Exploration via Epistemic Value Estimation

Simon Schmitt,[1,2] **John Shawe-Taylor,** [2] **Hado van Hasselt**[1]
[1]DeepMind
[2]University College London, UK
suschmitt@google.com

Abstract

How to efficiently explore in reinforcement learning is an open problem. Many exploration algorithms employ the epistemic uncertainty of their own value predictions – for instance to compute an exploration bonus or upper confidence bound. Unfortunately the required uncertainty is difficult to estimate in general with function approximation.

We propose epistemic value estimation (EVE): a recipe that is compatible with sequential decision making and with neural network function approximators. It equips agents with a tractable posterior over all their parameters from which epistemic value uncertainty can be computed efficiently.

We use the recipe to derive an epistemic Q-Learning agent and observe competitive performance on a series of benchmarks. Experiments confirm that the EVE recipe facilitates efficient exploration in hard exploration tasks.

1 Introduction

Reinforcement learning agents strive to maximize return in sequential decision making problems. To learn about the problem structure they take potentially costly exploratory actions. Ideally the price of exploration will be outweighed by future gains afforded by the gained information. Balancing those two conflicting objectives is called the *exploration versus exploitation trade-off*. It is at the heart of reinforcement research and has been studied extensively (Bellman 1957; Martin 1967; Duff 2002; Guez, Silver, and Dayan 2012). Many proposed exploration algorithms are uncertainty based: To explore efficiently the uncertainty of the value prediction is used – for instance as an exploration bonus or upper confidence interval (Auer, Cesa-Bianchi, and Fischer 2002), or for Thomson Sampling (Thompson 1933).

A Recipe for Uncertainty Estimation How to measure value uncertainty in deep reinforcement learning is actively researched and so far no consensus has been reached. We propose *epistemic value estimation* (EVE), a principled and computationally efficient recipe for uncertainty estimation with function approximation. It is compatible with neural networks and specifically supports off-policy learning and value-bootstrapping – key concepts distinguishing reinforcement learning from supervised learning.

EVE equips the agent with a tractable posterior over all its agent parameters, from which epistemic value uncertainty can be computed efficiently. Considering all agent parameters distinguishes it from prior approaches that are Bayesian only on the final network layer. The recipe reinterprets value prediction as density estimation of returns. Drawing on insights from parametric statistics it then approximates the posterior over agent parameters using a specifically structured Gaussian distribution. Besides being motivated in statistical theory this approximation is efficient to sample from and convenient to estimate using automatic differentiation frameworks. As a result it obtains favourable computational performance compared to ensemble methods that need to store and update multiple models concurrently.

When applied to Q-Learning, it matches the exploration performance of Bootstrapped DQN on the Deep Sea benchmark – providing encouraging evidence for our recipe.

What are Epistemic Values? As we are uncertain about the true value (i.e. the expected return) we can treat it as a random variable given the agents experience. We will call the corresponding random variable the *epistemic value* to emphasize that the value distribution differs from the return distribution. On average the epistemic value uncertainty decreases with more observed data while the return uncertainty is by definition invariant to it.

$$\underbrace{p(V|s,\mathcal{D})}_{\substack{\text{Posterior of } \mathbf{mean} \text{ returns } V \\ \text{captures } \mathbf{epistemic} \text{ uncertainty.}}} \neq \underbrace{p(Z|s,\mathcal{D})}_{\substack{\text{Density of } \mathbf{Monte{-}Carlo} \text{ returns } Z \\ \text{captures } \mathbf{aleatoric} \text{ uncertainty.}}}$$

For example the near-optimal UCB algorithm for bandits upper-bounds the uncertainty of the epistemic value V. If it were to instead upper-bound the uncertainty of return Z it would continually keep selecting the action with the noisiest return and cease to be optimal. The EVE recipe provides epistemic value estimates for sequential decision making problems with function approximation.

2 Background

We investigate exploration in Markov Decision Processes (Bellman 1957). We largely follow the notation from Sutton and Barto (1998) denoting actions A_t taken at states S_t yielding new states S_{t+1} and rewards R_{t+1}. When selecting actions $A_t \sim \pi(S_t)$ from a policy π the Monte-Carlo return at state S_t is $Z_t := \sum_{i=0}^{\infty} \gamma^i R_{t+i+1}$ to be distinguished from bootstrapped returns G_t. The expected return is

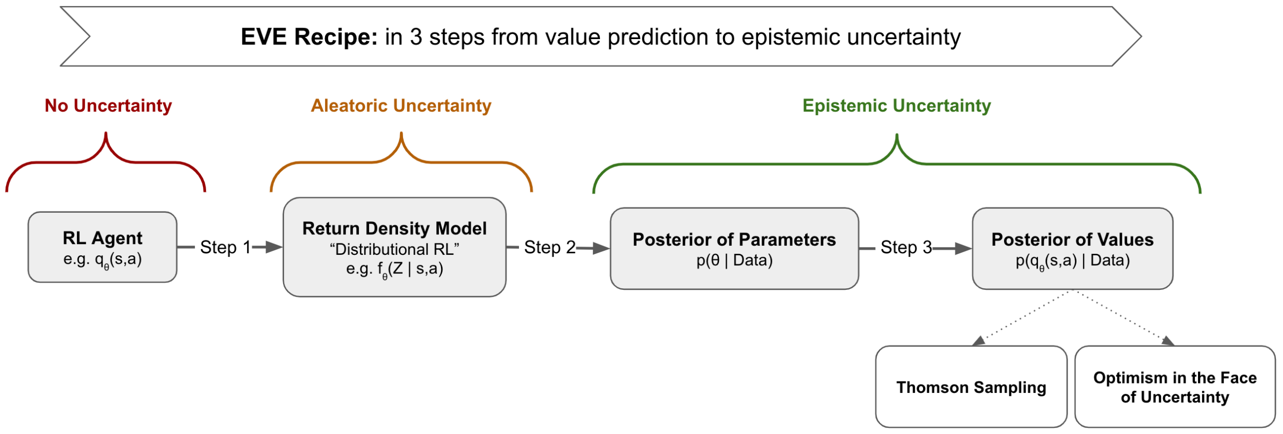

Figure 1: The Epistemic Value Estimation recipe (EVE) equips agents with a tractable posterior over all their parameters from which epistemic uncertainty can be computed efficiently. The first step converts the agent into a distributional agent, that captures the seemingly unrelated aleatoric return uncertainty. The second step approximates the posterior over all agent parameters from which the third step can obtain the epistemic value uncertainty $p(q_\theta(s,a)|\mathcal{D})$.

called the value $v_\pi(s) := \mathbb{E}_\pi\left[Z_t \mid S_t = s\right]$. Function approximation is often used to represent value estimates in large state spaces – typically by parametric functions $v_\theta(s)$. Given no prior knowledge about the MDP we strive to find a policy that selects actions to maximize the expected return.

We intend to estimate the uncertainty of predicted values $v_\theta(s)$ given limited data $\mathcal{D}$ comprised of n steps. In the process we estimate the distribution of Monte-Carlo returns Z. Recall from the introduction that the uncertainty of returns is different from the uncertainty of values (expected returns).

In general, consider some random variables X_i that are sampled i.i.d. from a distribution $f_{\theta^{\text{True}}}$ parameterized by an unknown parameter θ^{True}. We can construct a frequentist estimate of θ^{True} using the maximum likelihood estimator: $\theta_n^{\text{MLE}} := \arg\max_\theta p(X_1, \dots, X_n|\theta) = \arg\max_\theta \prod_i f_\theta(X_i)$. We can also adopt a Bayesian view, and assume a prior belief about the likelihood of parameters $p(\theta)$ and use Bayes rule to compute the posterior: $p(\theta|\mathcal{D}) \propto p(\mathcal{D}|\theta)p(\theta)$. The maximum a posteriori solution is then typically considered the best parameter point-estimate: $\theta_n^{\text{MAP}} := \arg\max_\theta p(\theta|\mathcal{D})$ For notational simplicity we drop the n from $\theta_n^{\text{MLE}}, \theta_n^{\text{MAP}}$. One can show that $\theta^{\text{MLE}} \to \theta^{\text{True}}$ and $\theta^{\text{MAP}} \to \theta^{\text{True}}$ when f and prior $p(\theta)$ satisfy appropriate regularity conditions (see van der Vaart 1998; Nickl 2012, for details). Finally, we use $x \odot y$ to refer to the Hadamard product of two vectors x, y.

3 A Scalable Recipe for Epistemic Uncertainty in RL

Our goal is to formalize a recipe that can provide deep RL agents with epistemic value uncertainty and posterior sampling capabilities in a way compatible with their large neural networks. Our recipe supports off-policy Q-Learning and k-step TD learning where an agent bootstraps on its own predictions – a requirement that differentiates it from supervised learning. We first state the three major steps in the recipe, and then explain each step in turn in the remainder of this section.

Intuition For an intuitive summary of the three steps consider Figure 1.

Steps

1. Use a suitable parametric density model f_θ to predict the distribution of returns: $f_\theta(Z|s) \approx p(Z|s)$.
2. Using f_θ, approximate the corresponding posterior of parameters $p(\theta|\mathcal{D})$ with a specifically constructed Gaussian distribution.
3. Finally, compute $p(v_\theta(s)|\mathcal{D})$ from $p(\theta|\mathcal{D})$.

3.1 Step 1: Model & Log-Likelihood

This step aims at setting up a parametric agent model that predicts the entire distribution of Monte-Carlo returns $p(Z|s,a)$. We require its log-likelihood gradient for model fitting and for the posterior approximation in Step 2.

Modelling Monte-Carlo Returns In reinforcement learning the Monte-Carlo return is not always available or desirable due to its high variance and on-policy nature. In fact many agents use k-step bootstrapped returns $G_t^k := \sum_{i=1}^k \gamma^{i-1} R_{t+i} + \gamma^{k-1} v_\theta(S_{t+k})$. This is an important difference to supervised learning with fixed-outputs pairs. For us this poses a challenge as they are differently distributed

$$p(Z|s) \neq p(G^k|s)$$

and predicting k-step returns would only capture the epistemic uncertainty of the first k steps and disregard the uncertainty of the bootstrap value $v_\theta(S_{t+k})$. A similar problem occurs with value iteration where returns are off-policy and differ from the observed Monte-Carlo return: $G_t^Q := R_{t+1} + \gamma \max_a Q(S_{t+1}, a)$.

A solution is to make the agent *distributional* (Dearden, Friedman, and Russell 1998; Bellemare, Dabney, and Munos 2017): We sample hypothetical Monte-Carlo returns Z' such that $p(Z') = p(Z)$ and use them instead of G^k or G^Q. Rather than bootstrapping G_t^k with the value $v_\theta(S_{t+k})$ we bootstrap using a Monte-Carlo return sample from our model:

$$\tilde{Z}_{t+k} \sim f_\theta(Z|S_{t+k}) \tag{1}$$

and define a *modelled return* Z'_t that preserves the distributions $p(Z'_t|S_t) \approx p(Z_t|S_t)$:

$$Z'_t := \sum_{i=1}^{k} \gamma^{i-1} R_{t+i} + \gamma^k \tilde{Z}_{t+k} \tag{2}$$

Sampling from f induces an approximation error, however Bellemare, Dabney, and Munos (2017) have observed strong empirical performance and provided a theoretical analysis for learning distributional value functions.

Log-Likelihood Estimation Given the density model f_θ and modelled or actual Monte-Carlo returns Z' at state S we can estimate the maximum likelihood solution θ^{MLE}. Typically this is achieved through stochastic gradient decent on $\nabla_\theta \log f_\theta(Z'|S)$. The same gradient will also be used in Step 2 to approximate the posterior.

3.2 Step 2: Posterior Approximation

We will now derive and approximate the posterior of the parametric density model of returns $f_\theta(Z|s)$ corresponding to the distributional agent from Step 1. Combining the standard Bayesian approach with the Bernstein-von Mises theorem, we will obtain a Gaussian posterior with a scaled inverse Fisher information as a covariance matrix. Finally we will observe that the posterior is efficient to approximate using log-likelihood gradients.

Bayesian Posterior Derivation The derivation follows the typical Bayesian approach where $p(\theta|\mathcal{D}) \propto p(\mathcal{D}|\theta)p(\theta)$. Our data $\mathcal{D}$ is comprised of n state-return pairs (Z_i, S_i) originating from a policy μ. We denote the set of all n observed returns as $\boldsymbol{Z^n}$ and states as $\boldsymbol{S^n}$. Akin to Bayesian regression we estimate the distribution of returns given input states:

$$\begin{aligned} p(\theta|\boldsymbol{Z^n}, \boldsymbol{S^n}) &\propto p(\boldsymbol{Z^n}|\boldsymbol{S^n}, \theta)p(\theta) \\ &\approx \prod_i f_\theta(Z_i|S_i)p(\theta) \end{aligned} \tag{3}$$

In Bayesian regression the last step would be exact because the likelihood can be factorized $p(\boldsymbol{Z^n}|\boldsymbol{S^n}, \theta) = \prod_i f_\theta(Z_i|S_i)$. This is only possible if the Z_i are conditionally independent given states, which is not the case for Monte-Carlo returns that are computed from overlapping reward sequences. Independence can be achieved when modelled k-step returns are used because all $Z'_i \sim f_\theta(Z|S_i)$ are conditionally independent given $\boldsymbol{S^n}$. Return overlap is then reduced to k-steps, enabling two approaches: (a) sub-sampling the data to each $(k+1)^{\text{th}}$ state will yield full independence and make Equation (3) exact (b) treating the factorization as an approximation that becomes more and more exact with smaller k. In the theoretical derivations we will assume independence e.g. achieved via (a). Incidentally such decorrelation via sub-sampling improves asymptotic agent performance in model based RL (Schrittwieser et al. 2020). In Algorithm 2 we chose (b) for simplicity and counter the potential overconfidence by rescaling the number of effective samples n in our posterior approximation by a factor ω.

Approach to Posterior Approximation For large models the exact posterior is intractable and needs to be approximated. Following the Bernstein-von Mises theorem (which is explained below) we represent it as a Gaussian centered around θ^{MLE}

$$p(\theta|\mathcal{D}) \approx \mathcal{N}\left(\theta^{\text{MLE}}, \frac{1}{n}\mathcal{I}(\theta^{\text{True}})^{-1}\right) \tag{4}$$

with covariance proportional to the inverse Fisher information matrix (which can be approximated as explained below):

$$\mathcal{I}(\theta^{\text{True}}) := \mathbb{E}_{X \sim f_{\theta^{\text{True}}}}\left[\nabla_\theta \log f_{\theta^{\text{True}}}(X) \nabla_\theta \log f_{\theta^{\text{True}}}(X)^\top\right]$$

A similar representation can be derived using the Laplace approximation (see appendix). The Gaussian structure permits efficient sampling.

Bernstein-von Mises Theorem The Bernstein-von Mises theorem states that the Bayesian posterior of a parametric density model $f_\theta(X)$ inferred from samples $X_i \sim f_{\theta^{\text{True}}}(X)$ from the true distribution becomes a Gaussian distribution:

$$p(\theta|X_1, ..., X_n) \to \mathcal{N}(\theta_n^{\text{MLE}}, \frac{1}{n}\mathcal{I}(\theta^{\text{True}})^{-1}) \tag{5}$$

Note that the Gaussian distribution is centered at the maximum likelihood solution θ_n^{MLE}. The covariance depends on the Fisher at the unknown true distribution parameters θ^{True}. Observe that the covariance shrinks with the number n of observed samples i.e. that the posterior gets narrower with $1/\sqrt{n}$ – a property resembling the central limit theorem. Note that we slightly abuse notation since both sides in the limit depend on n. More precisely stated the total variation norm between both distributions converges in probability to zero $\|p(\theta|X_1, ...X_n) - N(\theta_n^{\text{MLE}}, \frac{1}{n}\mathcal{I}(\theta^{\text{True}})^{-1})\|_{TV} \to 0$. For a short summary please consider the appendix; for a detailed exposition please consider van der Vaart (1998); Le Cam (1986); Nickl (2012) – who date the theorem's origins back to Laplace (1810) work on the central limit theorem and early work by von Mises (1931) and Bernstein (1917).

The Bernstein-von Mises theorem relates Bayesian and frequentist statistics and is typically used to show that $||\theta_n^{\text{MAP}} - \theta_n^{\text{MLE}}|| \to 0$ and to argue that the prior distribution does not matter in the limit (van der Vaart 1998). While not all theoretical assumptions can be satisfied in reinforcement learning with neural networks we observe favourable empirical results when employing it within the EVE recipe for value uncertainty estimation and exploration in Section 6.

Fisher Approximation We have θ^{MLE} from Step 1, but we are missing the Fisher information matrix $\mathcal{I}(\theta^{\text{True}})$ to employ the Bernstein-von Mises theorem. Its expectation can be computed using the observed samples $X_i \sim f_{\theta^{\text{True}}}$: $\mathcal{I}(\theta^{\text{True}}) \approx \frac{1}{n}\sum_{i=1}^n \nabla_\theta \log f_{\theta^{\text{True}}}(X_i) \nabla_\theta \log f_{\theta^{\text{True}}}(X_i)^\top$ However it needs a further approximation because we can not compute gradients at the unknown θ^{True}.

$$\hat{\mathcal{I}}(\theta^{\text{MLE}}) := \frac{1}{n}\sum_{i=1}^n \nabla_\theta \log f_{\theta^{\text{MLE}}}(X_i) \nabla_\theta \log f_{\theta^{\text{MLE}}}(X_i)^\top$$

Now the gradients can be computed using automatic differentiation frameworks. Unfortunately estimating the full

Algorithm 1: **Standard Q-Learning** with ϵ-greedy exploration.

```
Exploration Parameters: Epsilon-greedy $\epsilon$.
Regular Parameters: Learning rate $\alpha$, neural network $q_\theta$, discount $\gamma$.
Initialization: Vector $\theta$ random.
Acting:
  1: Play each action as $\arg\max_a q_\theta(s,a)$ with probability $1-\epsilon$ or uniformly random otherwise.
  2: Add resulting trajectory $\tau$ into experience replay $\mathcal{D}$.
Q-Learning Update with $\theta$:
  1: Sample one transition $S_t, A_t, R_{t+1}, S_{t+1}$ from $\mathcal{D}$
  2: $G = R_{t+1} + \gamma \max_a q_\theta(S_{t+1}, a)$
  3: $\theta \leftarrow \theta - \alpha \nabla_\theta (G - q_\theta(S_t, A_t))^2$
```

empirical Fisher information matrix $\hat{\mathcal{I}}(\theta^{\text{MLE}})$ is infeasible with large-scale function approximation due to its quadratic memory requirements. However it can be efficiently approximated and inverted using a diagonal or Kronecker-factored representation (Martens 2014).

3.3 Step 3: Epistemic Value Uncertainty

Using Step 2 we can now efficiently sample parameters θ' from our approximation to the posterior $p(\theta|\mathcal{D})$. We can use them for Thomson Sampling in parameter space, or to estimate the epistemic uncertainty of values. While $p(\theta|\mathcal{D})$ has a convenient Gaussian shape the nonlinearities in the model prevent us from deriving a similar analytic representation for $p(v_\theta(s)|\mathcal{D})$. Sampling epistemic values via $v'(s) \sim p(v_\theta(s)|\mathcal{D})$ is however easy and can for instance be used to estimate $\mathbb{V}\left[v_\theta(s)|\mathcal{D}\right]$ numerically. Sampling of $v'(s)$ can be achieved though sampling θ' and evaluating $v'(s) := v_{\theta'}(s)$ which is the mean of the predicted return distribution $f_{\theta'}(Z|s)$: $v_{\theta'}(s) = \int_Z Z f_{\theta'}(Z|s) dZ$. For Gaussian return models the predicted mean is the output of the networks forward pass and does not require integration.

4 Epistemic Q-Learning

The EVE recipe strives to provide deep RL agents with epistemic uncertainty estimates to improve their exploration. To provide a concrete example we use the recipe to convert a standard Q-Learning agent from Algorithm 1 into an illustrative epistemic Q-Learning agent (Algorithm 2).

In this section we strive for clarity over performance, hence where possible we prefer conceptually simpler approximations. As a result we managed to keep the difference minimal: the Q-Learning update is modified slightly and the Fisher estimation is added, which is implemented with an exponential average of squared gradients from a noisy Q-Learning loss. These changes result in a computational overhead (one additional gradient pass) that is moderate compared to ensemble methods that store and update multiple model copies. We discuss more advanced techniques for EVE such as distributional value functions, K-FAC approximations and variance reduction in the appendix. Nevertheless in Section 6 we already observe competitive results with this illustrative agent on common benchmarks.

Algorithm 2: **Epistemic Q-Learning using EVE** with diagonal Fisher approximation.

```
Exploration Parameters: Exploration scale $\omega$, return variance ${\sigma^{\text{Return}}}^2$, Fisher learning rate $\beta$, Fisher regularization $\epsilon$.
Regular Parameters: Learning rate $\alpha$, neural network $q_\theta$, discount $\gamma$.
Initialization: Vectors $\theta$ random, $f^{\text{diagonal}}$ zero. Scalar $n$ one.
Acting:
  1: Sample $\theta'$ from posterior.
  2: Play one episode with $\arg\max_a q_{\theta'}(s,a)$.
  3: Add resulting trajectory $\tau$ into experience replay $\mathcal{D}$.
  4: $n \leftarrow n + |\tau|$
Learning Step:
  1: Sample $\theta'$ from posterior.
  2: Q-Learning Update with $\theta'$:
  3:   Sample one transition $S_t, A_t, R_{t+1}, S_{t+1}$ from $\mathcal{D}$.
  4:   $G = R_{t+1} + \gamma \max_a q_{\theta'}(S_{t+1}, a)$
  5:   $\theta \leftarrow \theta - \alpha \nabla_\theta (G - q_\theta(S_t, A_t))^2$
  6: Fisher Update:
  7:   $\eta' \sim \mathcal{N}(0, \sigma^{\text{Return}})$
  8:   $Z' = G + \gamma \eta'$
  9:   $g^{\text{LogL}} = \nabla_\theta (Z' - q_\theta(S_t, A_t))^2$
 10:   $f^{\text{diagonal}} \leftarrow (1-\beta) f^{\text{diagonal}} + \beta g^{\text{LogL}} \odot g^{\text{LogL}}$
Posterior Sampling:
  1: Define the vectors $\sigma$, $z$ such that for all $i$:
  2: $\sigma_i = \frac{1}{\sqrt{(f^{\text{diagonal}})_i + \epsilon}}$
  3: Sample $z$ such that $z_i \sim \mathcal{N}(0,1)$.
  4: Return $\theta + \frac{1}{\sqrt{n\omega}} \sigma \odot z$
```

Standard Q-Learning Baseline The standard deep Q-Learning in Algorithm 1 uses a neural network q_θ to predict Q-values. The parameters θ are updated to minimize the squared prediction error towards targets $G_t = R_{t+1} + \gamma \max_a q_\theta(S_{t+1}, a)$:

$$\mathcal{L}^{Q-Prediction}(\theta) := (G_t - q_\theta(S_t, A_t))^2 \tag{6}$$

Note that the targets G_t are fixed and $\nabla_\theta G_t = 0$ (sometimes this is referred to as *stop gradient*). Q-Learning then acts with ϵ-greedy.

Introducing Thomson Sampling The first difference in Algorithm 2 is the introduction of Thomson Sampling at acting time. Furthermore we Thomson-sample at bootstrapping time by changing the target in line 4 of the learning step to:

$$\begin{aligned} G_t^{TS} &:= R_{t+1} + \gamma \max_a q_{\theta'}(S_{t+1}, a) \\ \theta' &\sim p(\theta|\mathcal{D}) \end{aligned} \tag{7}$$

4.1 Applying Step 1: Model & Log-Likelihood

In this step we need to make the agent distributional and obtain the corresponding log-likelihood gradient. Rather than predicting the expected return $q_\theta(s,a) \approx \mathbb{E}\left[G|s,a\right]$ it needs to predict the entire distribution of Monte-Carlo returns

$f_\theta(Z|s,a) \approx p(Z|s,a)$. Furthermore we require the ability to sample Monte-Carlo returns from the model: $Z' \sim f_\theta$.

A Distributional Interpretation for Standard Q-Learning To make the example agent distributional and at the same time minimize algorithmic differences we simply reinterpret Q-Learning as a Gaussian density model which represents the Monte-Carlo return distribution as a Gaussian with fixed variance centered around the predicted Q-values q_θ:

$$f_\theta(Z|s,a) = \mathcal{N}(Z|\mu = q_\theta(s,a), \sigma = 1) \qquad (8)$$

enabling us to use the same powerful neural network architecture q_θ as the Q-Learning baseline. This algorithmically convenient choice of f_θ yields a powerful predictor of mean values but a crude return density model as it disregards state-dependent differences in the return variance. While this leaves room for future work, we will see in Figure 5 that it is able to capture state dependent epistemic value uncertainty and exhibits favourable exploration performance in our experiments in Section 6.

Computing the Log-Likelihood Gradient The Gaussian model choice from Equation (8) implies that sampling bootstrap returns from $f_\theta(\cdot|S_{t+1},a)$ simplifies to $Z' = q_\theta(S_{t+1},a) + \eta'$ with $\eta' \sim \mathcal{N}(0,1)$. We can use that to derive the log-likelihood gradient for the model f_θ when it aims to predict the modelled return samples Z' of following the Thomson Sampling policy:

$$\begin{aligned} g_t^{\text{LogL}}(\theta) &:= \nabla_\theta \log f_\theta(Z'_t|S_t,A_t) \qquad (9)\\ &= \nabla_\theta \frac{1}{2}(Z'_t - q_\theta(S_t,A_t))^2 \end{aligned}$$

with a f_θ-modelled Monte-Carlo return sample

$$Z'_t = \underbrace{R_{t+1} + \gamma \max_a q_{\theta'}(S_{t+1},a)}_{G_t^{TS}} + \gamma\eta'$$

$$\theta' \sim p(\theta|\mathcal{D}),\ \eta' \sim \mathcal{N}(0,1)$$

4.2 Applying Step 2: Posterior Approximation

According to the recipe we approximate the posterior as $p(\theta|\mathcal{D}) \approx \mathcal{N}(\theta^{\text{MLE}}, \frac{1}{n}\hat{\mathcal{I}}(\theta^{\text{MLE}})^{-1})$ with

$$\begin{aligned} \hat{\mathcal{I}}(\theta^{\text{MLE}}) &:= \frac{1}{n}\sum_{t=1}^{n} \mathbb{E}_\eta \left[\nabla_\theta \log f_{\theta^{\text{MLE}}}(Z'_t) \nabla_\theta \log f_{\theta^{\text{MLE}}}(Z'_t)^\top\right]\\ &:= \frac{1}{n}\sum_{t=1}^{n} \mathbb{E}_\eta \left[g_t^{\text{LogL}}\ \ g_t^{\text{LogL}\top}\right] \end{aligned}$$

where g_t^{LogL} is short notation for the log-likelihood gradient from Equation (9) at the MLE estimate $g_t^{\text{LogL}}(\theta^{\text{MLE}})$.

Unfortunately estimating the full empirical Fisher information matrix is infeasible with large-scale function approximation due to its quadratic memory requirements. However it be efficiently approximated and inverted using diagonal or K-FAC representation (Martens 2014). Striving again for the most simple example agent in this section we use the diagonal approximation:

$$\hat{\mathcal{I}}^{\text{Diag}}(\theta^{\text{MLE}}) := \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_\eta \left[g^{\text{LogL}} \odot g^{\text{LogL}}\right]$$

4.3 Applying Step 3: Epistemic Value Uncertainty

Now estimation, inversion and sampling are efficiently possible using element-wise operations. Given a vector from a standard Normal $z \sim \mathcal{N}(0, I)$ we can obtain a sample from:

$$\theta' \sim \mathcal{N}(\theta^{\text{MLE}}, \frac{1}{n}\hat{\mathcal{I}}^{\text{Diag}}(\theta^{\text{MLE}})) \approx p(\theta|\mathcal{D})$$

by computing each component in the vector as $\theta'_i := (\theta^{\text{MLE}})_i + \frac{z_i}{\sqrt{n\hat{\mathcal{I}}^{\text{Diag}}(\theta^{\text{MLE}})_i}}$.

4.4 Optional Variance Reduction

The gradient $g_t^{\text{LogL}}(\theta) = \nabla_\theta \frac{1}{2}(Z'_t - q_\theta(S_t,A_t))^2$ is stochastic because of the random variable η' in $Z'_t = R_{t+1} + \gamma \max_a q_{\theta'}(S_{t+1},a) + \gamma\eta'$. We can make our updates more sample-efficient by considering the corresponding expected updates. This is straightforward for the θ^{MLE} maximum likelihood parameter estimation where we recover the classic Q-Learning update:

$$\begin{aligned} g_t^{\text{MLE}}(\theta) &:= \mathbb{E}_\eta\left[g_t^{\text{LogL}}(\theta)\right] \qquad (10)\\ &= (\mathbb{E}_\eta[Z'_t] - q_\theta(S_t,A_t))\nabla_\theta q_\theta(S_t,A_t)\\ &= (G_t^{TS} - q_\theta(S_t,A_t))\nabla_\theta q_\theta(S_t,A_t)\\ &= \nabla_\theta \frac{1}{2}(G_t^{TS} - q_\theta(S_t,A_t))^2 \end{aligned}$$

The variance of the Fisher update from line 10 in Algorithm 2 could also be reduced, but this is non-trivial (see appendix).

5 Related work

How to measure epistemic uncertainty in deep reinforcement learning is actively researched. Popular methods involve ensembles consisting of multiple independent neural networks or sharing parts of them (Osband et al. 2016) resulting in increased memory and computational requirements. Heuristics, like used by Burda et al. (2019), use the prediction error of an unknown random function target as an exploration bonus. Ostrovski et al. (2017) employed a generative model of states to compute pseudo-counts for exploration. Dearden, Friedman, and Russell (1998) approximated the posterior of state-action values for exploration in tabular MDPs. In non-tabular MDPs Deisenroth and Rasmussen (2011) employed techniques from kernel methods resulting in increased data-efficiency but also being limited by the large computational cost of kernel regression. Similarly Chua et al. (2018); Curi, Berkenkamp, and Krause (2020) approximate the environment dynamics with an ensemble of neural networks each parameterizing the next states probability with a Gaussian. In the field of Bayesian deep learning (cf. MacKay 1992; Hinton and van Camp 1993; Neal 1995, for early discussions of the principles) a series of innovations were proposed (Graves 2011; Blundell et al. 2015; Gal and Ghahramani 2016; Korattikara Balan et al. 2015; Lakshminarayanan, Pritzel, and Blundell 2017; Osband, Aslanides, and Cassirer 2018; Zhang et al. 2018; Ritter, Botev, and Barber 2018; Daxberger et al. 2021) (see Murphy 2023, for an overview). However there is no consensus of how to adapt from supervised learning to sequential decision-making processes, where value-bootstrapping and

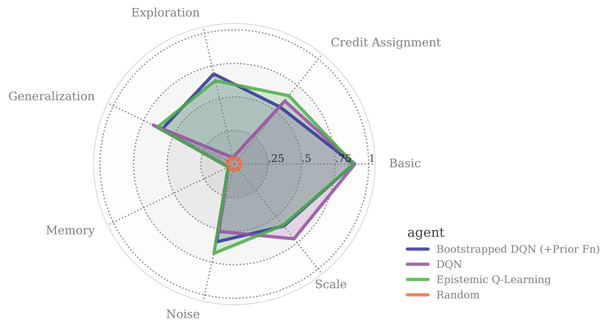

Figure 2: Performance breakdown of selected agents along the Bsuite capabilities. Observe that Epistemic Q-Learning and Bootstrapped DQN achieve high exploration scores while regular DQN barely exceeds a random agent. The score breakdown is aggregated from 468 Bsuite task variations averaged over 30 holdout seeds.

off-policy learning need to be considered. Tang and Agrawal (2018) use distributional reinforcement learning (Bellemare, Dabney, and Munos 2017) for exploration and provide a variational interpretation for Plappert et al. (2018) and Fortunato et al. (2018) that perturb the parameters of reinforcement learning agents with heuristically scaled Gaussian noise. In our work we derive an explicit approximation to the posterior that is motivated by the Bernstein von-Mises theorem from statistical theory. The posterior considers all agent parameters distinguishing it from last-layer methods such as Azizzadenesheli, Brunskill, and Anandkumar (2018). In particular it is compatible with neural network function approximation, efficient to estimate (using automatic differentiation methods) and efficient to evaluate. Compared to ensembles that maintain and update multiple models it requires fewer parameters and typically less compute. Furthermore it supports unlimited sampling of values, where ensembles only provide a constant number of values (one sample per parameter copy).

6 Experiments

We have proposed a general recipe for epistemic value estimation (EVE), derived a simple example agent from it in Section 4 and empirically evaluate it here. In Figure 2 we observe competitive performance on the Bsuite benchmarks (Osband et al. 2020), where our agent matches the state-of-the-art results from Osband, Aslanides, and Cassirer (2018) that employs an ensemble of 20 independent neural network copies each with their own copy of a random prior function, target network and optimizer state. In comparison epistemic Q-Learning with EVE requires only a single copy of network, target network, diagonal fisher and optimizer state – resulting in 20$\times$ fewer parameters. Furthermore we present ablations (Figure 7) and parameter studies (Figure 6) showing that our agent is robust to the choice of hyper-parameters.

6.1 Benchmark Environments

We use the behaviour suite benchmark (Bsuite) with special focus on the Deep Sea environment to empirically analyze our epistemic Q-Learning example agent from Section 4.

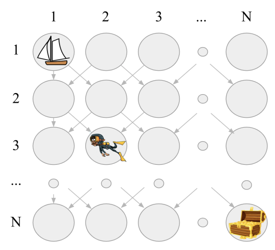

Figure 3: The Deep Sea MDP is given by an L by L sized grid, where a diver starts in the top left corner and may move down right or down left at each step. A treasure is on the far right side, but the agent can only reach it by moving right at all steps. Hence only a single among 2^{L-1} possible action sequences is successful. Trivial solutions are prevented by randomizing the action mapping. To discourage the correct sequence there is a penalty for each movement to the right.

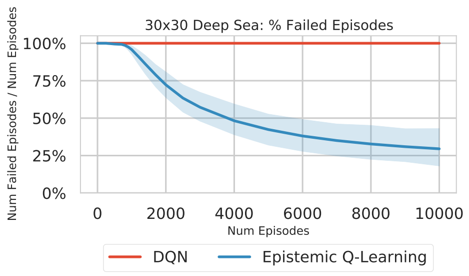

Figure 4: Epistemic Q-Learning solves the Deep Sea (30$\times$30) exploration benchmark: the failures among the total number of episodes diminish. DQN fails to solve it.

Behaviour Suite Bsuite was introduced by Osband et al. (2020) to facilitate the comparison of agents not just in terms of total score but across meaningful capabilities (such as exploration, credit assignment, memory, and generalization with function approximators). The standardized evaluation protocol facilitates direct comparisons between research papers. Bsuite consists of 13 environments (including versions of Deep Sea, Mountaincar, Cartpole, contextual bandits with MNIST images and T-maze environments) which are then varied in difficulty (such as problem size, reward sparsity, or stochasticity of transitions or reward) resulting in 468 task variations. Each agent is evaluated on all 468 tasks and the performance is grouped into 7 categories (called capabilities). We are most interested in the *exploration capability score* which considers the sparse reward tasks (Deep Sea, Stochastic Deep Sea and Cartpole Swingup) with various difficulty levels resulting in 62 task variations. The aggregate exploration score is the fraction of those 62 tasks where the sparse reward is consistently[1] obtained by the agent.

Deep Sea Osband, Aslanides, and Cassirer (2018) propose the Deep Sea environment (see Figure 3) to measure exploration performance. It is a 'needle in the haystack'-style

[1]In line with prior publications we use a harder evaluation metric than originally described in Osband et al. (2020) – see Appendix F.

Algorithm 3: **Epistemic Q-Learning using EVE** with diagonal Fisher approximation, burn-in, target networks, and ADAM optimization (omitting mini-batching details).

```
**Exploration Parameters:** Exploration scale $\omega$, return variance ${\sigma^{\text{Return}}}^2$, Fisher learning rate $\beta$, Fisher regularization $\epsilon$, burn-in steps $K_{\text{burnin}}$.
**Regular Parameters:** Learning rate $\alpha$, target network period $K_{\text{target}}$, neural network $q_\theta$, discount $\gamma$.
**Initialization:** Vectors $\theta, \bar{\theta}$ random, $f^{\text{diagonal}}$ zero. Scalars $m, n$ one.
**Acting:**
 1: Act uniformly for the first $K_{\text{burnin}}$ episodes; otherwise:
 2: Sample $\theta'$ from posterior.
 3: Play one episode with $\arg\max_a q_{\theta'}(s, a)$.
 4: Add resulting trajectory $\tau$ into experience replay $\mathcal{D}$.
 5: $n \leftarrow n + |\tau|$
**Learning Step:**
 1: Sample $\theta'$ from posterior.
 2: **Q-Learning Update with $\theta'$:**
 3:   Sample one transition $S_t, A_t, R_{t+1}, S_{t+1}$ from $\mathcal{D}$.
 4:   $G = R_{t+1} + \gamma \max_a q_{\theta'}(S_{t+1}, a)$
 5:   $g^{\text{MLE}} = \nabla_\theta (G - q_\theta(S_t, A_t))^2$
 6:   $\theta \leftarrow \theta - \alpha \,\text{ADAM}(g^{\text{MLE}})$
 7: **Fisher Update:**
 8:   $\eta' \sim \mathcal{N}(0, \sigma^{\text{Return}})$
 9:   $Z' = G + \gamma \eta'$
10:   $g^{\text{LogL}} = \nabla_\theta (Z' - q_\theta(S_t, A_t))^2$
11:   $f^{\text{diagonal}} \leftarrow (1-\beta) f^{\text{diagonal}} + g^{\text{LogL}} \odot g^{\text{LogL}}$
12:   $m \leftarrow (1-\beta) m + 1$
13: Every $K_{\text{target}}$ steps update the target network $\bar{\theta} \leftarrow \theta$.
**Posterior Sampling**:
 1: Define the vectors $f^{\text{unbiased}}$, $\sigma$, $z$ such that for all $i$:
 2: $(f^{\text{unbiased}})_i = (f^{\text{diagonal}})_i / m$
 3: $\sigma_i = \frac{1}{\sqrt{(f^{\text{unbiased}})_i + \epsilon}}$
 4: Sample $z$ such that $z_i \sim \mathcal{N}(0, 1)$.
 5: **Return** $\bar{\theta} + \frac{1}{\sqrt{n\omega}} \sigma \odot z$
```

problem where ϵ-greedy requires an exponential number of environment steps because it over-explores. It was proposed to motivate the the need for 'deep' or 'directed' exploration with polynomial exploration time. Figure 4 shows that DQN with its ϵ-greedy exploration is unable to find the treasure, while Epistemic Q-Learning obtains the treasure in more than 70% of the given 10000 episodes and 30 seeds.

6.2 Methodology

We build our agent on the reference agent implementations from Bsuite and strive to minimize all unrelated differences (e.g. neural network architecture, optimizer, target networks) to permit a clear comparison – see appendix for algorithmic details. Most notably we replace ReLU with Leaky-ReLU activations as they yield non-zero gradients almost everywhere while maintaining the ReLU benefits and confirm that the choice of activation does not change the DQN baseline performance. Exploration hyper-parameters were tuned among possible powers of 10 in the range $[10^{-15}, 10^{10}]$. We then sampled new holdout seeds for all figures (10 holdout seeds for the parameter studies and 30 for epistemic Q-Learning, all baselines and ablations). Hence a complete Bsuite evaluation in Figure 2 evaluates all 468 tasks 30 times with the exploration score aggregating 62×30 separate RL runs. Figures 6 and 7 summarize $468 \times 10 \times 7 \times 4$ and respectively $62 \times 30 \times 5$ separate RL runs.

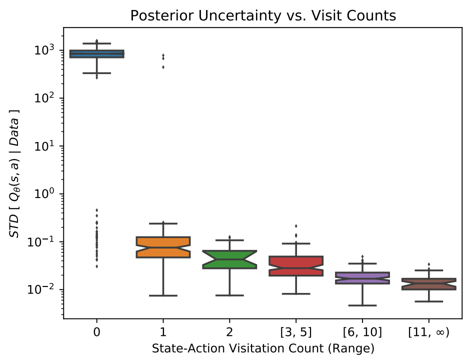

Figure 5: As desired, the predicted epistemic uncertainty $\sqrt{\mathbb{V}\left[q_\theta(s,a)|\mathcal{D}\right]}$ is lower at frequently visited and hence better explored states. Evaluated on a 30×30 Deep Sea.

6.3 Experimental Study on Bsuite

In Figure 2 we compare to the DQN and Bootstrapped DQN reference implementations from Bsuite. The latter uses an ensemble of 20 agents combined with random prior functions (Osband, Aslanides, and Cassirer 2018). Our agent matches Bootstrapped DQN in important parameters such as network capacity, number of training steps and optimizer settings. We strive to minimize changes for better comparability – see Algorithm 3 and appendix for implementation details. However note that Bootstrapped DQN maintains 20 separate agent copies in an ensemble increasing the number of parameters and computational requirements significantly.

Empirically we observe on-par performance of Epistemic Q-Learning and Bootstrapped DQN: Both score nearly equal across all dimensions. In particular they score high in the exploration dimension, where regular DQN is barely better than the random agent. On the other hand they are both a bit worse than DQN in the scale dimension. Epistemic Q-Learning can however trade off the performance on scale vs. exploration by hyper parameter selection (see Figure 6).

6.4 Probing for Epistemic Uncertainty

In Figure 5 we ask the fundamental question whether our predicted epistemic uncertainty makes sense empirically: i.e. if uncertainty is high at unknown states and low at frequently visited states. We observe affirmative evidence from the following sanity check: we consider a 30×30 Deep Sea and collect 100 episodes with a uniformly random policy, while estimating epistemic Q-values using the Epistemic Q-Learning algorithm. In Figure 5 we compare the actual number of visits (x-axis) with the posterior standard deviation of the Q-values

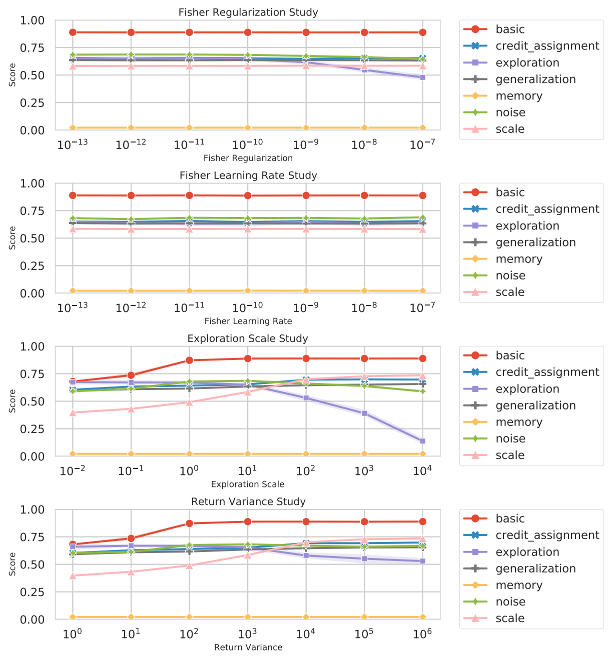

Figure 6: Parameter sensitivity study of Epistemic Q-Learning broken down by Bsuite capability using 10 holdout seeds. Centered values on the x-axis are the default parameters. Observe that parameters are robust across multiple orders of magnitude.

at each state (y-axis). To simplify the plot we group states into visit count ranges. One can observe a clear correlation where uncertainty roughly decreases the more often a state is visited. In particular unknown states with zero visitations exhibit a significantly larger uncertainty than visited states.

6.5 Parameter Stability

An ideal algorithm should be robust to hyper-parameters and consequently require minimal tuning. In Figure 6 we observe that Epistemic Q-Learning is robust with respect to all parameters: to both the Fisher regularization and Fisher learning rate across 5 orders of magnitude (from 10^{-13} to 10^{-8}). The exploration scale and return variance parameters trade-off exploration and exploitation i.e. too large values seem to hinder exploration while improving the scale capability. Good trade-offs are achieved in the range of 1 to 10^2 for the exploration scale and 10^2 to 10^6 for the return variance.

6.6 Ablations

In Figure 7 we ablate key components of the Epistemic Q-Learning agent (Algorithm 3) and evaluate the corresponding Bsuite exploration score to answer the following questions:

- **What happens if we replace the Thomson Sampling acting?** We replace it by ϵ-greedy as in DQN. Note that we keep the Thomson-Sampling inspired bootstrapping as is. We observe a drop in performance.
- **What happens if we replace the Thomson-Sampling inspired value-bootstrapping?** We replace line 4 in

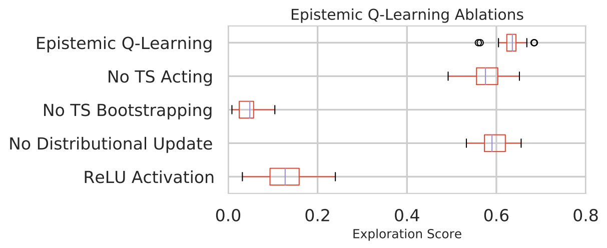

Figure 7: Various ablations of Epistemic Q-Learning applied to Bsuite with 30 seeds (experimental details in Section 6.6).

the learning step of Algorithm 3 by a distributional update with the Q-function at θ^{MLE} by setting: $Z_t = R_{t+1} + \gamma \max_a q_{\theta^{\text{MLE}}}(S_{t+1}, a) + \gamma\eta'$. We observe a large drop in performance indicating that bootstrapping from a posterior sampled $q_{\theta'}$ is important for exploration.
- **What happens if we remove the distributional update?** When estimating the Fisher in Algorithm 3 line 11 with a regular non-distributional Q-Learning update g_t^{MLE} we observe a drop in performance and increased variance.
- **What happens if we use ReLU activations?** We use Leaky-ReLU activations (Maas, Hannun, and Ng 2013; Xu et al. 2015) as they have a non-zero gradient almost everywhere. We observe a dramatic drop in performance if we use ReLU activations instead and hypothesize that this is due to the zero gradient it exhibits on all negative input values interacting with the posterior estimation. Note that the DQN-baseline performance does not improve with Leaky-ReLUs. Hence Leaky-ReLUs are a prerequisite for efficient exploration using EVE, but not its cause.

7 Conclusion

Motivated by statistical theory we introduced a principled recipe for posterior approximation (EVE) that is compatible with sequential decision making and with neural network function approximation. In practice it requires moderate computational and memory overhead and no architecture changes besides introducing Leaky-ReLU activations. We applied the recipe to compute epistemic value uncertainty in a Q-Learning agent and used it for Thomson-Sampling inspired exploration. The proposed epistemic Q-Learning agent exhibits competitive performance on a series of general reinforcement learning and dedicated exploration benchmarks. It is robust to hyper-parameters and matches the performance of Bootstrapped DQN on Bsuite with $20\times$ fewer agent parameters.

To facilitate understanding and analysis, the presented Epistemic Q-Learning agent was designed with a focus on simplicity. It can be extended in many ways: with more expressive distributional representations (Bellemare, Dabney, and Munos 2017), more-sophisticated Fisher approximations (Martens 2014), or better use of epistemic uncertainty in the policy construction (Nikolov et al. 2019). While we employed the approximate posterior for exploration, future work may evaluate its benefits for other applications such as safety in reinforcement learning or robust offline learning.

References

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

Azizzadenesheli, K.; Brunskill, E.; and Anandkumar, A. 2018. Efficient Exploration Through Bayesian Deep Q-Networks. In *2018 Information Theory and Applications Workshop (ITA)*, 1–9.

Bellemare, M. G.; Dabney, W.; and Munos, R. 2017. A Distributional Perspective on Reinforcement Learning. In Precup, D.; and Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, 449–458. PMLR.

Bellman, R. 1957. *Dynamic Programming*. Princeton University Press.

Bernstein, S. N. 1917. *The Theory of Probabilities*.

Blundell, C.; Cornebise, J.; Kavukcuoglu, K.; and Wierstra, D. 2015. Weight Uncertainty in Neural Network. In Bach, F.; and Blei, D., eds., *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, 1613–1622. Lille, France: PMLR.

Botev, A.; Ritter, H.; and Barber, D. 2017. Practical Gauss-Newton Optimisation for Deep Learning. In Precup, D.; and Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, 557–565. PMLR.

Burda, Y.; Edwards, H.; Storkey, A.; and Klimov, O. 2019. Exploration by random network distillation. In *International Conference on Learning Representations*.

Chua, K.; Calandra, R.; McAllister, R.; and Levine, S. 2018. Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Curi, S.; Berkenkamp, F.; and Krause, A. 2020. Efficient Model-Based Reinforcement Learning through Optimistic Policy Search and Planning. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 14156–14170. Curran Associates, Inc.

Daxberger, E.; Kristiadi, A.; Immer, A.; Eschenhagen, R.; Bauer, M.; and Hennig, P. 2021. Laplace Redux - Effortless Bayesian Deep Learning. In Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*.

Dearden, R.; Friedman, N.; and Russell, S. 1998. Bayesian Q-learning. In *Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, 761–768. American Association for Artificial Intelligence. ISBN 0262510987.

Deisenroth, M.; and Rasmussen, C. 2011. PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on Machine Learning*, 465–473.

Duff, M. 2002. *Optimal Learning: Computational procedures for Bayes-adaptive Markov decision processes*. Ph.D. thesis, University of Massachusetts Amherst.

Fortunato, M.; Azar, M. G.; Piot, B.; Menick, J.; Hessel, M.; Osband, I.; Graves, A.; Mnih, V.; Munos, R.; Hassabis, D.; Pietquin, O.; Blundell, C.; and Legg, S. 2018. Noisy Networks For Exploration. In *International Conference on Learning Representations*.

Gal, Y.; and Ghahramani, Z. 2016. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In Balcan, M. F.; and Weinberger, K. Q., eds., *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, 1050–1059. New York, New York, USA: PMLR.

Graves, A. 2011. Practical Variational Inference for Neural Networks. In Shawe-Taylor, J.; Zemel, R.; Bartlett, P.; Pereira, F.; and Weinberger, K., eds., *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc.

Grosse, R.; and Martens, J. 2016. A Kronecker-factored approximate Fisher matrix for convolution layers. In Balcan, M. F.; and Weinberger, K. Q., eds., *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, 573–582. New York, New York, USA: PMLR.

Guez, A.; Silver, D.; and Dayan, P. 2012. Efficient Bayes-Adaptive Reinforcement Learning using Sample-Based Search. In Pereira, F.; Burges, C.; Bottou, L.; and Weinberger, K., eds., *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Hinton, G. E.; and van Camp, D. 1993. Keeping the Neural Networks Simple by Minimizing the Description Length of the Weights. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, COLT '93, 5–13. New York, NY, USA: Association for Computing Machinery. ISBN 0897916115.

Kingma, D. P.; and Adam, J. B. 2015. A method for stochastic optimization. In *International Conference on Learning Representation*.

Korattikara Balan, A.; Rathod, V.; Murphy, K. P.; and Welling, M. 2015. Bayesian dark knowledge. In Cortes, C.; Lawrence, N.; Lee, D.; Sugiyama, M.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Lakshminarayanan, B.; Pritzel, A.; and Blundell, C. 2017. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Laplace, P. S. 1774. Mémoire sur la probabilité des causes par les événements. *Mémoires de Mathématique et de Physique*, 6.

Laplace, P. S. 1810. Mémoire sur les approximations des formules qui sont fonctions de très grands nombres et sur leur application aux probabilités. *Mémoires de l'Académie Royale des Sciences de Paris*, 10: 301–347.

Le Cam, L. 1953. *On some asymptotic properties of maximum likelihood estimates and related Baye's estimates*. University of California Press, Berkeley.

Le Cam, L. 1986. *Asymptotic Methods in Statistical Decision Theory*. Springer.

Maas, A. L.; Hannun, A. Y.; and Ng, A. Y. 2013. Rectifier nonlinearities improve neural network acoustic models. In *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*.

MacKay, D. J. C. 1992. A Practical Bayesian Framework for Backprop Networks. *Neural Computation*, 4: 448–472.

Martens, J. 2014. New perspectives on the natural gradient method. *CoRR*, abs/1412.1193.

Martens, J.; Ba, J.; and Johnson, M. 2018. Kronecker-factored Curvature Approximations for Recurrent Neural Networks. In *International Conference on Learning Representations*.

Martin, J. 1967. *Bayesian decision problems and Markov chains*. Wiley.

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. 2023. *Probabilistic Machine Learning: Advanced Topics*. MIT Press.

Neal, R. M. 1995. *Bayesian Learning for Neural Networks*. Ph.D. diss., University of Toronto.

Nickl, R. 2012. Statistical Theory.

Nikolov, N.; Kirschner, J.; Berkenkamp, F.; and Krause, A. 2019. Information-Directed Exploration for Deep Reinforcement Learning. In *International Conference on Learning Representations*.

Osband, I.; Aslanides, J.; and Cassirer, A. 2018. Randomized Prior Functions for Deep Reinforcement Learning. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Osband, I.; Blundell, C.; Pritzel, A.; and Van Roy, B. 2016. Deep Exploration via Bootstrapped DQN. *CoRR*, abs/1602.04621.

Osband, I.; Doron, Y.; Hessel, M.; Aslanides, J.; Sezener, E.; Saraiva, A.; McKinney, K.; Lattimore, T.; Szepesvari, C.; Singh, S.; Roy, B. V.; Sutton, R.; Silver, D.; and Hasselt, H. V. 2020. Behaviour Suite for Reinforcement Learning. In *International Conference on Learning Representations*.

Ostrovski, G.; Bellemare, M. G.; van den Oord, A.; and Munos, R. 2017. Count-Based Exploration with Neural Density Models. In Precup, D.; and Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, 2721–2730. PMLR.

Plappert, M.; Houthooft, R.; Dhariwal, P.; Sidor, S.; Chen, R. Y.; Chen, X.; Asfour, T.; Abbeel, P.; and Andrychowicz, M. 2018. Parameter Space Noise for Exploration. In *International Conference on Learning Representations*.

Ritter, H.; Botev, A.; and Barber, D. 2018. A Scalable Laplace Approximation for Neural Networks. In *International Conference on Learning Representations*.

Rowland, M.; Bellemare, M.; Dabney, W.; Munos, R.; and Teh, Y. W. 2018. An Analysis of Categorical Distributional Reinforcement Learning. In Storkey, A.; and Perez-Cruz, F., eds., *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, 29–37. PMLR.

Schrittwieser, J.; Antonoglou, I.; Hubert, T.; Simonyan, K.; Sifre, L.; Schmitt, S.; Guez, A.; Lockhart, E.; Hassabis, D.; Graepel, T.; Lillicrap, T.; and Silver, D. 2020. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588(7839): 604––609.

Schäcke, K. 2013. On the Kronecker product. Master's Thesis, University of Waterloo, Ontario.

Sutton, R. S.; and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. The MIT press, Cambridge MA.

Tang, Y.; and Agrawal, S. 2018. Exploration by Distributional Reinforcement Learning. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, 2710–2716. International Joint Conferences on Artificial Intelligence Organization.

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

van der Vaart, A. W. 1998. *Asymptotic Statistics*. Cambridge University Press.

von Mises, R. 1931. *Wahrscheinlichkeitsrechnung und ihre Anwendung in der Statistik und theoretischen Physik*, volume 1. Franz Deuticke.

Xu, B.; Wang, N.; Chen, T.; and Li, M. 2015. Empirical Evaluation of Rectified Activations in Convolutional Network. In *ICML Workshop on Deep Learning*.

Zhang, G.; Sun, S.; Duvenaud, D.; and Grosse, R. 2018. Noisy Natural Gradient as Variational Inference. In Dy, J.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 5852–5861. PMLR.

Appendix

A Implementation Details

We build our agent on the reference agent implementations from Bsuite. In particular we strive to minimize all unrelated differences to Bootstrapped DQN to obtain a clear comparison. Notable differences are replacing ReLU with Leaky-ReLU activations and using 10 small batches of size 12 rather than one large batch of size 128 and increasing the replay capacity such that the agent can remember all previous transitions. In particular we maintain the neural network architecture. While Bootstrapped DQN ensembles 20 copies Epistemic Q-Learning uses a single copy.

Compared to the high-level example Algorithm 2 in the paper it inherits a target network (Mnih et al. 2015) and

ADAM optimizer (Kingma and Adam 2015) with learning rate 10^{-3} from the Bsuite reference implementations. Furthermore it performs maximum likelihood optimization with the expected gradient (Equation 10) to reduce variance. See Algorithm 3 for a full description of the agent modulo mini-batching. To simplify the mini-batching implementation we use a single posterior sample θ' to compute all returns in the batch $G_t = R_{t+1} + \gamma \max_a q_{\theta'}(S_{t+1}, a)$. Furthermore we compute the gradient g^{LogL} by summing the prediction loss over all transitions batch $\nabla_\theta \sum_{t \in B}(Z'_t - q_\theta(S_t, A_t))^2$. A gold standard implementation would implement batching by sampling a different posterior sample θ' and computing the gradient g^{LogL} for each transition in the batch. We mitigate the resulting approximation error by using a smaller batch size compared to the baseline (12 rather than 128). Finally it acts uniformly random for the first $K_{\text{burnin}} = 100$ episodes to ensure there the Fisher information matrix has been estimated sufficiently before we sample.

B Variance Reduction of Fisher Estimation

In Step 2 of the recipe we approximate the posterior as $p(\theta|\mathcal{D}) \approx \mathcal{N}(\theta^{\text{MLE}}, \frac{1}{n}\hat{\mathcal{I}}(\theta^{\text{MLE}})^{-1})$. In Section 3 we have discussed how to obtain the θ^{MLE} and the Fisher information matrix $\hat{\mathcal{I}}(\theta^{\text{MLE}})$ by sampling hypothetical returns from the model $Z' \sim f_\theta$. Sampling introduces additional variance, which can however be removed for the Gaussian return model: We discussed variance reduction for the MLE estimation in Section 4.4 and will now focus on the Fisher information matrix:

$$\begin{aligned}\hat{\mathcal{I}}(\theta^{\text{MLE}}) &:= \frac{1}{n}\sum_{t=1}^{n} \mathbb{E}_{Z'_t}\left[\nabla_\theta \log f_{\theta^{\text{MLE}}}(Z'_t) \nabla_\theta \log f_{\theta^{\text{MLE}}}(Z'_t)^\top\right] \\ &:= \frac{1}{n}\sum_{t=1}^{n} \mathbb{E}_{Z'_t}\left[g_t^{\text{LogL}} \;\; g_t^{\text{LogL}\top}\right]\end{aligned}$$

This estimator would become quite expensive if we implemented it naively by summing over n transitions in the dataset and then evaluating each $\mathbb{E}_{Z'_t}[g_t^{\text{LogL}} \;\; g_t^{\text{LogL}\top}]$ numerically by sampling Z'_t say $m \gg 1$ times. Below we present a method to compute the expectation analytically at similar computational cost as $m = 1$.

As the approach below is mathematically a bit more involved one might be tempted to instead use $\frac{1}{n}\sum_{t=1}^{n} \mathbb{E}_{Z'_t}[g_t^{\text{LogL}}]\mathbb{E}_{Z'_t}[g_t^{\text{LogL}}]^\top = \frac{1}{n}\sum_{t=1}^{n} g_t^{\text{MLE}} g_t^{\text{MLE}\top}$. This is tempting as we already have g_t^{MLE} from Section 4.4. Unfortunately this is however not theoretically justified because $\mathbb{E}_{Z'_t}[g_t^{\text{LogL}} g_t^{\text{LogL}\top}] \neq \mathbb{E}_{Z'_t}[g_t^{\text{LogL}}]\mathbb{E}_{Z'_t}[g_t^{\text{LogL}}]^\top$ and empirically results in inferior performance (see Section 6.6, Figure 7: 'No Distributional Update').

Fisher Variance Reduction for the Gaussian Model When employing a Gaussian model $f_\theta(Z) = \mathcal{N}(Z|v_\theta, \sigma = 1)$ with k-step returns G^k we sample hypothetical returns as $Z' = G^k + \gamma^k \eta$ with $\eta \sim \mathcal{N}(0, 1)$. The log-likelihood gradient then simplifies as follows:

$$\begin{aligned}g_t^{\text{LogL}}(\theta) &:= \nabla_\theta \log f_\theta(Z'_t|S_t, A_t) \\ &= \nabla_\theta \frac{1}{2}(Z'_t - q_\theta(S_t, A_t))^2 \\ &= \frac{1}{2}(Z'_t - q_\theta(S_t, A_t))\nabla_\theta q_\theta(S_t, A_t) \\ &= \frac{1}{2}(\underbrace{G_t^k - q_\theta(S_t, A_t)}_{:=\delta_t^{TD}} + \gamma^k \eta)\nabla_\theta q_\theta(S_t, A_t)\end{aligned}$$

Hence we can pull the gradient computation out of the expectation: denoting $\delta_t^{TD} := G_t^k - q_\theta(S_t, A_t)$ and $\nabla_\theta q_\theta := \nabla_\theta q_\theta(S_t, A_t)$

$$\begin{aligned}\mathbb{E}_{Z'_t}\left[g_t^{\text{LogL}}(\theta) g_t^{\text{LogL}}(\theta)^\top\right] &= \mathbb{E}_\eta\left[\frac{1}{4}(\delta_t^{TD} + \gamma^k \eta)^2 \nabla_\theta q_\theta \nabla_\theta q_\theta{}^\top\right] \\ &= \mathbb{E}_\eta\left[\frac{1}{4}(\delta_t^{TD} + \gamma^k \eta)^2\right] \nabla_\theta q_\theta \nabla_\theta q_\theta{}^\top \\ &= \frac{1}{4}\left((\delta_t^{TD})^2 + \gamma^{2k}\right) \nabla_\theta q_\theta \nabla_\theta q_\theta{}^\top\end{aligned}$$

using $\mathbb{E}\left[X^2\right] = \mathbb{E}\left[X\right]^2 + \mathbb{V}\left[X\right]$ in the last step we can compute the expectation analytically. Overall we can now compute the expected Fisher update for transition t with a single gradient computation of $\nabla_\theta q_\theta$ and with n computations for the entire dataset. Furthermore we can share computation with the MLE optimization since both employ the same gradient $\nabla_\theta q_\theta$.

C Posterior Approximation

Exact posterior estimation is intractable in high dimensional parameter spaces hence approximations have been studied extensively (see Murphy 2023, for an overview). Historically the Bernstein-von Mises theorem is an extension and improvement of previous Laplace approximation based attempts to estimate the posterior of a parametric model.

Comparison of Bernstein-von Mises and Laplace Approximations Approximating the posterior with a Gaussian

$$p(\theta|\mathcal{D}) \approx \mathcal{N}(\theta^*, \Sigma)$$

can be justified in two similar ways. Inspired by the frequentist Bernstein-von Mises theorem:

$$p(\theta|\mathcal{D}) \to \mathcal{N}(\theta_n^{\text{MLE}}, \frac{1}{n}\mathcal{I}(\theta^{\text{True}})^{-1})$$

or using the Laplace approximation:

$$p(\theta|\mathcal{D}) \approx \mathcal{N}(\theta_n^{\text{MAP}}, \mathcal{H}(\theta_n^{\text{MAP}})^{-1})$$

Prior to von Mises (1931) it was common to heuristically approximate posteriors using the Laplace formula without sound theoretical understanding of how accurate the approximation is. Hence von Mises (1931) emphasizes that his work provides the missing clarity by proving that the posterior distribution convergences to a Gaussian – a result that he named *the second law of large numbers*. While originally limited to specific statistical models the result has been extended by Le Cam (1953) and referred to as *Bernstein-von Mises theorem*.

While the means in both formulas appear different note that they are equal in the presence of regularization $\theta_n^{\text{MLE}} = \theta_n^{\text{MAP}}$. Now let us compare the covariance matrices: $\mathcal{I}(\theta^{\text{True}})$ is the Fisher information at the unknown true parameters and needs to be approximated. $\mathcal{H}(\theta_n^{\text{MAP}})$ is the Hessian of the negative log-posterior and both hard to estimate and not guaranteed to be positive semi-definite, hence impractical as a covariance matrix. Usually it is heuristically replaced by something more suitable. Two common options are the Fisher information matrix and the Generalized Gauss Newton (Murphy 2023; Botev, Ritter, and Barber 2017; Ritter, Botev, and Barber 2018). Martens (2014) remarks that the Fisher and GGN matrix are equal for exponential families such as the Gaussian model used in our example agent from see Section 4.1. In Section 6 we observed favourable results with the empirical Fisher matrix. Optimizing the matrix choice is out of scope for this paper. We refer to Martens (2014) and Murphy (2023) for a discussion of the matrix choice and conclude that the posterior approximation in Step 2 can be justified both using the Bernstein-von Mises theorem and the Laplace approximation.

Bernstein-von Mises Theorem The Bernstein-von Mises theorem states that the posterior of a parametric model converges to a Gaussian distribution and that ultimately the prior does not matter. While it can be used to relate frequentist and Bayesian inference for large sample sizes, we use it simply for its parametric form.

At the core of frequentist statistics is the existence of an unknown true parameter θ^{True} parameterizing the true distribution from which the samples are observed:

$$X_i \sim f_{\theta^{\text{True}}}$$

The true distribution parameter θ^{True} is then estimated from a set of n samples – e.g. using the maximum likelihood estimator:

$$\theta_n^{\text{MLE}} := \arg\max_{\theta} \prod_{i=1}^{n} f_\theta(X_i) \tag{11}$$

The Bernstein-von Mises theorem states that the Bayesian posterior distribution $p(\theta|X_1, ..., X_n)$ of the parameters θ given the observations X_i and any reasonable Bayesian prior approaches a Gaussian around the maximum likelihood estimate θ^{MLE}:

$$\|p(\theta|X_1, ...X_n) - N(\theta_n^{\text{MLE}}, \frac{1}{n}\mathcal{I}(\theta^{\text{True}})^{-1})\|_{TV} \to 0 \tag{12}$$

In particular it says that for large enough n this holds for any reasonable Bayesian prior (that is continuous and non-zero in a neighbourhood of θ^{True}). Convergence is with respect to the total variation norm (i.e. the worst case event) and in probability (i.e. the chance of the TV norm exceeding any fixed threshold $\epsilon > 0$ converges to zero as the number of samples $X_i \sim f_{\theta^{\text{True}}}$ increases).

The theorem makes further assumptions about parametric models that are standard in frequentist statistics to ensure a convergence rate of the maximum likelihood estimator. Most notable are: $f_\theta(X)$ should be nonzero at all θ and X, f_θ twice continuously differentiable wrt. θ at all θ and X, the Fisher information at θ^{True} should be non-singular. Additionally it assumes a uniformly consistent estimator (see Nickl 2012, for full treatment) which exists for instance if among other conditions $\theta \neq \theta'$ implies $f_\theta \neq f_{\theta'}$ (van der Vaart 1998). Naturally not all of the above conditions can be satisfied with neural networks. In particular the weights of a neural network θ can be permuted while preserving the same function hence violating the final condition – a property that may be amplified with ReLU activations that map an entire input range to zero. Empirically we observed better results with Leaky-ReLU activations.

Laplace Approximation The Laplace approximation is used in Bayesian statistics to approximate the posterior with a Gaussian distribution centered at the maximum a posteriori estimate $\theta_n^{\text{MAP}} := \arg\max_\theta \prod_{i=1}^{n} f_\theta(X_i)p(\theta)$. It uses a second order Taylor expansion of the negative log-posterior for which it requires the Hessian $\mathcal{H}(\theta) := -\nabla_\theta^2 \log p(\theta|\mathcal{D})$.

$$p(\theta|\mathcal{D}) \approx \mathcal{N}(\theta_n^{\text{MAP}}, \mathcal{H}(\theta_n^{\text{MAP}})^{-1})$$

Unfortunately the Hessian is difficult to estimate and may not be positive definite hence it is frequently approximated by one of the following: the generalized Gauss-Newton matrix, the empirical Fisher or a Fisher information matrix estimate at θ^{MAP} – see Martens (2014); Botev, Ritter, and Barber (2017); Murphy (2023) for details. The Laplace approximation as a tool for integration dates back to Laplace (1774), has been applied to neural networks by MacKay (1992), and can be applied to large neural networks using K-FAC (Ritter, Botev, and Barber 2018; Daxberger et al. 2021). When originally proposed by Laplace the function (in our case the posterior) was assumed to have a single maximum so that the Gaussian shape can cover it well.

D Kronecker Factorizing the Fisher information

In the context of a second order optimization Martens (2014) proposes to estimate the Fisher information matrix of neural networks using a block diagonal structure where each block corresponds to a layer in the neural network. Furthermore each block is represented by the Kronecker product of two matrices $A \otimes G$ that are of similar size as the weight matrix W of that layer ($a \times a$ and $g \times g$ for a $g \times a$ sized W). As a consequence the memory and computational complexity of Fisher estimation is similar to the regular gradient computation. The proposed factorization assumes that the gradient per layer is statistically independent of the gradients from all other layers. This permits efficient updates of A and G during the forward and backward pass of the backpropagation algorithm for neural networks. The K-FAC approach supports linear, convolutional and recurrent layers by making further assumptions about the statistical independence of the layers input activations and the gradients of the layers outputs see Martens (2014); Grosse and Martens (2016); Martens, Ba, and Johnson (2018) for details.

This factorization does not only permit efficient estimation, but also efficient inversion and sampling.

Useful Kronecker Identities Let us recall the the following identities (see Schäcke 2013, for an overview): Matrix-vector products can be written using the operator vec that reshapes matrices into vectors and mat that shapes vectors back into matrices:

$$(A \otimes G)x = \mathrm{vec}(G\mathrm{mat}(x)A^\top)$$

Inversion can be computed on each factor:

$$(A \otimes G)^{-1} = A^{-1} \otimes G^{-1}$$

Eigendecompositions can be computed on each factor, where Λ is a diagonal matrix containing the eigenvalues and E contains the eigenvectors:

$$(E_A \Lambda_A E_A^{-1}) \otimes (E_G \Lambda_G E_G^{-1}) = (E_A \otimes E_G)(\Lambda_A \otimes \Lambda_G)(E_A \otimes E_G)^{-1}$$

where $\Lambda_A \otimes \Lambda_G$ is a diagonal matrix containing the eigenvalues of the Kronecker product $A \otimes G$. Now let us define a few helpful quantities: For any eigenvalue matrix Λ let $\Lambda^{-1/2}$ be the diagonal matrix with the inverse of the square root of Λs eigenvalues: $(\Lambda^{-1/2})_{i,i} = 1/\sqrt{\Lambda_{i,i}}$. Finally define $B_A := E_A \Lambda_A^{-1/2}$ and $B_G := E_G \Lambda_G^{-1/2}$.

Efficient Sampling Given the identities from above, we can now sample from $\mathcal{N}(\mu, (A \otimes G)^{-1})$ as follows:

$$z \sim \mathcal{N}(0, I_{ag \times ag})$$
$$\theta' = \mu + (B_A \otimes B_G)z = \mu + \mathrm{vec}(B_G \mathrm{mat}(z) {B_A}^\top)$$

Note that this involves only products of matrices with at most $\max(a, g)$ rows and columns. It also requires eigenvalue decomposition of A and G which can be amortized as in K-FAC over multiple iterations.

E Distributional RL

Distributional RL (Bellemare, Dabney, and Munos 2017) learns a distribution over returns Z. Originally referred to as the 'value distribution' Rowland et al. (2018) argues that 'return distribution function' is a technically more correct name. The latter name emphasizes the difference between returns and values: Recall from the introduction that learning a distribution over returns captures aleatoric uncertainty while a distribution over values captures epistemic uncertainty. By learning a distribution over returns distributional RL captures aleatoric uncertainty.

Bellemare, Dabney, and Munos (2017) introduce a distributional Bellman operator and prove its convergence for policy evaluation with respect to the Wasserstein metric. As the Wassertein metric can not be estimated empirically from sampled transitions a sample Bellman update is proposed that minimizes the KL distance between the predicted return distribution $f_\theta(Z|s, a)$ and the target that estimates $p(Z|s, a)$. See Bellemare, Dabney, and Munos (2017) for details and note that they use different notation: The learnt distribution over returns that we call $f_\theta(Z|s, a)$ corresponds to the random value $Z_\theta(s, a)$ (a random value whose distribution is parameterized by θ e.g. through a neural network). In particular they consider a categorical representation for $f_\theta(Z|s, a)$. By mapping the returns into 51 intervals the model resembles multi-class classification. While the categories result in a discretization error the empirical performance of the corresponding 'C51' agent was state-of-the-art at the time.

Recall that our example agent does not use the categorical return model from C51 and hence does not reap its empirical benefits that Bellemare, Dabney, and Munos (2017) attribute to a combination of representation learning, reduced state-aliasing and better optimization properties of the categorical loss. Instead we use the Gaussian model, whose loss resembles the regular Q-Learning loss (see Section 4.4). We leave the combination of both approaches to future work.

F Miscellaneous

Deep Sea Evaluation Metric Note that the Bsuite developers recently fixed a typo in the Github implementation. This bugfix makes the Deep Sea task easier by reducing the threshold from 20% to only 10% required optimal episodes. Note that this change impacts comparability of papers before and after the change. All experiments and figures in this paper use the more difficult setting that requires 20% successes among all episodes. The presented results are hence directly comparable to all previous publications that used Bsuite but not necessarily to Bsuite experiments conducted after October 5th, 2022.