Efficient exploration with Double Uncertain Value Networks

Moerland, Thomas; Broekens, Joost; Jonker, Catholijn

Publication date
2017

Document Version
Final published version

Published in
Deep Reinforcement Learning Symposium, NIPS 2017

Citation (APA)
Moerland, T., Broekens, J., & Jonker, C. (2017). Efficient exploration with Double Uncertain Value Networks. In *Deep Reinforcement Learning Symposium, NIPS 2017* (pp. 1-17) https://sites.google.com/view/deeprl-symposium-nips2017/home

Efficient exploration with Double Uncertain Value Networks

Thomas M. Moerland, Joost Broekens and Catholijn M. Jonker
Department of Computer Science
Delft University of Technology, The Netherlands
{T.M.Moerland,D.J.Broekens,C.M.Jonker}@tudelft.nl

Abstract

This paper studies directed exploration for reinforcement learning agents by tracking uncertainty about the value of each available action. We identify two sources of uncertainty that are relevant for exploration. The first originates from limited data (*parametric uncertainty*), while the second originates from the distribution of the returns (*return uncertainty*). We identify methods to learn these distributions with deep neural networks, where we estimate parametric uncertainty with Bayesian drop-out, while return uncertainty is propagated through the Bellman equation as a Gaussian distribution. Then, we identify that both can be jointly estimated in one network, which we call the Double Uncertain Value Network. The policy is directly derived from the learned distributions based on Thompson sampling. Experimental results show that both types of uncertainty may vastly improve learning in domains with a strong exploration challenge.

1 Introduction

Reinforcement learning (RL) is the dominant class of algorithms to learn sequential decision-making from data. In RL we start with zero prior knowledge and need to actively collect our own data. Therefore, we should not settle on a policy too early, instead of trying out actions we have not properly explored yet. However, we neither want to continue exploring sub-optimal actions, when we already know what is best. This challenge is known as the exploration/exploitation trade-off.

Most state-of-the-art reinforcement learning implementations use *undirected* forms of exploration, such as ϵ-greedy or Boltzmann exploration. These methods act on *point estimates* of the mean action-value, usually applying some random perturbation to avoid only selecting the currently optimal action. However, these undirected methods are known to be highly inefficient (Osband et al., 2014). By only tracking point estimates of the mean state-action value, these algorithms lack the information to, for example, discriminate between an action that has never been tried before (and requires exploration) and an action that has been tried extensively and deemed sub-optimal (and can be avoided).

A natural solution to this problem originates from tracking uncertainties/distributions. The intuition is that with limited data and large uncertainty there is reason to explore, while narrow distributions naturally transfer to exploitation (see Appendix C for a detailed illustration). For this work we identify two types of uncertainties/distributions that are interesting for exploration:

- *Parametric uncertainty*: This is the classical statistical uncertainty which is a function of the number of available data points. The cardinal example is the posterior distribution of the mean (action-value).
- *Return uncertainty*: This is the distribution over returns from a state-action pair given the policy. For this work we focus on deterministic domains, which makes the return distribution entirely induced by the (exploratory) stochastic policy.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

We argue that - for deterministic environments - we can explore by acting probabilistically optimal with respect to *both* distributions (see Section 3). We identify neural network methods to estimate each of them separately, and subsequently show that both can be combined in one network, which we call the Double Uncertain Value Network (DUVN). To the best of our knowledge, we are the first to 1) distinguish between uncertainty due to limited data (parametric) and uncertainty over the return distribution, 2) propagate both through the Bellman equation, 3) track both with neural networks (i.e., high-capacity function approximators), and 4) use both to improve exploration.[1]

The remainder of this paper is organized as follows. In Section 2 we provide a general introduction to Bayesian deep learning and distributional reinforcement learning. In Section 3, we discuss parametric and return uncertainty, and identify their potential for exploration. Section 4 discusses their implementations for policy evaluation with neural networks, and also discusses how to derive a policy from the learned distributions based on Thompson sampling. Sections 4, 5 and 6 show experimental results, discuss future work, and draw conclusions, respectively.

2 Pre-liminaries

2.1 Bayesian deep learning

Bayesian neural networks (MacKay, 2003) represent the uncertainty in the model through posterior distributions over the model parameters. Assume we observe some random variables X and Y and are interested in the conditional distribution $p(Y|X)$. We introduced a neural network $p_\phi(Y|X)$ with parameters $\phi \in \Phi$ to estimate this conditional distribution. In the Bayesian setting, we treat the model parameters ϕ as random variables themselves. Given an observed dataset $\mathcal{H}$, we may use the posterior distribution over model parameters $p(\phi|\mathcal{H})$ to obtain the posterior predictive distribution

$$p(y^\star|x^\star, \mathcal{H}) = \int p(y^\star|x^\star, \phi)p(\phi|\mathcal{H})\mathrm{d}\phi \tag{1}$$

for a new observed datapoint $x^\star$. In the non-linear neural networks of practical interest, the posterior distribution $p(\phi|\mathcal{H})$ is analytically intractable. Gal and Ghahramani (2016) showed that the well-known empirical procedure drop-out actually produces a Monte-Carlo approximation to Eq. 1, providing samples from the posterior predictive distribution by simply retaining drop-out during test time (prediction). We use this technique in this paper, and discuss alternative methods for Bayesian inference in neural networks in the Future work section.

2.2 Distributional reinforcement learning

In reinforcement learning (RL) (Sutton and Barto, 1998) agents are studied that interact with an unknown environment with the goal to optimize some long-term performance measure. The framework adopts a Markov Decision Process (MDP) given by the tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma\}$. At every time-step t we observe a state $s_t \in \mathcal{S}$ and pick an action $a_t \in \mathcal{A} = \{1...N_\mathcal{A}\}$, for $N_\mathcal{A}$ available discrete actions. The MDP follows the transition dynamics $s_{t+1} = \mathcal{T}(\cdot|s_t, a_t) \in \mathcal{S}$ and returns rewards $r_t = \mathcal{R}(s_t, a_t) \in \mathbb{R}$. For this work, we assume a discrete action space and deterministic transition and reward functions.

We act in the MDP according to a stochastic policy π, i.e. $a \sim \pi(\cdot|s) \in \mathcal{P}(\mathcal{A})$. The discounted return $Z^\pi(s, a)$ from a state-action pair (s, a) is a *random process* given by

$$Z^\pi(s, a) = \sum_{t=0}^{\infty} \gamma^t r_t, \qquad s_{t+1} = \mathcal{T}(\cdot|s_t, a_t), a_{t+1} \sim \pi(\cdot|s_{t+1}), s_0 = s, a_0 = a \tag{2}$$

for discount factor $\gamma \in [0, 1]$. We emphasize that the return Z^π is a random variable, where the distribution of Z^π is induced by the stochastic policy (as we assume a deterministic environment). We

[1] As a side contribution, we introduce the Initial Return Entropy (IRE) as a measure of task exploration difficulty. See Appendix B.

may rewrite equation 2 into a recursive form, known as the *distributional Bellman equation* (omitting the π superscript from now on):

$$Z(s,a) = r_t + \gamma Z(s',a'), \qquad s' = \mathcal{T}(\cdot|s,a), a' \sim \pi(\cdot|s'). \tag{3}$$

Note that the equality sign represents *distributional equality* here (Engel et al., 2005). We are now ready to define the action-value function. Denote by $\mathbb{E}_\pi$ the expectation over all traces induced by the policy π. Applying this operator to $Z(s,a)$ defines the state-action value $Q(s,a) = \mathbb{E}_\pi[Z(s,a)]$. Applying this operator to Eq. 2 gives

$$Q(s,a) = r_t + \gamma \mathbb{E}_{s'=\mathcal{T}(\cdot|s,a),a'\sim\pi(\cdot|s')}[Q(s',a')] \tag{4}$$

which is known as the Bellman equation (Sutton and Barto, 1998). Most RL papers actually start-off from Eq. 4. We present the current introduction to emphasize that the mean action value $Q(s,a)$ is a quantity that we estimate by sampling from an underlying return distribution $p(Z|s,a)$.[2]

We approximate the action-value (distribution) with a deep neural network. We write $Q_\phi(s,a)$ for a network predicting a (point estimate) action-value, and $p_\phi(Z|s,a)$ for a network approximating the entire return distribution. To learn the state-action value RL algorithms follow variants of a scheme known as *generalized policy iteration* (GPI) (Sutton and Barto, 1998). GPI iterates between policy evaluation, in which we calculate a new estimates $\Psi(s,a)$ of the state-action value based on (new) sample data (e.g., for one-step SARSA $\Psi(s,a) = r(s,a) + \gamma Q(s',a')$), and policy improvement, in which we use the estimate $\Psi(s,a)$ to improve the policy (whether with a value-based, actor-critic or policy gradient algorithm).

3 Distributional perspective on exploration

We will now argue for a probabilistic perspective on value functions and exploration. There are two distributions that might be useful from an exploration point of view: 1) the statistical parametric uncertainty of the mean action value, and 2) the distribution of the return.

Parametric uncertainty of the mean Given a policy the state-action value $Q(s,a)$ is a scalar number by definition, as it is an expectation over all possible future traces. However, from a statistical point of view it makes sense to treat our *estimate* of $Q(s,a)$ as a random variable, as we need to approximate it from a finite number of samples. We call this the parametric uncertainty.
Parametric exploration, i.e. acting optimistic with respect to the uncertainty of the mean action-value, has been very successful in the bandit setting. However, it has been sparsely applied to RL (see Appendix A for related work). We believe this is due to a fundamental complication regarding uncertainties in RL, which has only been identified by Dearden et al. (1998) before. Bandits are one-step decision problems with pay-offs originating from a stationary distribution, which makes the value approximation an ordinary supervised learning problem. However, in RL the target distribution is highly non-stationary. A standard target like $r + \gamma Q(s',a')$ falsely assumes that $Q(s',a')$ is known, while it is actually uncertain itself. Therefore, repeatedly visiting a state-action pair should not makes us certain about its value if we are still uncertain about what to do next. In other words: *the state-action value certainty depends on the future policy certainty*. Standard parametric uncertainty cannot account for this problem (the 'local' parametric uncertainty will converge as if it is supervised learning), and we somehow need to propagate the uncertainty of future state-action pairs' value (the 'global' uncertainty) back through the Bellman equation. An illustration is seen in Fig. 1b right, where the uncertainty in ϕ influences both the current and future value estimates (ignoring the Z distributions in that graph for now, as the need to propagate the parametric uncertainty $p(\phi)$ over timesteps already applies when we learn mean action-values).

[2] We empirically observe that the shape of this return distribution strongly differs between domains. This matters because the shape of the return distribution also influences how easily we can estimate its expectation, or some other quantity like an upper confidence bound, from samples. For example, a long, thin right tail in the return distribution - as frequently the case in RL with only a few 'good' traces - may give our mean estimate high variance (it would actually need importance sampling). In Appendix B we visualize return distributions for some well-known domains, and also introduce the *Initial Return Entropy* as a measure of task exploration difficulty.

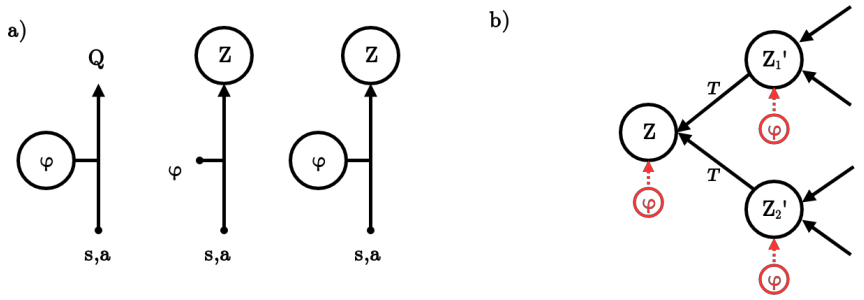

Figure 1: **a)** Three types of neural networks with different uncertanties/probabilitiy distributions. Circles are probabilistic nodes. Left: parametric uncertainty over the mean action-value. Middle: propagating (return) distributions for point estimate parameters. Right: parametric uncertainty over propagating distribution (= Double Uncertain Value Network). **b)** Illustration of propagating distributions. Subscripts identify unique state-action pairs. We initialize all state-action pairs with a prior parametric uncertainty $p(\phi)$ and prior output distribution $p_\phi(Z)$. Then, for a new observed transition, we want to update our estimates of $p_\phi(Z)$ at the current state action pair by propagating the distribution of the next node $p(Z')$ through the Bellman operator T (instead of just propagating the mean). For this work, we consider two quantities to propagate: i) the return distribution at the next node $p_\phi(Z')$ (for point estimate ϕ), or ii) the parametric uncertain return distribution at the next node $p(Z') = \int p_\phi(Z')p(\phi|\mathcal{H})\mathrm{d}\phi$. Arrows point backwards because we focus on the direction of uncertainty propagation/back-up (which runs in the different direction than our exploration).

Return distribution Standard RL, and also the parametric uncertainty introduced above, usually deal with the *mean* action-value $Q(s, a)$. However, from an exploration point of view, it makes more sense to learn the full *return distribution* $p(Z|s, a)$. Note that we still focus on deterministic environments. Therefore, the distribution over returns is solely induced by our own policy. As we may modify our own policy, it makes sense to act optimistically with respect to the return distribution we observe. As an illustration, consider a state-action pair with particular mean action value estimate $Q(s, a)$. It really matters whether this average originates from a highly varying return, or from consistently the same return. It matters because our policy may *influence* the shape of this distribution, i.e. for the highly varying returns we may actively transform the distribution towards the good returns. In other words, what we really care about in deterministic domains is the best return, or the upper end of the return distribution.[3]

It turns out that both challenges focus around propagating either parametric uncertainty and/or return distributions through the Bellman equation (Fig 1b). The overall idea is to memorize the propagating *global* MDP uncertainties in a neural network, which makes them *locally* available at action selection time. We thereby avoid the need for any forward planning (to get global information), and our approach is entirely model-free.

4 Double Uncertain Value Networks

4.1 Policy evaluation

We now discuss three probabilistic policy evaluation approaches that incorporate the uncertainties introduced in the previous section: 1) (local) parametric uncertainty only, 2) return distribution only, and 3) both combined. The respective network structures are illustrated in Fig. 1a. Implementation details are provided in Appendix E.

Parametric uncertainty only To estimate our parametric uncertainty we may use any type of Bayesian inference method suitable for neural networks. For this paper we consider the Bayesian dropout (Gal et al., 2016), as it has a very simple practical implementation (see Sec. 2.1). This gives us a sample from the posterior predictive distribution of the mean action value: $p(Q|s, a, \mathcal{H}) = \int Q_\phi(s, a)p(\phi|\mathcal{H})\mathrm{d}\phi$. The associated network structure is visualized in Figure 1a, left.

[3]For stochastic domains the return distribution has additional noise for which we do want to act on the expectation.

Return distribution only We next consider the problem of learning return distributions instead of mean action-values. For this work we will assume that the return distribution $p(Z|s,a)$ can be approximated by a Gaussian. Therefore, we modify our neural network to output the distribution parameters $\mu^Z(s,a)$ and $\sigma^Z(s,a)$, where clearly $\mu^Z(s,a) = Q(s,a)$. Note that the network parameters ϕ are point estimates now. The associated network structure is visualized in Figure 1a, middle. During policy evaluation we need to estimate distributional targets instead of point estimate targets. We will construct bootstrap estimators based on the distributional Bellman equation (Eq. 3). The derivation for the mean $\mu^Z(s,a) = Q(s,a)$ is well-known from standard RL, so we focus on propagating the return standard deviation through the distributional Bellman equation:

$$\mathrm{Sd}\Big[p(Z|s,a)\Big] = \mathrm{Sd}\Big[r(s,a) + \gamma \sum_{a' \in \mathcal{A}} \pi(a'|s') p(Z|s',a')\Big] = \gamma \sum_{a' \in \mathcal{A}} \pi(a'|s') \mathrm{Sd}\Big[p(Z|s',a')\Big] \quad (5)$$

because $\gamma \geq 0$, $\pi(a|s) \geq 0$, and we assume the next state distributions are independent so we may ignore the covariances.[4] We see that the standard deviation of $p(Z|s,a)$ is a linear combination of the standard deviations $p(Z|s',a')$ (one timestep ahead), reweighted by the policy probabilities and shrunken by γ. We approximate the sum over the policy probabilities $\pi(a'|s')$ by sampling from our policy (as is the usual solution in RL, which will be right in expectation over multiple traces). The network may then be trained to move the current predictions closer to these targets, for example with a squared loss

$$L(\phi) = \Big(r(s,a) + \gamma \mu^Z_\phi(s',a') - \mu^Z_\phi(s,a)\Big)^2 + \Big(\gamma \sigma^Z_\phi(s',a') - \sigma^Z_\phi(s,a)\Big)^2 \quad (6)$$

where we as usual fix the bootstrap predictions at (s',a'), i.e. the training gradients w.r.t. ϕ are blocked there. This approach can be seen as a form of analytic approximate return propagation with a (heuristic) distributional loss (see Appendix A.2 for other distributional losses). Similar ideas with approximate return propagation were recently explored with discrete network output distributions (Bellemare et al., 2017), which may also accommodate for propagating multimodality.

A second, more simple propagation method which we also experimented with is sampling-based propagation. In that setting we sample M values $z_m(s',a') \sim p_\phi(Z|s',a')$, push these through the Bellman operator to construct $\Psi_m(s,a) = r(s,a) + \gamma z_m(s',a')$, and train our network on this collection of samples with, e.g., a maximum likelihood loss. This may require more samples and be less accurate, but it will also work for complicated network output distributions (like deep generative models) for which analytic propagation and projection is infeasible. Results of this approach are not shown, but were comparable to the results with approximate return propagation shown in Section 5.

Parametric uncertainty over return distributions We finish with the observation that both ideas may actually naturally be combined in one function approximator (Fig 1a, right). Note that we can now propagate both the return distribution and its parametric uncertainty at the next timestep, i.e. we are effectively propagating *uncertain return distributions* (the parametric uncertainty over the network output distribution). Starting from a sampled transition now, we want to propagate the return distribution weighted over the parametric uncertainty at the next timestep:

$$Z(s,a) = \int \Big[r + \gamma Z_\phi(s',a')\Big] p(\phi|\mathcal{H}) \mathrm{d}\phi = r + \gamma \int Z_\phi(s',a') p(\phi|\mathcal{H}) \mathrm{d}\phi \quad (7)$$

Besides that, the same distributional Bellman propagating machinery applies as above.[5] We refer to the general mechanism of uncertainty propagation (parametric, return or both) as *Bellman uncertainty*. The appearance of the network, with both uncertainty over the network parameters ϕ and over the output distribution to track the propagating (uncertain) return distributions, makes us refer to it as the

[4] For random variables X, Y and scalar constants a, b, c we have: $\mathrm{Var}[a + bX + cY] = b^2\ \mathrm{Var}[X] + c^2\ \mathrm{Var}[Y] + 2bc\ \mathrm{Cov}[X,Y]$.

[5] Sample from $p(\phi|\mathcal{H})$ at the next time-step, make network predictions $\mu^Z_\phi(s',a')$ and $\sigma^Z_\phi(s',a')$, and do the Bellman propagation. Repeated sampling of ϕ does Monte Carlo integration over $p(\phi|\mathcal{H})$, as a numerical integration like in Dearden et al. (1998) is infeasible for the neural network setting.

Double Uncertain Value Network (DUVN) (Fig 1a, right). The intuition is that during early learning we will mostly be propagating uncertainty, while with converging distributions we will eventually start propagating true return distributions.

In summary, we identified three types of probabilistic policy evaluation algorithms (with the three associated network structures visualized in Fig. 1a):[6]

1. The (local) parametric uncertainty of the mean value: $p(Q|s,a,\mathcal{H}) = \int Q_\phi(s,a)p(\phi|\mathcal{H})\mathrm{d}\phi$.
2. The (propagating) distribution of the return: $p_\phi(Z|s,a,\mathcal{H})$ (with point parameters ϕ).
3. Both, (propagating) uncertain return distr.: $p(Z|s,a,\mathcal{H}) = \int p_\phi(Z|s,a,\mathcal{H})p(\phi|\mathcal{H})\mathrm{d}\phi$.

4.2 Policy improvement

We now describe how to use any of these distributions to naturally balance exploration versus exploitation, based on Thompson sampling (Thompson, 1933) (see Appendix C as well). To generalize notation, we introduce a new random variable Θ with distribution $p(\Theta|s,a)$ to capture any of the three policy evaluation distributions introduced in the previous section. We write $p(\mathbf{\Theta}|s) = \prod_{a^\star \in \mathcal{A}} p(\Theta|s,a^\star)$ for the joint action-value distribution in a state s, where we assume the posterior distributions per action are independent. Thompson sampling selects action a with probability equal to:

$$\pi(a|s) = \int p(\Theta_a > \Theta_{a^\star \neq a}) p(\mathbf{\Theta}|s)\mathrm{d}\mathbf{\Theta} \tag{8}$$

where $\Theta_a = \Theta(s,a)$ and $\Theta_{a^\star \neq a}$ notational convention for $\Theta(s,a^\star)\forall a^\star \in \mathcal{A}, a^\star \neq a$. In words, we choose action a with probability equal to the probability that the specific action is the optimal one when averaging over all uncertainty in the joint distribution $p(\mathbf{\Theta}|s)$. The practical implementation of Thompson sampling is very simple, as we may just sample from $p(\Theta|s,a)$ for every a and argmax over these values:

1. Sample $\phi \sim p(\phi)$ (or equivalently a dropout mask).
2. Sample $Z(s,a^\star) \sim p_\phi(Z|s,a^\star) \quad \forall \quad a^\star \in \mathcal{A}$.
3. Select $a = \arg\max_{a^\star \in \mathcal{A}} Z(s,a^\star)$.

If we do not consider parametric uncertainty, then we ignore the first sampling step and just use the current parameter point estimates. If we do not consider the Bellman uncertainty, then we replace the second sampling step with a deterministic prediction $Q_\phi(s,a)$.

Thompson sampling is not the only possible choice to make decisions under uncertainty, but it has shown good empirical performance in the bandit literature (Chapelle and Li, 2011). It naturally performs policy improvement, as it gradually starts to prefer the better actions when the distributions start narrowing/converging. We thereby hope to improve on the instability of greedy policy improvement (see also Bellemare et al. (2017)) or undirected exploration. Ideally, the uncertain return distribution would gradually narrow and for a deterministic environment eventually converge to a Dirac distribution on the optimal value function.

5 Experiments

We now evaluate the different types of probabilistic policy evaluation in combination with Thompson sampling exploration. We refer to Thompson sampling on the three types of discussed policy evaluation as *parametric exploration*, *return exploration*, and *uncertain return exploration*. Experimental details are provided in Appendix E.

We first consider the Chain domain (Appendix D, Figure 6). The domain consists of a chain of states of length N, with two available actions at each state. The only trace giving a positive, non-zero

[6] We could think of another algorithm that does propagate (i.e., has a probabilistic network output), but only propagates the parametric uncertainty of the mean at the next time-step $p(Q'|\mathcal{H}) = \int Q'_\phi p(\phi|\mathcal{H})\mathrm{d}\phi$ (and not the entire return distribution). We did not come up with such an algorithm, but concurrently with our work, O'Donoghue et al. (2017) did focus on this problem. See Related Work.

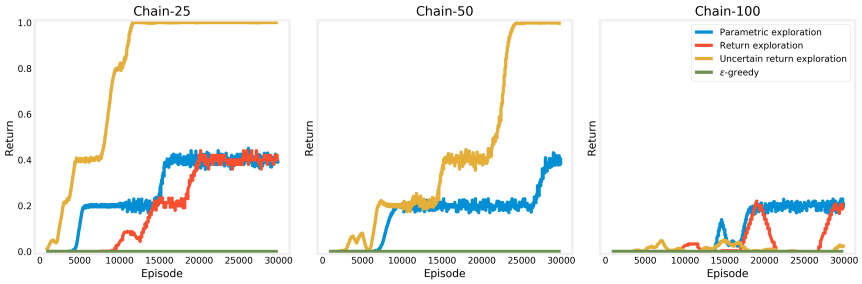

Figure 2: Learning curves on Chain domain for Thompson sampling on parametric uncertainty, return distribution and uncertain return distribution versus ϵ-greedy exploration ($\epsilon = 0.05$). Plots progress row-wise for increased depth of the Chain, i.e. increased exploration difficulty. Note that the correct action at each state in the chain is initialized at random (i.e. not always action 2, as in the visualization in Fig. 6). Results averaged over 5 repetitions.

reward is to select the 'correct' action at every step. The correct action per step is determined at domain initialization by sampling from a uniform Bernoulli. The domain has a strong exploration challenge, which grows exponentially with the length of the chain (see Appendix D).

Learning curves for the Chain domain are shown in Fig. 2, for different lengths of the chain. First of all, we note that the ϵ-greedy strategy does not learn in this domain at all (not even for the short length). The three probabilistic approaches do explore, with best performance for the uncertain return exploration. In the longest chain, of length 100, all probabilistic exploration methods also get trouble solving the domain. However, they do see some rewards, which makes us hypothesize this could be an issue of stabilization (more than that the exploration does not work). See Appendix D.1 for results when the correct action is always the same, as in the original variants of this problem (Osband et al., 2016).

We next test our method on a set of tasks from the OpenAI Gym repository (Fig. 3). We see that our exploration methods manage to learn on all domains. The achieved end policies all reflect good policies on each problem. ϵ-greedy exploration is a bit unstable on some domains (CartPole, LunarLander), but generally performs reasonable as well. We note that the uncertainty exploration methods, which have a completely different exploration mechanism compared to ϵ-greedy exploration, never really perform worse on these domains.

We hypothesize these domains have too much structure and are not challenging enough to show the same exploration difference as seen for the Chain domain. Future work should address more challenging (high-dimensional) exploration problems. We also want to stress that probabilistic exploration will not always outperform undirected methods, especially not on domains with relatively simple exploration. Uncertainty methods will generally create a cautious agent, that first wants to properly verify all parts of the domain. In contrast, undirected exploration agents may exploit sooner, which can be beneficial in domains with non-deep exploration (i.e., with quick rewards).

6 Future work

We identify several directions for future work:

1. **Other types of Bayesian inference in neural networks** (for parametric uncertainty): we hypothesize that the Bayesian drop-out may be too unstable and tedious to tune, as we sometimes observed in our experiments as well. Potentially, different methods to approximate the posterior over the network parameters (e.g., Welling and Teh (2011)) may improve estimation of parametric uncertainty.
2. **More expressive output distributions** (for Bellman uncertainty propagation): for this work we only experimented with Gaussian distributions for propagation. Recently, Bellemare et al. (2017) studied return distribution propagation with categorical distributions, which more naturally accommodate for multi-modality (see Related Work as well). Another extension could involve more expressive continuous network distributions, e.g. based on conditional variational inference (Moerland et al., 2017).

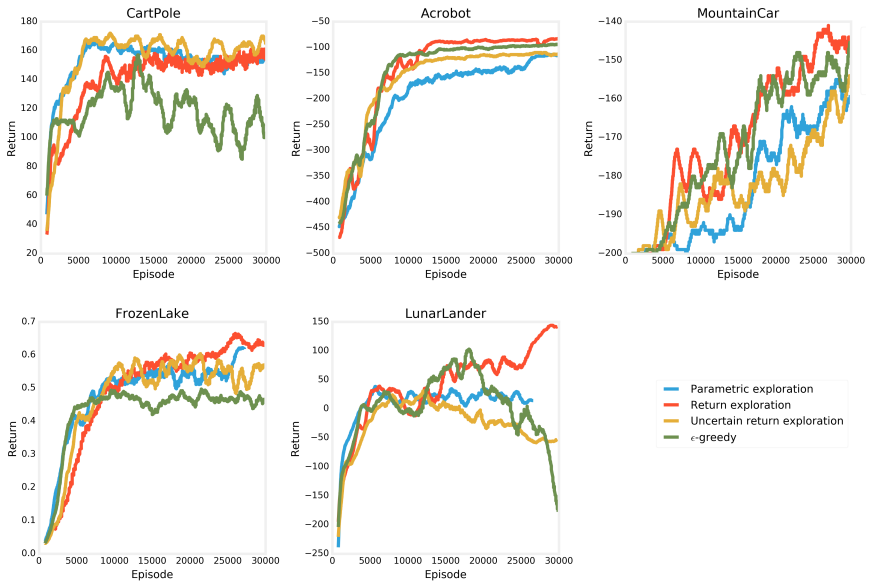

Figure 3: Learning curves for parametric exploration, return exploration and uncertain return exploration on different OpenAI Gym environments. Results averaged over 5 repetitions.

3. **Continuous action-spaces**: the current implementations only focussed on discrete action spaces, where Thompson sampling can easily be applied by maintaining a distribution per action and enumerating all actions for action selection. Extension to the continuous setting would require either directly propagating policy uncertainty, or learning a parametric policy whose distribution mimics the uncertainty in the value function.
4. **Stochastic environments**: this paper entirely focussed on domains with deterministic reward and transition functions, which makes the return distribution only induced by the stochastic policy. In stochastic domains the return distribution will have additional noise for which we do want to act on the expectation, to prevent continuing to act optimistically with respect to something we can't influence.

Finally, we want to stress that the RL algorithms in this paper are entirely model-free. The uncertainty theme also appears in model-based RL, where it is useful/necessary to track the uncertainty on an estimated transition and/or reward function (Deisenroth and Rasmussen, 2011; Depeweg et al., 2017). This parametric *model* uncertainty is different from the parametric *value/policy* uncertainty studied in this work, but our ideas may be extended to the model-based setting as well (which would add another source of uncertainty).

7 Conclusion

This paper introduced Double Uncertain Value Networks (DUVN), which, to the best of our knowledge, is the first algorithm that 1) distinguishes between uncertainty due to limited data (parametric) and uncertainty over the return distribution, 2) propagates both through the Bellman equation, 3) tracks both with neural networks (i.e., high-capacity function approximators), and 4) uses both to improve exploration. We implemented the DUVN algorithm with Bayesian dropout for the parametric uncertainty and a Gaussian distribution for the Bellman uncertainty propagation. The main appeal of this implementation is its simplicity: any deep Q-network implementation can be easily extended as in this work by adding drop-out to the neural network layers and specifying a Gaussian output distribution instead of a mean-only prediction. This should take no more then a few lines of code in most automatic differentiation software packages. We showed that, even for the vanilla implementation, we at least match or improve undirected exploration performance on a variety of problems, and drastically improve performance on an exploration heavy domain (Chain). We believe further improvements in the distributional approach to RL, e.g. with more expressive network output distributions that capture multi-modality, is a promising direction for RL exploration research.

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.

Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. (2016). Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479.

Bellemare, M. G., Dabney, W., and Munos, R. (2017). A distributional perspective on reinforcement learning. *arXiv preprint arXiv:1707.06887*.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*.

Brafman, R. I. and Tennenholtz, M. (2002). R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct):213–231.

Chapelle, O. and Li, L. (2011). An empirical evaluation of thompson sampling. In *Advances in neural information processing systems*, pages 2249–2257.

Dearden, R., Friedman, N., and Andre, D. (1999). Model based Bayesian exploration. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 150–159. Morgan Kaufmann Publishers Inc.

Dearden, R., Friedman, N., and Russell, S. (1998). Bayesian Q-learning. In *AAAI/IAAI*, pages 761–768.

Deisenroth, M. and Rasmussen, C. E. (2011). PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472.

Depeweg, S., Hernández-Lobato, J. M., Doshi-Velez, F., and Udluft, S. (2017). Uncertainty Decomposition in Bayesian Neural Networks with Latent Variables. *arXiv preprint arXiv:1706.08495*.

Engel, Y., Mannor, S., and Meir, R. (2003). Bayes meets Bellman: The Gaussian process approach to temporal difference learning. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 154–161.

Engel, Y., Mannor, S., and Meir, R. (2005). Reinforcement learning with Gaussian processes. In *Proceedings of the 22nd international conference on Machine learning*, pages 201–208. ACM.

Gal, Y. (2016). *Uncertainty in deep learning*. PhD thesis, PhD thesis, University of Cambridge.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059.

Gal, Y., McAllister, R. T., and Rasmussen, C. E. (2016). Improving PILCO with bayesian neural network dynamics models. In *Data-Efficient Machine Learning workshop*, volume 951, page 2016.

Ghavamzadeh, M. and Engel, Y. (2007a). Bayesian actor-critic algorithms. In *Proceedings of the 24th international conference on Machine learning*, pages 297–304. ACM.

Ghavamzadeh, M. and Engel, Y. (2007b). Bayesian policy gradient algorithms. In *Advances in neural information processing systems*, pages 457–464.

Guez, A., Silver, D., and Dayan, P. (2012). Efficient Bayes-adaptive reinforcement learning using sample-based search. In *Advances in Neural Information Processing Systems*, pages 1025–1033.

Houthooft, R., Chen, X., Duan, Y., Schulman, J., De Turck, F., and Abbeel, P. (2016). Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pages 1109–1117.

Kearns, M. and Singh, S. (2002). Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2-3):209–232.

Kocsis, L. and Szepesvári, C. (2006). Bandit based monte-carlo planning. In *ECML*, volume 6, pages 282–293. Springer.

MacKay, D. J. (2003). *Information theory, inference and learning algorithms*. Cambridge university press.

Mannor, S., Simester, D., Sun, P., and Tsitsiklis, J. N. (2004). Bias and variance in value function estimation. In *Proceedings of the twenty-first international conference on Machine learning*, page 72. ACM.

Mannor, S., Simester, D., Sun, P., and Tsitsiklis, J. N. (2007). Bias and variance approximation in value function estimates. *Management Science*, 53(2):308–322.

Mannor, S. and Tsitsiklis, J. (2011). Mean-variance optimization in Markov decision processes. *arXiv preprint arXiv:1104.5601*.

Matthias, P., Rein, H., Prafulla, D., Szymon, S., Richard Y., C., Xi, C., Tamim, A., Pieter, A., and Marcin, A. (2017). Parameter Space Noise for Exploration. *arXiv preprint arXiv:1706.01905*.

Moerland, T. M., Broekens, J., and Jonker, C. M. (2017). Learning Multimodal Transition Dynamics for Model-Based Reinforcement Learning. *arXiv preprint arXiv:1705.00470*.

Morimura, T., Sugiyama, M., Kashima, H., Hachiya, H., and Tanaka, T. (2012). Parametric return density estimation for reinforcement learning. *arXiv preprint arXiv:1203.3497*.

O'Donoghue, B., Osband, I., Munos, R., and Mnih, V. (2017). The Uncertainty Bellman Equation and Exploration. *arXiv preprint arXiv:1709.05380*.

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. (2016). Deep exploration via bootstrapped DQN. In *Advances in Neural Information Processing Systems*, pages 4026–4034.

Osband, I., Van Roy, B., and Wen, Z. (2014). Generalization and exploration via randomized value functions. *arXiv preprint arXiv:1402.0635*.

Rasmussen, C. E., Kuss, M., et al. (2003). Gaussian Processes in Reinforcement Learning. In *NIPS*, volume 4, page 1.

Sobel, M. J. (1982). The variance of discounted Markov decision processes. *Journal of Applied Probability*, 19(4):794–802.

Stadie, B. C., Levine, S., and Abbeel, P. (2015). Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Tamar, A., Di Castro, D., and Mannor, S. (2016). Learning the variance of the reward-to-go. *Journal of Machine Learning Research*, 17(13):1–36.

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

van Hoof, H., Tanneberg, D., and Peters, J. (2017). Generalized exploration in policy search. *Machine Learning*, 106(9-10):1705–1724.

Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 681–688.

White, D. (1988). Mean, variance, and probabilistic criteria in finite Markov decision processes: a review. *Journal of Optimization Theory and Applications*, 56(1):1–29.

A Related work

Exploration is a widely studied topic in reinforcement learning. We will discuss work based on parametric (value/policy) uncertainty, return distributions/uncertainty, and add some context on other exploration approaches (count-based/intrinsic motivation) and other uncertainty methods in RL (uncertainty in model-based RL).

A.1 Parametric uncertainty of the mean

This research direction uses the uncertainty of the mean action value, either from a frequentist or Bayesian direction, to direct exploration. Exploration is usually based on 'optimism under uncertainty'. Parametric uncertainty has been extensively studied in the *bandit* setting, which are environments with a single state, multiple actions and unknown, stochastic rewards. Succesful approaches are UCB Auer et al. (2002), which acts on the upper confidence bound of a frequentist confidence interval, and Thompson sampling Thompson (1933), which is also studied in this paper.

There are a few extensions of these ideas to the RL/MDP setting. The first occurrence of parametric uncertainty in RL seems to be Bayesian Q-learning by Dearden et al. (1998). The authors use tabular Q-learning with normal distributions and conjugate updating. They are also the only ones that explicitly identify the necessity to propagate parametric uncertainty from future states. Their exploration is based on either Thompson sampling (which they call Q-value sampling), while they also consider myopic value of perfect information (VPI) as another exploration strategy.

Osband et al. (2014) extended these ideas to the linear function approximation setting with randomized least-squares value iteration (RLSVI). In the neural network context, parametric uncertainty based on variational inference was studied for bandits by Blundell et al. (2015). Gal and Ghahramani (2016) studied the use of dropout uncertainty for parametric value uncertainty similar to our work, but did not consider any propagation, nor the distribution over returns. Osband et al. (2016) also studied parametric exploration in RL with neural networks, using the non-parametric bootstrap (i.e., a frequentist approach to uncertainty estimation, not to be confused with the use of the term 'bootstrapping' in RL).

Concurrently with the present paper, O'Donoghue et al. (2017) also identified the need to propagate parametric uncertainty over timesteps. Their approach is based on a variance estimate, which has a similar role as the σ in our Gaussian uncertainty propagation. Their neural network implementation derives the local parametric uncertainty estimates from the linearity of their last network layer and frequentist uncertainty estimates known from linear regression. This contrasts to our Bayesian approach to parametric uncertainty. Moreover, they still propagate uncertainty about the mean action value, and do not consider the returns as in our paper.

There is more work that does track uncertainty for policy evaluation, but does not use these for policy improvement / exploration. Most of these have focussed on Gaussian Process regression (Engel et al., 2003). Rasmussen et al. (2003) uses two Gaussian Processes: one to track the parametric uncertainty in the value, and a second one to model the uncertainty in the transition model. However, the paper still uses a greedy policy improvement. The Gaussian Process approach was also extended to continuous action spaces. There are actor-critic (Ghavamzadeh and Engel, 2007a) and policy search (Ghavamzadeh and Engel, 2007b) algorithms that track the uncertainty in the *gradients*, but again only to stabalize the update, not to direct exploration.

The idea of exploration based on parametric uncertainty also connects to the difference between *action space* and *parameter space* / episode-based exploration (Matthias et al., 2017; van Hoof et al., 2017). Most (undirected) exploration methods, like ϵ-greedy and Boltzmann, inject exploration noise at the action space level. However, it can be beneficial to inject the noise at parameter level instead, usually because it allows you to retain a particular noise setting over multiple steps (e.g. an entire episode). The risk of action-space exploration noise is that the agent has to redecide at every timestep and therefore cannot stick with an exploration decision. The effect might be jittering behaviour between exploration and exploitation steps. This has also been identified as the challenge of ensuring 'deep' exploration Osband et al. (2016). We have not considered this problem in this paper, but it could for example be implemented by fixing the dropout mask over an entire episode.

We also want to note that the exact same exploration problem occurs in classical (tree) search. Succesful Monte Carlo Tree Search (MCTS) algorithms, like Upper Confidence Bounds for Trees

(UCT) Kocsis and Szepesvári (2006), act on the upper confidence bound of a frequentist confidence interval of the value at each state-action pair. The overlap between reinforcement learning and (model-based) search has been identified for long Sutton and Barto (1998), where RL i) does not assume an a-priori known environment model, and ii) usually includes a parametric function approximator to represent the value/policy, while search stores these in a tree structure (which is technically an effective sparse form of tabular representation). But besides that, the same exploration themes appear in both fields.

A.2 Return uncertainty

While the distributional Bellman equation (Eq. 3) is certainly not new Sobel (1982); White (1988), nearly all RL research has focussed on the mean action-value. Most papers that do study the underlying return distribution study the 'variance of the return'. Engel et al. (2005) learned the distribution of the return with Gaussian Processes, but did not use it for exploration. Tamar et al. (2016) studied the variance of the return with linear function approximation. Mannor and Tsitsiklis (2011) theoretically studies policies that bound the variance of the return.

The variance of the return does not need to be used with 'optimism under uncertainty', and actually has more frequently been considered for *risk-sensitive RL*. In several scenarios we may want to avoid incidental large negative pay-offs, which can e.g. be desastrous for a real-world robot, or in a financial portfolio. Morimura et al. (2012) studied parametric return distribution propagation as well. They do risk-sensitive exploration by softmax exploration over *quantile* Q-functions (also known as the *Value-at-Risk* (VaR) in financial management literature). Their distribution losses are based on KL-divergences (including Normal, Laplace and skewed Laplace distributions), which could be a better distributional loss than the heuristic loss in Eq. 6. However, their implementations do remain in the tabular setting.

Recently, Bellemare et al. (2017) theoretically studied the distributional Bellman operator. The authors show that the operator is still a contraction in the policy evaluation setting, but not a contraction in any distribution metric for the control setting. They hypothesize this is due to the 'inherent instability of greedy updates' in the Bellman optimality operator. Their algorithm (called C51) uses a categorical distribution to propagate returns distributions, which may more easily accommodate for multimodality compared to the Gaussian distribution used in this work. C51 backs-up the complete Bellman distributions, but they do not use these for exploration. Their methods nevertheless improves over all other previous deep Q-networks on Atari games.

A.3 Count-based Exploration & Intrinsic Motivation

Count-based exploration uses a slightly different incentive for exploration, focussing or rewarding regions of state-space that have not been visited (often). These ideas were extensively studied in the *tabula rasa* setting, e.g. R-max Brafman and Tennenholtz (2002) and Explicit-Explore or Exploit (E^3) Kearns and Singh (2002). Guez et al. (2012) explicitly plans ahead using Monte Carlo Tree Search over uncertain transition dynamics models. Applications in high-dimensional domains include Stadie et al. (2015) and Bellemare et al. (2016).

Intrinsic motivation generalizes this notion of novelty to any internal reward for *domain-independent* characteristics, i.e. next to the domain-dependent external reward function. An example is rewarding actions that decreases the parametric uncertainty in the transition model (Houthooft et al., 2016). Alltogether, this class of exploration methods usually depends on the ability to learn good transition models (from limited data), a problem which is not trivial itself (Deisenroth and Rasmussen, 2011; Depeweg et al., 2017; Moerland et al., 2017).

A theoretical problem with count-based / intrinsic motivation approaches can be that they change the RL objective itself. For example, bonuses on novelty might make an agent continue to visit a region of state-space where the value functions are already very certain, yet not all states are frequently visited yet (like continuously walking around a room to view it from all angles, which gives a new visual state each time). Nevertheless, intrinsic motivation-based approaches hold the state-of-the-art on challenging exploration problems like Montezuma's Revenge (Bellemare et al., 2016).

A.4 Uncertainty in model-based RL

All work in this paper only considered model-free RL. However, similar issues with uncertainty appear when learning the transition and/or reward function, known as 'model-based RL'. Dearden et al. (1999) was again the first to address this problem for the tabular setting. Mannor et al. (2004, 2007) studied two environment sources of variance that may influence the return distribution: the 'internal variance' due to a stochastic environment, and the 'parametric (model) variance' due to bias in the environment model. Neither of these were considered in this work, but both may be added. These ideas were studied in neural networks by Depeweg et al. (2017), who instead uses the terms *empistemic uncertainty* for the model bias and *aleatoric uncertainty* for the inherent environment noise/stochasticity. Their approach, which infers distributions on neural network parameters to capture model bias, and uses expressive output distributions to capture true environment stochasticity, actually has a similar structure as our Double Uncertain Value Network (in some sense, they learn a 'Double Uncertain Transition Network'). Of course, transition model learning does not involve any uncertainty propagation (it is a well-defined supervised learning problem). Finally, Gal et al. (2016) used Bayesian drop-out, as considered in this work for parametric value uncertainty, to track parametric model uncertainty.

B Initial Return Entropy (IRE) as a measure of initial exploration difficulty

Define the initial return distribution (IRD) $p^{\text{init}}(Z)$ as the distribution over trace returns $Z \in \mathbb{R}$ when sampling an initial state s^{init} from some initial state distribution and following a uniform random policy from there on. For undirected exploration, the uniform policy is the best policy we can specify until we start encountering varying returns. Define the initial return entropy (IRE) as the entropy of this distribution:

$$H(Z) = \int p^{\text{init}}(Z) \cdot \log \left(p^{\text{init}}(Z) \right) \mathrm{d}Z \tag{9}$$

We propose that the IRE is an interesting measure of the domain exploration difficulty, where lower values indicate a higher exploration challenge. Figure 4 shows the IRD and IRE for various domains from the OpenAI Gym repository. We see quite large differences between the shape of these distributions. Importantly, some domains with hard exploration, for example the Atari game Montezuma's Revenge, show a very spiked IRD and therefore low IRE. The challenge of such domains is that *nearly all initial traces give the same return*, which makes it hard to ever find a first indication of where to go. In many of such domains nearly all traces then give 0 reward, but for example Mountain Car shows all traces giving -200 reward (there is a -1 penalty per timestep, and Gym caps MountainCar episodes at length 200 by default). The entropy of the return distribution is of course robust against such reward function translations, making it a stable measure of initial exploration difficulty.

We do not propose this is the only measure of domain exploration difficulty. For example, a well-known exploration challenge is choosing between a small suboptimal pay-off and exploring further to obtain a potential higher reward. This type of exploration challenge is not accurately reflected in the IRE, as simple early rewards spread out the initial return distribution and may falsely suggest the domain is easy. There are of course many more dimensions that influence the RL task difficulty, like the state and action space cardinality, but the IRE nicely illustrates why a low-dimensional task like the Chain (Appendix D) can actually be quite challenging.

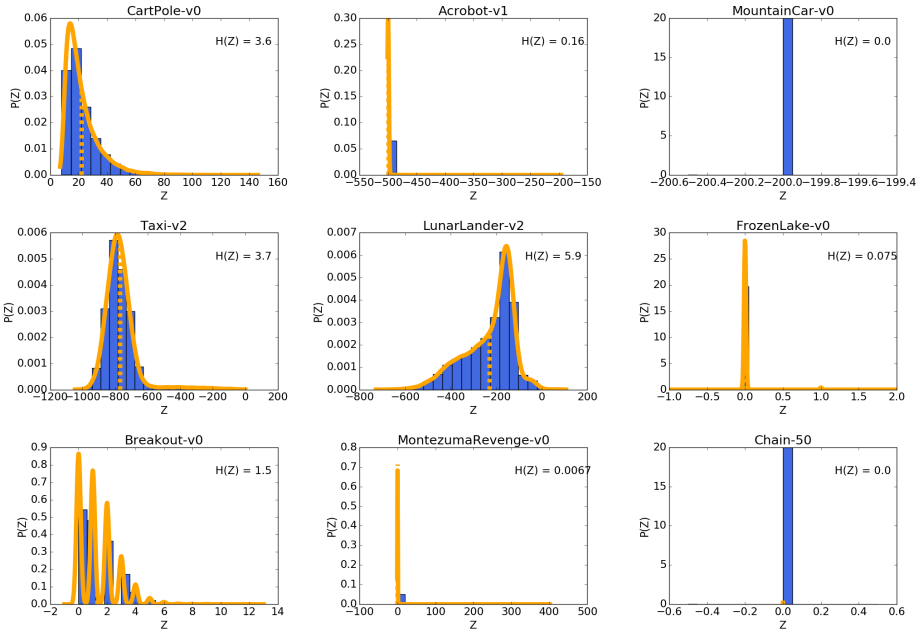

Figure 4: Return distributions from the initial state in different environments for a uniform random policy. Histogram produced over 50.000 traces of maximum 500 steps. The first 8 domains are directly taken from the OpenAI Gym. The Chain domain is introduced in Appendix D. Orange line is a kernel density estimate, with the vertical dashed line its empirical mean (a Monte Carlo estimate of $Q(s, a)$ under a uniform random policy. The top-right display the initial return entropy (IRE) estimate for the domain.

C Illustration of undirected versus directed exploration

We will quickly elaborate on the difference between undirected exploration methods, like ϵ-greedy and Boltzmann exploration, and directed methods, like Thompson sampling, in a theoretical example. Consider two available actions which, given some observed data $\mathcal{H}$, both have some posterior action-value distribution $p(Q|\mathcal{H})$. Figure 5 shows two scenario's, I (left) and II (right). The only difference between both scenario's is our uncertainty about the value of action a_1: in the second scenario we are much more uncertain about its true value. We now compare how ϵ-greedy, Boltzmann and Thompson sampling will act in both scenario's. The main point will be that undirected methods cannot leverage the uncertainty information.

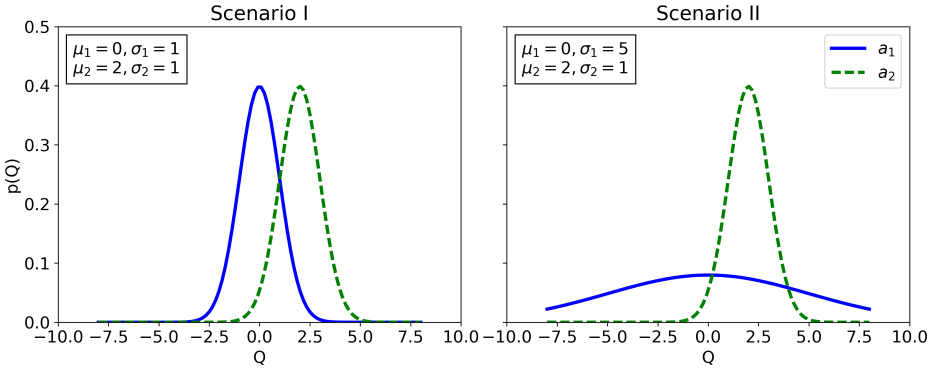

Figure 5: Example posterior value distributions for two available actions. Scenario I (left): Action 1 (blue solid line) has $\mu_1 = 0$, $\sigma_1 = 1$, Action 2 (green dashed line) has $\mu_2 = 2$, $\sigma_2 = 1$. Scenario II (right): The same except for $\sigma_1 = 5$.

1. ϵ-greedy exploration only uses the distribution means and will act the same in both scenarios, preferring action 2 and selecting action 1 with (small) probability ϵ.

2. Boltzmann (soft-max) exploration is usually seen as more subtle, gradually preferring actions with a higher pay-off. Boltzmann does consider the numerical scale of the action means, including their difference (something ϵ-greedy ignores):

$$\pi_{Boltzmann}(a|s) = \frac{e^{\mu_a}}{\sum_{a^\star \in \mathcal{A}} e^{\mu_{a^\star}}}$$

However, it still acts the same in scenario I and II, because Boltzmann approximates a distribution over both actions by still *only considering their means*. Although the softmax returns a probability distribution over actions, this should can **not** be interpreted as their uncertainty.[7] Another problem with Boltzmann action selection is that it is non-robust against translation of the reward function, making it tedious to tune. Therefore, many undirected implementations still prefer ϵ-greedy exploration.

3. Thompson sampling, a directed exploration method, uses $\pi(a_1|s) = p(Q_1 > Q_2)$. For the example with normal random variables $Q_1 \sim \mathcal{N}(\cdot|\mu_1, \sigma_1)$ and $Q_2 \sim \mathcal{N}(\cdot|\mu_2, \sigma_2)$, we can analytically calculate $P(Q_1 > Q_2) = P(Q_1 - Q_2 > 0)$. Define $X = Q_1 - Q_2$, then X will still have a normal distribution, and by standard laws of probability, $\mathbb{E}[X] = \mu_1 - \mu_2$, and $\text{Sd}[X] = \sqrt{(\sigma_1)^2 + (\sigma_2)^2}$. Applying this to the example, gives us for Scenario I (left) $\pi(a_1) = \mathcal{N}(Q_1 - Q_2 > 0|\mu_X = -1, \sigma_X = \sqrt{2}) \approx 0.08$, and for Scenario II (right) $\pi(a_1) = \mathcal{N}(Q_1 - Q_2 > 0|\mu_X = -1, \sigma_X = \sqrt{26}) \approx 0.35$. Note how Thompson sampling naturally assigns extra probability mass to action a_1 in Scenario II, where we are much more uncertain about its potential value.

[7] A similar phenomenon happens with the softmax and cross-entropy loss in classification tasks. The outputs of this softmax are also frequently falsely interpreted as a measure of uncertainty over classes (Gal, 2016). However, when we extrapolate (far) away from our observed data, one of the classes usually gets a high probability. It thereby appears as we are very certain, but since we have not observed any data in this region of input space, we should actually be very uncertain. This illustrates how point estimates over a discrete set *cannot* be transformed to uncertainties (what we need is an entire uncertainty/distribution per class output).

D Illustration of exploration challenge: Chain domain

We now study the Chain domain, an example MDP (Fig. 6) that illustrates the difficulty of exploration with sparse rewards. This domain is also empirically studied in the results section of this paper. The MDP consists of a chain of states $\mathcal{S} = \{1, 2..., N\}$. At each time step the agent has two available actions: a_1 ('left') and a_2 ('right'). At every step, one of both actions is the 'correct' one, which deterministically moves the agent one step further in the chain. The wrong action terminates the episode. All states have zero reward except the final chain state N, which has $r = 1$. Variants of these problem have been studied more frequently in RL (Osband et al., 2014). In the 'ordered' implementation, the correct action is always the same (e.g. a_2), and the optimal policy is to always walk right. This is the variant illustrated in Fig. 6 as well.

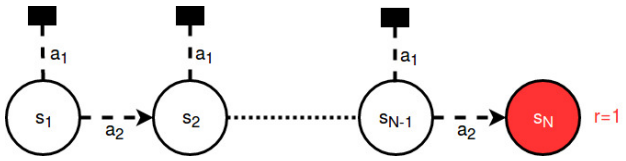

Figure 6: Chain domain. Example MDP where undirected exploration is highly inefficient. Based on (Osband et al., 2014).

Osband et al. (2014) studied the expected regret for a variant of this scenario. We here present a different illustration, where we show the expected time until the first visit to the terminal state (i.e. the first non-zero trace in this domain).

Example 1. *Let l denote the number of episodes before we first reach state N. Clearly, before we reach N for the first time, we have seen no reward information, and undirected exploration will follow a uniform random policy. The probability of a trace reaching state N under the uniform policy is $p = 2^{-(N-1)}$. Therefore, the number of episodes until we first reach N follows a negative binomial distribution with success probability p, i.e. $l \sim NB(1, p)$. It follows that $\mathbb{E}[l] = \frac{1-p}{p} = 2^{(N-1)} - 1$.*

Example 1 shows that, for undirected exploration on point estimates, the required number of exploratory episodes scales *exponentially* with the exploration depth N. Although this is clearly a simplified domain, it is important to note that this setting is actually very representative of the exploration problem in sparse reward domains. This is well visible in Fig. 4, where we can see the Chain domain having similar initial return distributions as for example Montezuma's Revenge, a game notorious for its challenging exploration.

D.1 Additional results for ordered Chain

The experiments section in the paper discusses the unordered Chain, where the correct action at every step is randomized. We here compare to the 'ordered' Chain, where the correct action at every step is always a_2. Although this is the standard implementation in literature, we believe there is a systematic bias in this domain that makes them not really exponentially challenging for exploration. The problem is that the optimal policy has a network predicting a_2 at every step. This will happen too easily in a function approximator (like a neural network) due to its natural tendency to generalize.

We show the results on the ordered Chain in Fig. 7. First of all, compared to the results in Fig. 2, we see that exploration is indeed much easier in the ordered problem. For example, ϵ-greedy now also solves the problem, something which we would not expect from the exponential exploration time discussed above. Nevertheless, we see that the probabilistic exploration methods still outperform ϵ-greedy, especially when the length of the chain increases.

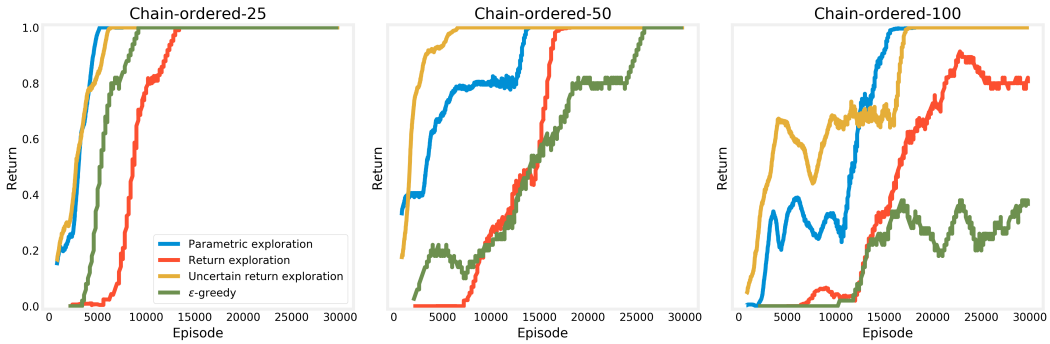

Figure 7: Learning curves on the ordered Chain domain.

E Implementation Details

Network architecture consists of a 3 layer network *for each discrete action* with 128 nodes in each hidden layer and ReLu activations. For parametric uncertainty, each hidden layer has drop-out applied to its output, with p_{keep} probability to keep a node. We use separate subnetworks per action to explicitly separate their uncertainty. For larger problems, the initial representation layers may be shared. Learning rates are fixed at 0.001 on all experiments. Optimization is performed with stochastic gradient descent using Adam updates in Tensorflow. For the experiments with parametric exploration (parametric uncertainty only) we train on a standard squared loss between new target and predicted mean action value, i.e. the first half of Eq. 6. We use a target network and replay database, where we replay 10% of times in a prioritized way (by maintaining a separate prioritized replay queue based on the total temporal difference error in the previous time this trace was trained on). All domains (except for the Chain) are taken from the OpenAI Gym repository available at `https://github.com/openai/gym`.

All ϵ-greedy experiments have ϵ fixed at 0.05 throughout learning. Drop-out rates were either $p_{keep} = 0.75$ (for parametric uncertainty only) or $p_{keep} = 0.90$ (for uncertain returns). All experiments used one-step SARSA (ie., on-policy updates with eligibility traces parameter $\lambda = 0$ (Sutton and Barto, 1998)), except for the MountainCar experiments, which use $\lambda = 0.9$. Note that the ideas about eligibility traces and cutting traces equally apply to the propagation of distributions, i.e. they allow for quicker propagation over multiple timesteps (always at the risk of propagating on-policy, exploratory results too quickly/far).[8]
Thompson sampling uses the same policy for exploration and evaluation. In some sense, proper uncertainty policies somewhat blur the line between on- and off-policy (where the behavioural/exploration policy differs from the target/evaluation policy), as there is just on reasonable probabilistic policy incorporating all uncertainty. Nevertheless, we could consider Thompson sampling exploration while evaluating with a policy that does act on some mean value again.

[8]We do believe that the uncertainty-based policies, like Thompson sampling, may also benefit work on cutting traces. Trace cutting is usually based on importance sampling ratios between the exploratory policy and the target policy. Thompson sampling may provide more realistic probabilities for exploratory actions, which may allow for more natural trace cutting. For example, ϵ-greedy always strongly cuts a trace for every exploratory step, no matter whether the exploratory action is very close to the best one, or known to be very bad. In contrast, probabilistic policies will cut traces when other possible actions in the state have much uncertainty left, which should indeed stop the speed of our back-ups. A challenge is that our neural network implementation naturally samples a next action, but the associated probability of each action is not directly available. Of course, for a small discrete action space we could approximate it by repeatedly sampling from our policy.