Exploring Uncertainty in Distributional Reinforcement Learning

Georgy Antonov
georgy.antonov@tuebingen.mpg.de
Max Planck Institute for Biological Cybernetics, Tübingen
Graduate Training Centre for Neuroscience, University of Tübingen

Peter Dayan
peter.dayan@tuebingen.mpg.de
Max Planck Institute for Biological Cybernetics, Tübingen
University of Tübingen

Abstract

Epistemic uncertainty, which stems from what a learning algorithm does not know, is the natural signal for exploration. Capturing and exploiting epistemic uncertainty for efficient exploration is conceptually straightforward for model-based methods. However, it is computationally ruinous, prompting a search for model-free approaches. One of the most seminal and venerable such is Bayesian Q-learning, which maintains and updates an approximation to the distribution of the long run returns associated with state-action pairs. However, this approximation can be rather severe. Recent work on distributional reinforcement learning (DRL) provides many powerful methods for modelling return distributions, which offer the prospect of improving upon Bayesian Q-learning's parametric scheme, but have not been fully investigated for their exploratory potential. Here, we examine the characteristics of a number of DRL algorithms in the context of exploration and propose a novel Bayesian analogue of the categorical temporal-difference algorithm. We show that this works well, converging appropriately to a close approximation to the true return distribution.

1 Introduction

Efficient exploration remains a challenging problem in reinforcement learning (RL). Optimal solutions to the explore-exploit trade-off involve planning in continuous belief spaces (Kaelbling et al., 1998) whereby the long-run costs and benefits of acquiring information are calculated and balanced off against exploitation of the current knowledge. Belief states correspond to full probability distributions which characterise the *epistemic* state of knowledge of a learning algorithm, indicating what it knows it knows and what it knows it does not know. The incentive for exploration thus naturally arises from the resulting opportunity for learning that there might be a better alternative.

However, computing this incentive is notoriously challenging (Gittins, 1979; Deisenroth et al., 2009; Silver & Veness, 2010; Guez et al., 2012). Heuristic approaches typically collapse such distributional information onto single statistics, such as constraints on action values associated with the popular upper confidence bound policy (Auer, 2002). However, representing epistemic uncertainty with a full distribution, rather than a single statistic, allows exploration to be directed to the precise task of reward-efficient policy improvement. That is, one can calculate the *value of perfect information* (Howard, 1966) that one expects to gain through exploring a given action, and balance this against the expected cost of doing so. This provides an (admittedly myopic) guarantee that exploration is directed towards improving the subsequent behaviour of the algorithm, and can thus be employed when appropriate.

Along these lines, Dearden et al. (1998) proposed the seminal Bayesian Q-learning (BQL) algorithm which maintains a full (albeit approximate) belief distribution over the possible long-run expected values of actions. BQL makes a number of strong assumptions, including that the return distribution is Gaussian; however, it performs very well, with highly sample-efficient exploration informed by a rich representation of subjective uncertainty.

Distributional RL (Bellemare et al., 2023) offers a formal framework for learning the whole distribution over returns. In this work, however, the uncertainty in the return is of the objective rather than subjective kind: it comes from the *aleatoric* nature of the environment, and is therefore irreducible. Here, we examine both the merits and perils of maintaining and using such distributional representations for exploration. We derive a novel Bayesian distributional RL algorithm which explicitly models its subjective uncertainty and, as a result, exhibits stable learning and efficient exploration.

2 Background

2.1 Setting

We consider the setting in which an agent interacts with an environment modelled as a Markov decision process (MDP). An MDP is defined as a tuple $\langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a stochastic reward function, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a stochastic transition kernel, and $\gamma \in [0, 1)$ is the discount factor. The agent is equipped with a policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ which, for any given state, outputs a probability distribution over actions.

The random return $Z_{s,a}$[1] is defined as the discounted cumulative reward collected by the agent conditioned on starting in state s and executing action a, that is $Z_{s,a} = \sum_{t=0}^{\infty} \gamma^t R_t \mid S_0 = s, A_0 = a$. The value of a given policy π is given by the state-action value function $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, defined as the expected return accrued when choosing action a in state s and following π thereafter, $Q^\pi(s, a) = \mathbb{E}[Z_{s,a}]$.

One main goal of an RL agent is to learn an optimal policy which maximises the expected return for all state-action pairs, that is $\pi^* = \operatorname{argmax}_\pi \mathbb{E}[Z_{s,a}]\ \forall\ (s, a) \in \mathcal{S} \times \mathcal{A}$. The state-action value function for such an optimal policy is denoted by Q^{π^*}. The main difficulty lies in the agent not knowing R and T, as well as having a limited interaction with the environment. Throughout this paper, we focus on model-free algorithms which learn the Q-values without learning R and T.

2.2 Exploration

The strategy that the agent uses to select actions is critical for its performance, since an appropriate amount of initial exploration is necessary to learn a good policy. Directed exploration is typically achieved by selecting actions based on their current value estimates with an additional exploration bonus. For instance, upper confidence bound (Auer, 2002) is a popular count-based heuristic which estimates an upper bound on the action values given the number of sampled experiences.

On the other hand, if one had access to the full probability distribution over action values, $p(Q)$, one could calculate a more informative exploration bonus. Here, we consider an exploration bonus based on the *value of perfect information* (Howard, 1966; Dearden et al., 1998).

2.3 Value of Perfect Information

The utility of exploration can be formalised with the notion of value of perfect information (VPI) which quantifies how much more reward one expects to accrue if one were to change one's policy in the light of learning the true value, $q^*(a)$, associated with the outcome of an action. The gain from

[1] We use notation consistent with the distributional RL literature.

learning the true value is defined as:

$$Gain_{s,a}(q^*(a)) = \begin{cases} \mathbb{E}[Q(s,a_2)] - q^*(a) & \text{if } a = a_1 \text{ and } q^*(a) < \mathbb{E}[Q(s,a_2)] \\ q^*(a) - \mathbb{E}[Q(s,a_1)] & \text{if } a \neq a_1 \text{ and } q^*(a) > \mathbb{E}[Q(s,a_1)] \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

where a_1 is the action currently estimated to be best, and a_2 is the action currently estimated to be second-best. Of course, the agent has to calculate the gain (2.1) *before* finding out the true value $q^*(a)$; it does this by calculating the expected gain under its current belief. The value of perfect information is thus $VPI(s,a) = \int_{-\infty}^{\infty} Gain_{s,a}(x)p(Q(s,a) = x)dx$. The myopic VPI exploratory policy at state s is therefore to select an action as $a = \text{argmax}_b \left[\mathbb{E}[Q(s,b)] + VPI(s,b)\right]$.

2.4 Bayesian Q-learning

Dearden et al. (1998) proposed a Bayesian variant of Q-learning (Watkins & Dayan, 1992) which maintains uncertainty about the return distribution associated with taking an action at a state. Bayesian Q-learning (BQL) assumes that the return has a Gaussian distribution with an unknown mean and precision; that is, $Z_{s,a} \sim \mathcal{N}(\mu_{s,a}, \tau_{s,a}^{-1})$. The bivariate normal-gamma distribution is a conjugate prior for the Gaussian likelihood implied by this assumption, and therefore BQL models its prior belief as $p(\mu_{s,a}, \tau_{s,a}) = NG(\mu_{s,a}, \alpha_{s,a}, \beta_{s,a}, \lambda_{s,a})$. We can see that $\mu_{s,a} = \mathbb{E}[Z_{s,a}] = Q(s,a)$, and therefore the marginal distribution $p(\mu_{s,a})$ corresponds to epistemic uncertainty about action values. The choice of representation makes it possible to derive VPI for BQL in closed form.[2]

BQL updates its prior belief with every experience, $\langle s, a, r, s' \rangle$, sampled from the environment. Notice that the belief specifies a distribution over the full return, whereas in an online setting the agent only has access to local rewards and its belief at the next state. Dearden et al. (1998) define the mixture posterior $p_{mix}(\mu_{s,a}, \tau_{s,a}) = \int_{-\infty}^{\infty} p(\mu_{s,a}, \tau_{s,a} \mid r + \gamma x)p(Z_{s',a'} = x)dx$, where $a' = \text{argmax}_b \mathbb{E}[Q_{s',b}]$. Since the mixture posterior is not a member of the normal-gamma family, it is approximated by a projection which corresponds to the closest normal-gamma distribution in Kullback-Leibler (KL) divergence. Projections are, of course, a common thread in distributional RL.

2.5 Distributional reinforcement learning

Instead of learning the expected return, distributional RL algorithms (Bellemare et al., 2023) also attempt to learn the entire distribution of the random return, which we denote by $\eta(s,a) = \mathcal{D}(Z_{s,a})$. Various algorithms have been proposed, which differ most in the way they represent the return distribution. In this paper, we focus on the categorical (Bellemare et al., 2017) and quantile representations (Dabney et al., 2018). Put crudely, regular distributional RL maintains and updates parameters associated with these representations. Our form of Bayesian distributional RL maintains and updates simple *distributions* over these parameters, and thereby captures epistemic uncertainty. Through appropriate learning, the uncertainty about these parameters vanishes, leaving certain knowledge of the return distributions.

2.5.1 Categorical representation

The categorical representation for each state-action pair (s,a) consists of a finite set of M regularly-spaced locations (or atoms) and the probabilities associated with each location: $\{(\theta_1^{s,a}, p_1^{s,a}), ..., (\theta_M^{s,a}, p_M^{s,a})\}$.[3] The return distribution is then a weighted mixture of Dirac deltas at the represented locations, written as $\eta(s,a) = \sum_{m=1}^{M} p_m^{s,a}\delta_{\theta_m^{s,a}}$ where $0 \leq p_m^{s,a} \leq 1$ and $\sum_{m=1}^{M} p_m^{s,a} = 1$.

The true return, however, need not be categorical. This requires an additional projection step to find the closest approximation within the representational class to the required return distribution.

[2]Note that our result, with derivations appearing in Appendix A, and consistent with extensive simulations, slightly differs from that reported by Dearden et al. (1998).

[3]The superscripts will occasionally be omitted for brevity.

For a probability atom at g, the categorical projection operator, Π_C, is defined as

$$\Pi_C(\delta_g) = \begin{cases} \delta_{\theta_1} & \text{if } g \leq \theta_1 \\ \frac{\theta_{m+1}-g}{\theta_{m+1}-\theta_m}\delta_{\theta_m} + \frac{g-\theta_m}{\theta_{m+1}-\theta_m}\delta_{\theta_{m+1}} & \text{if } \theta_m < g < \theta_{m+1} \\ \delta_{\theta_M} & \text{if } g \geq \theta_M \end{cases} \tag{2.2}$$

In response to the sample experience $\langle s, a, r, s' \rangle$, the categorical Q-learning (CatQL) algorithm updates its return distribution estimate as:

$$\eta(s,a) \leftarrow \eta(s,a) + \alpha \left[\sum_{m=1}^{M} p_m^{s,a} \Pi_C \left(\delta_{r+\gamma\theta_m^{s',a'}} \right) - \eta(s,a) \right]$$

where $a' = \text{argmax}_b \, \mathbb{E}[Z_{s',b}]$ and $\alpha \in [0,1]$ is a learning rate parameter.

2.5.2 Quantile representation

The quantile representation consists of a finite set of M evenly spaced quantile levels, $\tau_m = \frac{2m-1}{2M}$, and the corresponding quantiles: $\{(\theta_1^{s,a}, \tau_1^{s,a}), ..., (\theta_M^{s,a}, \tau_M^{s,a})\}$. In general, the $\tau^{\text{th}} \in (0,1)$ quantile of a random variable Z is defined as $\inf\{z \in \mathbb{R} : F_Z(z) \geq \tau\}$ where F_Z is the distribution function of Z. Moreover, with the uniform spacing of quantile levels, the return distribution is a uniform mixture of quantiles, written as $\eta(s,a) = \frac{1}{M}\sum_{m=1}^{M} \delta_{\theta_m^{s,a}}$.

An equivalent of CatQL can be derived for the quantile representation, which we abbreviate as QQL. We use the quantile projection introduced by Dabney et al. (2018), which finds a quantile representation which minimises the 1-Wasserstein distance to an arbitrary return distribution.

3 Bayesian Categorical Q-learning

In this section we present our algorithm, Bayesian Categorical Q-learning (BCQL), which extends the idea of maintaining epistemic uncertainty whilst learning a return distribution using a more flexible, categorical return representation.

3.1 Dirichlet prior for the categorical return

BCQL uses the categorical return representation described in 2.5.1. That is, the support locations $\{\theta_1^{s,a}, ..., \theta_M^{s,a}\}$ are fixed and the problem is to learn the associated return probabilities $\{p_1^{s,a}, ..., p_M^{s,a}\}$. Given the categorical representation, a Dirichlet distribution over the return probabilities is a natural choice for capturing epistemic uncertainty about the return distribution. Thus, BCQL's prior belief state for each state-action pair is $p(p_1^{s,a}, ..., p_M^{s,a}) = Dir(\alpha_1^{s,a}, ..., \alpha_M^{s,a})$. Together with the fixed locations, we write BCQL's representation as $\{(\theta_1^{s,a}, \alpha_1^{s,a}), ..., (\theta_M^{s,a}, \alpha_M^{s,a})\}$.

3.2 Epistemic uncertainty in the expected return

The prior belief of BCQL corresponds to uncertainty over the return probabilities, which in turn implies epistemic uncertainty in the expected return. Unfortunately, there is no closed-form expression for the distribution of the expected return. In practice, we approximate it with a Gaussian centred at the expected return, $\sum_{m=1}^{M} \theta_m \mathbb{E}[P_m]$, and whose variance is given by $\sum_{m=1}^{M}\sum_{n=1}^{M} \theta_m\theta_n Cov(P_m, P_n)$.

Note that although BCQL approximates its epistemic uncertainty with a Gaussian distribution, its underlying return representation is more flexible than BQL (Dearden et al., 1998), allowing it to learn a better approximation to the true return distribution and thus a better approximation of the epistemic uncertainty. Furthermore, with this approximation, the expression for VPI can be reduced to closed form.

Proposition 1. *Assuming that the Bayesian Categorical Q-learning algorithm approximates its epistemic uncertainty as $Q(s,a) \sim \mathcal{N}(\mathbb{E}[Q(s,a)], Var[Q(s,a)])$, $VPI(s,a)$ is equal to*

$$VPI(s,a) = \begin{cases} (\mathbb{E}[Q(s,a_2)] - \mathbb{E}[Q(s,a)])p(Q(s,a) < \mathbb{E}[Q(s,a_2)]) + c(a_2) & \text{if } a = a_1 \\ (\mathbb{E}[Q(s,a)] - \mathbb{E}[Q(s,a_1)])p(Q(s,a) > \mathbb{E}[Q(s,a_1)]) + c(a_1) & \text{if } a \neq a_1 \end{cases}$$

where $c(b) = \sqrt{Var[Q(s,a)]}p(Q(s,a) = \mathbb{E}[Q(s,b)])$.

In Fig E1 we present simulation results which show that the quality of such an approximation improves as the number of categories, M, increases.

3.3 Distributional updates

BCQL faces a similar problem to BQL: namely, the prior belief is specified for the full return, but the agent only ever observes local sample rewards and its own beliefs. We approach it in the same way as Dearden et al. (1998) by defining a mixture posterior resulting from the possible posterior distributions under the current belief over the corresponding return realisations. Moreover, because of the discrete categorical representation, BCQL's mixture posterior is a sum rather than an integral:

$$p_{mix}(p_1^{s,a}, ..., p_M^{s,a}) = \sum_{m=1}^{M} p\left(p_1^{s,a}, ..., p_M^{s,a} \mid \Pi_C\left(\delta_{r+\gamma\theta_m^{s',a'}}\right)\right) \mathbb{E}[p(Z_{s',a'} = \theta_m^{s',a'})] \tag{3.1}$$

where Π_C is the categorical projection defined in (2.2).

Since the mixture posterior (3.1) does not belong to the Dirichlet family, we find the best Dirichlet approximation which minimises the KL divergence to the true mixture posterior distribution.

Theorem 1. *Let $P_1, ..., P_M$ be jointly distributed random probabilities with a density measure p. Then, the parameters of the Dirichlet distribution $q(p_1, ..., p_M) = Dir(\alpha_1, ..., \alpha_M)$ which minimise the KL divergence $D_{KL}(p||q)$ are given by:*

$$\alpha_m = \psi^{-1}\left(\psi\left(\sum_{i=1}^{M} \alpha_i\right) + \mathbb{E}_p[\log P_m]\right)$$

where $\psi(x) = \frac{\Gamma'(x)}{\Gamma(x)}$ is the digamma function and $\psi^{-1}(y)$ is its inverse.

Two caveats arise which are in essence identical to those in BQL (Dearden et al., 1998). The first is that Theorem 1 does not give a closed-form solution for the parameters of the approximating distribution. Minka (2000) derived an iterative procedure for a numerical solution, which we employed in our simulations. The second is that one has to calculate expectations of $\log P_m$ with respect to the mixture posterior p_{mix}, which requires another approximation by instead using the sufficient statistics of the projected Dirichlet distribution, which take a particularly simple form. We provide a pseudocode for BCQL's distributional updates in Algorithm C1, which details all the necessary approximations involved. Moreover, we demonstrate empirically that BCQL's belief state converges appropriately to a close approximation to the true return distribution in a simple evaluation environment (Fig E2).

4 Results

To compare learning and exploration capacities of the distributional agents, we performed simulations in a modified version of the loop environment (Watkins, 1989; Dearden et al., 1998) shown in Figure 1. Here, all transitions are deterministic, and what makes this environment challenging for exploration is that along the left loop, one of the available actions in each state brings the agent back to the start state with zero reward. This means that insufficient exploration would likely result in suboptimal policies favouring the right loop even when its expected reward is lower.

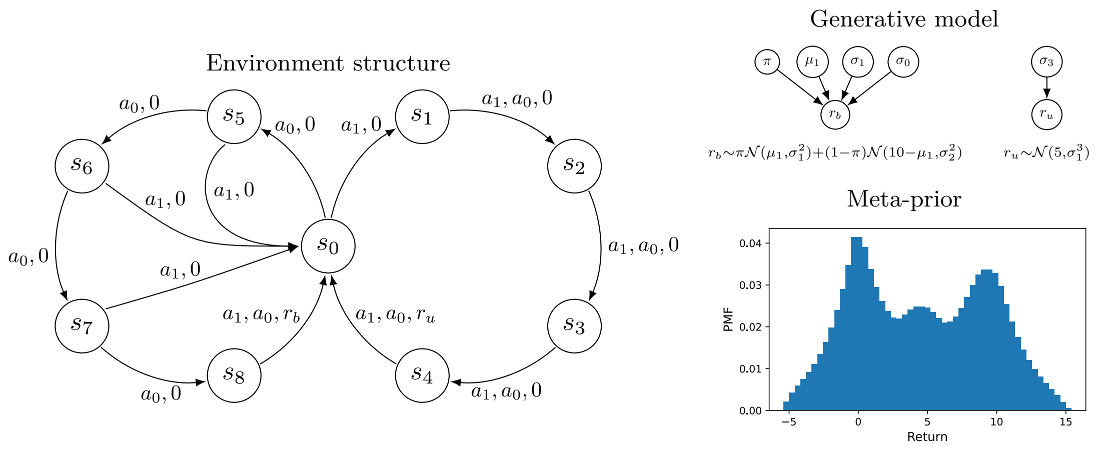


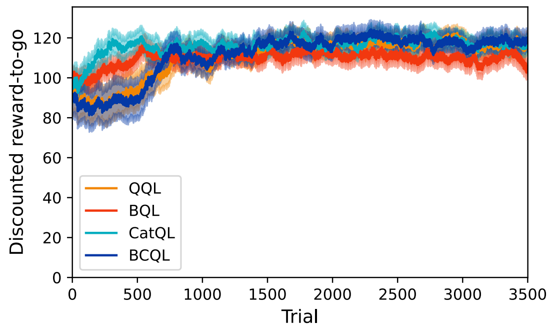
Figure 1: Modified loop environment from Watkins (1989). The starting state is s_0. In the right loop, both actions a_0 and a_1 in each state deterministically transition the agent towards the starting state. In the left loop, the transitions are also deterministic but the two actions in each state can either progress the agent further along the loop (a_0) or bring it back to the start state (a_1). The reward for each action is 0 except for the actions from states s_4 and s_8. For the former, r_u is sampled from a unimodal Gaussian and for the latter r_b is sampled from a bimodal Gaussian. The generative model of the environment is shown to the right of the transition structure. The prior was $\pi \sim \text{Beta}(1, 1)$, $\mu_1 \sim \text{Uniform}([0, 10])$, $\sigma_i \sim \text{Gamma}(3)$ for $i = 1, ..., 3$. In addition, we show the meta-prior for the optimal return distribution in this environment, accounting for the possible realisations of the generative model.

We introduced a number of modifications to this environment to make it even more challenging, and to take full advantage of the representational capacity of distributional agents. First, the reward at the end of the left loop followed a bimodal Gaussian distribution, and the reward at the end of the right loop had a unimodal Gaussian distribution (Fig 1). Second, we noticed that the agents could cheat by choosing high learning rates in the cases where the expected reward from the bimodal loop was lower and effectively collapsing their distributional estimates at occasional samples from the lower mode. To avoid this, we specified a prior distribution over the parameters of the two reward functions which included their means, variances, and mixing coefficient for the bimodal reward function (Fig 1). This resulted in environment samples for which such a collapse would have been catastrophic, and thus enforced more gradual learning. The free parameters of all agents were optimised on environments sampled from this distribution, and the performance was subsequently evaluated on unseen environment samples from the same distribution.

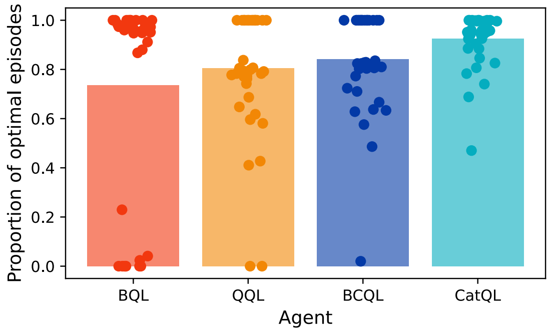
The prior return distributions for all agents were initialised to the average true optimal return distribution under the specified prior over the environments (which we refer to as the 'meta-prior'; Fig 1). This was done to reduce the number of free parameters to optimise, as well as to make a fair comparison of the learning process and eliminate confounds associated with lucky prior initialisations.

We report a number of performance measures in Fig 2. These include reward-based measures, such as i) discounted reward-to-go until the end of the run, calculated for each trial t as $\sum_{i=t}^{T} \gamma^{i-t} r_i$ where $T = 4000$ (Fig 2A); and ii) total reward collected in the first 2000 and second 2000 trials (Fig 2C). Because of the high variance of reward distributions in the sampled environments, these reward-based measures were also highly variable (Fig 2A,C). Thus, we additionally report a performance measure based on the fraction of trials in which the agents executed the optimal sequence of actions from the start state (Fig 2B,D). According to this measure, BCQL and CatQL agents scored highest in their ability to discover optimal paths (Fig 2B,D).

A



B



C

Agent	1st phase Mean	Std	2nd phase Mean	Std
QQL	2090.4	738.8	2353.5	532.3
BQL	2178.0	432.9	2212.0	446.2
CatQL	**2276.8**	591.4	2360.1	507.1
BCQL	2093.5	735.4	**2377.9**	512.5

D

Agent	Proportion Mean	Std
QQL	0.80	0.25
BQL	0.74	0.41
CatQL	**0.93**	0.11
BCQL	0.84	0.20

Figure 2: Exploration performance. A) Discounted reward-to-go for each agent averaged over 40 simulations. Each agent was simulated in the same 40 environment samples. The shaded areas show standard error of the mean. B) Proportion of optimal episodes executed by each agent. The individual dots show the total fraction of times an agent executed the optimal trajectory from the start state in that environment sample. The height of the bar indicates the average fraction over the same 40 simulations. C) Total reward collected by each agent in the first and second 2000 trials, averaged over the same 40 simulations. D) Summary of C).

The performance of CatQL and BCQL did not differ substantially, most likely because our modified loop environment was insufficiently challenging. To gain a deeper insight into the learning processes, we examined the underlying learning by simulating all agents on the same stream of experience in the same environment sample (Fig 3). This revealed some of the pathological modes of learning in the non-Bayesian distributional algorithms which we had expected would affect their performance (Fig 3). First, due to the non-adaptive nature of their learning rates, we observed large fluctuations in the distributional estimates in response to sample experiences (Fig 3A). This concerned both the estimated mean (Fig 3A; left), VPI (Fig 3B), and variance of the expected return distribution (Fig 3; right). BCQL and BQL did not suffer from such fluctuations since, being Bayesian algorithms, they are able to adapt their learning rates optimally.

Second, the non-Bayesian distributional algorithms consistently overestimated VPI (Fig 3A; middle) because of the irreducible aleatoric uncertainty in the return distribution (Fig 3B,C). By contrast, BCQL and BQL correctly calculated VPI based on their epistemic uncertainty which reduced with experience, as reflected in the vanishing values of VPI (Fig 3A; middle).

Lastly, BQL learnt the worst approximation to the true return distribution (Fig 3B,C) because of its assumption that the return had a Gaussian distribution, whereas it was actually bimodal (Fig 3C; bottom). On the other hand, BCQL correctly learnt the bimodal structure of the return distribution (Fig 3B,C), with the learning being most stable across all the other algorithms. All of these observations are even more apparent when one considers the immediately rewarding action in state s_8 (Fig E3).

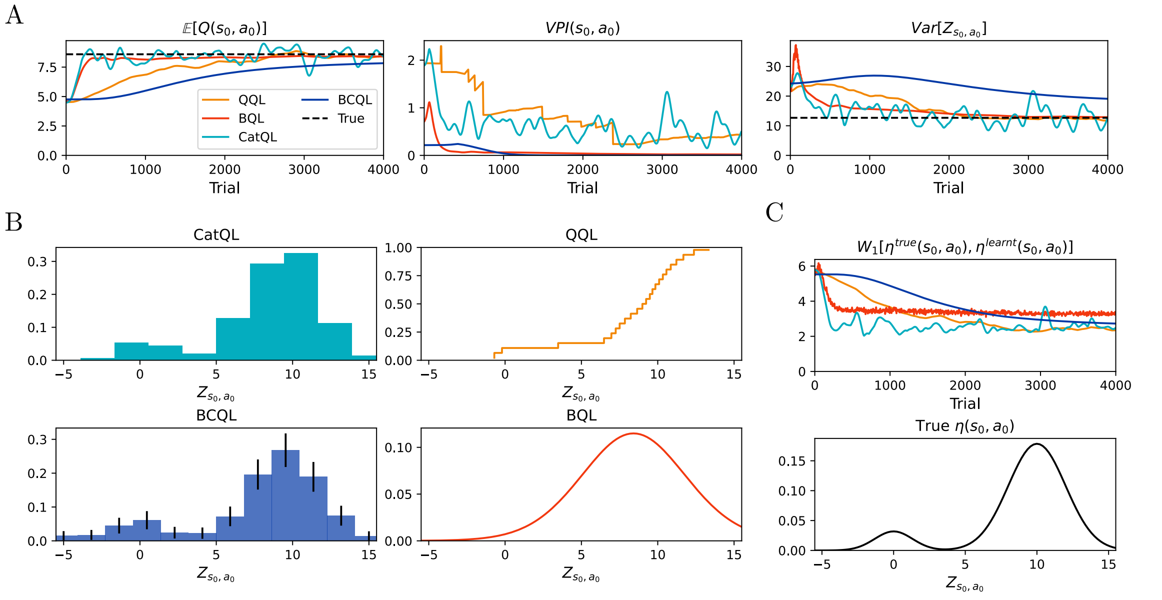


Figure 3: Learning distributional representations. A) Left: expected Q-values at the start state s_0 for the optimal action a_0 as estimated by each agent. The black dotted line denotes the true optimal Q-value. Middle: Value of perfect information for the same action. Notice how VPI for the Bayesian agents drops to nearly zero, whereas for the other agents it keeps fluctuating at relatively high values. Right: variance of the estimated expected return distribution. B) Final distributions learnt by the agents for the same action at the end of the run. The solid black lines for the BCQL agent indicate standard deviation (i.e., epistemic uncertainty) for each return category. C) Top: Wasserstein distance between the estimated expected return distribution and the true return distribution (bottom) over the course of learning.

5 Discussion

We present a novel Bayesian distributional RL algorithm, Bayesian Categorical Q-learning. BCQL improves on the seminal Bayesian Q-learning algorithm (Dearden et al., 1998) by relaxing the assumption that the return distribution is Gaussian. This is indeed often violated. Instead, BCQL represents the return distribution using a more flexible, categorical representation, which allows it to learn a better approximation to the true return distribution. Moreover, BCQL is significantly faster than BQL because its return representation assumes discrete support which replaces expensive numerical integration with finite sums.

The main difference between our Bayesian approach and regular distributional RL (Bellemare et al., 2023) is the notion of subjective, or epistemic, uncertainty, captured by the prior belief about the return distribution. BCQL models its epistemic uncertainty using a Dirichlet prior over return probabilities, which it updates incrementally with every experience by approximating optimal Bayesian updates. This uncertainty is exactly what model-free algorithms should exploit to arrange for efficient exploration, insofar as it provides rich signals for learning opportunities.

We used a challenging exploration task to compare the exploration abilities of BCQL and BQL to two regular distributional RL algorithms which used categorical and quantile return representations. The Bayesian agents were significantly more competent learners, manifested in their ability to adapt their learning rates optimally as a function of their uncertainty about the learnt distributions. As a result, the learning of distributional representations was smoother and more stable. One could equally imagine a decaying learning rate schedule for the regular distributional RL algorithms, which is in fact needed for convergence; however, such a schedule would need to be heuristically fine-tuned to each environment.

Prior work has suggested the use of distributional information for exploration (Tang & Agrawal, 2018; Mavrin et al., 2019), albeit without exploiting the full richness of learnt distributions and still relying on count-based heuristics. Here, we operationalised exploration by means of value of perfect information (Howard, 1966; Dearden et al., 1998). We showed that the Bayesian agents correctly calculated VPI. This allowed them to learn efficiently up to the point when all subjective uncertainty was reduced to certainty, yielding policies which could be efficiently exploited. By contrast, regular distributional RL algorithms overestimated VPI because of the objective, irreducible uncertainty in their learnt distributional representations. Thus the resulting policies tended to favour exploration that continued even after the optimal paths had been discovered.

The myopic nature of VPI means that it does not explicitly account for the statistical interactions between the whole collection of state-action pairs. However, the propagation of subjective uncertainty via experience and learning does implicitly capture at least a part of this dependency structure. Extending our methods to DYNA-like architectures (Sutton, 1991) could be a powerful means of arranging for efficient presbyopic exploration.

Distributional RL has uncovered powerful algorithms for learning rich representations of uncertainty useful in a wide array of applications including, for instance, risk-sensitive control (Gagne & Dayan, 2021; Shen & Dayan, 2024). Combined with general function approximators, these have furthermore shown substantial performance gains on challenging control tasks relative to classic RL methods (Lyle et al., 2019). In this work, we have examined the use of distributional information for exploration, and argued that capturing a different source of uncertainty, which is subjective in nature, is critical for efficient and stable learning.

References

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

Necdet Batir. Inequalities for the inverses of the polygamma functions. *arXiv preprint arXiv:1705.06547*, 2017.

Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International conference on machine learning*, pp. 449–458. PMLR, 2017.

Marc G Bellemare, Will Dabney, and Mark Rowland. *Distributional reinforcement learning*. MIT Press, 2023.

Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Richard Dearden, Nir Friedman, and Stuart Russell. Bayesian q-learning. *Aaai/iaai*, 1998:761–768, 1998.

Marc Peter Deisenroth, Carl Edward Rasmussen, and Jan Peters. Gaussian process dynamic programming. *Neurocomputing*, 72(7-9):1508–1524, 2009.

Christopher Gagne and Peter Dayan. Two steps to risk sensitivity. *Advances in Neural Information Processing Systems*, 34:22209–22220, 2021.

John C Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 41(2):148–164, 1979.

Arthur Guez, David Silver, and Peter Dayan. Efficient bayes-adaptive reinforcement learning using sample-based search. *Advances in neural information processing systems*, 25, 2012.

Ronald A Howard. Information value theory. *IEEE Transactions on systems science and cybernetics*, 2(1):22–26, 1966.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

Clare Lyle, Marc G Bellemare, and Pablo Samuel Castro. A comparative analysis of expected and distributional reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4504–4511, 2019.

Borislav Mavrin, Hengshuai Yao, Linglong Kong, Kaiwen Wu, and Yaoliang Yu. Distributional reinforcement learning for efficient exploration. In *International conference on machine learning*, pp. 4424–4434. PMLR, 2019.

Thomas Minka. Estimating a dirichlet distribution, 2000.

Tingke Shen and Peter Dayan. Risking your tail: Modeling individual differences in risk-sensitive exploration using conditional value at risk and bayes adaptive markov decision processes. *bioRxiv*, pp. 2024–01, 2024.

David Silver and Joel Veness. Monte-carlo planning in large pomdps. *Advances in neural information processing systems*, 23, 2010.

Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

Yunhao Tang and Shipra Agrawal. Exploration by distributional reinforcement learning. *arXiv preprint arXiv:1805.01907*, 2018.

Christopher JCH Watkins. Learning from delayed rewards. 1989.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

A Derivations

In this Appendix, we provide some useful derivations which will be used in supplying proofs. Moreover, we show how VPI was calculated for the non-Bayesian agents by expressing it in terms of their representations (since the non-Bayesian agents incorrectly used the objective uncertainty in the return distribution).

A.1 VPI

We first repeat the definition of the Value of Perfect Information. The gain from learning the true value is defined as:

$$Gain_{s,a}(q^*(a)) = \begin{cases} \mathbb{E}[Q(s,a_2)] - q^*(a) & \text{if } a = a_1 \text{ and } q^*(a) < \mathbb{E}[Q(s,a_2)] \\ q^*(a) - \mathbb{E}[Q(s,a_1)] & \text{if } a \neq a_1 \text{ and } q^*(a) > \mathbb{E}[Q(s,a_1)] \\ 0 & \text{otherwise} \end{cases}$$

where a_1 is the action currently estimated to be best, and a_2 is the action currently estimated to be second-best. The value of perfect information is thus $VPI(s,a) = \int_{-\infty}^{\infty} Gain_{s,a}(x) p(Q(s,a) = x) dx$ which can be written out for each case individually:

$$VPI(s,a) = \begin{cases} \mathbb{E}[Q(s,a_2)] p(Q(s,a) < \mathbb{E}[Q(s,a_2)]) - \int_{-\infty}^{\mathbb{E}[Q(s,a_2)]} x p(Q(s,a) = x) dx & \text{if } a = a_1 \\ \int_{\mathbb{E}[Q(s,a_1)]}^{\infty} x p(Q(s,a) = x) dx - \mathbb{E}[Q(s,a_1)] p(Q(s,a) > \mathbb{E}[Q(s,a_1)]) & \text{if } a \neq a_1 \end{cases} \quad \text{(A.1)}$$

A.2 VPI for the Quantile representation

Consider the case that $a = a_1$. We can re-write the integral which appears in the definition of VPI by using the following fact:

$$\int_{-\infty}^{\mathbb{E}[Q(s,a_2)]} p(Q(s,a) < x)dx = [xp(Q(s,a) < x)]_{-\infty}^{\mathbb{E}[Q(s,a_2)]} - \int_{-\infty}^{\mathbb{E}[Q(s,a_2)]} xp(Q(s,a) = x)dx$$
$$= \mathbb{E}[Q(s,a_2)]p(Q(s,a) < \mathbb{E}[Q(s,a_2)]) - \int_{-\infty}^{\mathbb{E}[Q(s,a_2)]} xp(Q(s,a) = x)dx$$

And therefore we get:

$$\int_{-\infty}^{\mathbb{E}[Q(s,a_2)]} xp(Q(s,a) = x)dx = \mathbb{E}[Q(s,a_2)]p(Q(s,a) < \mathbb{E}[Q(s,a_2)]) - \int_{-\infty}^{\mathbb{E}[Q(s,a_2)]} p(Q(s,a) < x)dx$$

Plugging this into the expression for VPI we obtain:

$$VPI(s,a) = \int_{-\infty}^{\mathbb{E}[Q(s,a_2)]} p(Q(s,a) < x)dx$$

The calculations for the case when $a \neq a_1$ are similar. Overall, we can express VPI as:

$$VPI(s,a) = \begin{cases} \int_{-\infty}^{\mathbb{E}[Q(s,a_2)]} p(Q(s,a) < x)dx & \text{if } a = a_1 \\ \int_{\mathbb{E}[Q(s,a_1)]}^{\infty} p(Q(s,a) > x)dx & \text{if } a \neq a_1 \end{cases}$$

We use this expression to approximate VPI for the QQL agent numerically, since it learns the inverse of the distribution function.

A.3 VPI for the Categorical representation

Since the categorical representation has a discrete support, VPI for the CQL agent was calculated as:

$$VPI(s,a) = \sum_{m=1}^{M} Gain_{s,a}(\theta_m(s,a))p_m(s,a)$$

A.4 VPI for Bayesian Q-learning

Lemma 3.2 from Dearden et al. (1998) establishes that, if $p(\mu,\tau) = NG(\mu_0,\lambda,\alpha,\beta)$, then the marginal density of μ is:

$$p(\mu) = \sqrt{\frac{\lambda}{2\pi}}\beta^{\alpha}\frac{\Gamma(\alpha+\frac{1}{2})}{\Gamma(\alpha)}\left(\beta + \frac{1}{2}\lambda(\mu-\mu_0)^2\right)^{-(\alpha+\frac{1}{2})}$$

and the marginal distribution of μ is:

$$p(\mu < x) = T((x-\mu_0)\sqrt{\frac{\lambda\alpha}{\beta}} : 2\alpha)$$

where $T(x : d)$ is the cumulative t-distribution with d degrees of freedom:

$$T(x : d) = \frac{\Gamma\left(\frac{d+1}{2}\right)}{\Gamma\left(\frac{d}{2}\right)\sqrt{d\pi}}\int_{-\infty}^{x}\left(1+\frac{y^2}{d}\right)^{-\frac{d+1}{2}} dy$$

We consider the case when $a = a_1$ and focus on the integral in (A.1). By plugging in the marginal density, we can write:

$$\int_{-\infty}^{\mathbb{E}[Q(s,a_2)]} xp(Q(s,a) = x)dx = \sqrt{\frac{\lambda}{2\pi}}\beta^{\alpha}\frac{\Gamma(\alpha+\frac{1}{2})}{\Gamma(\alpha)}\int_{-\infty}^{\mathbb{E}[Q(s,a_2)]} x\left(\beta+\frac{1}{2}\lambda(x-\mathbb{E}[Q(s,a)])^2\right)^{-(\alpha+\frac{1}{2})} dx$$

Next, we make the substitution $u(x) = (x - \mathbb{E}[Q(s,a)])\sqrt{\frac{\lambda\alpha}{\beta}}$, which leads to:

$$\int_{-\infty}^{\mathbb{E}[Q(s,a_2)]} xp(Q(s,a)=x)dx = \sqrt{\frac{\lambda}{2\pi}}\beta^{\alpha}\frac{\Gamma(\alpha+\frac{1}{2})}{\Gamma(\alpha)}\int_{-\infty}^{u(\mathbb{E}[Q(s,a_2)])} u\left(\beta+\frac{1}{2}\lambda\left(\frac{u^2\beta}{\lambda\alpha}\right)\right)^{-(\alpha+\frac{1}{2})}\sqrt{\frac{\beta}{\lambda\alpha}}du$$

$$= \frac{\Gamma(\alpha+\frac{1}{2})}{\Gamma(\alpha)\sqrt{2\pi\alpha}}\int_{-\infty}^{u(\mathbb{E}[Q(s,a_2)])}\left(u\sqrt{\frac{\beta}{\lambda\alpha}}+\mathbb{E}[Q(s,a)]\right)\left(1+\frac{u^2}{2\alpha}\right)^{-(\alpha+\frac{1}{2})}du$$

$$= \frac{\Gamma(\alpha+\frac{1}{2})}{\Gamma(\alpha)\sqrt{2\pi\alpha}}\int_{-\infty}^{u(\mathbb{E}[Q(s,a_2)])} u\sqrt{\frac{\beta}{\lambda\alpha}}\left(1+\frac{u^2}{2\alpha}\right)^{-(\alpha+\frac{1}{2})}du$$
$$+\mathbb{E}[Q(s,a)]\frac{\Gamma(\alpha+\frac{1}{2})}{\Gamma(\alpha)\sqrt{2\pi\alpha}}\int_{-\infty}^{u(\mathbb{E}[Q(s,a_2)])}\left(1+\frac{u^2}{2\alpha}\right)^{-(\alpha+\frac{1}{2})}du$$

In the last line we recognise the distribution function $p(Q(s,a) < \mathbb{E}[Q(s,a_2)])$. As for the other integral, we again solve it by substitution, this time using $h(u) = 1 + \frac{u^2}{2\alpha}$:

$$\frac{\Gamma(\alpha+\frac{1}{2})\sqrt{\beta}}{\Gamma(\alpha)\alpha\sqrt{2\pi\lambda}}\int_{-\infty}^{u(\mathbb{E}[Q(s,a_2)])} u\left(1+\frac{u^2}{2\alpha}\right)^{-(\alpha+\frac{1}{2})}du = \frac{\Gamma(\alpha+\frac{1}{2})\sqrt{\beta}}{\Gamma(\alpha)\alpha\sqrt{2\pi\lambda}}\int_{-\infty}^{h(u(\mathbb{E}[Q(s,a_2)]))}\alpha h^{-(\alpha+\frac{1}{2})}dh$$
$$= \frac{\Gamma(\alpha+\frac{1}{2})\sqrt{\beta}}{\Gamma(\alpha)\alpha\sqrt{2\pi\lambda}}\left[\frac{\alpha h^{\frac{1}{2}-\alpha}}{\frac{1}{2}-\alpha}\right]_{-\infty}^{h(u(\mathbb{E}[Q(s,a_2)]))}$$
$$= -\frac{\Gamma(\alpha+\frac{1}{2})\sqrt{\beta}}{(\alpha-\frac{1}{2})\Gamma(\alpha)\sqrt{2\pi\lambda}}\left[\left(1+\frac{u^2}{2\alpha}\right)^{\frac{1}{2}-\alpha}\right]_{-\infty}^{u(\mathbb{E}[Q(s,a_2)])}$$
$$= -\frac{\Gamma(\alpha+\frac{1}{2})\sqrt{\beta}}{(\alpha-\frac{1}{2})\Gamma(\alpha)\sqrt{2\pi\lambda}}\left(1+\frac{(\mathbb{E}[Q(s,a_2)]-\mathbb{E}[Q(s,a)])^2\frac{\lambda\alpha}{\beta}}{2\alpha}\right)^{\frac{1}{2}-\alpha}$$
$$= -\frac{\Gamma(\alpha+\frac{1}{2})\sqrt{\beta}}{(\alpha-\frac{1}{2})\Gamma(\alpha)\sqrt{2\pi\lambda}}\left(1+\frac{(\mathbb{E}[Q(s,a_2)]-\mathbb{E}[Q(s,a)])^2\lambda}{2\beta}\right)^{\frac{1}{2}-\alpha}$$

Performing similar algebraic manipulations for the case when $a \neq a_1$ gives the following final expression for VPI:

$$EVPI(s,a) = \begin{cases}(\mathbb{E}[Q(s,a_2)]-\mathbb{E}[Q(s,a)])p(Q(s,a)<\mathbb{E}[Q(s,a_2)])+c(a_2) & \text{if } a = a_1\\ (\mathbb{E}[Q(s,a)]-\mathbb{E}[Q(s,a_1)])p(Q(s,a)>\mathbb{E}[Q(s,a_1)])+c(a_1) & \text{if } a \neq a_1\end{cases}$$

where

$$c(b) = \frac{\Gamma(\alpha+\frac{1}{2})\sqrt{\beta}}{(\alpha-\frac{1}{2})\Gamma(\alpha)\Gamma(\frac{1}{2})\sqrt{2\lambda}}\left(1+\frac{(\mathbb{E}[Q(s,b)]-\mathbb{E}[Q(s,a)])^2\lambda}{2\beta}\right)^{\frac{1}{2}-\alpha}$$

B Proofs

Proposition 1. *Assuming that the Bayesian Categorical Q-learning algorithm approximates its epistemic uncertainty as $Q(s,a) \sim \mathcal{N}(\mathbb{E}[Q(s,a)], Var[Q(s,a)])$, $VPI(s,a)$ is equal to*

$$VPI(s,a) = \begin{cases}(\mathbb{E}[Q(s,a_2)]-\mathbb{E}[Q(s,a)])p(Q(s,a)<\mathbb{E}[Q(s,a_2)])+c(a_2) & \textit{if } a = a_1\\ (\mathbb{E}[Q(s,a)]-\mathbb{E}[Q(s,a_1)])p(Q(s,a)>\mathbb{E}[Q(s,a_1)])+c(a_1) & \textit{if } a \neq a_1\end{cases}$$

where $c(b) = \sqrt{Var[Q(s,a)]}p(Q(s,a) = \mathbb{E}[Q(s,b)])$.

Proof of Proposition 1. Consider the case where $a = a_1$ in (A.1) and focus on the integral. To ease the notation, let $\mu_{s,a} = \mathbb{E}[Q(s,a)]$ and $\sigma^2_{s,a} = Var[Q(s,a)]$ where the subscripts index a state and an action. Assuming the Gaussian approximation, the random expected return is $Q(s,a) \sim \mathcal{N}(\mu_{s,a}, \sigma^2_{s,a})$. We can re-write it as the following transformation:

$$Q(s,a) = \sigma_{s,a} Z + \mu_{s,a}$$

where Z is the standard normal variable with cdf $\Phi(z)$ and pdf $\phi(z)$. Furthermore, by letting $t = \frac{\mu_{s,a_2} - \mu_{s,a}}{\sigma_{s,a}}$, we can write the distribution of the random expected return $Q(s,a)$ in terms of the distribution of Z:

$$p(Q(s,a) \leq \mu_{s,a_2}) = p(Z \leq t) = \Phi(t)$$

Now consider the following conditional expectation:

$$\begin{aligned}
\mathbb{E}[Q(s,a) \mid Q(s,a) \leq \mu_{s,a_2}] =& \mathbb{E}[\sigma_{s,a} Z \mid Z \leq t] + \mathbb{E}[\mu_{s,a} \mid Z \leq t] \\
=& \frac{\sigma_{s,a} \int_{-\infty}^{t} z\phi(z)dz + \mu_{s,a} \int_{-\infty}^{t} \phi(z)dz}{\Phi(t)} \\
=& \frac{-\sigma_{s,a} \int_{-\infty}^{t} \phi'(z)dz + \mu_{s,a}\Phi(t)}{\Phi(t)} \\
=& \mu_{s,a} - \sigma_{s,a} \frac{\phi(t)}{\Phi(t)}
\end{aligned}$$

where the first equality is due to the linearity of conditional expectations, and in the third line we used the fact that $\phi(z) = -z\phi'(z)$. We can now write out the original integral using this expression for the conditional expectation:

$$\begin{aligned}
\int_{-\infty}^{\mathbb{E}[Q(s,a_2)]} xp(Q(s,a) = x)dx =& \mathbb{E}[Q(s,a) \mid Q(s,a) \leq \mu_{s,a_2}]p(Q(s,a) \leq \mu_{s,a_2}) \\
=& \left(\mu_{s,a} - \sigma_{s,a} \frac{\phi(t)}{\Phi(t)}\right) \Phi(t) \\
=& \mu_{s,a}\Phi(t) - \sigma_{s,a}\phi(t)
\end{aligned}$$

which, in terms of our original notation, can be written as

$$\int_{-\infty}^{\mathbb{E}[Q(s,a_2)]} xp(Q(s,a) = x)dx = \mathbb{E}[Q(s,a)]p(Q(s,a) \leq \mathbb{E}[Q(s,a_2)]) - \sqrt{Var[Q(s,a)]}p(Q(s,a) = \mathbb{E}[Q(s,a_2)])$$

The calculations for the case with $a \neq a_1$ are similar. Finally, this leads to:

$$VPI(s,a) = \begin{cases} (\mathbb{E}[Q(s,a_2)] - \mathbb{E}[Q(s,a)])p(Q(s,a) < \mathbb{E}[Q(s,a_2)]) + c(a_2) & \text{if } a = a_1 \\ (\mathbb{E}[Q(s,a)] - \mathbb{E}[Q(s,a_1)])p(Q(s,a) > \mathbb{E}[Q(s,a_1)]) + c(a_1) & \text{if } a \neq a_1 \end{cases}$$

where $c(y) = \sqrt{Var[Q(s,a)]}p(Q(s,a) = \mathbb{E}[Q(s,y)])$. □

Theorem 1. *Let $P_1, ..., P_M$ be jointly distributed random probabilities with a density measure p. Then, the parameters of the Dirichlet distribution $q(p_1, ..., p_M) = Dir(\alpha_1, ..., \alpha_M)$ which minimise the KL divergence $D_{KL}(p||q)$ are given by:*

$$\alpha_m = \psi^{-1}\left(\psi\left(\sum_{i=1}^{M} \alpha_i\right) + \mathbb{E}_p[\log p_m]\right)$$

where $\psi(x) = \frac{\Gamma'(x)}{\Gamma(x)}$ is the digamma function and $\psi^{-1}(y)$ is its inverse.

Proof of Theorem 1. The Kullback-Leibler (KL) divergence between two distributions p and q is defined as $D_{KL}(p||q) = \mathbb{E}_p[\log p/q]$. To minimise the divergence, we find the parameters of q so that $dD_{KL}(p||q)/dq = 0$. Notice further that $D_{KL}(p||q) = \mathbb{E}_p[\log p] - \mathbb{E}_p[\log q]$. The first term does not depend on q, and so we only need to differentiate $\mathbb{E}_p[\log q]$. Moreover, $d\{\mathbb{E}_p[\log q]\}/dq = \mathbb{E}_p[d\{\log q\}/dq]$, and so we can first differentiate the log-density and then take the expectation.

The Dirichlet density $q(p_1, ..., p_M) = Dir(\alpha_1, ..., \alpha_M)$ is defined as:

$$q(p_1, ..., p_M) = \frac{\Gamma\left(\sum_{i=1}^{M} \alpha_i\right)}{\prod_{i=1}^{M} \Gamma(\alpha_i)} \prod_{i=1}^{M} p_i^{\alpha_i - 1}$$

The log-density is therefore:

$$\log q(p_1, ..., p_M) = \log \Gamma\left(\sum_{i=1}^{M} \alpha_i\right) - \sum_{i=1}^{M} \log \Gamma(\alpha_i) + \sum_{i=1}^{M} (\alpha_i - 1) \log p_i$$

We can now take the partial derivatives to obtain:

$$\begin{aligned}\frac{\partial \log q(p_1, ..., p_M)}{\partial \alpha_m} &= \frac{\partial}{\partial \alpha_m} \log \Gamma\left(\sum_{i=1}^{M} \alpha_i\right) - \frac{\partial}{\partial \alpha_m} \sum_{i=1}^{M} \log \Gamma(\alpha_i) + \frac{\partial}{\partial \alpha_m} \sum_{i=1}^{M} (\alpha_i - 1) \log p_i \\ &= \frac{\partial}{\partial \alpha_m} \log \Gamma\left(\sum_{i=1}^{M} \alpha_i\right) - \frac{\partial}{\partial \alpha_m} \log \Gamma(\alpha_m) + \log p_m \\ &= \psi\left(\sum_{i=1}^{M} \alpha_i\right) - \psi(\alpha_m) + \log p_m\end{aligned}$$

where $\psi(x) = \frac{d \log \Gamma(x)}{dx}$ is the digamma function.

Setting the derivatives to zero and taking the expectation, we get

$$\alpha_m = \psi^{-1}\left(\psi\left(\sum_{i=1}^{M} \alpha_i\right) + \mathbb{E}_p[\log p_m]\right)$$

as required. □

C Pseudocode

The pseudocode for BCQL mixture posterior update appears in Algorithm C1.

Algorithm C1 Bayesian Categorical Q-learning (BCQL) mixture belief update

```
Input: Current state $s$, prior belief state of BCQL $\{(\theta_1^{s',a'}, \alpha_1^{s',a'}), ..., (\theta_M^{s',a'}, \alpha_M^{s',a'})\}$ for each $(s', a')$,
    discount factor $\gamma$, tolerance threshold $\epsilon$
 1: procedure BeliefUpdate($s, a, g$)
 2:     if $r \leq \theta_1$ then                                   ▷ If $g$ lies outside of the support
 3:         $\alpha_1^{s,a} \leftarrow \alpha_1^{s,a} + 1$
 4:     else if $r \geq \theta_M$ then
 5:         $\alpha_M^{s,a} \leftarrow \alpha_M^{s,a} + 1$
 6:     else                                        ▷ Dirichlet prior update with $\Pi_C$ projection
 7:         $i^* \leftarrow \text{argmax}_{i \in \{1,...,M\}} \{\theta_i : \theta_i \leq g\}$
 8:         $\zeta \leftarrow \frac{r - \theta_{i^*}}{\theta_{i^*+1} - \theta_{i^*}}$
 9:         $\alpha_{i^*}^{s,a} \leftarrow \alpha_{i^*}^{s,a} + 1 - \zeta$
10:         $\alpha_{i^*+1}^{s,a} \leftarrow \alpha_{i^*+1}^{s,a} + \zeta$
11:     return $\{\alpha_m^{s,a}\}_{m=1,...,M}$
12: procedure MixtureBeliefUpdate($s, a, r, s'$)
13:     if $s'$ == goal state then
14:         $\{\alpha_m^{s,a}\}_{m=1,...,M} \leftarrow$ BeliefUpdate($s, a, r$)
15:     else                                                           ▷ Mixture update
16:         $a' \leftarrow \text{argmax}_b \, \mathbb{E}[Q(s', b)]$
17:         $\mathbb{E}_p[\log P_m] \leftarrow 0$ for each $m = 1, ..., M$
18:         for each $i$ in $m = 1, ..., M$ do
19:             for each $j$ in $m = 1, ..., M$ do
20:                 $\{\alpha_m^{s,a}\}_{m=1,...,M} \leftarrow$ BeliefUpdate($s, a, r + \gamma\theta_j$)
21:                 $\mathbb{E}_p[\log P_i] \leftarrow \left[\psi(\alpha_i^{s,a}) - \psi\left(\sum_{k=1}^M \alpha_k^{s,a}\right)\right] \alpha_j^{s',a'} / \sum_{k=1}^M \alpha_k^{s',a'}$
22:     $\delta \leftarrow 1$
23:     $\alpha_{m,old}^{s,a} \leftarrow \alpha_m^{s,a}$ for each $m = 1, ..., M$
24:     $\alpha_{m,new}^{s,a} \leftarrow 0$ for each $m = 1, ..., M$
25:     while $\delta \geq \epsilon$ do
26:         for each $i$ in $m = 1, ..., M$ do
27:             $\alpha_{m,new}^{s,a} \leftarrow \psi^{-1}\left(\psi\left(\sum_{k=1}^M \alpha_{k,old}^{s,a}\right) + \mathbb{E}_p[\log P_m]\right)$
28:         $\delta \leftarrow \max_{m=1,...,M}(|\alpha_{m,new}^{s,a} - \alpha_{m,old}^{s,a}|)$
29:         $\alpha_{m,old}^{s,a} \leftarrow \alpha_{m,new}^{s,a}$ for each $m = 1, ..., M$
30:     for each $i$ in $m = 1, ..., M$ do
31:         if $\alpha_{m,old}^{s,a} < 1$ then
32:             $\alpha_{m,new}^{s,a} \leftarrow 1$
33:         else
34:             $\alpha_{m,new}^{s,a} \leftarrow \alpha_{m,old}^{s,a}$
35:     return $\{\alpha_{m,new}^{s,a}\}_{m=1,...,M}$
36: $a \leftarrow \text{argmax}_b \{\mathbb{E}[Q(s, b)] + VPI(s, b)\}$
37: Sample next state $s'$ and reward $r$
38: $\{\alpha_m^{s,a}\}_{m=1,...,M} \leftarrow$ MixtureBeliefUpdate($s, a, r, s'$)
```

The digamma inverse function ψ^{-1} in line 27 of Algorithm C1 was implemented with Newton's method (with a tolerance of 10^{-5}) using its analytical derivative. To spare computation and speed up convergence, recent evaluations of this inverse were memoized. For values which had not been memoized, the initial point was chosen as a lower bound on ψ^{-1} (Batir, 2017). The additional constraint on the Dirichlet hyperparameters in lines 30-34, namely $\alpha \geq 1$, was added to enforce the maximal uncertainty of the algorithm to correspond to the uniform distribution, and to counteract the approximation errors of the numerical procedures.

D Simulation Details

D.1 Parameter optimisation

The prior distributions for all algorithms were initialised as described in D.2. The remaining free parameters of all algorithms except BQL were optimised using the `gp_minimize` Bayesian optimisation routine from Python's `scikit-optimize` package. For each parameter sample, we ran each algorithm on 40 random environment samples, and took the average total reward collected as the maximisation objective.

The learning rates of the regular distributional RL algorithms were restricted to $\alpha \in (0,1)$. For CatQL and QQL we additionally optimised for the number of atoms and number of quantiles respectively, both of which were limited to the interval $[10, 50]$. The return support represented by CatQL was limited to the interval $[-5, 15]$, as described in D.2.

For BCQL, the number of atoms and the return support were limited to the same range as for CatQL. BCQL had an additional parameter, $N \in [1, 10]$, which controlled the initial amount of epistemic uncertainty about the prior return distribution. This was operationalised by multiplying the hyperparameters of the Dirichlet prior by this constant number.

BQL's parameters were optimised by performing grid search. We specified 40 regularly-spaced variance values for the algorithm's epistemic uncertainty about the expected return, which ranged from 0.1 to 10 (this corresponded to the same range of uncertainties as for BCQL). BQL's prior is specified by 4 parameters, $\mu_{s,a}, \lambda_{s,a}, \alpha_{s,a}, \beta_{s,a}$, where $\mu_{s,a}$ corresponds to the mean of the return distribution, which was always initilised to the mean of the meta-prior (D.2). The remaining parameters, $\lambda_{s,a}, \alpha_{s,a}, \beta_{s,a}$, were chosen by solving a minimisation problem such that the variance of the marginal return distribution, $Var[p(Z_{s,a})]$, where $p(Z_{s,a}) = \int p(Z_{s,a} \mid \mu_{s,a}, \tau_{s,a}) p(\mu_{s,a}, \tau_{s,a}) d\mu d\tau$ was as close as possible to the variance of the meta-prior. The minimisation was implemented using the `minimize` routine from Python's `SciPy` package.

The final optimised parameters for all algorithms are reported in Table D1. Due to space constraints, we only report optimised parameters for BQL for one state-action pair; however, the procedure described above was applied to all state-action pairs.

Agent	Parameter	Value
QQL	Number of quantiles	23
	α	0.57
CatQL	Number of atoms	10
	α	0.17
BCQL	Number of atoms	12
	N	1.02
BQL	$Var[\mu_{s_0,a_0}]$	6.45
	μ_{s_0,a_0}	4.59
	λ_{s_0,a_0}	2.95
	α_{s_0,a_0}	1.51
	β_{s_0,a_0}	9.68

Table D1: Optimised parameter values.

The discount factor for all algorithms and in all simulations was set to $\gamma = 0.99$. Moreover, the tolerance threshold for BCQL's mixture posterior updates was set to $\epsilon = 10^{-5}$.

D.2 Prior initialisations

The prior distributions of all algorithms were initialised to the 'meta-prior', which was obtained by averaging the true optimal return distributions for 1000 environment samples, repeated for each state-action pair. Each environment sample corresponded to sampling the parameters of the reward distributions in the loop environment (Fig 1).

The optimal return distribution at each state-action pair was found for each environment sample using categorical dynamic programming (CDP) (Bellemare et al., 2023). That is, we ran CDP for each integer number of atoms in the range $[10, 50]$, where the atoms were regularly-spaced on a grid ranging from -5 to 15 (we empirically verified that this covered most of the support of the meta-prior; this can be seen in Fig 1). The resulting optimal return distributions were then averaged across the environment samples (more specifically, we averaged the estimated probabilities for each atom) and re-normalised to obtain the meta-prior for the optimal return distribution.

The prior for the QQL algorithm was obtained by performing quantile regression over the resulting categorical meta-prior with 50 atoms (Dabney et al., 2018).

The prior for CatQL was simply set to the meta-prior with the same number of atoms as came out of the optimisation procedure.

The prior for BCQL was set in the same way as for CatQL, but we additionally divided it by the smallest probability so that the smallest Dirichlet hyperparameter was equal to 1. However, this was then multiplied by the fit parameter N as described in D.1.

E Supplementary Figures

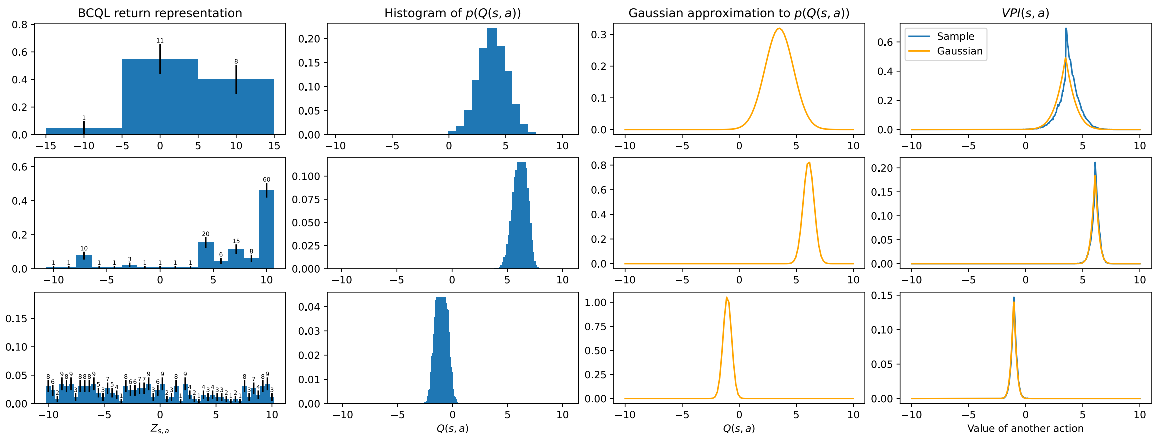


Figure E1: Quality of the Gaussian approximation to BCQL's epistemic uncertainty. The left column shows BCQL's prior belief over return distributions for various number of locations ($M = 3, 15, 50$ top to bottom), as well as the Dirichlet parameters ($\{\alpha_m\}_{m=1,\ldots,M}$, which are written on top of the bars). The individual bars show the expected probabilities and the vertical black lines show 1 standard deviation above and below the mean (that is, BCQL's epistemic uncertainty). The second column shows the empirical probability distribution of the Q-value associated with the return distribution in the previous column obtained by sampling from the prior over the return distributions and taking the mean for each sample. The next column shows a Gaussian approximation to this distribution, which was obtained as described in 3.2. Finally, the last column compares VPI for the action associated with this prior belief based on the empirical and Gaussian approximations to the true distribution over the Q-values, as a function of the expected Q-value of an alternative action. As expected, the Gaussian approximation improves with increasing the number of locations M.

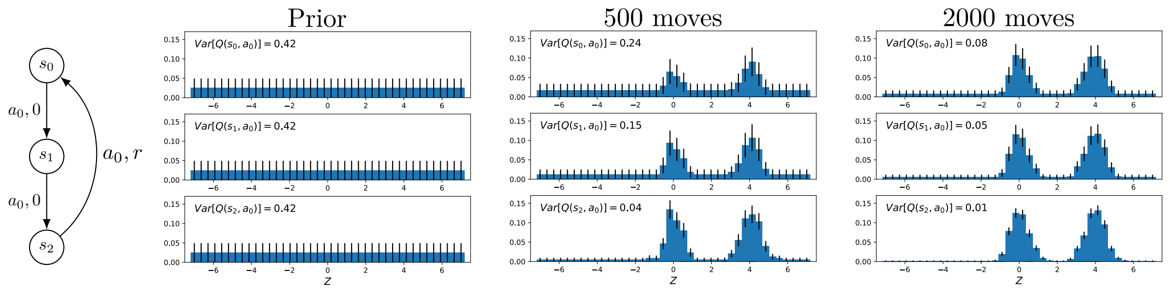


Figure E2: Evaluation of the return distribution by BCQL. The diagram on the left shows the simple 3-state Markov chain in which the agent was simulated. There was a single action a_0 available in each state which occasioned a deterministic transition to the next state. The reward was zero for executing this action in every state except s_2, where it was distributed according to a bimodal Gaussian distribution, that is $r \sim 0.5\mathcal{N}(0, 0.5) + 0.5\mathcal{N}(4, 0.5)$. The plots to the right show the evolution of BCQL's belief state over the course of learning in this environment. In each plot, the height of the bars shows the expected return probabilities for the associated locations. The vertical black lines show BCQL's epistemic uncertainty, namely 1 standard deviation around the mean value. The rows show BCQL's belief about the return distribution for states s_0, s_1, s_2 (top to bottom, on the same level as in the chain diagram). We see that after 2000 moves BCQL's belief state corresponds to a close approximation to the true return distribution in this environment. Moreover, its epistemic uncertainty is reduced appropriately with experience, as can be seen from the variance of the expected return (approximated as described in 3.2).

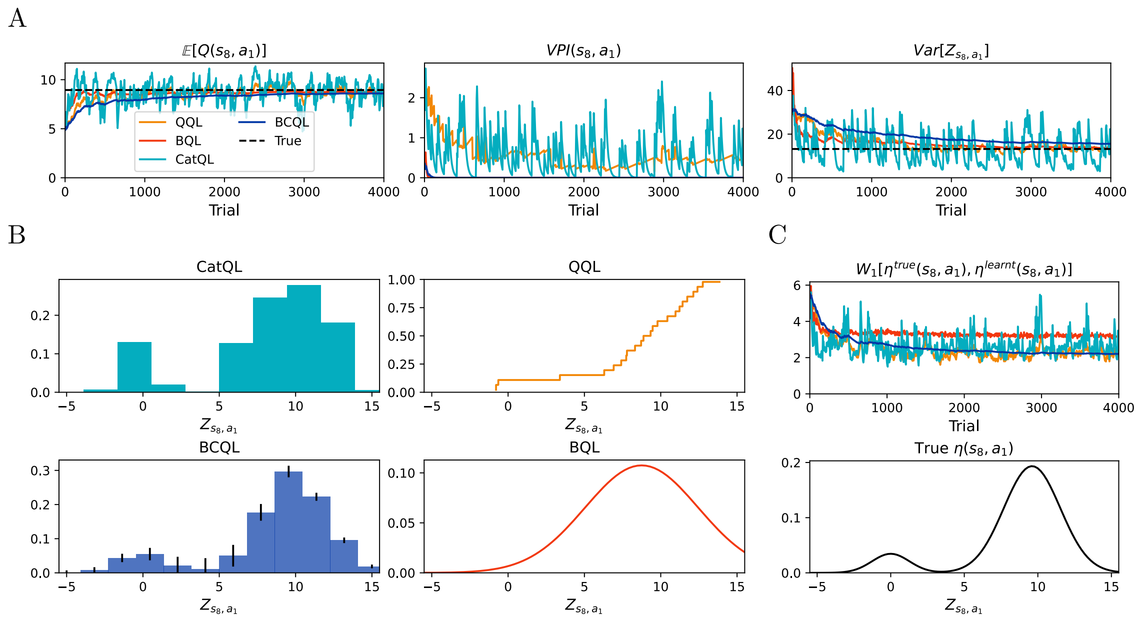


Figure E3: Learning distributional representations at the final state. The layout is the same as in Fig 3 but here we show learning in state s_8. The Wasserstein distance in C) as well as in Fig 3C) was computed using the available method in `SciPy`.