SUCCESSOR UNCERTAINTIES: EXPLORATION AND UNCERTAINTY IN TEMPORAL DIFFERENCE LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

We consider the problem of balancing exploration and exploitation in sequential decision making problems. To explore efficiently, it is vital to consider the uncertainty over all consequences of a decision, and not just those that follow immediately; the uncertainties need to be propagated according to the dynamics of the problem. To this end, we develop Successor Uncertainties, a probabilistic model for the state-action function of a Markov Decision Process that propagates uncertainties in a coherent and scalable way. Our model achieves this by combining successor features and online Bayesian uncertainty estimation. We relate our approach to other classical and contemporary methods for exploration and present an empirical analysis of successor uncertainties.

1 INTRODUCTION

We consider the sequential decision making problem in which an agent interacts with an unknown environment, modelled as a Markov Decision Process (MDP), with the goal of maximising the expected cumulative reward it receives. To do so, the agent has to balance exploration of the environment and exploitation of the information it has already obtained. For instance, it may valuate the possible actions it can perform in any state in terms of an estimate of achievable cumulative reward (the Q-function) and a measure of uncertainty over that return. It is natural to approach this trade-off with a Bayesian mindset and the statistics of a Bayesian posterior of the Q-function to encourage exploration. Intuitively, the agent should explore regions of the state-space with high uncertainty according to the posterior in order to get a better estimate of the potential rewards therein.

However, defining such a generative model for MDPs is challenging. Unlike in a bandit setting, where each decision is independent, learning in MDPs requires us to consider the structure of the state space; that is, uncertainty about long term consequences of an action selection needs to propagate to the point of decision-making. We illustrate this concept of propagation in figure 1 and define it formally in section 3. While there exist classic Bayesian methods that enable such propagation (Strens, 2000; Guez et al., 2012), these methods tend to scale poorly and are difficult to combine with modern deep learning approaches.

We introduce a probabilistic model called *Successor Uncertainties*, which builds on successor features (Dayan, 1993; Barreto et al., 2017) and is based on Bayesian principles to enable coherent propagation of local uncertainty estimates – uncertainties associated with each state-action pair in isolation. The successor features represent each state by the expected (and discounted) future state representations under a given policy and can thus be leveraged to compute the *accumulated* local uncertainty an agent is expected to encounter upon taking a particular action. This enables us to effectively steer exploration to regions of the state space that we are most uncertain about.

Our paper will introduce the necessary notation (section 2) and then precisely define the concept of uncertainty propagation and review related exploration methods (section 3). In section 4 we derive and discuss the implementation of Successor Uncertainties. We then present an empirical analysis of successor uncertainties as compared with common methods from previous work and conclude with a discussion.

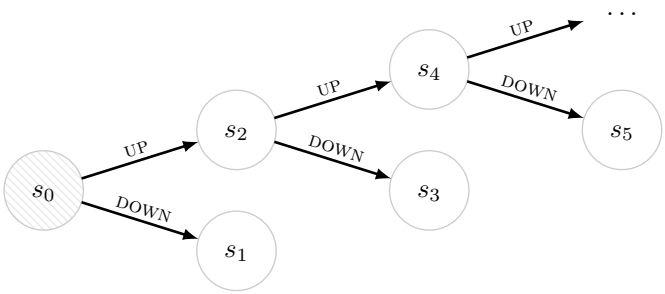

Figure 1: Binary TREE MDP; in each state the agent can either move UP or DOWN. States with odd indices are terminal. Reward is only obtained after selecting a sequence of many UP actions. Suppose that selecting between action UP versus DOWN in state s_0 is treated as a bandit problem, with no considerations of further opportunities for exploration; in that context, with no reward yet discovered, exploration policies will mix between the two actions 50-50. On the other hand, if the exploration method takes into account the many states lying beyond (s_0, UP), action UP becomes preferable. We refer an agent performing the latter as propagating exploration bonuses/uncertainty.

2 REINFORCEMENT LEARNING

We are interested in solving Markov Decision Processes (MDP) defined by a state space $\mathcal{S}$, a finite action space $\mathcal{A}$, a transition probability kernel $\mathcal{T} \colon \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$, a reward function $R \colon \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ and a discount factor $\gamma \in [0, 1)$. The task of solving an MDP is that of finding a policy $\pi \colon \mathcal{S} \to \mathcal{P}(\mathcal{A})$ that maximises the expected return $J^\pi = \mathbb{E}_{\mathcal{P}(s_0),\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$ where $r_t = r(s_t, a_t)$, where $\mathcal{P}(s_0)$ is a given distribution over the starting state and $\mathbb{E}_\pi[\cdot]$ denotes an expectation with respect to both the transition model $\mathcal{T}$ and the policy π.

The approaches to exploration we examine are all based on the temporal difference methods of solving MDPs, where an estimate of action-value return is kept for each state and the optimal policy is greedy with respect to that action-value function for each state; this state-action value function (the Q-function) is defined as $Q^\pi(s, a) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a\right]$; the state-value function is the marginal of the state-action value function, denoted $V^\pi(s) = \mathbb{E}_{a\sim\pi(s)}\left[Q^\pi(s, a)\right]$. For any given policy π, the Bellman equation states that $Q^\pi(s, a) = r(s, a) + \mathbb{E}_{s'\sim\mathcal{T}}\left[\gamma V^\pi(s')\right]$.

Frequently, the Q-function is estimated using a neural network. This network is then trained end-to-end such that it (approximately) obeys the Bellman equation on observed data: the error on a given tuple (s, a, s', r) is thus a function of $\delta_{\text{td}} = Q(s, a) - r - \gamma V(s')$, a quantity known as the temporal difference error. Implicitly Monte Carlo sampling is used to approximate the relevant expectations. The observations are typically stored in an experience buffer (Watkins, 1989), such that batch training can occur – this has the benefit of reducing the correlations in the data seen during training.

Where the Q-function estimate $\hat{Q}$ is maintained for the greedy policy $\pi^\star(s) \propto \mathbb{1}\left\{\text{argmax}_{a\in\mathcal{A}} \hat{Q}(s, a)\right\}$, the temporal difference update is known as Q-learning (Watkins & Dayan, 1992); as $\pi^\star$ merely exploits the rewards already found and does not perform explicit exploration (albeit estimator structure and training dynamics may provide implicit exploration, see Dauparas et al. (2018)) a separate policy π^E is then kept to interact with and explore the environment. If the update is taken with respect to the exploration policy π^E, (with the requirement that $\pi^E \to \pi^\star$ as $t \to \infty$), the temporal difference update is known as Expected SARSA (E-SARSA, Sutton et al., 1998; Rummery & Niranjan, 1994).

3 EXPLORATION IN REINFORCEMENT LEARNING

In this section we will discuss the existing literature on exploration in reinforcement learning. Given an estimate of the Q-function and a prior over plausible values, these tend to then use statistics of the posterior over Q in order to balance exploration and exploitation. There are two types of statistics

commonly used: state-action visitation counts, which may be used to infer upper bounds on the Q-function, and variance estimates for state-action pairs, which enable both the estimation of upper bounds and exploration using samples from the Q-distribution. We will briefly discuss visitation count estimation and how these estimates are propagated within MDPs; these easy to understand methods serve to clarify the concept of propagation. We will then move to methods that explicitly model the uncertainty in the Q-function estimates, propose a definition of propagation of uncertainty, discuss recent work that we believe does not correctly propagate uncertainty and the practical effects of failing to do so.

3.1 Exploring using visitation counts

State-action visitation counts, denoted n_{sa}, are straightforward to obtain in the tabular contextual bandit case. Probable upper bounds on the Q-values may then be estimated, and take the form of $Q(s,a) + f(n_{sa})$ where f is some monotonically decreasing function,[1] a result presented in Auer et al. (2002) for the multi-armed bandit problem. Exploration then consists of selecting actions that maximise this upper bound, consequently either selecting the optimal action, or driving the upper bound down.

In order to scale the visitation count approach from a bandit task to a reinforcement learning environment, the structure of the state-space needs to be accounted for. By this we mean that bonuses from state-action pairs that are poorly explored must be propagated to the state-action pairs required to reach them. This can be done within the standard Q-learning framework, by augmenting the rewards with the associated visitation counts, that is r_{sa} is replaced with $r'_{sa} = r_{sa} + f(n_{sa})$. Then state-action values reflect the visitation count exploration bonuses of their downstream children. In particular

$$\hat{Q}^{\pi}(s,a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^t r_t + \sum_{t=0}^{\infty}\gamma^t f(n_t) \mid s_0 = s, a_0 = a\right] \tag{1}$$

where $n_t = n(s_t, a_t)$. We discuss approaches of extending visitation count methods to continuous domains in appendix A.

3.2 Estimating uncertainty

Estimating uncertainty in a bandit, or contextual bandit problem is straightforward. States can be encoded using one-hot representation and a Bayesian linear model used to provide estimates of the Q-function mean μ_{sa} and variance σ^2_{sa} for each state-action pair (Riquelme et al., 2018). When moving to a reinforcement learning problem, where one deals with a structured state space, we believe that it is vital that the uncertainty estimates are propagated through the MDP. In definition 1 we propose what we believe is a suitable generalisation of the concept of propagation from the domain of visitation count bonuses (as per equation (1)) to uncertainty estimates. We note that whilst a model of the Q-function that follows from applying Bayes' rule to the relevant quantities propagate uncertainty in the thus defined sense, a Bayesian approach is not required.

Definition 1 (Propagation of uncertainty) *Given a fixed data set of observations sampled from an MDP M, denoted $\mathcal{D} = \{s_t, a_t, s_{t+1}, r_t \mid t = 1, \ldots, T\}$ and a probabilistic model of estimators $\mathcal{P}(\hat{Q}^{\pi})$ for M under policy π, we say the model propagates uncertainty if for all samples $\hat{q} \sim \mathcal{P}(\hat{Q}^{\pi})$, $\hat{q}$ satisfies the Bellman equation with respect to π for a restricted MDP M' defined by the empirical distribution of $\mathcal{D}$.*

In order to transition from a tabular contextual bandit scenario to general reinforcement learning settings, methods in recent literature attempt to either: 1) model the Q-function as a Bayesian neural network and perform approximate inference by considering the temporal difference updates as if these were real observations of the Q-values in a regression problem (for example Lipton et al., 2016; Gal, 2016), or 2) use a neural network to learn a state embedding ϕ^Q_s and then perform Bayesian linear regression with ϕ^Q_s as input to predict μ_{sa} and σ^2_{sa} estimates for each state-action pair (as used in Azizzadenesheli et al., 2018; Moerland et al., 2017). Neither approach results in a probabilistic

[1] Expressions for optimal f exist for various analytic problems, however for this exposition we keep the notation generic, as these tend not to carry over to generic reinforcement learning settings.

model for Q-functions that correctly propagates uncertainty as per definition 1. Consequently, the approaches still treat the problems as a contextual bandit.

We finish with two examples of methods that propagate uncertainty and can be scaled to work with complex state spaces. Osband et al. (2016) introduce structure to the exploration by maintaining a bootstrap-based distribution over plausible state-action value functions. It is, however, far removed from the framework of Bayesian probabilities, and the quality of bootstrap uncertainty estimates for exploration has been questioned in Riquelme et al. (2018) (in the context of contextual bandits). Work presented in O'Donoghue et al. (2018) introduces a dynamic programming rule for estimates to Q-function uncertainty; the method accumulates uncertainties from local Bayesian linear models, performing very well on the Atari benchmark suite. We discuss the relation between our proposed method and this work in detail in section 4.6.

4 Uncertainty propagation through Successor Features

We have seen how visitation count bonuses can be trivially propagated along an MDP through adding a bonus to the observed rewards (see Equation (1)). Here, we develop a Bayesian model for Q-function estimation that correctly (as per definition 1) propagates uncertainties. We call the resulting model and uncertainties 'successor uncertainties'. Our derivations begin with a probabilistic model for rewards, which we then extend to a model over Q-functions. To do so, we use the so-called successor representations (Dayan, 1993), a notion of predicted future state-action visitation counts for a given policy. Successor representations implicitly allow the model to consider downstream uncertainty in reinforcement learning contexts. Successor features (Kulkarni et al., 2016; Zhang et al., 2017; Barreto et al., 2017), the natural extension of successor representations to neural networks, allow our successor uncertainties to benefit from the latest methodological advances in deep learning.

4.1 The Successor Uncertainty Model

Specifically, given a state-action pair embedding ϕ^l_{sa}, we predict its reward as an inner product with a weight vector w, that is $\hat{r}(s,a) = \langle w, \phi^l_{sa} \rangle$. We model w probabilistically, with a Gaussian prior and likelihood, resulting in a standard Bayesian linear regression model. The embedding ϕ^l_{sa} can either be one-hot, in which case the method is applicable to tabular problems, or learnt using a neural network. To learn the embedding, we minimise the squared reward prediction loss $\delta_r = (r - \hat{r})^2$ over observed data, where $\hat{r}$ is the output of neural network that mapping (s,a) to ϕ^l_{sa} and then ϕ^l_{sa} to $\hat{r}$. This architecture is illustrated in figure 2.

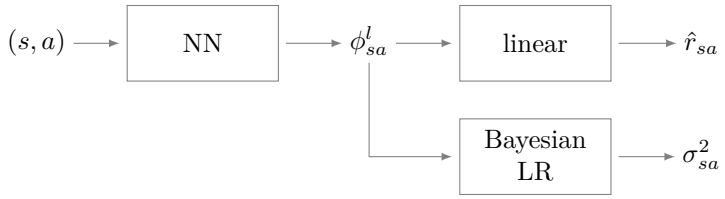

Figure 2: Architecture for computing successor uncertainties

To extend the probabilistic model from individual rewards to Q-functions, we need to consider the temporal structure within the predictions induced by the transition function and the policy π. Substituting our model for rewards into the definition of the Q-function in equation (2) we see that the Q-function can be computed as the inner product of w and $\mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \phi_t \mid s_0 = s, a_0 = a\right] = \psi_{sa}$, the discounted sum of the expected feature embeddings under π:

$$Q(s,a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a\right] \approx \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \langle \phi_t, w \rangle \mid s_0 = s, a_0 = a\right] \tag{2}$$
$$= \left\langle \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \phi_t \mid s_0 = s, a_0 = a\right], w \right\rangle = \hat{Q}^{\text{SF}}(s,a)$$

with $\hat{Q}^{\text{SF}}(s,a)$ giving the μ_{sa} used by our method. The features ψ_{sa} are what is known as the successor representation/features (Dayan, 1993; Barreto et al., 2017). We estimate ψ_{sa} using a second neural network by noting that it obeys a Bellman equality

$$\psi_t = \phi_t^l + \gamma \mathbb{E}_\pi \left[\psi_{t+1}\right] \tag{3}$$

where $\psi_t = \psi_{s_t a_t}$, and can thus be trained with a standard temporal difference approach (as described in section 2). The resulting successor features ψ_{sa} and a distribution over the weights w induce our successor uncertainty model of for estimating the Q-function, given by

$$\hat{Q}^{\text{SF}}(s,a) \sim \mathcal{N}(\langle w, \psi_{sa}\rangle, \psi_{sa}\Sigma_w\psi_{sa}^T) \tag{4}$$

where Σ_w is the weight covariance of the Bayesian linear model. Importantly, equation (4) provides a posterior over Q-functions that correctly propagate uncertainty as per definition 1 (given that the successor features ψ_{sa} are converged).

4.2 Temporal difference updates

Our approach uses E-SARSA as its update mechanism (equation (3)). This is a vital design choice for two reasons. First, successor features must be learnt for the exploration policy π^E. If ψ_{sa} were instead trained for the estimated optimal policy $\hat{\pi}^\star$ (e.g., using Q-learning updates), this would result in features that provide information only about the successive state-action pairs already estimated to be optimal, and would thus not carry sufficient information for exploration. Second, a Q-learning update would result in a drastic change of the successor features each time the estimated optimal action at a given state changes. This may make optimisation challenging; E-SARSA combined with a non-degenerate policy provides smoother optimisation dynamics.

4.3 Bayesian Linear model & low rank updates

The slow convergence of online uncertainty estimates in variational deep learning has been shown to give poor results in the contextual bandit case (Riquelme et al., 2018). The Linear Bayesian model, which admits an analytic online update to its variance estimates, may thus be preferable in the reinforcement learning setting.

Suppose the data consists of points $\phi_1, \ldots, \phi_t$ where $\phi_t \in \mathbb{R}^m$ with corresponding regression outputs $r_t \in \mathbb{R}$. The posterior distribution in a Gaussian linear model satisfies $p(w|(\phi_1, r_1), \ldots, (\phi_t, r_t)) = \mathcal{N}(\mu_w, \Sigma_w)$; the predictive variance does not depend on μ_w, we only show how to perform online updates for Σ_w. The weight covariance matrix has the closed form expression $\Sigma_w = \left(\Sigma_p^{-1} + \sigma_n^{-2}\sum_{i=1}^n \phi_i\phi_i^T\right)^{-1}$ where Σ_p is the prior weight covariance and σ_n^2 is the observation noise. Denoting the covariance after observing t data points as Σ_w^t, by the Sherman-Morrison formula (Sherman & Morrison, 1950), upon observing an additional data point ϕ_{t+1} we obtain

$$\Sigma_w^{t+1} = \Sigma_w^t - \frac{(\Sigma_w^t\phi_{t+1})(\Sigma_w^t\phi_{t+1})^T}{1 + \phi_{t+1}^T\Sigma_w^t\phi_{t+1}} \tag{5}$$

at a cost quadratic in the dimension of ϕ_t. The predictive variance for an input $x \in \mathbb{R}^m$ then given by $\sigma^2(x) = x\Sigma_w x^T$.

4.4 Exploration policies

Successor uncertainties may be used either through posterior sampling or through a method relying on a probable upper bound on the Q-values. Here we describe one of each type of exploration policies.

Posterior sampling The posterior sampling exploration policy π^E is defined by the procedure of sampling from a posterior over $Q(s,a)$ for each action and executing action with the maximal sample value. However, we note that the uncertainties output by the Bayesian linear model are sensitive to the re-scaling of the input features; since the scale of the learnt neural network features is largely arbitrary, we introduce an uncertainty scale parameter β absorbs the reward observation noise term σ_n^2 and sample from $\hat{Q}(s,a) \sim \mathcal{N}(\mu_{sa}, \beta\sigma_{sa})$. We estimate expectation with respect to a posterior sampling policy that is required for E-SARSA updates using simple Monte Carlo.

Upper confidence bound We use a upper confidence bound (UCB) style policy that we refer to as UCB-Boltzmann (UCB-B) of the form $\pi^E(a) \propto \exp(\frac{\mu_{sa}+\beta\sigma_{sa}}{\tau})$, where τ is a temperature parameter that affects the exploration-exploitation trade-off. This usage of a UCB bonus as the parameter for a Boltzmann distribution transforms what would otherwise be a degenerate policy to a policy that changes continuously as a function of its parameters – an important factor for the learning dynamics of successor features.

4.5 ALGORITHM

From an implementation point of view, successor uncertainty is very similar to standard Q-learning and the various recent methods that employ some form of Bayesian inference. Algorithm 1 provides pseudo-code for using successor uncertainties with posterior sampling.

Algorithm 1 Exploration through posterior sampling with Successor Uncertainties

procedure TRAIN
  for episode **in** episodes **do**
    done $\leftarrow$ false
    while not done **do**
      $a \leftarrow$ POSTERIORSAMPLE(s)
      $(s',\ r,\ \text{done}) \leftarrow$ ENV.INTERACT(a)
      UPDATE_UNCERTAINTY(s, a) ▷ See section 4.3 for details
      $\delta_r \leftarrow ||r - \langle \phi^l_{sa}, w\rangle||_2^2$
      minimise δ_r wrt ϕ^l parameters and w vector
      $\delta_{\text{td}} \leftarrow$ E-SARSA(s, a, s')
      minimise δ_{td} with respect to ψ parameters

function POSTERIORSAMPLE(s)
  for a **in** $\mathcal{A}$ **do**
    $\mu_{sa} \leftarrow \langle w, \psi(s, a)\rangle$
    $\sigma_{sa} \leftarrow \psi(s, a)\Sigma_w\psi(s, a)^T$ ▷ Σ_w is obtained from the Bayesian linear model
    sample $q_{sa} \sim \mathcal{N}(\mu_{sa}, \sigma_{sa})$
  return $\text{argmax}_{a\in\mathcal{A}}\, q_{sa}$

function E-SARSA(s, a, s')
  $\psi(s') = \mathbb{E}_{a'\sim\pi}\left[\psi(s', a')\right]$ ▷ Expectation done through simple Monte Carlo
  return $||\psi(s, a) - \phi(s, a) - \gamma\psi(s')||_2^2$

4.6 RELATION TO UNCERTAINTY BELLMAN EQUATION

Work presented in O'Donoghue et al. (2018), the Uncertainty Bellman Equation (UBE), sets up a Bayesian linear model on-top of the penultimate layer of a neural network predictor of the Q-function, such that $\hat{Q} = \langle \phi^Q, w\rangle$ and utilise a Bayesian linear model to estimate σ^2_{sa}. They recognise that this uncertainty is not propagated correctly, and thus estimate a new quantity $u_{sa} = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^{2t}\sigma_t^2\right]$ using temporal difference learning. This new quantity then serves as an uncertainty estimate that is correctly propagated to use for exploration.

You can consider our method as estimating the Q-function uncertainty using an update based on $\hat{\mathbb{V}}(Q(s, a)) = \hat{\mathbb{V}}(r(s, a)) + \gamma^2\mathbb{E}_{a'\sim\pi}\hat{\mathbb{V}}(Q(s', a'))$, On the other hand the UBE method performs an update of the form $\hat{\mathbb{V}}(Q(s, a)) = \tilde{\mathbb{V}}(Q(s, a)) + \gamma^2\mathbb{E}_{a'\sim\pi}\hat{\mathbb{V}}(Q(s', a'))$, where $\tilde{\mathbb{V}}(Q)$ is some other model of Q-function uncertainty that need not correctly propagate uncertainty.

The practical implication of this conceptual difference between the two methods is that the features used to learn the Bayesian linear model within the UBE framework, $\phi^Q(s, a)$, correspond to the Q-value estimate for the tuple (s, a). They must thus in some manner carry information about state-action pairs that follow on from (s, a) under some policy that changes throughout training. These changes in policy may thus drastically change the embedding $\phi^Q(s, a)$ of any given state-action pair,

and negatively affect the quality of the uncertainty estimates output by the Bayesian linear model. In the case of successor uncertainties, the inputs to the Bayesian linear model are local features $\phi^l(s, a)$. These features only predict the reward associated with the tuple (s, a), and as this reward does not change as a function of π, we can expect $\phi^l(s, a)$ to more stable and thus to provide more reliable uncertainty estimates.

Within sparse reward problems, such as those presented in our experiment section, there is very little difference between Q-function predictions and reward predictions for most of the training process. In that context, therefore, both methods thus lead to an equivalent propagation of uncertainty estimates.

5 EXPERIMENTS

Throughout the experiments we will benchmark three methods: *ϵ-greedy* ($\epsilon = 0.1$), or *uniform* ($\epsilon = 1.0$) policy with Q-learning updates; *local uncertainties*, consisting of a Bayesian linear model on Q-estimator embeddings with Q-learning temporal difference updates; and our proposed method, *successor uncertainties* with E-SARSA updates.

Learning rates were selected from $\{0.1, 0.01, 0.001\}$. The neural networks feature a single hidden layer with a width equal to twice the size of the corresponding state-space for all experiments; this setting was close to optimal for all algorithms. As the tasks focus on exploration rather than learning, the experiments are not very sensitive to network architecture and related hyperparameters. For all experiments $\beta = 1.0$, $\mu = 1$. These parameters affect on both local and successor uncertainties similarly and due to the simple reward structures, they have little impact on performance.

5.1 TREE, SEQUENTIAL BINARY DECISIONS

Our first experiment involves the very example we used to introduce the concept of propagation in the introduction, pictured in figure 1. The agent progresses through L binary decisions; the action DOWN terminates the episode with no reward. A sequence of L UP actions results in a reward of 1. Propagation of uncertainty is crucial for efficient exploration within this environment, as shown by our results in table 1 and figure 5. We were unable to obtain results for local uncertainties for $L = 20$ and $L = 30$, as these required too great a number of episodes. The uniform policy numbers for $L = 20$ and $L = 30$ were calculated analytically. Figure 4 shows the uncertainty that successor uncertainties and local uncertainties assign to each action, UP and DOWN through the first 100 episodes of training on a short tree. We see that the propagation of uncertainty causes our method to assign a great deal more uncertainty to the UP action. Local uncertainty fails to account for the long-term consequences for exploration from action UP and is thus inefficient.

5.2 CHAIN, THE CYCLIC EXPLORATION PROBLEM

We also consider the CHAIN environment introduced in Osband et al. (2016). It consists of states $\mathcal{S} = \{s_{-1}, s_0, \ldots, s_L\}$, where L is the length of the chain, $\mathcal{A} = \{a_1, a_2\}$ corresponding to deterministic transitions to the RIGHT and LEFT respectively. The reward for any state-action pair that leads to s_{-1} is $1/1000$ and for any state-action that leads to s_L is $1/10$. Any other state-action gives zero reward. The horizon $H = L + 9$, such that the maximal reward that can be obtained in an episode is 1. The states are presented to the algorithms as one-hot encoded. This is thus a problem with very sparse reward structure, with a sub-optimal greedy optimum given by the policy LEFT always.

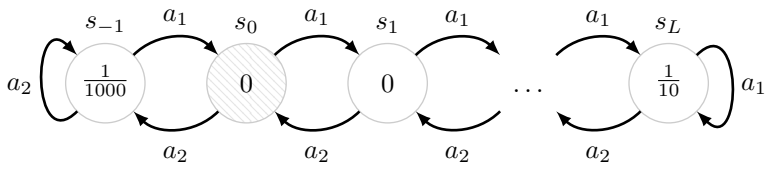

Figure 3: The CHAIN MDP illustrated

Table 1: Approximate number of episodes until 99% of the runs learn a policy with a return of 1 on TREE of size L

	L		
method	10	20	30
successor uncert.	100	1.0e3	1.2e4
local uncert.	1.4e3	–	–
uniform	5e3	5e6	5e9

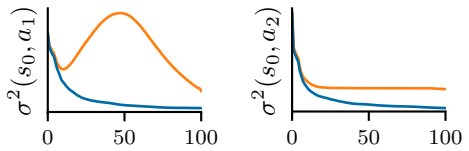

Figure 4: Uncertainties for action a_1 and a_2 at state s_0 of the TREE $L = 10$ for successor uncertainties (orange) and local uncertainties (blue). Episodes on x-axis.

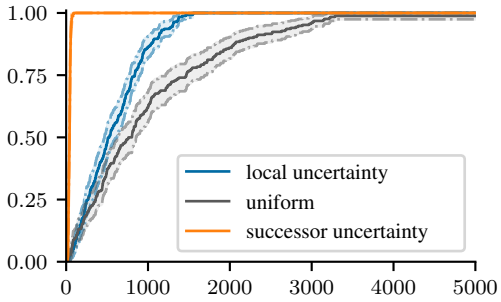

Figure 5: Test reward on TREE with L=10, average over 250 runs plotted; 95% CI shaded. Uncertainty based methods use UCB-B policy with $\tau = 0.1$. Episodes on x-axis.

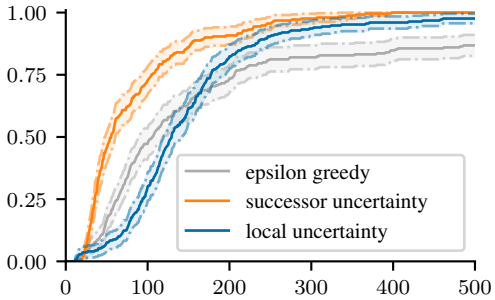

Figure 6: Test reward on CHAIN with L=10, average over 250 runs plotted; 95% CI shaded. Uncertainty based methods use posterior sampling. Episodes on x-axis.

In figure 6 we show the performance of local uncertainties, successor uncertainties and epsilon greedy on such a chain of length $L = 10$. Here, encouraging exploration using successor uncertainties converges faster and more reliably than the two benchmark methods. Insight into the improved performance of successor uncertainties over local uncertainties may be obtained by examining figure 7, where we visualise the uncertainties given to the action RIGHT at various points through training. Whereas the local uncertainties collapse near the beginning of the chain, successor uncertainties avoid this pathology by propagating uncertainties back from states further down the chain.

6 DISCUSSION

In this paper, we approached to solve a shortcoming of some previous work on probabilistic exploratory techniques within model free reinforcement learning. Specifically, some probabilistic exploration methods presented within the deep reinforcement learning literature do not correctly define a Bayesian model for the state-action value; instead, they perform approximate Bayesian inference on-top of bootstrapped Q-value estimates. These may then fail to coherently propagate uncertainties, at least according to a definition of uncertainty propagation we propose.

We address this shortcoming by introducing Successor Uncertainties, a simple generative model for the Q-values based on successor features that can be scaled to work in the same contexts as the previous work with little additional overhead. It provides a clear separation between the local uncertainties that we can reason about, and features learnt by temporal difference approaches, which are treated as deterministic. The immediate benefit of employing successor uncertainties is that they propagate correctly along the states space of the MDP. Our method is thus a reinforcement learning exploration solution, rather than an application of a contextual bandit method.

We present a set of simple experiments where we examine and describe the effects of using successor uncertainties versus using local uncertainty estimates. Quantitatively, successor uncertainties

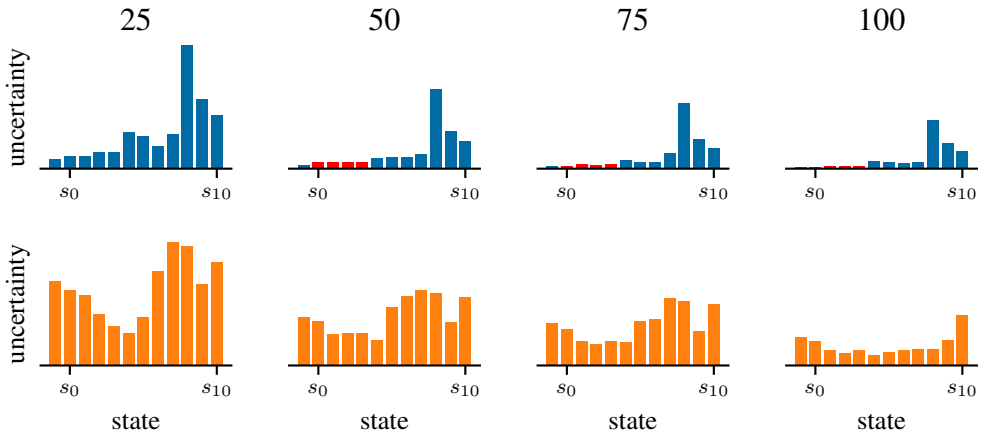

Figure 7: Local (top, blue) versus Successor (bottom, orange) uncertainties through time on length 10 chain. Bars represents the uncertainty over action RIGHT in each state at 25, 50, 75 and 100 episodes. Note that local uncertainty estimates for action RIGHT in the neighbourhood of s_0 rapidly collapse (highlighted in red).

perform better on this small set of tasks. The successor uncertainty estimates are also qualitatively very different to the estimates provided by local methods.

To finish, we note that the key limitation of our method is that it does not consider uncertainty in the and the transition dynamics. The key avenue for future work is a principled approach to modelling uncertainty in the successor features themselves. A second direction that strikes us as interesting is to note that we deal with state-action embeddings, rather than state embeddings with discrete predictors for each action. There may well be a natural extension of this work to continuous action domains, and if so, the same benefits as this work provides to the finite cardinality action space environments could be carried over. Furthermore such embeddings may be beneficial in domains with large discrete correlated action spaces.

REFERENCES

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

Kamyar Azizzadenesheli, Emma Brunskill, and Animashree Anandkumar. Efficient Exploration through Bayesian Deep Q-Networks. *arXiv preprint arXiv:1802.04412*, 2018.

André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. In *Advances in neural information processing systems (NIPS)*, pp. 4055–4065, 2017.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 1471–1479, 2016.

Justas Dauparas, Ryota Tomioka, and Katja Hofmann. Depth and nonlinearity induce implicit exploration for rl. *arXiv preprint arXiv:1805.11711*, 2018.

Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993.

Yarin Gal. Uncertainty in deep learning. *University of Cambridge*, 2016.

Arthur Guez, David Silver, and Peter Dayan. Efficient bayes-adaptive reinforcement learning using sample-based search. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 1025–1033, 2012.

Tejas D Kulkarni, Ardavan Saeedi, Simanta Gautam, and Samuel J Gershman. Deep successor reinforcement learning. *arXiv preprint arXiv:1606.02396*, 2016.

Zachary C Lipton, Jianfeng Gao, Lihong Li, Xiujun Li, Faisal Ahmed, and Li Deng. Efficient exploration for dialogue policy learning with bbq networks & replay buffer spiking. *arXiv preprint arXiv:1608.05081*, 2016.

Jarryd Martin, Suraj Narayanan Sasikumar, Tom Everitt, and Marcus Hutter. Count-based exploration in feature space for reinforcement learning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2017.

Thomas M Moerland, Joost Broekens, and Catholijn M Jonker. Efficient exploration with double uncertain value networks. *arXiv preprint arXiv:1711.10789*, 2017.

Brendan O'Donoghue, Ian Osband, Remi Munos, and Volodymyr Mnih. The Uncertainty Bellman Equation and Exploration. In *International Conference on Machine Learning (ICML)*, 2018.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In *Advances in neural information processing systems (NIPS)*, pp. 4026–4034, 2016.

Georg Ostrovski, Marc G Bellemare, Aaron van den Oord, and Rémi Munos. Count-based exploration with neural density models. *arXiv preprint arXiv:1703.01310*, 2017.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning (ICML)*, volume 2017, 2017.

Carlos Riquelme, George Tucker, and Jasper Snoek. Deep Bayesian Bandits Showdown: An Empirical Comparison of Bayesian Deep Networks for Thompson Sampling. *arXiv preprint arXiv:1802.09127*, 2018.

Gavin A Rummery and Mahesan Niranjan. *On-line Q-learning using connectionist systems*, volume 37. 1994.

Jack Sherman and Winifred J Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, 21(1):124–127, 1950.

Malcolm Strens. A Bayesian framework for reinforcement learning. In *Conference on Machine Learning (ICML)*, pp. 943–950, 2000.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*. MIT press, 1998.

Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 2753–2762, 2017.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. PhD thesis, King's College, Cambridge, 1989.

Jingwei Zhang, Jost Tobias Springenberg, Joschka Boedecker, and Wolfram Burgard. Deep reinforcement learning with successor features for navigation across similar environments. In *International Conference on Intelligent Robots and Systems (IROS)*, pp. 2371–2378. IEEE, 2017.

A EXTENDING VISITATION COUNTS TO CONTINUOUS DOMAINS

Visitation counts are a simple and powerful concept for discrete state-action spaces. One of two groups of methods is typically used to extend counts to continuous domains. First, one may create a discrete embedding for the state-space, be it through hand-crafted features (Martin et al., 2017) or by using autoencoders with discrete hashing functions (Tang et al., 2017), and apply the usual concept of vistation counts to these embeddings. These methods are heavily reliant on the quality of the discretisation. The other approach involves relaxing and generalising the concept of visitation count to a pseudo-count function estimator $\hat{n}(s, a)$, inferred from a density model over the state space (Bellemare et al., 2016). These models are able to exploit advances in domain-specific modelling (e.g. vision and video models) and thus scale very well (Ostrovski et al., 2017, for details). Pseudo-counts do however impose additional constraints on the way in which learning is performed, and utilising these in practice may not be simple. Both pseudo-counts and state-space discretisation methods tend not to consider the reward and dynamics structure of the MDP. Thus they may be inefficient in the presence of features that have no bearing on the optimal policy, like an irrelevant background flicker in a video game (Pathak et al., 2017).