, x, s'', \cdot)\}$$

is the posterior after adding s'' to the observed transitions from state s_t and action x.

After obtaining the reduction in uncertainty, the KG heuristic is easy to compute using equation 4.2, with the generalisations of ϑ_x^t and $\tilde{\sigma}_x^t$ described in the beginning of this section. The KG policy in state s_t is to choose the action that maximizes the KG heuristic:

$$a_t = \underset{a'}{\mathrm{argmax}}\, \nu_x^{KG,t}$$

We call this version of the Knowledge Gradient the Generalised Knowledge Gradient (GKG).

4.4.3 Monte Carlo sampling versions of the Knowledge Gradient

The simple KG idea of the agent choosing an action that maximizes the future value of the knowledge state, is computed by looking at expected trajectories given the world models. It is also possible to use the same idea, but rather than looking at the expectation of the future knowledge state, use Monte Carlo sampling to *simulate* possible world scenarios. In the following section we present algorithms that achieve that for the KG heuristic. We also extend the Monte Carlo KG idea to simulating N exploratory steps ahead.

The intuitive idea behind the Knowledge Gradient, is that the agent imagines taking one last exploratory step, before committing to acting greedily for the rest of the episode. The agent wants to pick an action which will make its greedy action the most profitable, so that the long-term return is maximized. This intuition leads to a equivalent formulation of the KG

policy:

$$a_t|s_t = \underset{a}{\operatorname{argmax}} \Big[\underbrace{\mathbb{E}_{\boldsymbol{\theta}_t} \left[\mathbb{E}_{\pi, \mathcal{W}\,|\,\boldsymbol{\theta}_t}[R_{t+1}|s_t, a, \boldsymbol{\theta}_t] \right]}_{\text{Exploratory step}} + \gamma \underbrace{\max_{a'} \mathbb{E}_{\boldsymbol{\theta}_{t+1}}[Q^{*,|\,\boldsymbol{\theta}_{t+1}}(s_{t+1}, a')]}_{\text{All subsequent actions taken greedily}} \Big] \quad (4.7)$$

We can see that this policy is exactly equivalent to maximizing the expected mean of return $G_{\pi,t}$, with the parameters $\boldsymbol{\theta}_t$ allowed to be updated after taking the first step. In the 1-step Monte Carlo Knowledge Gradient (MCKG-1), we estimate $\max_{a'} \mathbb{E}_{\boldsymbol{\theta}_{t+1}}[Q^{*,|\,\boldsymbol{\theta}_{t+1}}(s_{t+1}, a')]$ by sampling the reward R_{t+1} and next state s_{t+1} from the models $\mathcal{W}\,|\,\boldsymbol{\theta}_t$, recomputing the posteriors

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}\,|\,\big[\mathcal{D}_t \cup \{(s_t, a, R_{t+1}, s_{t+1})\}\big],$$

calculating $\mathbb{E}_{\boldsymbol{\theta}_{t+1}}[Q^{*,\mathcal{W}\,|\,\boldsymbol{\theta}_{t+1}}]$ and taking its maximal value in state s_{t+1}. This process is summarized in Algorithm 6.

A KG heuristic using Monte Carlo sampling was first introduced in (Ryzhov and Powell (2009)), however, there the authors considered it only for the MAB environments. To the best of our knowledge, our implementation is the first stateful version of the MCKG-1.

Algorithm 6 Monte Carlo Knowledge Gradient

Input: state s_t, the world models $W\,|\,\boldsymbol{\theta}_t$ and discount factor γ.
Output: action to perform at s_t.
procedure MCKG-1($s_t, \mathcal{W}\,|\,\boldsymbol{\theta}_t, \gamma$)
Initialize $KG \in \mathbb{R}^{|\mathcal{A}|}$ to a vector of zeros.
    for a in $\mathcal{A}$ **do**
        Sample $r \sim \mathbb{E}_{\boldsymbol{\theta}_t}[R_{t+1}|s_t, a, \boldsymbol{\theta}_t]$ from the predictive reward model.
        Sample $s_{t+1} \sim \mathbb{E}_{\boldsymbol{\theta}_t}[s_{t+1}|s_t, a, \boldsymbol{\theta}_t]$ from the predictive transition model.
        Compute new temporary posterior $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}\,|\,\big[\mathcal{D}_t \cup \{(s_t, a, r, s_{t+1})\}\big]$.
        Compute $\mathbb{E}_{\boldsymbol{\theta}_{t+1}}[Q^{*,\mathcal{W}\,|\,\boldsymbol{\theta}_{t+1}}]$ using policy iteration.
        $KG_a = \mathbb{E}_{\boldsymbol{\theta}_t}\left[\mathbb{E}_{*,\mathcal{W}\,|\,\boldsymbol{\theta}_t}[R_{t+1}|s_t, a_t = a, \boldsymbol{\theta}_t]\right] + \gamma \max_{a'} \mathbb{E}_{\boldsymbol{\theta}_{t+1}}[Q^{*,\mathcal{W}\,|\,\boldsymbol{\theta}_{t+1}}(s_{t+1}, a')]$
    end for
    return $\operatorname{argmax}_{a'} KG$

N-step Monte Carlo Knowledge Gradient

The KG idea of taking one exploration step and then following the updated greedy policy can easily be generalized into making N exploratory steps instead of one. In order to fully grasp the idea behind this heuristic, let us define a N-KG return, the return we expect by taking N

exploration steps and then forever acting greedily:

$$G_t^{\text{N-KG}} = \underbrace{R_{t+1} + \gamma R_{t+2} + ... + \gamma^{N-1} R_{t+N-1}}_{\text{Exploration steps}} + \gamma^N \underbrace{G_{\pi^*, t+N-1}}_{\text{Greedy steps}} \tag{4.8}$$

In the above formulation, the actions $a_t, a_{t+1}, ..., a_{t+N-2}$ are taken to be exploratory and π^* represents a greedy policy after we have seen the rewards and transitions of $N-1$ previous actions. In order to truly calculate the maximal expected mean of the N-KG return, we would need to consider all possible trajectories of the initial $N-1$ exploratory actions, meaning we would need to perform $|\mathcal{A}|^N$ simulations, which quickly would become unpractical in large action spaces.

Algorithm 7 N-step Monte Carlo Knowledge Gradient

Input: N number of exploratory steps, state s_t, world models $W|\,\boldsymbol{\theta}_t$ and discount factor γ.
Output: action to perform at s_t.
procedure MCKG-N($N, s_t, \mathcal{W}\,|\,\boldsymbol{\theta}_t, \gamma$)
Initialize $KG \in \mathbb{R}^{|\mathcal{A}|}$ to a vector of zeros.
    for a in $\mathcal{A}$ **do**
        $\mathcal{D}_{\text{temp}} \leftarrow \mathcal{D}_t$
        $s_{\text{first}} \leftarrow s_t$, $a_{\text{first}} \leftarrow a$
        for i in $0, 1, ..., N-1$ **do**
            $KG_a = KG_a + \gamma^i\, \mathbb{E}_{\boldsymbol{\theta}_{t+i}} \left[\mathbb{E}_{*, \mathcal{W}\,|\,\boldsymbol{\theta}_{t+i}}[R_{t+1+i} | s_{\text{first}}, a_{\text{first}}, \boldsymbol{\theta}_{t+i}] \right]$
            Sample $r \sim \mathbb{E}_{\boldsymbol{\theta}_{t+i}}[R_{t+i} | s_{\text{first}}, a_{\text{first}}, \boldsymbol{\theta}_t]$ from the predictive reward model.
            Sample $s_{\text{next}} \sim \mathbb{E}_{\boldsymbol{\theta}_t}[s_{t+1+i} | s_{\text{first}}, a_{\text{first}}, \boldsymbol{\theta}_t]$ from the predictive transition model.
            Add to temporary data buffer $\mathcal{D}_{\text{temp}} \leftarrow \mathcal{D}_{\text{temp}} \cup \{(s_{\text{first}}, a_{\text{first}}, r, s_{\text{next}})\}$
            Compute new temporary posterior $\boldsymbol{\theta}_{\text{temp}} = \boldsymbol{\theta}\,|\mathcal{D}_{\text{temp}}$.
            Compute $\mathbb{E}_{\boldsymbol{\theta}_{\text{temp}}}[Q^{*, \mathcal{W}\,|\,\boldsymbol{\theta}_{\text{temp}}}]$ using policy iteration.
            Compute next action

$$a_{\text{next}} = \underset{a'}{\text{argmax}}[KG_{a'} + \gamma^{i+1} \max_{a''}[\mathbb{E}_{\boldsymbol{\theta}_{\text{temp}}}[Q^{*, \mathcal{W}\,|\,\boldsymbol{\theta}_{\text{temp}}}(s_{\text{next}}, a'')]]]$$

            Set current state and action $s_{\text{first}} \leftarrow s_{\text{next}}$, $a_{\text{first}} \leftarrow a_{\text{next}}$
        end for
        $KG_a = KG_a + \gamma^N \max_{a''}[\mathbb{E}_{\boldsymbol{\theta}_{\text{temp}}}[Q^{*, \mathcal{W}\,|\,\boldsymbol{\theta}_{\text{temp}}}(s_{\text{next}}, a'')]]$
    end for
    return $\text{argmax}_{a'}\, KG$

Instead of computing the entire trajectory, we propose computing N-KG return only for each first step actions a_t and letting the agent pick actions that are optimal with respect to the i-KG return at the $t + i^{\text{th}}$ step. The algorithm, which we call N-step Monte Carlo Knowledge Gradient (MCKG-N), for doing just that, is given in Algorithm 7.

Chapter 5

Experimental methods

In this section we describe the experimental methods that we employ to test both the epistemic uncertainty of return estimates from section 3.3 and the action selection algorithms from chapter 4. As a testing-bed, we propose several MDPs which vary in structure and difficulty. We also describe metrics that we use to assess the performance of the agents in these environments.

5.1 Environments

5.1.1 Random MDP

Random MDP is an environment in which both the reward and transition dynamics are chosen randomly at initialization. The MDP consists of $|\mathcal{S}| = N_s$ states and at each state one of $|\mathcal{A}| = N_a$ actions is possible. This environment was first introduced in (Osband and Van Roy (2017)) and represents a world where there is absolutely no structure. This offers a unique opportunity to scrutinize the agents performance without the bias of human priors.

The transition dynamics in the Random MDP are Categorical, with the parameters sampled from a Dirichlet distribution at creation:

$$s_{t+1}|s_t, a_t \sim \text{Cat}(\boldsymbol{\alpha}_{s_t,a_t})$$

where $\boldsymbol{\alpha}_{s_t,a_t} \sim \text{Dirichlet}(\mathbf{1})$. The rewards are Gaussian:

$$R_{t+1}|s_t, a_t \sim \mathcal{N}(\omega_{s_t,a_t}, \varphi^{-1}_{s_t,a_t})$$

with the Gaussian parameters sampled from a Normal Gamma at creation:

$$\omega_{s_t,a_t}, \varphi_{s_t,a_t} \sim NG(0, 0.1, 2, 2)$$

We chose these parameters because they induce a very wide sampling range of MDPs, both with regards to structure and to reward distributions.

5.1.2 Perfect Binary Tree (PBT)

The Perfect Binary Tree (PBT) environment is a modification of both the binary tree MDP from Figure 2.3 (Janz et al. (2018)) and the chain environment proposed in (Osband et al. (2017)). The schematic of the PBT world is given in Figure 5.1.

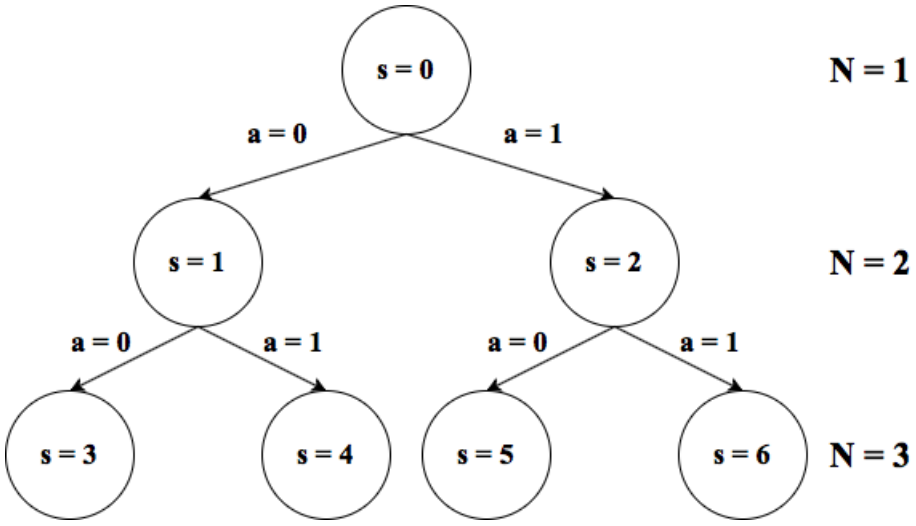


Fig. 5.1 The illustration of the Perfect Binary Tree (PBT) environment, for depth $N = 3$.

We use the tree depth N, to fully describe the MDP. For a tree of depth $N = n$, the number of states is $|\mathcal{S}| = 2^n - 1$ and in each state the agent can either go *right* ($a = 1$) or *left* ($a = 0$), with transitions fully deterministic as shown in Figure 5.1. Additionally, following the example of the chain MDPs in (Osband et al. (2017)), we introduce a mask $\boldsymbol{W} = \text{Bernoulli}(0.5)^{|\mathcal{S}|}$. The mask, sampled at the initialization of PBT, has the effect of shuffling the left-right actions, so that if $W_{s'} = 1$, then in state s', $a = 0$ represents traversing to the 'right' of the tree and $a = 1$ to the 'left' (and vice-versa if $W_{s'} = 0$).

The agent always starts in state $s_0 = 0$ at the top of the tree. Each 'left' action is associated with 0 reward, whereas each 'right' action, except one, carries a reward penalty of $-\frac{9}{N-1}$. The only action that carries a positive reward, is in the right-most state $s = 2^N - 2$, where the

reward for going 'right' is 10. After reaching the 'bottom row' of the tree, so for states s' such that $2^N - 2 - 2^{N-1} < s' \leq 2^N - 2$, any action takes the agent back to the starting $s = 0$.

This environment presents a huge exploratory challenge, as it is easy for the agent to simply learn to pick 'left' at each state, leading him to have 0 reward. Finding the profitable right-most state requires the agent to go 'right' N times, accruing a total reward of $10 - 9 = 1 > 0$ in the process. The optimal policy is hence to go 'right' in all states in the right part of the tree (to get to the profitable state) and 'left' in the left part of the tree (to accrue no negative reward).

5.1.3 Corridor MABs

The Corridor Multi Armed Bandit (Corridor MAB) MDP is a natural generalisation of the MAB environment, to worlds that contain states and transitions. A schematic illustration of a N state Corridor MAB is given in Figure 5.2.

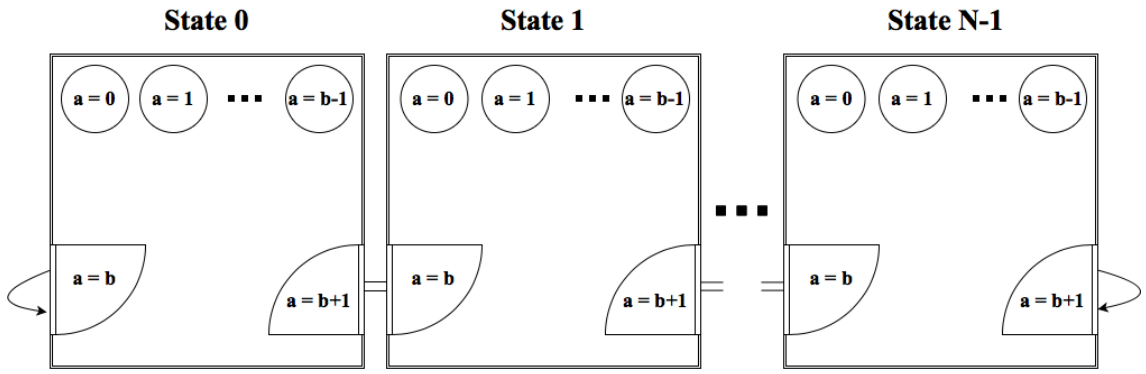


Fig. 5.2 Illustration of the Corridor Multi Armed Bandit (Corridor MAB) MDP, with N states.

Each state represents a 'room' full of 'bandits'. Each room has the same amount of 'bandits', given by b, and they can be activated by playing actions $a \in \{0, ..., b-1\}$. Each bandit returns a reward sampled from a Gaussian, so that:

$$R_{t+1}|s_t, a_t \sim \mathcal{N}(\omega_{s_t,a_t}, \varphi_{s_t,a_t}), \quad \forall a_t \in \{0, ..., b-1\}$$

Actions $a \in \{b, b+1\}$ are associated with traversing the 'corridor', so that $a = b$ in $s = s'$ will transition the agent to state $s' - 1$ or to s' if $s' = 0$. Similarly, playing $a = b+1$ in $s = s'$ transitions the agent to state $s' + 1$ or to s' if $s' = N - 1$.

The agent always starts at the initial state $s = 0$. To make things harder, we also introduce a penalty for moving, so that:

$$R_{t+1}|s_t, a_t = c, \quad \forall \in \{b, b+1\}$$

with $c < 0$.

Next we present two variants of the Corridor MAB, the Light at the End of Tunnel (LET) and Exploration will Pay off (EWP) environments.

Light at the End of Tunnel (LET)

In the LET environment, $b = 4$ and $N = 10$. The 'bandit' distributions in each room are designed so that they get worse and worse as the agent goes down the corridor. In each room, the 'best bandit', so the one with the highest mean reward, is at $a = a_0 = 0$. The 'bandit' distributions in LET are such that:

$$\begin{aligned} \omega_{s_t,a_t} &= -(s_t - 1), \quad \forall s_t < N - 1 \\ \omega_{N-1,a_0} &= 10 \\ \varphi_{s_t,a_0} &= 1, \quad \forall s_t \in \mathcal{S} \end{aligned}$$

meaning that the last room, contains the overall 'best bandit' in the MDP. All the other bandits in a given room are 'worse' than the best bandit, in that their mean reward is always strictly smaller and their variance is allowed to be anything up to 10.

The LET environment represents an interesting and hard exploratory challenge, as achieving optimal behaviour requires the agent to keep traversing the corridor until the end, even though the encountered 'bandits' get worse and worse and cumulative reward suffers from the penalty for moving. Once the agent arrives in the last room, it should stay there forever playing the best bandit at a_0.

Exploration will Pay off (EWP)

In the EWP world, $b = 4$ and $N = 10$. The only difference to LET, is that the 'bandit' distributions are shaped so that there is a bait in the first room $s = 0$. This means that:

$$\begin{aligned} \omega_{0,a_0} &= 5 \\ \omega_{s_t,a_t} &= -(s_t - 1), \quad \forall 0 < s_t < N - 1 \\ \omega_{N-1,a_0} &= 10 \\ \varphi_{s_t,a_0} &= 1, \quad \forall s_t \in \mathcal{S} \end{aligned}$$

where a_0 still represents the 'best bandit' in a given room. Similarly to LET, all the other 'bandits' in a given room must have strictly lower mean rewards than the best one and their variance is allowed to be anything up to 10.

The EWP environment represents a slightly harder exploratory task to the LET, because the agent is tempted to stay in the initial state, where he can get a positive mean reward of 5. However, the best strategy is to ignore the high reward in $s = 0$ and keep traversing the corridor until the last room and play the a_0 bandit forever.

5.2 Performance metrics

We call an algorithm 'successful', if it is able to find the true optimal policy π^* of the MDP and we also report the time (in iterations) it requires to become 'successful'. Finding the optimal π^*, means that the algorithm's greedy policy with respect to its Q-value estimation $\mathbb{E}_{\boldsymbol{\theta}}[Q^{*,\mathcal{W}\,|\,\boldsymbol{\theta}}]$, is identical to the optimal π^*. By keeping track of that, we are making sure that the agent truly understands the world and knows what the optimal thing to do is in every state. The faster the agent becomes 'successful', the better he has done at guiding exploration into areas that uncovered the true dynamics of the environment.

We use the notion of online regret (Blundell et al. (2015)) to test how much of potential reward the agent is losing out on. Online regret is measured as the difference between the cumulative reward of a tested agent and the cumulative reward of a optimally behaving agent, which we call the oracle. The oracle agent simply follows the optimal policy π^* at any step. In detail, online regret of an agent A at time t in MDP $\mathcal{W}$, is given by:

$$\text{Online regret}(t) = \sum_{i=0}^{t} (R_i|\pi^*, \mathcal{W}) - (R_i|A, \mathcal{W})$$

where $R_i|A, \mathcal{W}$ represents the i^{th} reward obtained by agent A in world $\mathcal{W}$. The smaller the final regret of an agent, the better.

Lastly, we track the proportion of time a optimal action was chosen. This simply means:

$$\text{Proportion of best actions}(t) = \sum_{i=0}^{t} \frac{I[a_t = a_t^*]}{t}$$

where $I[\cdot]$ is a indicator function and a_t^* represents the optimal action at time t.

5.3 General notes

Bayesian priors

Choosing a right prior in Bayesian models is crucial and can impact learning efficiency and performance. In our experiments, we chose priors that best describe the lack of knowledge that the agent has about the world at initialization.

In order to account for most scenarios, we choose loose priors, so that before the agent actually gets data from a particular state-action pair, the prior renders a highly variable output. Since most environments that we test the algorithms on have a highest potential one-step reward of 10, we chose the reward priors so that:

$$\begin{aligned}\mathbb{E}_{\boldsymbol{\theta}}\left[\mathbb{E}_{\mathcal{W}}[R_{t+1}|s_t,a_t,\boldsymbol{\theta}]\right] &\approx 0 \pm 10, \quad \forall s_t \in \mathcal{S}, a_t \in \mathcal{A}\\ \text{Var}_{\boldsymbol{\theta}}\left[\mathbb{E}_{\mathcal{W}}[R_{t+1}|s_t,a_t,\boldsymbol{\theta}]\right] &\approx 1 \pm 1, \quad \forall s_t \in \mathcal{S}, a_t \in \mathcal{A}\end{aligned}$$

which induces a prior:

$$\mu_{s_t,a_t}, \tau_{s_t,a_t} \sim NG(m_0, \lambda_0, \alpha_0, \beta_0) = NG(0, 0.02, 2, 2), \quad \forall s_t \in \mathcal{S}, a_t \in \mathcal{A}$$

We choose a very loose prior on the transition dynamics as well:

$$\boldsymbol{a}_{s_t,a_t} = \mathbf{0.01}, \quad \forall s_t \in \mathcal{S}, a_t \in \mathcal{A}$$

where $\mathbf{0.01} \in \mathbb{R}^{|\mathcal{S}|}$. This causes the posterior distribution to be biased a lot more towards the actual counts from the data, with the prior information quickly becoming dominated once the counts become more than 2. This is desirable, as we do not want the agent to have any unnecessary beliefs about the structure of the world, but rather infer it directly from observations.

$Q_{\max}$ term in UBE

In the original paper (O'Donoghue et al. (2017)), the authors derive the UBE for the episodic case and say that the $Q_{\max}$ term is roughly $Q_{\max} \approx HR_{\max}$, where $R_{\max}$ is the maximal reward in the MDP and H is the number of steps the agent does per episode. The analogous estimation for continuing tasks is $Q_{\max} \approx \frac{\gamma}{1-\gamma} R_{\max}$, which we employ.

Discount factor

The discount factor plays a big role in planning, as it determines the length of the horizon the algorithm looks forward to. Since all of our experiments are continuing tasks, we used $\gamma = 0.98$ for all of them, in order to represent agents that look far into the future.

Parameter selection

When testing several action selection algorithms from chapter 5, we need to adjust their parameters (especially the β parameter in UCB and Thompson sampling). We do not fine-tune these parameters by doing a grid search or using BO, but rather check a couple of plausible values and report the performance on the best ones.

Chapter 6

Results and discussion

6.1 Epistemic uncertainty estimates comparison

In this section we present the direct comparison of the epistemic uncertainty estimates introduced in chapter 3. Because the Monte Carlo estimate is the only variance approximator that is guaranteed to converge to the *true* value as we increase the sample size N, we directly compare the UBE, EUDV and EUB estimates to it.

In all of the following experiments, we use small versions of the MDPs introduced in chapter 5 and use random action selection. In order for the comparison to be valid, we make sure that in all of the seed runs, the actions taken and rewards and transitions obtained from the environment are the same at each step for all of the estimates. We use $N = 2000$ posterior samples in order to compute the Monte Carlo estimate, a number we deem is large enough to accurately represent the Q-value posterior variance in the small worlds that we consider.

We compare the variance estimates on the LET(N=3,c=-5,b=1) environment (meaning LET environment with 3 rooms, 1 bandit per each room and move penalty of -5) and the Random MDP($N_s = 3, N_a = 3$), to best capture the difference between the estimates in a both structured and unstructured worlds.

6.1.1 Experiments on LET (N=3,c=-5,b=1)

We begin by comparing the UBE to the Monte Carlo baseline, which we can see on Figure 6.1. The UBE greatly overestimates the epistemic uncertainty in this environment, believing the uncertainty to be around 2.5 times above the Monte Carlo estimate in the opening iterations. The estimate goes down with time, as required of epistemic uncertainty, but does so very slowly. Even after the optimal Q's are found and the Monte Carlo variance shrinks rapidly,

the UBE keeps decreasing at the same, slow rate, rendering it almost 100 times as much as the Monte Carlo in the later training stages.

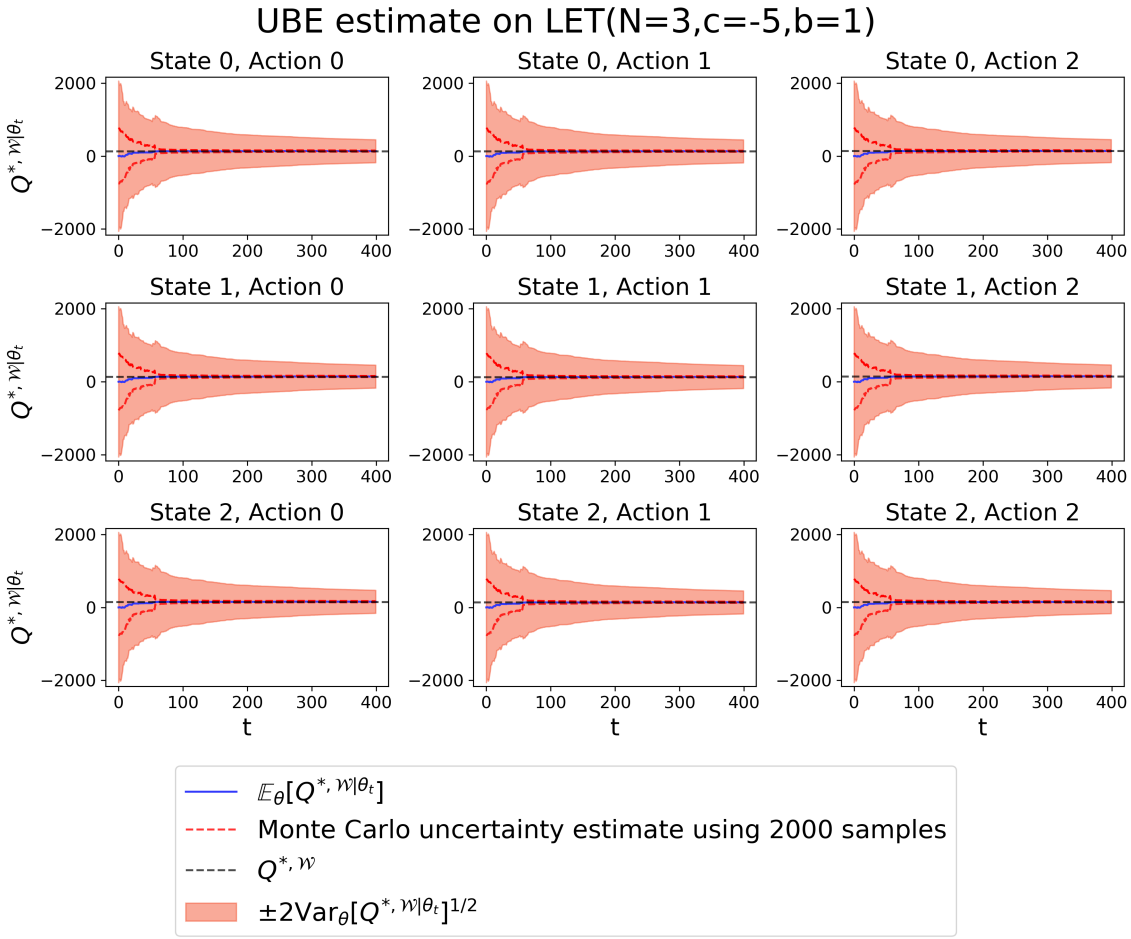


Fig. 6.1 Comparison of the UBE uncertainty estimate to the Monte Carlo (N=2000) baseline on the LET(N=3,c=-5,b=1) environment, averaged for 10 seed runs.

This 'looseness' of the UBE estimate is due to the local uncertainty part of the estimate, which contains the $Q^2_{\max}$ term. We estimate the term as $Q_{\max} \approx \frac{3\times 0.98}{1-0.98} = 147$ (as described in chapter 5), which actually is an accurate estimate of the maximal Q-value in this world. This term squared, has the dominating effect on the UBE estimate and hence its inability to accurately represent the true epistemic uncertainty caused by the parametrization of $\mathcal{W}$. We believe that this effect could be mitigated by tuning the $Q_{\max}$ term, which we will not do here.

Figure 6.2 shows the analogous comparison for the EUDV estimate. In this case, the EUDV underestimates the uncertainty in the opening iterations, then follows the Monte Carlo baseline quite tightly for around 150 iterations and then shrinks much faster to almost 0.

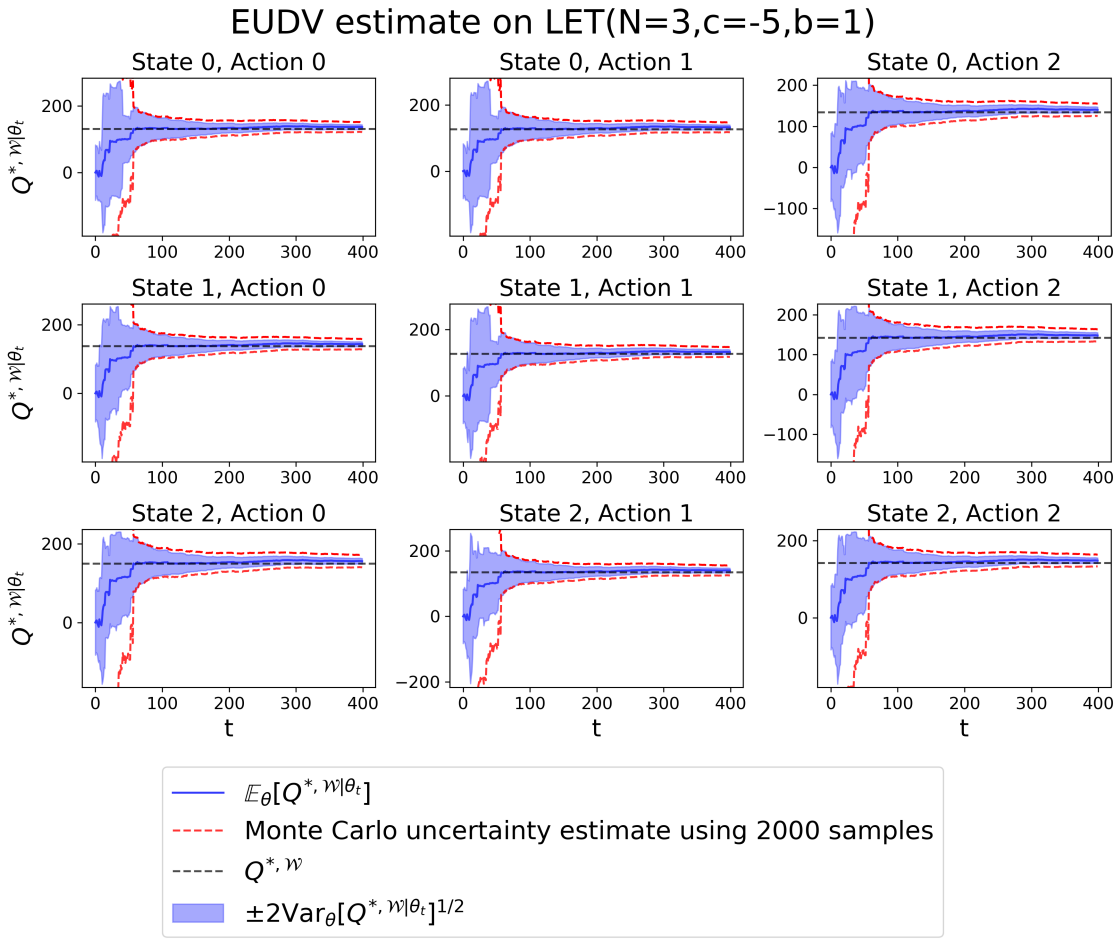


Fig. 6.2 Comparison of the EUDV uncertainty estimate to the Monte Carlo (N=2000) baseline on the LET(N=3,c=-5,b=1) environment, averaged for 10 seed runs.

The EUDV estimate shrinks much less smoothly than the UBE. One interesting behaviour, is that it increases momentarily once the Q-estimate gets closer to the optimal value, which can be explained by the $\mathbb{E}_{\boldsymbol{\theta}}[Q^{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}}]^2$ factor in its recursive definition. A big advantage of the EUDV is that it actually shrinks very rapidly once the algorithm settles around the optimal Q-value, faster than the Monte Carlo baseline, which we will see has a beneficial GLIE effect in action selection algorithms but can also cause hinder learning because of over-confidence.

Figure 6.3 shows that the EUB estimate represents a sort of combination between the UBE and EUDV. It does a better job at estimating the initial variance before the optimal Q's are found, overestimating the Monte Carlo baseline by a slight margin. It is also much more smooth while shrinking, making it resemble the UBE estimate. Unfortunately, similarly to the UBE, it greatly overestimates the uncertainty in the later stages of training, which can be explained by the additional always-positive correlation factor that the EUB includes.

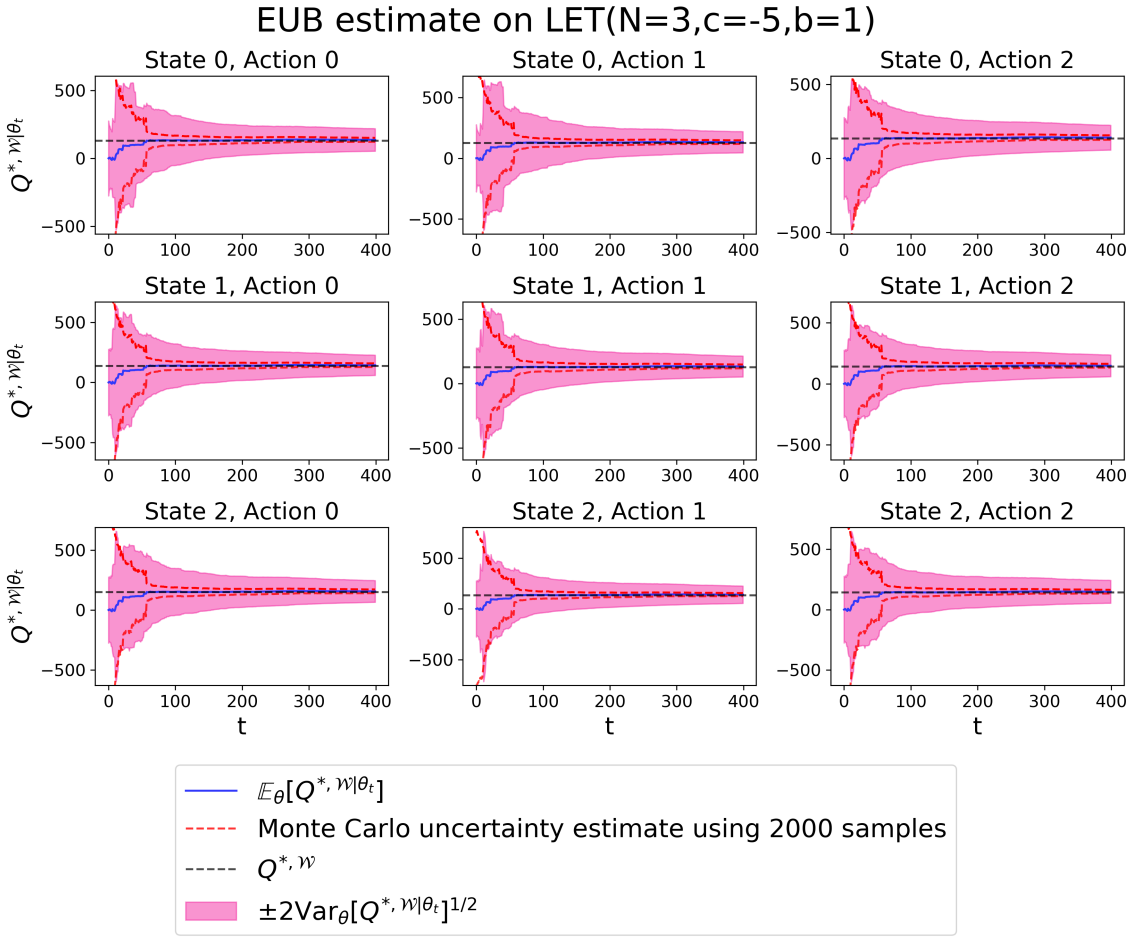


Fig. 6.3 Comparison of the EUB uncertainty estimate to the Monte Carlo (N=2000) baseline on the LET(N=3,c=-5,b=1) environment, averaged for 10 seed runs.

The effect of the correlation factor is very visible on Figure 6.4, where we plot the EUDV and EUB estimates next to each other. Since the EUB is essentially the EUDV with the added correlation factor, the difference between them represents exactly its magnitude. We can see that although the EUDV shrinks to almost 0, the correlation coefficient stays roughly constant, which can be explained by the fact that once the training stabilizes, the $\mathbb{E}_{\boldsymbol{\theta}}[Q^{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}}]$ factor, which is multiplied by the state transition covariance, stays roughly the same. The same squared factor in the EUDV is multiplied by the epistemic uncertainty of the transition dynamics, and so has less and less effect as the training progresses. Even though the state transition covariance also shrinks (as it is marginalized over parameters $\boldsymbol{\theta}$), it does so at a lot slower rate.

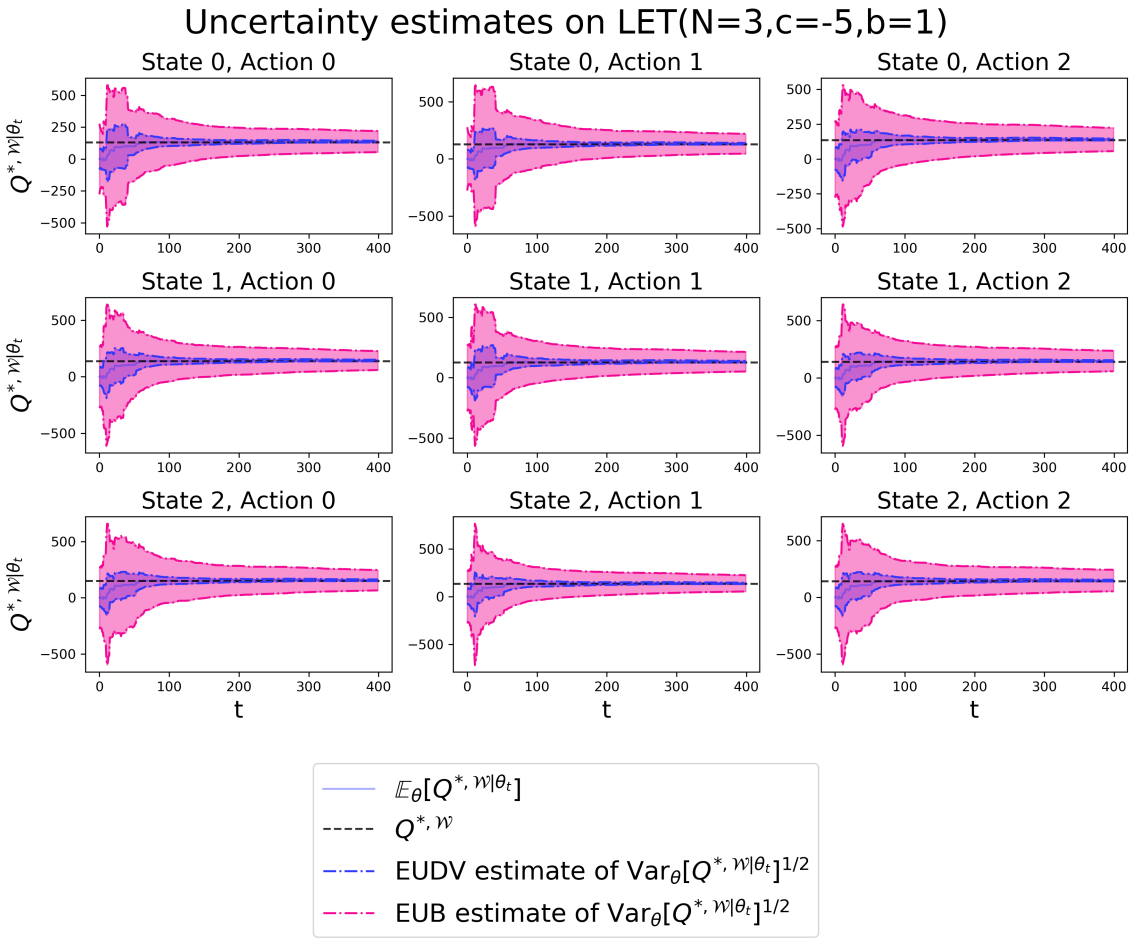


Fig. 6.4 Comparison of the EUDV and EUB uncertainty estimates on the LET(N=3,c=-5,b=1) environment, averaged for 10 seed runs.

6.1.2 Experiments on Random MDP$(N_s = 3, N_a = 3)$

Comparing the uncertainty estimates on the Random MDP environment yields very similar conclusions to above. The only difference that we observed, is that the EUDV estimate shrinks a lot slower, overestimating the Monte Carlo baseline for later iterations. We believe that this is because the Random MDP has stochastic transitions (compared to LETs deterministic) and consequently reducing the epistemic uncertainty of the transition dynamics takes many more iterations. Figures supporting this analysis can be found in Appendix B.1.

6.2 Action selection performance comparison

6.2.1 Results on Perfect Binary Tree (N=4)

In this section we will present performance results that the action selection algorithms introduced in chapter 4 achieve on the PBT(N=4) environment. We test all of the algorithms on all 3 epistemic uncertainty estimates (UBE, EUDV and EUB) and give a rough idea of the hyperparameters that work with each.

Upper Confidence Bound (UCB)

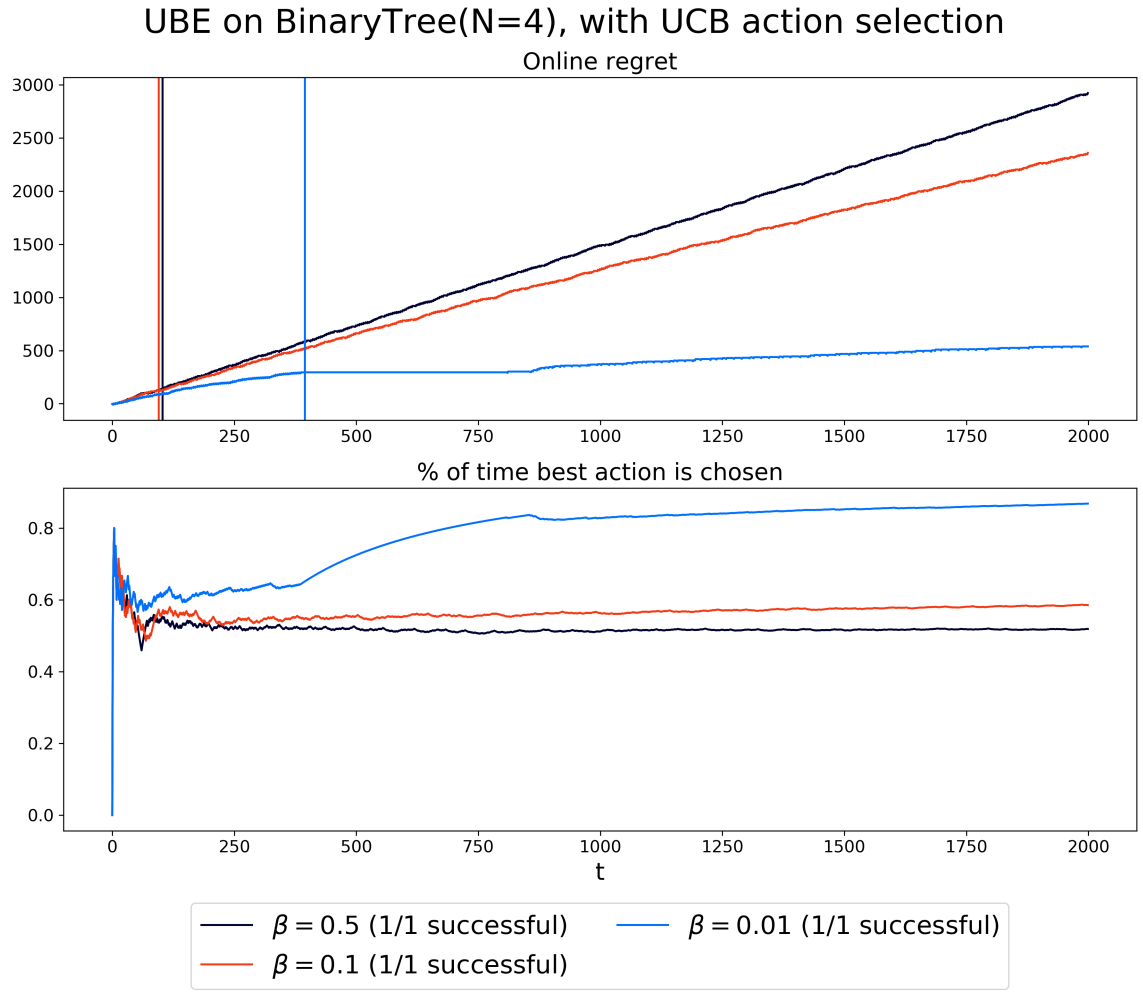


Fig. 6.5 UCB action selection performance on the PBT(N=4) environment, with UBE as variance estimate.

Figure 6.5 shows the regret and proportion of best action chosen for the UBE estimate, for a couple of β values. The β value has a radical impact on the performance of the algorithm, with the regret for $\beta = 0.01$ quickly plateauing at around 250 and regrets for other β's constantly rising due to over-exploration. We perform just 1 seed run for all the UCB experiments, since both UCB and PBT are deterministic.

The vertical lines represent the average time the algorithm takes to become 'successful' (see section 5.2). It is unsurprising that UBE requires very small values for β for the regret to plateau in reasonable time, as we saw in section 6.1 that it massively overestimates the uncertainty.

We can also see the adverse effect of the UBE shrinking very slowly on Figure 6.6, where action 1 frequency (corresponding to traversing 'right' in the tree) stops growing around iteration 850. This is coupled with the regret increasing, suggesting that after finding the optmial strategy, the UBE uncertainty of suboptimal actions is big enough to lure the agent into going back to exploration.

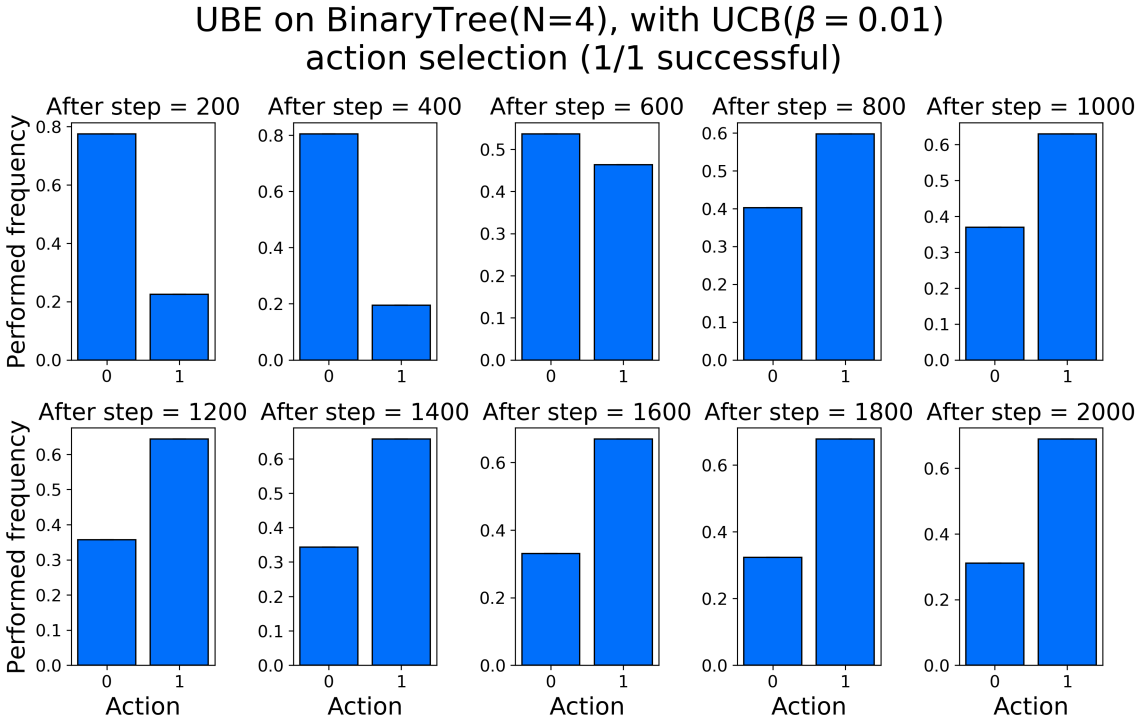


Fig. 6.6 Action frequencies for UCB($\beta = 0.01$) on the PBT(N=4) environment, with UBE as variance estimate.

Figure 6.7 shows the regret for the EUDV estimate, for a couple of values of β. Once again, we see that the β value needs to be heavily tuned in order for the algorithm to be 'successful' and have a reasonable regret. The algorithm fails to find the optimal strategy

for $\beta = 1$ and $\beta = 5$ and instead keeps playing 'left' all the time, accruing no reward at all (this still makes their regret smaller than the 'successful' version $\beta = 10$ because of the reward accruing at a rate of 1 per 4 time-steps for the oracle agent). A vertical line at the last time-step of the plot, always indicates that the algorithm has not been 'successful'.

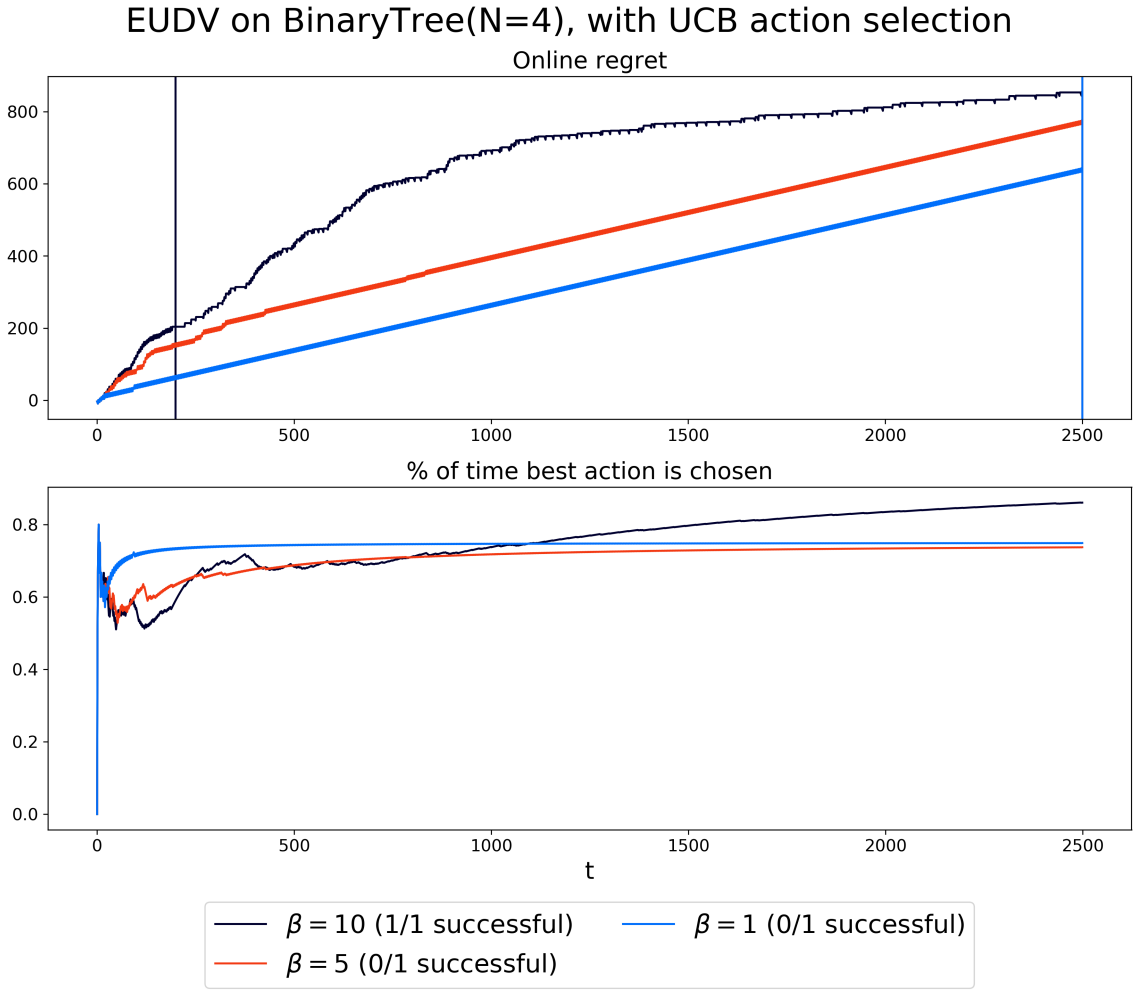


Fig. 6.7 UCB action selection on the PBT(N=4) environment, with EUDV as variance estimate.

Figure 6.8 closely highlights where the EUDV can fail with the UCB action selection. Prior to reaching state 14 (so the right-most state in the tree) the agent has not yet seen a single positive reward and since playing 'right' always gives a negative reward, its estimation for the Q-values of going 'right' will be significantly lower from 'left' actions. Moreover, since the EUDV shrinks fast, the uncertainty for going 'right' will also become smaller provided that the agent played 'right' a couple of times, which makes it even harder for it to keep playing 'right' in the previously visited states (which is required to reach the profitable state 14). We believe that this effect is magnified by the fact that the dynamics priors are very loose

(meaning that reaching state 14 from any visited state will quickly carry almost 0 probability as the counts of actually witnessed transitions greatly dominate the prior). This failure is mitigated in this example by using a high β value, which artificially increases the uncertainty inviting the agent to keep trying to play 'right' for a little while longer, allowing it to reach state 14.

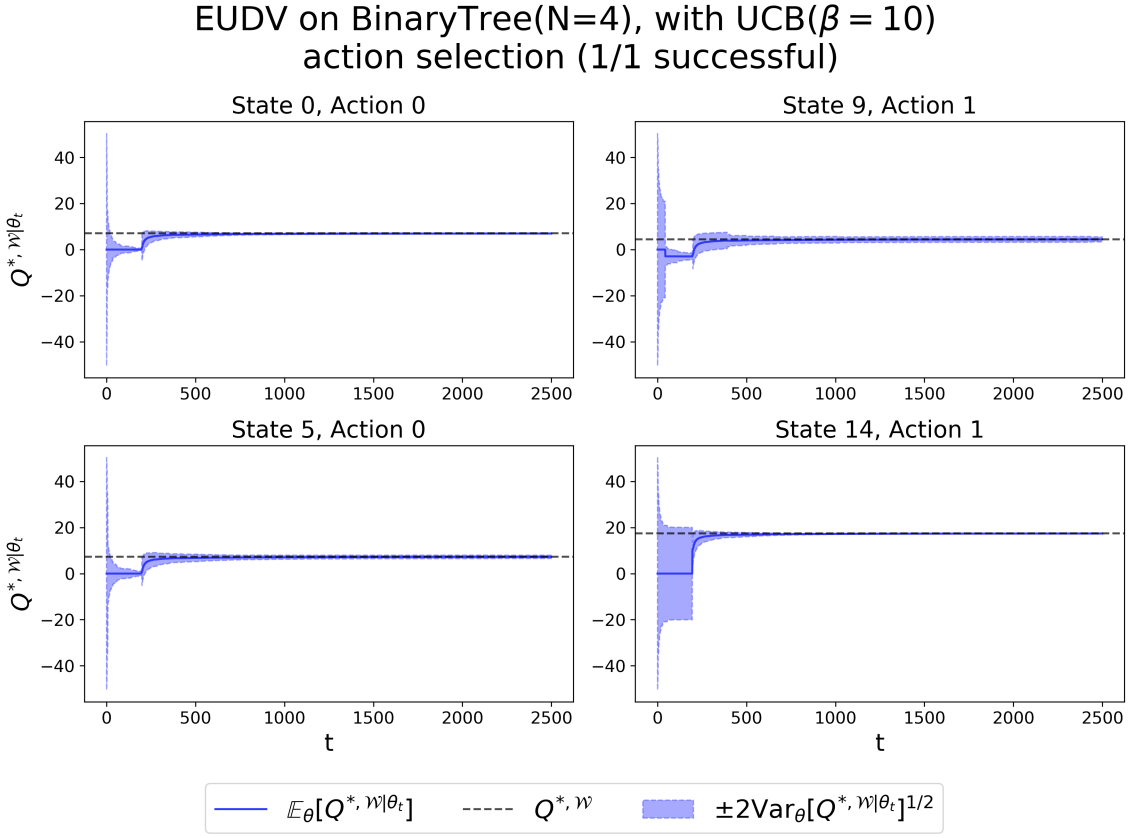


Fig. 6.8 Several Q-values and their variance, for UCB action selection on the PBT(N=4) environment, with EUDV as variance estimate.

As we saw in the previous section, EUB represents a combination of the features of UBE and EUDV. It is hence unsurprising that it performs quite well with UCB action selection. The regrets for a couple of β parameters with the EUB estimate are shown in Figure 6.9. The EUB seems to share the catastrophic behaviour with the EUDV, because of the algorithm with $\beta = 1$ not being able to learn. It is however clear, that the extra correlation term mitigates this effect and hence the algorithm is able to learn the optimal policy with $\beta = 5$ and higher.

Concluding, UCB action selection is a powerful method, which is able to find optimal strategies and plateau the regret with all considered uncertainty estimates. However, its

performance is strictly linked to tuning the β hyperparameter, which can prove much harder in more complex environments.

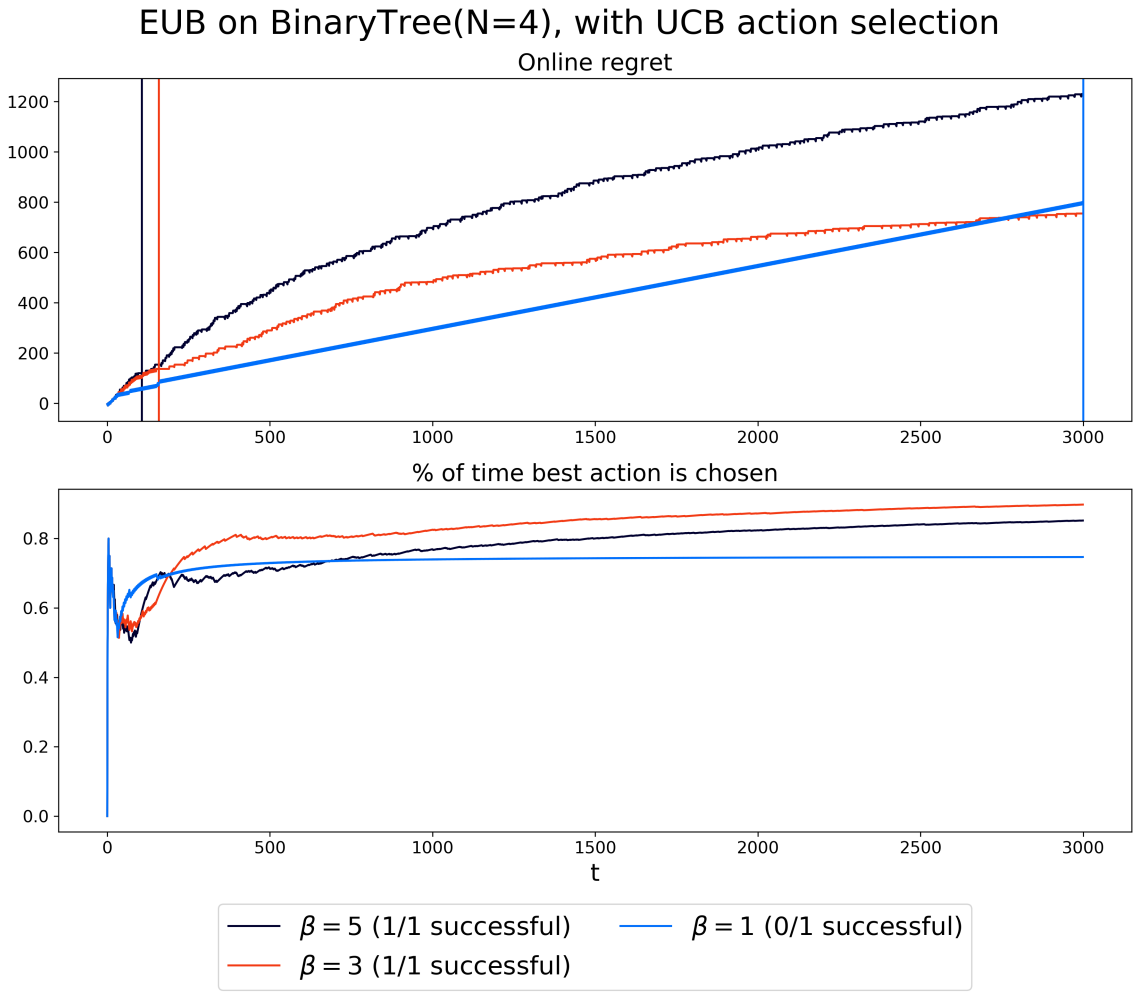
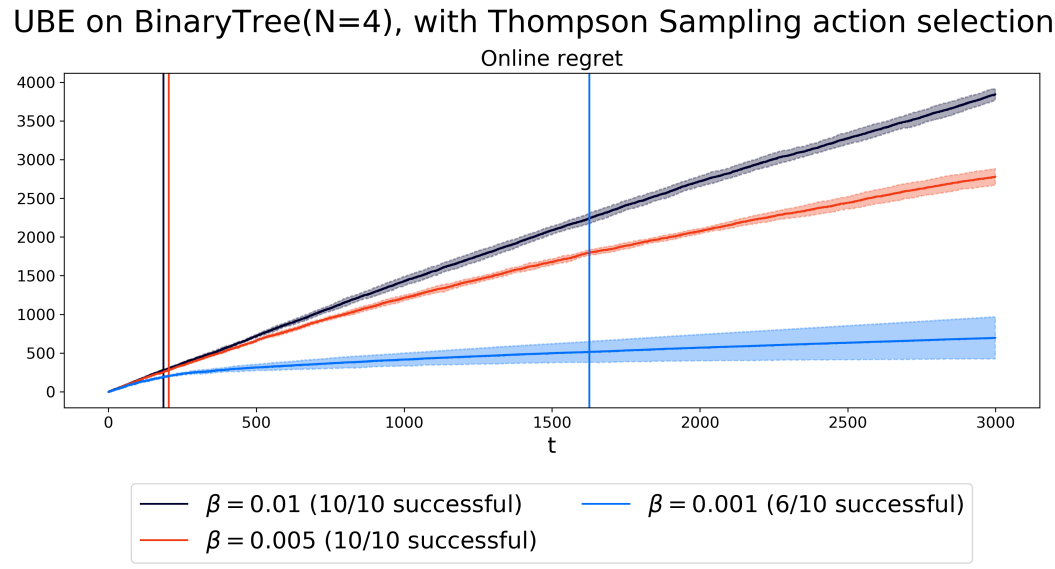
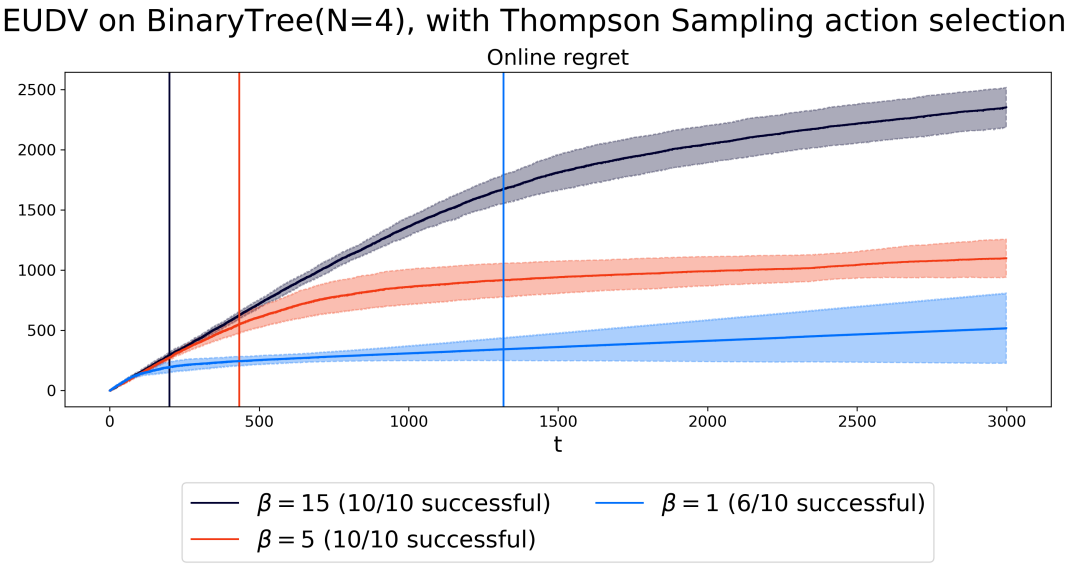


Fig. 6.9 UCB action selection on the PBT(N=4) environment, with EUB as variance estimate.

Thompson sampling

Thompson sampling also suffers from the problem of having to tune the hyperparameter β. Similarly to the UCB, it is not informative to directly compare the best performance of UBE, EUDV and EUB with Thompson sampling action selection, because it greatly depends on the choice of β. In theory, we could make all three uncertainty estimates achieve identical regrets by specifically fine tuning β for each, which we will not do here. Instead, we give a more qualitative analysis, showing how difficult for a human experimenter it is to pick a good value for β. Figure 6.10 shows the regrets achieved by Thompson sampling action selection, for a couple of β parameters, for the UBE, EUDV and EUB variance estimates respectively.





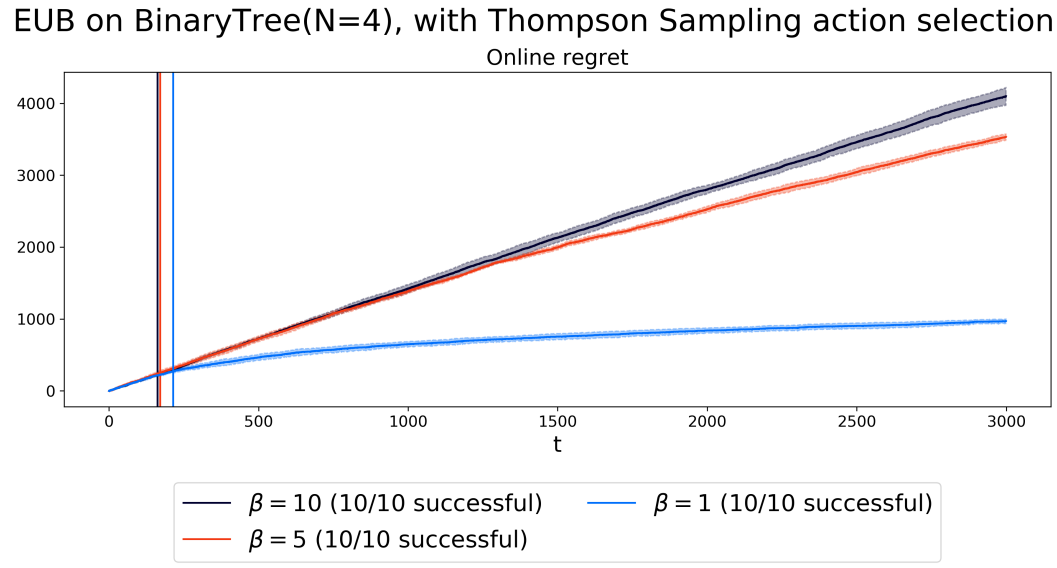


Fig. 6.10 Thompson sampling action selection performance on the PBT(N=4) environment, with UBE, EUDV and EUB as variance estimates.

We conclude that Thompson sampling is a far more stable action selection process than UCB, as the regrets plateau for a far wider range of the β parameter. Because Thompson sampling is inherently based on randomness rather than strict rules like UCB, it can pick actions that have theoretically less potential (in terms of the sum of Q-value and its epistemic uncertainty). This is why the EUDV behaves much more stable for Thompson sampling, since its underestimated variance can still have an effect if the Q-values are close together as we are sampling the optimal values from Gaussians.

In contrast, picking the right β to suit the UBE estimate is a lot harder. Since the UBE overestimates the uncertainty, β needs to be sufficiently small to reduce its effect in order to prevent the algorithm from over-exploring (as on Figure 6.10 with $\beta = 0.01$), but choosing a too small β can cause under-exploration (like with $\beta = 0.001$, where 4 seeds are not 'successful').

Fig. 6.11 PSRL action selection performance on the PBT(N=4) environment.

PSRL

Figure 6.11 shows the average performance of the PSRL algorithm for 10 seed runs. The algorithm achieves very good mean regret, which starts plateauing only after the 300th step. Only 1 of 10 runs ends up 'unsuccessful', showing that PSRL does not just find the optimal strategy and exploits it, but rather fully explores the environment (to be 'successful' the agent needs to know what the best action is in every state).

Generalized Knowledge Gradient (GKG)

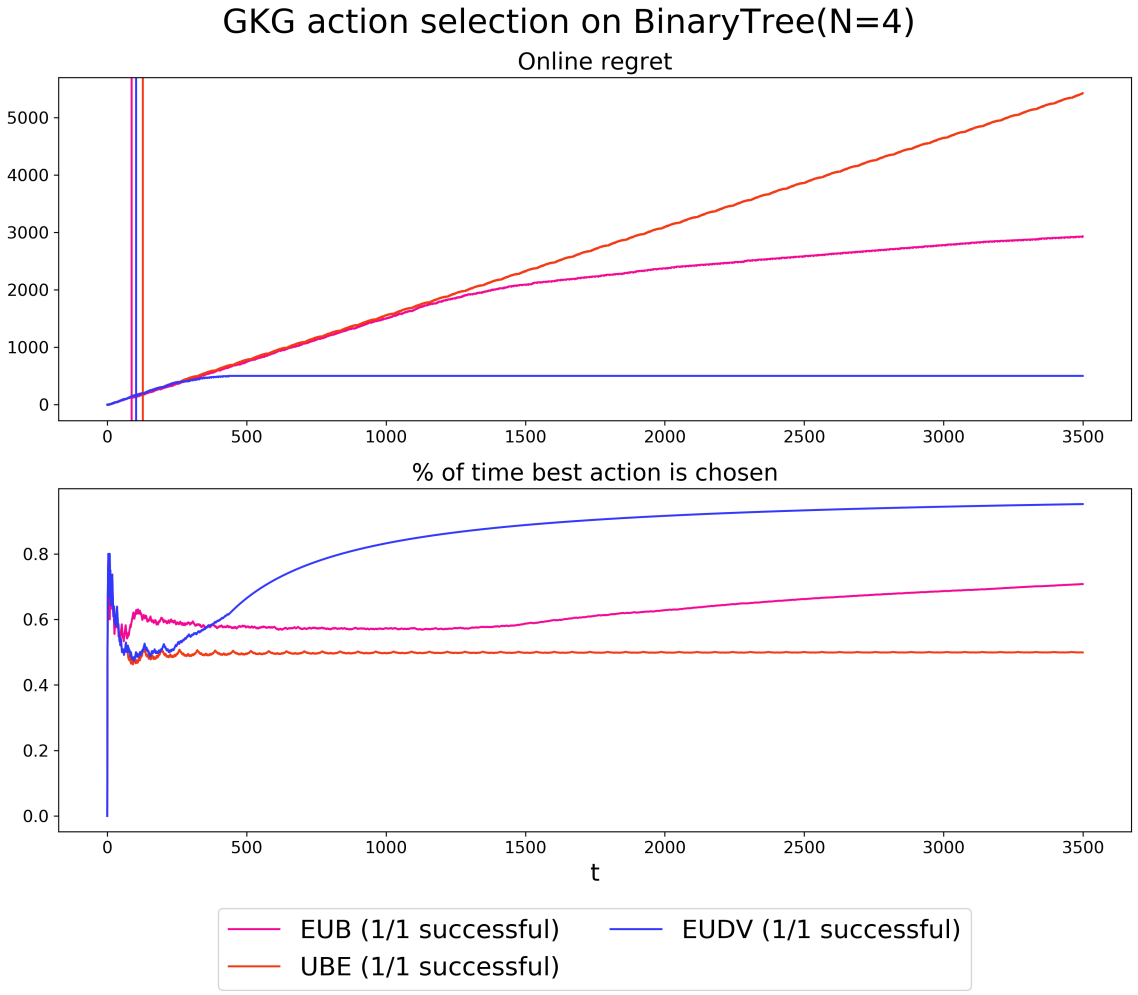


Fig. 6.12 Generalized Knowledge Gradient (GKG) action selection performance on the PBT(N=4) environment, with UBE, EUDV and EUB as variance estimates.

Since similarly to UCB, the GKG policy does not involve sampling, it is sufficient to consider just 1 seed run to discover its performance on PBT. The immediate advantage of GKG is that it does not require hyperparameter tuning, as it simply relies on the reduction in uncertainty.

Figure 6.12 shows the regrets achieved by GKG using all three epistemic uncertainty estimates. GKG is 'successful' with all of the estimates. The EUDV estimate achieves the best final regret and after only 400 iterations the algorithm starts being fully greedy, with the regret line having no slope at all (meaning that the agent always behaves optimally). Being the second most variance-overestimating estimate, EUB achieves the second best performance in terms of regret, which starts plateauing at around iteration 1200. UBE's performance is poor and we do not see the regret leveling-off in the first 3500 steps.

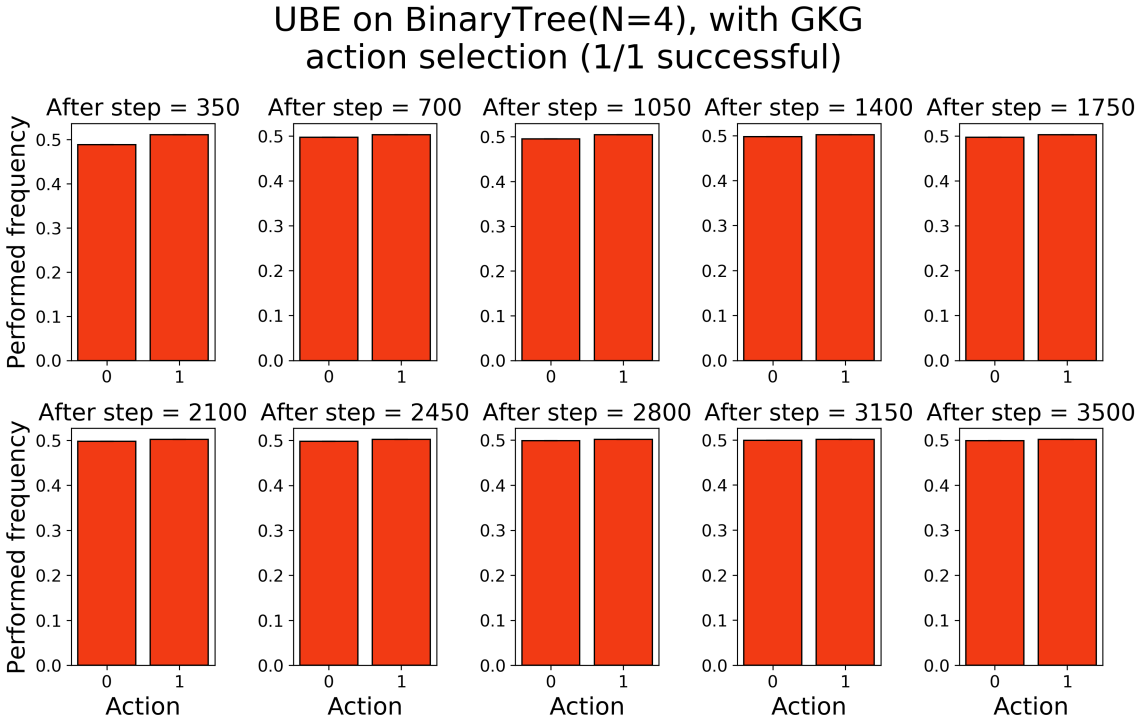


Fig. 6.13 Action frequencies for GKG on the PBT(N=4) environment, with UBE as variance estimate.

The performance of the three estimates does not come as a surprise, if we consider the findings of section 6.1. Since UBE significantly overestimates the epistemic uncertainty, the reduction in the uncertainty stays significant for a very long time, since any action that the agent takes will cause the uncertainty to reduce by a big margin. In effect, the agent plays essentially random actions throughout the episode, which is confirmed by Figure 6.13. The aforementioned effect is magnified by the fact that the UBE is generally slow at shrinking.

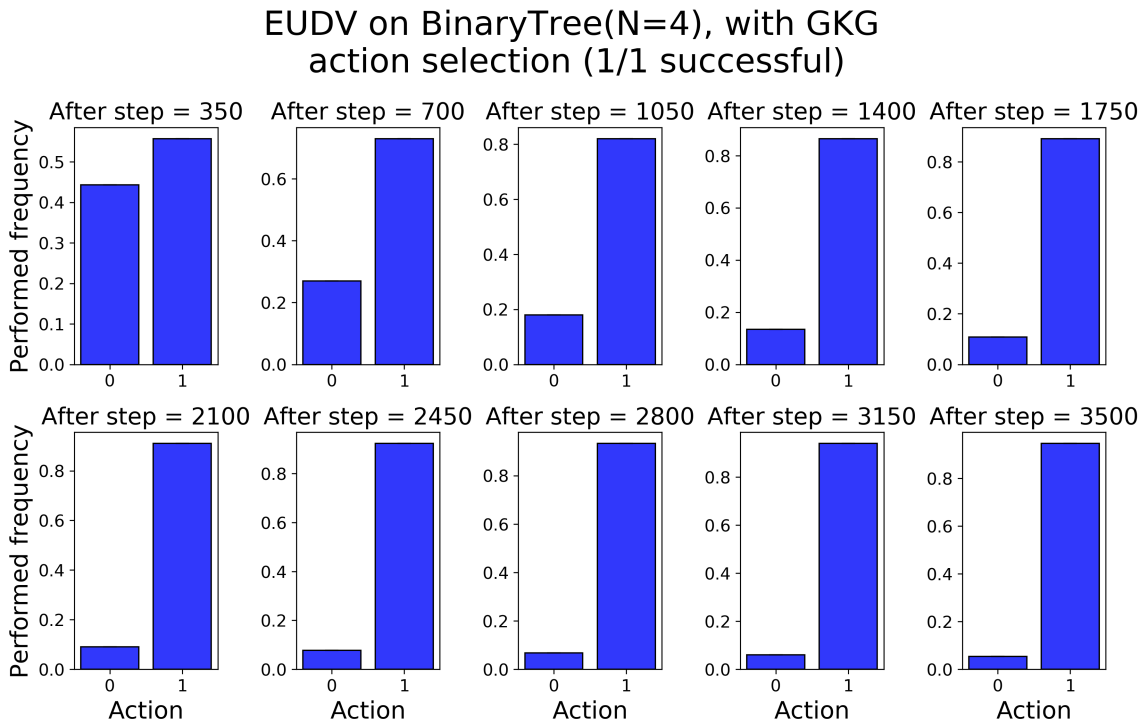


Fig. 6.14 Action frequencies for GKG on the PBT(N=4) environment, with EUDV as variance estimate.

In contrast, EUDV and EUB seem to be perfectly suited for the GKG action selection. Since the learning is driven by trying to reduce the uncertainty, the EUDV does not suffer a similar problem as in UCB, where the uncertainty of 'right' actions got so small that it discouraged ever going 'right'. Instead, the GKG does not care about the outright uncertainty, but rather the reduction in it, causing the 'right' actions to have comparative value as the 'left' ones, before the uncertainties of both become sufficiently small. The blue vertical line arrives around 300 time-steps before the regret plateaus, which means that the algorithm with EUDV takes 300 steps of exploratory uncertainty reduction after it found the optimal strategy, before it realises that going 'right' all the time is the best policy. For the EUB, that number of steps is around 1400.

N-step Monte Carlo Knowledge Gradient (MCKG-N)

Figure 6.15 shows that, unfortunately, the MCKG-N action selection method is unable to learn the optimal strategy in the PBT environment. For all tested values of N, the algorithm quickly settles on the sub-optimal strategy of traversing 'left' all the time, hence the stable slope of the regret line (as it accrues a stable -1 regret per 4 time-steps).

MCKG-N achieves the lowest mean regret for $N = 2$, but it feels useless to compare the performance of different N's, since none actually find the optimal policy. The reason why MCKG fails in this particular example, are similar to why EUDV failed with UCB for not big enough β. If the agent is not lucky to play 4 consecutive 'right' actions, it gets progressively harder for it to ever pick 'right', since its sampled next-step rewards will come from a tighter distribution and hence it will be less likely to draw a 'surprising' high reward.

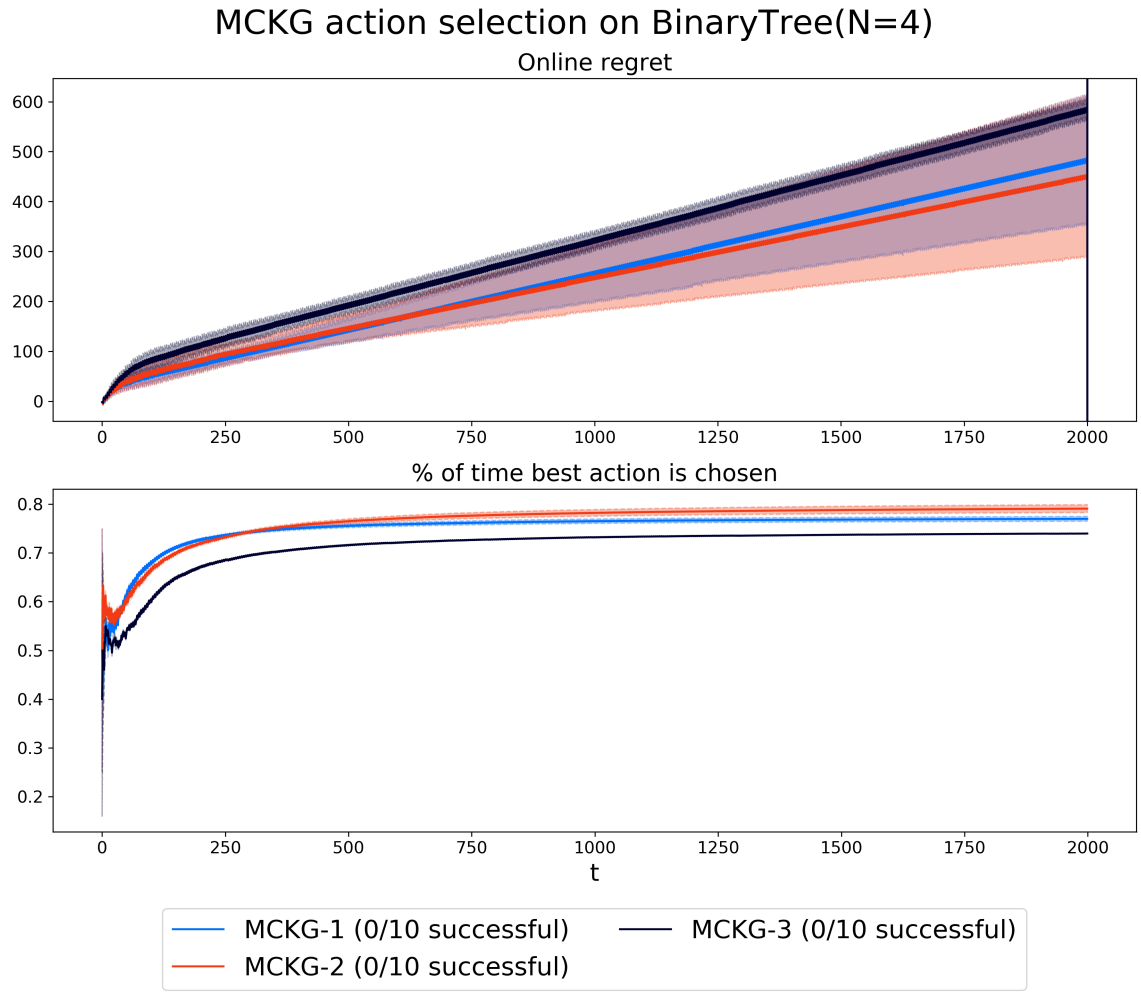


Fig. 6.15 Action frequencies for MCKG-N on the PBT(N=4) environment.

The performance of MCKG-N can be dramatically improved by adhering to the OFU principle, which in the Bayesian case means setting the reward priors to have high means. Figure 6.16 shows that the MCKG-N regret actually plateaus quite quickly, if we set the reward priors to $NG(10, 0.02, 2, 2)$, meaning that the prior mean reward becomes 10 ± 10. Changing the inherent 'optimism' of the algorithm allows it to overcome the aforementioned problem, because it quickly becomes important for the agent to visit states it previously has

not reached (like state 14) and hence it quickly tries out all the possible action combinations to achieve that.

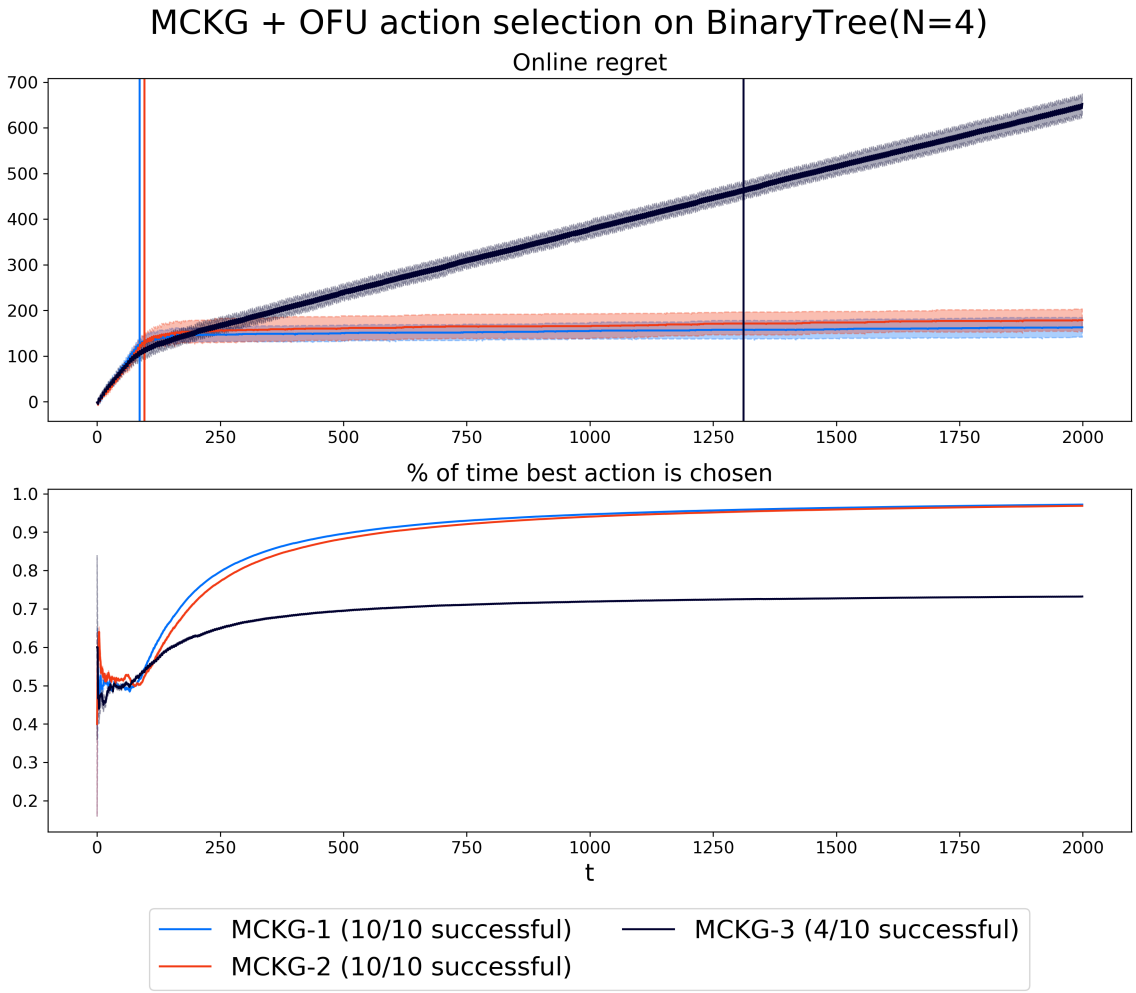


Fig. 6.16 Action frequencies for MCKG-N on the PBT(N=4) environment, with priors adhering to the OFU principle (see text).

The MCKG-1 and MLKG-2 algorithms very quickly learn the optimal policy and then once found, continue to exploit that knowledge, rendering the regrets flat. The MCKG-3 algorithm succeeds in only 4/10 runs, suggesting that increasing the number of exploratory steps N beyond 2 increases the performance instability.

Table 6.1 shows the summary of the results on PBT($N = 4$). For UCB and Thompson sampling, the results are given for β's which achieved the smallest regret at the same time being 'successful' more than 50% of time. Note that in the mean 'successful' time column, if a run is not 'successful', it is given a time of 3000 (maximal iteration time in all experiments). Table 6.1 informs us that the MCKG-1 algorithm with optimistic priors achieves the best

overall performance, leading PSRL and GKG with EUDV by a landslide. The algorithm also explores the world enough to be 'successful' at every single one of 10 seed runs. The fact that PSRL has been beaten in all metrics by a different sampling method is reassuring, considering that the authors of PSRL praised it as the most efficient method for tabular environments to date (Osband et al. (2013)).

Action selection method	**Uncertainty estimate**	**Mean regret after 3000 steps**	**Mean % of best action after 3000 steps**	**% of 'successful' runs**	**Mean 'successful' time**
UCB	UBE	659 ± 0	87.9 ± 0.0	100	395
	EUDV	865 ± 0	86.2 ± 0.0	100	199
	EUB	755 ± 0	87.1 ± 0.0	100	159
Thompson sampling	UBE	858 ± 238	78.1 ± 0.2	50	1943
	EUDV	495 ± 297	88.6 ± 0.2	60	1316
	EUB	947 ± 31	76.9 ± 0.1	80	716
GKG	UBE	4648 ± 0	52.3 ± 0.0	100	128
	EUDV	499 ± 0	89.1 ± 0.0	100	104
	EUB	2778 ± 0	74.1 ± 0.0	100	88
MCKG-1	-	780 ± 9	74.1 ± 0.2	0	3000
MCKG-2	-	742 ± 129	76.8 ± 0.1	10	2802
MCKG-3	-	848 ± 28	70.0 ± 0.9	0	3000
MCKG-1 + OFU	-	**171 ± 36**	**98.1 ± 0.8**	100	**86**
MCKG-2 + OFU	-	179 ± 21	97.8 ± 1.2	100	96
MCKG-3 + OFU	-	1001 ± 244	74.2 ± 1.1	40	1181
PSRL	-	252 ± 33	96.4 ± 1.0	90	222

Table 6.1 Summary of the final performance of algorithms in the PBT($N = 4$) environment.

6.2.2 Results on LET(N=10,c=-5)

In this section we give the performance results on the LET($N = 10, c = -5$) environment. Similarly to the previous table, UCB and Thompson sampling results are given for β's achieving the smallest regret and being 'successful' more than 50% of the time.

Action selection method	**Uncertainty estimate**	**Mean regret after 3000 steps**	**Mean % of best action after 3000 steps**	**% of 'successful' runs**	**Mean 'successful' time**
UCB	UBE	872 ± 48	96.1 ± 0.1	100	**64**
	EUDV	**859 ± 37**	**96.3 ± 0.2**	100	65
	EUB	908 ± 54	95.2 ± 0.0	100	**64**
Thompson sampling	UBE	38573 ± 1938	24.1 ± 1.9	100	496
	EUDV	11110 ± 9240	63.1 ± 5.1	50	1586
	EUB	33967 ± 843	28.4 ± 0.6	100	419
GKG	UBE	17636 ± 89	32.1 ± 0.1	100	73
	EUDV	2406 ± 67	93.8 ± 0.2	100	67
	EUB	16194 ± 86	41.4 ± 0.0	100	72
MCKG-1	-	7765 ± 12620	81.1 ± 3.2	70	1043
MCKG-2	-	13793 ± 15715	64.1 ± 2.1	50	1534
MCKG-3	-	2993 ± 4561	90.4 ± 4.9	50	1684
MCKG-1 + OFU	-	1009 ± 113	95.0 ± 0.9	90	376
MCKG-2 + OFU	-	982 ± 308	95.1 ± 0.3	80	660
MCKG-3 + OFU	-	1413 ± 221	94.3 ± 0.6	80	670
PSRL	-	3616 ± 1599	91.4 ± 1.0	60	222

Table 6.2 Summary of the final performance of algorithms in the LET(N=10,c=-5) environment.

Table 6.2 shows that UCB with the EUDV estimate achieved the best performance in terms of regret, with all runs 'successful'. Such a good performance of the UCB inevitably means that in our experiments we were quite lucky in picking the right β values. Thompson sampling, on the other hand, achieves appalling performance, suggesting that the β values were chosen particularly badly, making the agents over-explore.

Once again, GKG action selection provides very stable learning, with all uncertainty estimates being 100% 'successful'. The perfromance on regret is once again best for the EUDV estimate, achieving a competitive regret of only 3 times more than the best performing UCB method.

Out of methods based on posterior sampling, the best performing ones are the 1 and 2 step MCKG adhering to the OFU principle. MCKG-2 with OFU achieves a good regret of just 100 more than the best method but does so with being 80% 'successful', suggesting that it often storms through the rooms without actually exploring them and being content about the last room's rewards once reached. It is surprising to see PSRL struggle so much with the simple LET environment, achieving very variable regret and a low 'success' rate. We believe that the performance of PSRL could also be improved by making the priors adhere to the OFU principle.

6.2.3 Results on EWP(N=10,c=-5)

Table 6.3 shows the performance achieved on the EWP(N=10,c=-5) environment, with again the β values for UCB and Thompson sampling chosen as in previous experiments. We can immediately see that the introduction of the bait in the first room makes all of the agents perform worse than in the LET environment. As in the LET environment, the UCB action selection with EUDV achieves the best performance. We can see that the mean 'successful' time for the UCB methods is almost twice as high as in the LET environment, suggesting that the agents stay in the first room with the bait for a significant amount of time before they are encouraged to venture to other rooms. Again, Thompson sampling achieves terrible performance, which can be attributed to the β parameters not being sufficiently tuned.

Once again, the GKG method is reliable in finding the optimal strategy for every single uncertainty estimate. The regret follows the same ordering as for the previously discussed environments, with the EUDV delivering the best performance, followed by EUB and UBE. The regrets for GKG are much higher than in LET, suggesting that GKG exploits the initial room for a much longer time, before exploring other states and finding the optimal one. This is confirmed by the mean 'successful' times being much higher (around 4x) than in the LET environment and in Figure 6.17, where we see that the agent stays in the first room for a long time before starting to venture around the corridor.

Action selection method	**Uncertainty estimate**	**Mean regret after 3000 steps**	**Mean % of best action after 3000 steps**	**% of 'successful' runs**	**Mean 'successful' time**
UCB	UBE	7845 ± 69	64.1 ± 0.1	100	**92**
	EUDV	$\mathbf{1912 \pm 65}$	$\mathbf{92.1 \pm 0.0}$	100	112
	EUB	6693 ± 55	43.6 ± 0.0	100	97
Thompson sampling	UBE	24854 ± 2953	22.7 ± 0.6	90	836
	EUDV	10466 ± 1392	52.3 ± 1.9	80	838
	EUB	34266 ± 1280	20.1 ± 0.3	100	467
GKG	UBE	30328 ± 266	21.3 ± 0.0	100	304
	EUDV	17427 ± 74	56.2 ± 0.0	100	360
	EUB	27315 ± 266	22.5 ± 0.0	100	198
MCKG-1	-	15007 ± 68	10.9 ± 0.4	0	3000
MCKG-2	-	14993 ± 62	11.0 ± 0.3	0	3000
MCKG-3	-	13832 ± 4446	11.4 ± 4.2	0	3000
MCKG-1 + OFU	-	3847 ± 5759	83.1 ± 0.1	10	2714
MCKG-2 + OFU	-	3868 ± 5770	83.1 ± 0.1	20	2416
MCKG-3 + OFU	-	7146 ± 6932	67.2 ± 0.3	30	2135
PSRL	-	6659 ± 6862	86.3 ± 5.2	10	2714

Table 6.3 Summary of the final performance of algorithms in the EWP(N=10,c=-5) environment.

It is quite surprising to see all of the sampling methods struggle so much. High proportions of choosing the best action in MCKG-1 and MCKG-2, suggests that the algorithms find themselves in the right room at last, but low proportions of 'successful' seed runs suggests that they under-explore on the way and fail to identify which action is actually the best in each room. Nevertheless, the MCKG-2 outperforms PSRL in terms of regret and proportion

of 'successful' seed runs, which once again shows that it is a superior among action selection methods based on sampling from the dynamics posteriors.

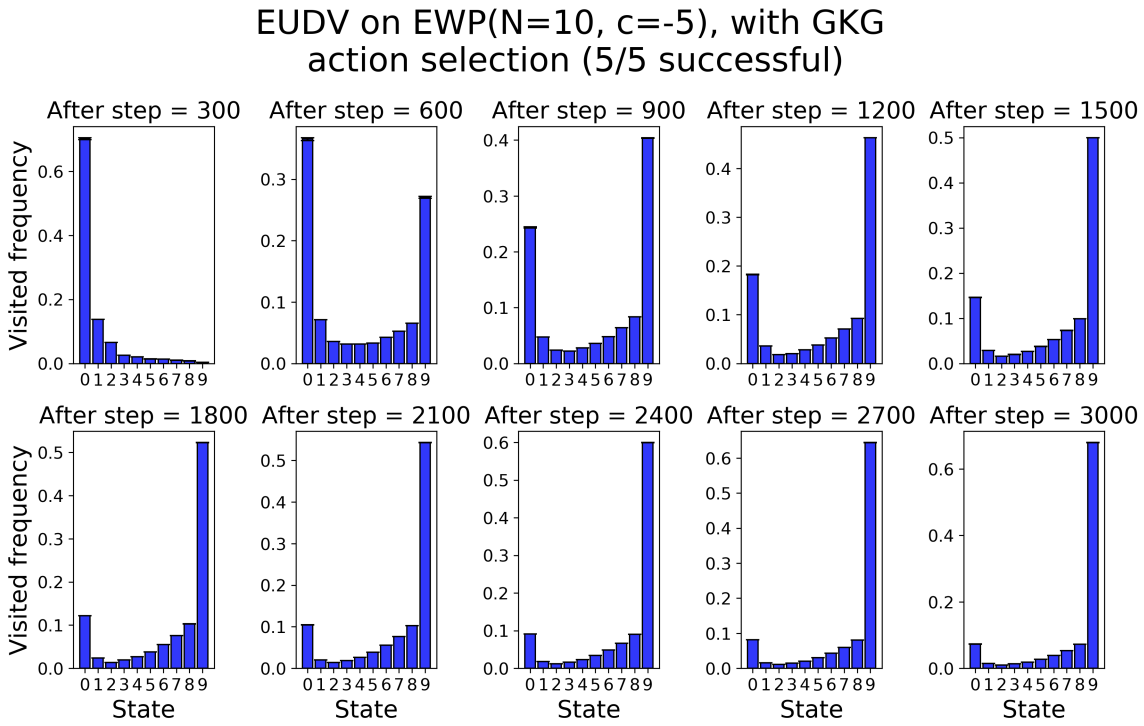


Fig. 6.17 State visit frequencies for GKG with EUDV estimate, on the LET(N=10,c=-5) environment.

Chapter 7

Conclusions

In this work, we investigated Bayesian approaches to tackle the exploration-exploitation trade-off in RL. We gave a brief but thorough overview of the theory and fundamental methods in RL and argued that popular action selection methods suffer from conceptual problems, that often prevent the agent from finding optimal policies in environments which are inherently hard to explore.

We saw that the Bayesian framework offers a natural way of computing the epistemic uncertainty of the world dynamics models. The argument followed that truly *directed* exploration should use the information about this uncertainty and that one way to incorporate it, is by considering the posterior variance of the value function. We proceeded to give several estimates of this uncertainty, one expensive sampling method (Monte Carlo), one method from the literature (Uncertainty Bellman equation (UBE)) and two novel methods directly leveraging the Bellman consistency (Epistemic Uncertainty of Decorrelated Value (EUDV) and Epistemic Uncertainty Bound (EUB)).

We then stressed the importance of having an action selection scheme that would leverage the posterior variance in order to make decisions that would efficiently balance exploration and exploitation. We gave an overview of several standard methods from the literature (UCB, Thompson sampling and Posterior Sampling for Reinforcement Learning (PSRL)) and discussed the theory of the Knowledge Gradient, a highly promising method never before deployed on stateful RL environments. We followed by introducing a novel generalization of the Knowledge Gradient as well as its sampling version, which we extended further to involve more than 1 'imagined' exploratory step.

We tested all the aforementioned methods on tabular environments by firstly comparing the epistemic uncertainty of return estimates to the Monte Carlo baseline. We found that our methods (EUDV and EUB) are much more accurate estimations than the UBE, which

massively overestimates the variance because of including a stationary maximal-value term. For the action selection methods dependent directly on these estimates we found that:

- The UCB and Thompson sampling performance is extremely dependent on the hyper-parameter β and therefore very hard to compare directly. In general, it is much easier to tune β with the EUDV and EUB estimates.
- The Generalized Knowledge Gradient (GKG) method is a very reliable method that does not require tuning of any hyperparameters and its performance depends directly on the accuracy of the epistemic uncertainty estimate.

For the action selection methods dependent on directly sampling from the dynamics posteriors, we found that:

- Increasing the step parameter N beyond 2 in the N-step Monte Carlo Knowledge Gradient (MCKG-N) does not improve performance.
- MCKG-N combined with the OFU principle outperforms the current state-of-the-art PSRL on all tested environments.

References

Atan, O., Jordon, J., and van der Schaar, M. (2018). Deep-treat: Learning optimal personalized treatments from observational data using neural networks.

Banach, S. (1922). Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fundamenta Mathematicae*, page 133–181.

Bellemare, M. G., Dabney, W., and Munos, R. (2017). A distributional perspective on reinforcement learning. pages 449–458.

Bellman, R. (1957). *Dynamic Programming*. Dover Publications.

Bertsekas, D. P., Bertsekas, D. P., Bertsekas, D. P., and Bertsekas, D. P. (1995). *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 1613–1622. JMLR.org.

Dabney, W., Ostrovski, G., Silver, D., and Munos, R. (2018). Implicit quantile networks for distributional reinforcement learning. 80:1096–1105.

DeGroot, M. (1970). *Optimal statistical decisions*. McGraw-Hill, New York, NY [u.a].

Deisenroth, M. (2009). Efficient reinforcement learning for motor control.

Ferreira, E. and Lefèvre, F. (2015). Reinforcement-learning based dialogue system for human–robot interactions with socially-inspired rewards. *Computer Speech Language*, 34(1):256 – 274.

Filos, A., Tigas, P., McAllister, R., Rhinehart, N., Levine, S., and Gal, Y. (2020). Can autonomous vehicles identify, recover from, and adapt to distribution shifts?

Frazier, P. (2009). Knowledge-gradient methods for statistical learning.

Ghavamzadeh, M., Mannor, S., Pineau, J., and Tamar, A. (2016). Bayesian reinforcement learning: A survey. cite arxiv:1609.04436.

Gu, S., Holly, E., Lillicrap, T., and Levine, S. (2017). Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *Proceedings 2017 IEEE International Conference on Robotics and Automation (ICRA)*, Piscataway, NJ, USA. IEEE.

Gupta, S. and Miescke, K. (1996). Bayesian look ahead one-stage sampling allocations for selection of the best population. *Journal of statistical planning and inference*, 54(2):229–244.

Janz, D., Hron, J., Hernández-Lobato, J. M., Hofmann, K., and Tschiatschek, S. (2018). Successor uncertainties: exploration and uncertainty in temporal difference learning. *CoRR*, abs/1810.06530.

Jaynes, E. T. (2003). *Probability theory: The logic of science*. Cambridge University Press, Cambridge.

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

Levine, S., Finn, C., Darrell, T., and Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *J. Mach. Learn. Res.*, 17(1):1334–1373.

Mavrin, B., Zhang, S., Yao, H., Kong, L., Wu, K., and Yu, Y. (2019). Distributional reinforcement learning for efficient exploration.

Mingxi Li, Wouter M.Koolen, T. v. E. (2017). Thompson sampling for monte carlo tree search and maxi-min action identification.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Murphy, K. P. (2007). Conjugate bayesian analysis of the gaussian distribution. Technical report.

O'Donoghue, B. (2018). Variational bayesian reinforcement learning with regret bounds. *CoRR*, abs/1807.09647.

O'Donoghue, B., Osband, I., Munos, R., and Mnih, V. (2017). The uncertainty bellman equation and exploration. *CoRR*, abs/1709.05380.

Osband, I., Russo, D., and Van Roy, B. (2013). (more) efficient reinforcement learning via posterior sampling. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26*, pages 3003–3011. Curran Associates, Inc.

Osband, I., Russo, D., Wen, Z., and Roy, B. V. (2017). Deep exploration via randomized value functions. *CoRR*, abs/1703.07608.

Osband, I. and Van Roy, B. (2017). Why is posterior sampling better than optimism for reinforcement learning? In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2701–2710, International Convention Centre, Sydney, Australia. PMLR.

Powell, W. and Ryzhov, I. (2012). *Optimal Learning*. Wiley Series in Probability and Statistics. Wiley.

Rasmussen, C. and Kuss, M. (2004). Gaussian processes in reinforcement learning. *Advances in Neural Information Processing Systems 16*, pages 751–759.

Ryzhov, I. and Powell, W. (2009). A monte carlo knowledge gradient method for learning abatement potential of emissions reduction technologies. pages 1492–1502.

Silver, D. (2015). *Reinforcement Learning course*. University College London.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis, D. (2017). Mastering the game of go without human knowledge. *Nature*, 550:354–.

Singh, S., Jaakkola, T., Littman, M. L., and Szepesvári, C. (2000a). Convergence results for single-step on-policyreinforcement-learning algorithms. *Mach. Learn.*, 38(3):287–308.

Singh, S. P., Kearns, M. J., Litman, D. J., and Walker, M. A. (2000b). Reinforcement learning for spoken dialogue systems. In Solla, S. A., Leen, T. K., and Müller, K., editors, *Advances in Neural Information Processing Systems 12*, pages 956–962. MIT Press.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. The MIT Press, second edition.

Szepesvari, C. (2010). *Algorithms for Reinforcement Learning*. Morgan and Claypool Publishers.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294.

Thrun, S. (1992). Efficient exploration in reinforcement learning. Technical Report CMU-CS-92-102, Carnegie Mellon University, Pittsburgh, PA.

Wang, Y. and Powell, W. (2016). Finite-time analysis for the knowledge-gradient policy.

Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3):279–292.

Weiss, N., Holmes, P., and Hardy, M. (2006). *A Course in Probability*. Pearson Addison Wesley.

Appendix A

Supporting proofs and derivations

A.1 Environment models

A.1.1 Reward model results

Here we present derivations of the essential quantities given in section 3.1.1. First, we show that the marginalized mean of next reward from state-action (s_t, a_t), is given by the m_{s_t,a_t} parameter of the parameter posterior:

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{\theta}}\left[\mathbb{E}_{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}}[R_{t+1}|s_t,a_t,\boldsymbol{\theta}]\right] &= \mathbb{E}_{\boldsymbol{\theta}}[m_{s_t,a_t}] \\
&= \int\int m_{s_t,a_t} p(m_{s_t,a_t},\tau_{s_t,a_t}) d\mu_{s_t,a_t} d\tau_{s_t,a_t} \\
&= \int\int m_{s_t,a_t} p(m_{s_t,a_t}|\tau_{s_t,a_t}) p(\tau_{s_t,a_t}) dm_{s_t,a_t} d\tau_{s_t,a_t} \\
&= \int m_{s_t,a_t} p(\tau_{s_t,a_t}) d\tau_{s_t,a_t} = m_{s_t,a_t}
\end{aligned}$$

Next, the marginalized second moment of the reward:

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{\theta}}\left[\mathbb{E}_{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}}[R_{t+1}^2|s_t,a_t,\boldsymbol{\theta}]\right] &= \mathbb{E}_{\boldsymbol{\theta}}[m_{s_t,a_t}^2 + \tau_{s_t,a_t}^{-1}] \\
&= \mathbb{E}_{\boldsymbol{\theta}}[m_{s_t,a_t}^2] + \mathbb{E}_{\boldsymbol{\theta}}[\tau_{s_t,a_t}^{-1}]
\end{aligned}$$

And similarly to before:

$$\begin{aligned}\mathbb{E}_{\boldsymbol{\theta}}[m^2_{s_t,a_t}] &= \int\int m^2_{s_t,a_t} p(m_{s_t,a_t}|\tau_{s_t,a_t}) p(\tau_{s_t,a_t}) dm_{s_t,a_t} d\tau_{s_t,a_t} \\ &= m^2_{s_t,a_t} + \frac{1}{\lambda_{s_t,a_t}} \int \tau^{-1}_{s_t,a_t} p(\tau_{s_t,a_t}) d\tau_{s_t,a_t} \\ &= m^2_{s_t,a_t} + \frac{\beta_{s_t,a_t}\Gamma(\alpha_{s_t,a_t}-1)}{\lambda_{s_t,a_t}\Gamma(\alpha_{s_t,a_t})}\end{aligned}$$

And by plugging in the pdf of NG in the integral we also realise:

$$\mathbb{E}_{\boldsymbol{\theta}}[\tau^{-1}_{s_t,a_t}] = \frac{\Gamma(\alpha_{s_t,a_t}-1)}{\Gamma(\alpha_{s_t,a_t})}\beta_{s_t,a_t}$$

Hence by using the property of the gamma function:

$$\mathbb{E}_{\boldsymbol{\theta}}\left[\mathbb{E}_{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}}[R^2_{t+1}|s_t,a_t,\boldsymbol{\theta}]\right] = m^2_{s_t,a_t} + \frac{\beta_{s_t,a_t}}{\alpha_{s_t,a_t}-1}\Big(1+\frac{1}{\lambda_{s_t,a_t}}\Big)$$

The last quantity of interest, is the epistemic uncertainty of reward, so its marginalized variance:

$$\begin{aligned}\mathrm{Var}_{\boldsymbol{\theta}}\left[\mathbb{E}_{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}}[R_{t+1}|s_t,a_t,\boldsymbol{\theta}]\right] &= \mathbb{E}_{\boldsymbol{\theta}}\left[\mathbb{E}_{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}}[R_{t+1}|s_t,a_t,\boldsymbol{\theta}]^2\right] \\ &\qquad - \mathbb{E}_{\boldsymbol{\theta}}\left[\mathbb{E}_{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}}[R_{t+1}|s_t,a_t,\boldsymbol{\theta}]\right]^2 \\ &= \mathbb{E}_{\boldsymbol{\theta}}[m^2_{s_t,a_t}] - m^2_{s_t,a_t} \\ &= \frac{\beta_{s_t,a_t}}{\lambda_{s_t,a_t}(\alpha_{s_t,a_t}-1)}\end{aligned}$$

A.1.2 Transition dynamics model results

In order to derive the marginal mean, variance and covariance of the next state probabilities, it is sufficient to notice that since we model the i^{th} transition probability to be Categorical and its parameters to be Dirichlet, the mean and variance of this probability will be the mean and variance of the underlying Dirichlet posterior. In detail, for the mean:

$$\mathbb{E}_{\boldsymbol{\theta}}[p(s_{t+1}|s_t,a_t,\boldsymbol{\theta})] = \int [\boldsymbol{\kappa}_{s_t,a_t}]_{s_{t+1}} p(\boldsymbol{\kappa}_{s_t,a_t}) d\boldsymbol{\kappa}_{s_t,a_t} = \frac{[\boldsymbol{a}_{s,a}+\boldsymbol{c}_{s,a}]_{s'}}{\mathbf{1}^T(\boldsymbol{a}_{s,a}+\boldsymbol{c}_{s,a})}$$

And similarly for the variance and covariance, they will just be the variance and covariance of the underlying posterior Dirichlet.

A.2 Epistemic uncertainty estimates

A.2.1 Direct derivation using Bellman consistency (EUDV and EUB)

In order to arrive at equation 3.25, from which we later obtain EUDV and EUB, we start by restating the Bellman equation 3.24 obeyed by Q-value in the parametrized MDP $\mathcal{W} \mid \boldsymbol{\theta}$:

$$Q^{\pi,\mathcal{W}\mid\boldsymbol{\theta}}(s_t, a_t) = \mathbb{E}_{\pi,\mathcal{W}\mid\boldsymbol{\theta}}[R_{t+1}|s_t, a_t, \boldsymbol{\theta}] + \sum_{s',a'} \pi(a'|s')p(s'|s_t, a_t, \boldsymbol{\theta})Q^{\pi,\mathcal{W}\mid\boldsymbol{\theta}}(s', a')$$

Taking variances with respect to the model parameters $\boldsymbol{\theta}$ of both sides we get:

$$\begin{aligned}\text{Var}_{\boldsymbol{\theta}}[Q^{\pi,\mathcal{W}\mid\boldsymbol{\theta}}(s_t, a_t)] = \text{Var}_{\boldsymbol{\theta}}\left[\mathbb{E}_{\pi,\mathcal{W}\mid\boldsymbol{\theta}}[R_{t+1}|s_t, a_t, \boldsymbol{\theta}]\right] \\ + \gamma^2 \text{Var}_{\boldsymbol{\theta}}\left[\sum_{a',s'} \pi(a'|s')p(s'|s_t, a_t, \boldsymbol{\theta})Q^{\pi,\mathcal{W}\mid\boldsymbol{\theta}}(s', a')\right]\end{aligned}$$

where we utilize the DAG assumption, since if the agent is not permitted to return back to (s_t, a_t) then:

$$\mathbb{E}_{\pi,\mathcal{W}\mid\boldsymbol{\theta}}[R_{t+1}|s_t, a_t, \boldsymbol{\theta}] \perp p(s'|s_t, a_t, \boldsymbol{\theta})Q^{\pi,\mathcal{W}\mid\boldsymbol{\theta}}(s', a'), \quad \forall s' \in \mathcal{S}, a' \in \mathcal{A}$$

This can be decomposed further:

$$\begin{aligned}\text{Var}_{\boldsymbol{\theta}}[Q^{\pi,\mathcal{W}\mid\boldsymbol{\theta}}(s_t, a_t)] &= \text{Var}_{\boldsymbol{\theta}}\left[\mathbb{E}_{\pi,\mathcal{W}\mid\boldsymbol{\theta}}[R_{t+1}|s_t, a_t, \boldsymbol{\theta}]\right] \\ &\quad + \gamma^2 \sum_{a',s'} \pi(a'|s')^2 \text{Var}_{\boldsymbol{\theta}}\left[p(s'|s_t, a_t, \boldsymbol{\theta})Q^{\pi,\mathcal{W}\mid\boldsymbol{\theta}}(s', a')\right] \\ &\quad + 2\gamma^2 \sum_{\substack{a',s' \\ a''\neq a', s''\neq s'}} \pi(a'|s')^2 \text{Cov}_{\boldsymbol{\theta}}\big[p(s'|s_t, a_t, \boldsymbol{\theta})Q^{\pi,\mathcal{W}\mid\boldsymbol{\theta}}(s', a'), \\ &\qquad\qquad p(s''|s_t, a_t, \boldsymbol{\theta})Q^{\pi,\mathcal{W}\mid\boldsymbol{\theta}}(s'', a'')\big] \\ &= \text{Var}_{\boldsymbol{\theta}}\left[\mathbb{E}_{\pi,\mathcal{W}\mid\boldsymbol{\theta}}[R_{t+1}|s_t, a_t, \boldsymbol{\theta}]\right] \\ &\quad + \gamma^2 \sum_{a',s'} \pi(a'|s')^2 \text{Var}_{\boldsymbol{\theta}}\left[p(s'|s_t, a_t, \boldsymbol{\theta})Q^{\pi,\mathcal{W}\mid\boldsymbol{\theta}}(s', a')\right] \\ &\quad + 2\gamma^2 \sum_{\substack{a',s' \\ a''\neq a', s''\neq s'}} \pi(a'|s')^2 C(s', a', s'', a''|s_t, a_t, \boldsymbol{\theta})\end{aligned}$$

where we introduced the value correlation variable:

$$C(s', a', s'', a''|s_t, a_t, \boldsymbol{\theta}) = \text{Cov}_{\boldsymbol{\theta}}\big[p(s'|s_t, a_t, \boldsymbol{\theta})Q^{\pi,\mathcal{W}|\boldsymbol{\theta}}(s', a') \\ p(s''|s_t, a_t, \boldsymbol{\theta})Q^{\pi,\mathcal{W}|\boldsymbol{\theta}}(s'', a'')\big]$$

The DAG assumption also implies that:

$$p(s'|s_t, a_t, \boldsymbol{\theta}) \perp Q^{\pi,\mathcal{W}|\boldsymbol{\theta}}(s', a')$$

allowing us to make the following decomposition:

$$\text{Var}_{\boldsymbol{\theta}}\left[p(s'|s_t, a_t, \boldsymbol{\theta})Q^{\pi,\mathcal{W}|\boldsymbol{\theta}}(s', a')\right] = \\ \mathbb{E}_{\boldsymbol{\theta}}\left[Q^{\pi,\mathcal{W}|\boldsymbol{\theta}}(s', a')\right]^2 \text{Var}_{\boldsymbol{\theta}}[p(s'|s_t, a_t, \boldsymbol{\theta})] \\ + \text{Var}_{\boldsymbol{\theta}}\left[Q^{\pi,\mathcal{W}|\boldsymbol{\theta}}(s', a')\right]\left[\mathbb{E}_{\boldsymbol{\theta}}[p(s'|s_t, a_t, \boldsymbol{\theta})]^2 + \text{Var}_{\boldsymbol{\theta}}[p(s'|s_t, a_t, \boldsymbol{\theta})\right]$$

using the formula for the variance of the product of two independent random variables.

A.2.2 Epistemic Uncertainty Bound (EUB)

In the main text, we saw a proof that the recursive formula for the epistemic uncertainty has a unique fixed point assuming $C(s', a', s'', a''|s_t, a_t, \boldsymbol{\theta}) = 0$, which gave us the Epistemic Uncertainty of Decorrelated Value (EUDV). Here we focus on deriving the EUB, which does not explicitly assume that term to be 0.

To derive the EUB, we make use of the following assumption:

Assumption 3. *The value function is independent of the transition probabilities:*

$$Q^{\pi,\mathcal{W}|\boldsymbol{\theta}}(s', a') \perp p(s''|s_t, a_t, \boldsymbol{\theta}) \quad \forall s_t, s', s'' \in \mathcal{S}, s' \neq s'' \tag{A.1}$$

$$\forall a_t, a' \in \mathcal{A} \tag{A.2}$$

For simplicity, let us call the s', a', s'', a'' elements of C as follows:

$$A = p(s'|s_t, a_t, \boldsymbol{\theta})$$
$$B = p(s''|s_t, a_t, \boldsymbol{\theta})$$
$$X = Q^{\pi,\mathcal{W}|\boldsymbol{\theta}}(s', a')$$
$$Y = Q^{\pi,\mathcal{W}|\boldsymbol{\theta}}(s'', a'')$$

Using the DAG assumption and Assumption 3 above, we have the following independencies:

$$
\begin{aligned}
A \perp X \quad & \text{by DAG} \\
A \perp Y \quad & \text{by Assumption 3} \\
B \perp X \quad & \text{by Assumption 3} \\
B \perp Y \quad & \text{by DAG}
\end{aligned}
$$

We can then make the following decomposition:

$$
\begin{aligned}
C(s', a', s'', a''|s_t, a_t, \boldsymbol{\theta}) &= \operatorname{Cov}_{\boldsymbol{\theta}}[AX, BY] = \\
&= \mathbb{E}_{\boldsymbol{\theta}}[AXBY] - \mathbb{E}_{\boldsymbol{\theta}}[AX]\,\mathbb{E}_{\boldsymbol{\theta}}[BY] \\
&= \mathbb{E}_{\boldsymbol{\theta}}[AB]\,\mathbb{E}_{\boldsymbol{\theta}}[XY] - \mathbb{E}_{\boldsymbol{\theta}}[A]\,\mathbb{E}_{\boldsymbol{\theta}}[B]\,\mathbb{E}_{\boldsymbol{\theta}}[X]\,\mathbb{E}_{\boldsymbol{\theta}}[Y] \\
&= \big(\mathbb{E}_{\boldsymbol{\theta}}[AB] - \mathbb{E}_{\boldsymbol{\theta}}[A]\,\mathbb{E}_{\boldsymbol{\theta}}[B]\big)\big(\mathbb{E}_{\boldsymbol{\theta}}[XY] - \mathbb{E}_{\boldsymbol{\theta}}[X]\,\mathbb{E}_{\boldsymbol{\theta}}[Y]\big) \\
&\quad + \mathbb{E}_{\boldsymbol{\theta}}[A]\,\mathbb{E}_{\boldsymbol{\theta}}[B]\big(\mathbb{E}_{\boldsymbol{\theta}}[XY] - \mathbb{E}_{\boldsymbol{\theta}}[X]\,\mathbb{E}_{\boldsymbol{\theta}}[Y]\big) \\
&\quad + \mathbb{E}_{\boldsymbol{\theta}}[XY]\big(\mathbb{E}_{\boldsymbol{\theta}}[AB] - \mathbb{E}_{\boldsymbol{\theta}}[A]\,\mathbb{E}_{\boldsymbol{\theta}}[B]\big) \\
&= \operatorname{Cov}_{\boldsymbol{\theta}}[A, B]\operatorname{Cov}_{\boldsymbol{\theta}}[X, Y] + \mathbb{E}_{\boldsymbol{\theta}}[A]\,\mathbb{E}_{\boldsymbol{\theta}}[B]\operatorname{Cov}_{\boldsymbol{\theta}}[X, Y] \\
&\quad + \mathbb{E}_{\boldsymbol{\theta}}[X]\,\mathbb{E}_{\boldsymbol{\theta}}[Y]\operatorname{Cov}_{\boldsymbol{\theta}}[A, B] \\
&= \operatorname{Cov}_{\boldsymbol{\theta}}[X, Y]\big(\operatorname{Cov}_{\boldsymbol{\theta}}[A, B] + \mathbb{E}_{\boldsymbol{\theta}}[A]\,\mathbb{E}_{\boldsymbol{\theta}}[B]\big) \\
&\quad + \mathbb{E}_{\boldsymbol{\theta}}[X]\,\mathbb{E}_{\boldsymbol{\theta}}[Y]\operatorname{Cov}_{\boldsymbol{\theta}}[A, B]
\end{aligned}
$$

We can further bound this term, using the Cauchy-Schwartz inequality:

$$
\begin{aligned}
\big|C(s', a', s'', a''|s_t, a_t, \boldsymbol{\theta})\big| \leq \big|\operatorname{Cov}_{\boldsymbol{\theta}}[X, Y]\big|\big(\big|\operatorname{Cov}_{\boldsymbol{\theta}}[A, B]\big| + \big|\mathbb{E}_{\boldsymbol{\theta}}[A]\,\mathbb{E}_{\boldsymbol{\theta}}[B]\big|\big) \\
+ \big|\mathbb{E}_{\boldsymbol{\theta}}[X]\,\mathbb{E}_{\boldsymbol{\theta}}[Y]\big|\big|\operatorname{Cov}_{\boldsymbol{\theta}}[A, B]\big|
\end{aligned}
$$

And using the fact that $\big|\operatorname{Cov}_{\boldsymbol{\theta}}[X, Y]\big| \leq \sqrt{\operatorname{Var}_{\boldsymbol{\theta}}[X]\operatorname{Var}_{\boldsymbol{\theta}}[Y]}$ we get:

$$
\begin{aligned}
\big|C(s', a', s'', a''|s_t, a_t, \boldsymbol{\theta})\big| \leq \sqrt{\operatorname{Var}_{\boldsymbol{\theta}}[X]\operatorname{Var}_{\boldsymbol{\theta}}[Y]}\big(\big|\operatorname{Cov}_{\boldsymbol{\theta}}[A, B]\big| + \big|\mathbb{E}_{\boldsymbol{\theta}}[A]\,\mathbb{E}_{\boldsymbol{\theta}}[B]\big|\big) \\
+ \big|\mathbb{E}_{\boldsymbol{\theta}}[X]\,\mathbb{E}_{\boldsymbol{\theta}}[Y]\big|\big|\operatorname{Cov}_{\boldsymbol{\theta}}[A, B]\big|
\end{aligned}
$$

as required.

A.3 Generalized Knowledge gradient

A.3.1 Reduction in uncertainty

In this section we derive the equations for the reduction in uncertainty for the EUDV estimate.

We start by reminding ourselves the definition of the reduction in uncertainty:

$$\tilde{\sigma}^t_{s_t,x} = \text{Var}_{\boldsymbol{\theta}_t}[Q^{*,\mathcal{W}\,|\,\boldsymbol{\theta}_t}(s_t,x)] - \mathbb{E}\left[\,\text{Var}_{\boldsymbol{\theta}_{t+1}}[Q^{*,\mathcal{W}\,|\,\boldsymbol{\theta}_{t+1}}(s_t,x)]\right]$$

where the full description of the expectation is given in the main text.

EUDV

The EUDV at time t is given by the following recursion:

$$\begin{aligned} u^{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}_t}(s_t,a_t) = \text{Var}_{\boldsymbol{\theta}_t}&\left[\,\mathbb{E}_{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}_t}[R_{t+1}|s_t,a_t,\boldsymbol{\theta}_t]\right] \\ &+ \gamma^2 \sum_{a',s'} \pi(a'|s')^2 \Big[\,\mathbb{E}_{\boldsymbol{\theta}_t}\left[Q^{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}_t}(s',a')\right]^2 \text{Var}_{\boldsymbol{\theta}}[p(s'|s_t,a_t,\boldsymbol{\theta})] \\ &\quad + u^{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}_t}(s',a')\left[\,\mathbb{E}_{\boldsymbol{\theta}_t}[p(s'|s_t,a_t,\boldsymbol{\theta}_t)]^2 + \text{Var}_{\boldsymbol{\theta}_t}[p(s'|s_t,a_t,\boldsymbol{\theta}_t)]\right]\Big] \end{aligned} \tag{A.3}$$

where

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}\,|\mathcal{D}_t$$

is the parameter posterior after t steps in the environment and where we have made the shortcut $u^{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}_t}(s_t,a_t) = \text{Var}_{\boldsymbol{\theta}_t}[Q^{*,\mathcal{W}\,|\,\boldsymbol{\theta}_t}(s_t,a_t)]$. The derivation of the reduction in uncertainty formula is possible by noticing two facts:

1. The marginalized mean Q-value does not change after we simulate mean observations:

$$\mathbb{E}_{\boldsymbol{\theta}_t}[Q^{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}_t}(s',a')] = \mathbb{E}\left[\,\mathbb{E}_{\boldsymbol{\theta}_{t+1}}[Q^{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}_{t+1}}(s',a')]\right]$$

   This is because when we marginalize over the parameters to obtain $\mathbb{E}_{\boldsymbol{\theta}_t}[Q^{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}_t}(s',a')]$ we only take into account the **means** of reward and transitions, which do not change when we add fake mean observations to the data.

2. The reduction in uncertainty is only non-zero for the state and action we are considering, so:

$$\tilde{\sigma}^t_{s',a'} = 0, \quad \forall s' \neq s_t, a' \neq x$$

   This follows directly from the fact, that if in state s_t and simulating action $a_t = x$, only reward variance for (s_t, x) will be able to go down, with all the rest states and actions

unchanged. This makes intuitive sense, as we expect the uncertainty to go down only for the state and actions that we consider exploring.

Writing out the reduction in variance as given in equation A.3, we get:

$$
\begin{aligned}
\tilde{\sigma}^t_{s_t,x} &= u^{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}_t}(s_t,x) - \mathbb{E}[u^{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}_{t+1}}(s_t,x)] \\
&= \underbrace{\text{Var}_{\boldsymbol{\theta}_t}\left[\mathbb{E}_{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}_t}[R_{t+1}|s,x,\boldsymbol{\theta}_t]\right] - \mathbb{E}\left[\text{Var}_{\boldsymbol{\theta}_{t+1}}\left[\mathbb{E}_{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}_{t+1}}[R_{t+1}|s,x,\boldsymbol{\theta}_{t+1}]\right]\right]}_{\text{Reduction in reward variance = \textbf{(1)}}} \\
&+ \gamma^2 \sum_{a',s'} \pi(a'|s')^2 \Big[\\
&\mathbb{E}_{\boldsymbol{\theta}_t}[Q^{\pi,\mathcal{W}\,|\,\boldsymbol{\theta}_t}(s',a')]^2 \underbrace{\Big[\text{Var}_{\boldsymbol{\theta}_t}[p(s'|s,a,\boldsymbol{\theta}_t)] - \mathbb{E}\left[\text{Var}_{\boldsymbol{\theta}_{t+1}}[p(s'|s,a,\boldsymbol{\theta}_{t+1})]\right]\Big]}_{\text{Reduction in dynamics variance = \textbf{(2)}}} \\
&+ \tilde{\sigma}^t_{s',a'} \Big[\underbrace{\text{Var}_{\boldsymbol{\theta}_t}[p(s'|s,a,\boldsymbol{\theta}_t)] - \mathbb{E}\left[\text{Var}_{\boldsymbol{\theta}_{t+1}}[p(s'|s,a,\boldsymbol{\theta}_{t+1})]\right]}_{\text{Reduction in dynamics variance = \textbf{(2)}}} \Big] \\
&+ \underbrace{\mathbb{E}_{\boldsymbol{\theta}_t}[p(s'|s,a,\boldsymbol{\theta}_t)]^2 - \mathbb{E}\left[\mathbb{E}_{\boldsymbol{\theta}_{t+1}}[p(s'|s,a,\boldsymbol{\theta}_{t+1})]^2\right]}_{\text{Increase in mean probability of next state = \textbf{(3)}}} \Big] \Big]
\end{aligned}
$$

Where we already made use of the stated fact 1 above. Using fact 2, we realise that the above becomes a linear equation in $\tilde{\sigma}^t_{s_t,x}$, which we can solve directly as presented in the main text.

Appendix B

Supplementary experimental material

B.1 Comparison of uncertainty estimates.

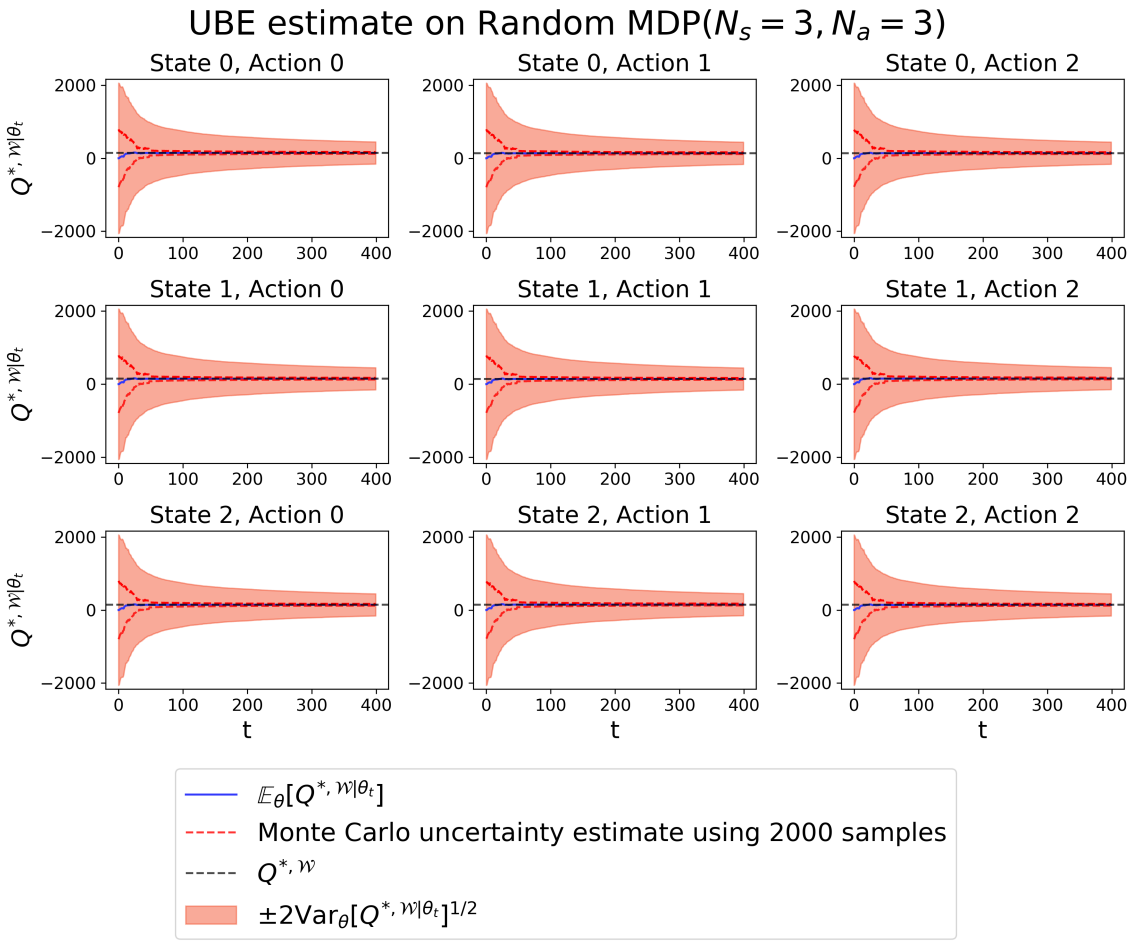


Fig. B.1 Comparison of the UBE uncertainty estimate to the Monte Carlo (N=2000) baseline on the Random MDP $(N_s = 3, N_a = 3)$ environment, averaged for 10 seed runs.

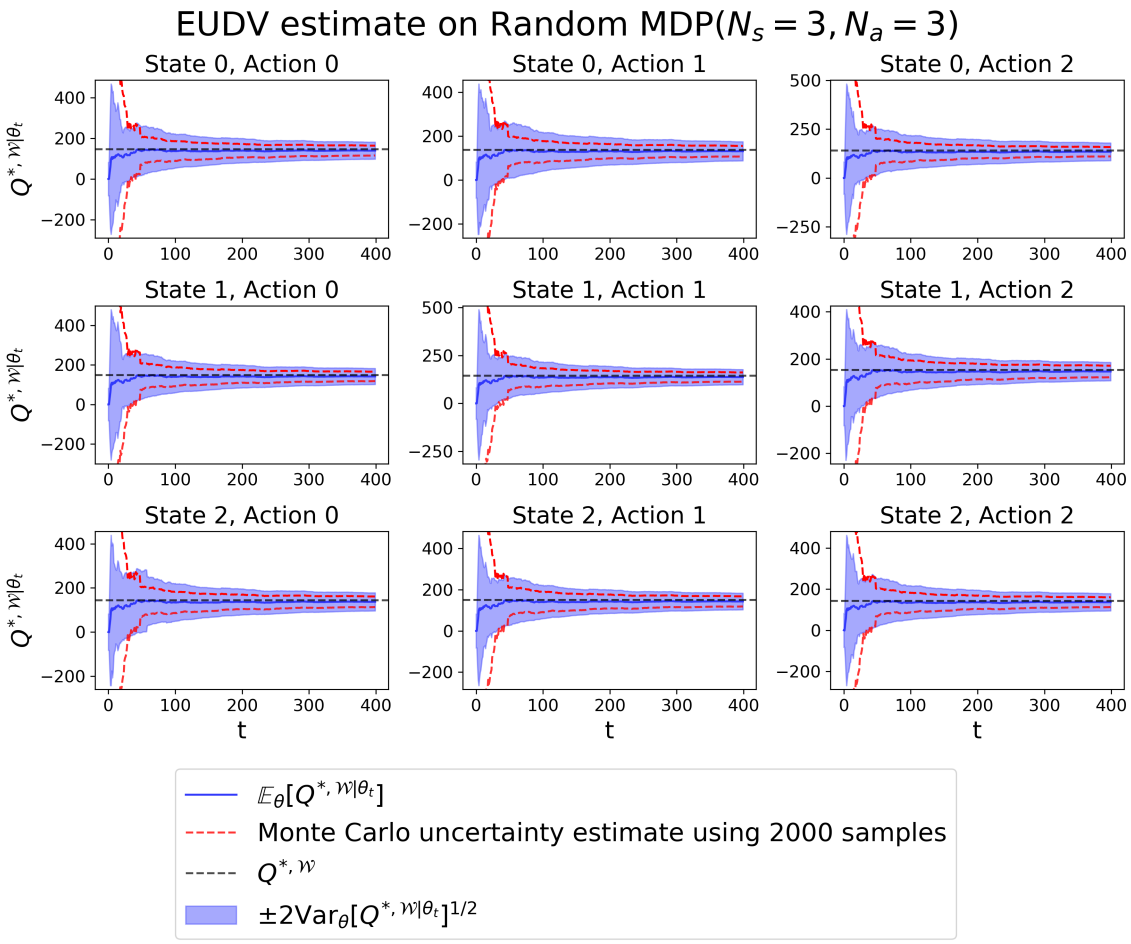


Fig. B.2 Comparison of the EUDV uncertainty estimate to the Monte Carlo (N=2000) baseline on the Random MDP $(N_s = 3, N_a = 3)$ environment, averaged for 10 seed runs.

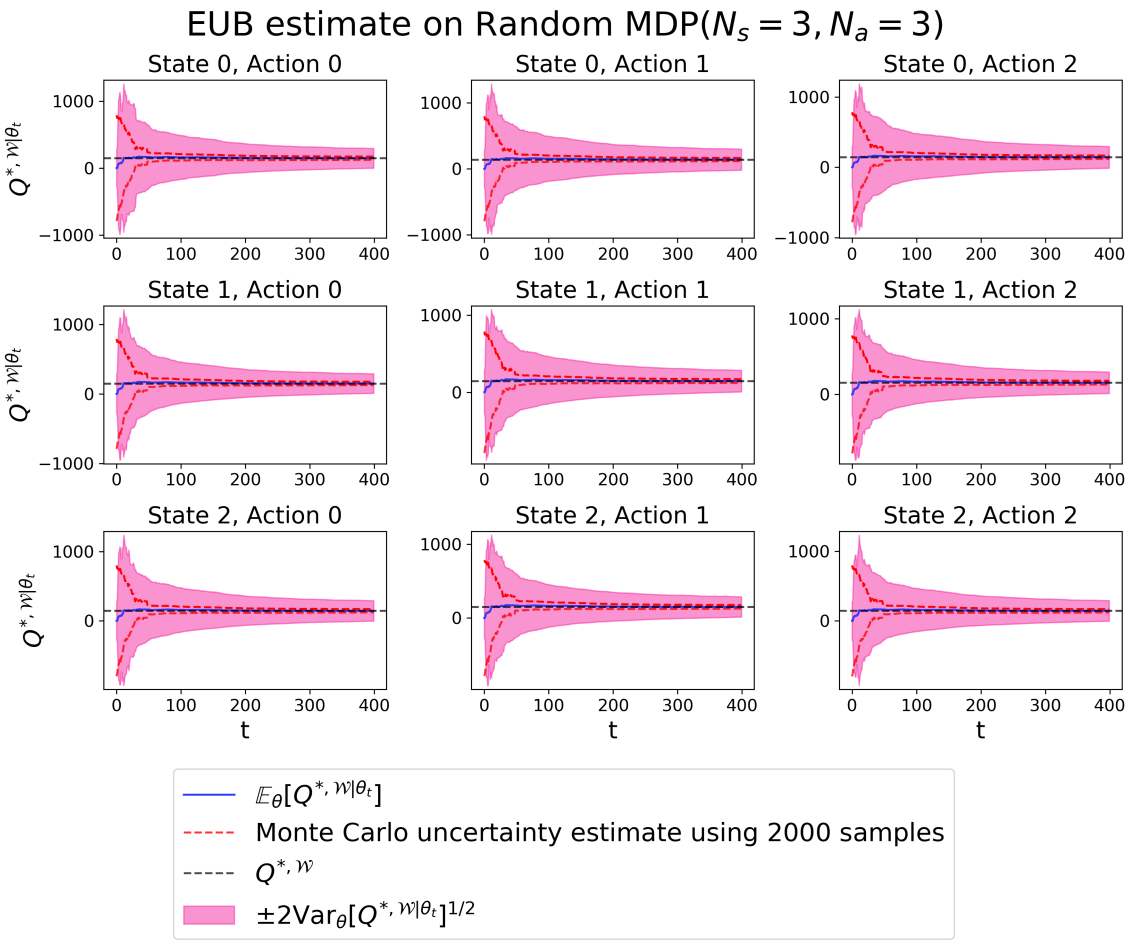


Fig. B.3 Comparison of the EUB uncertainty estimate to the Monte Carlo (N=2000) baseline on the Random MDP $(N_s = 3, N_a = 3)$ environment, averaged for 10 seed runs.