Model-Based Active Exploration

Pranav Shyam
NNAISENSE
Lugano, Switzerland
pranav@nnaisense.com

Wojciech Jaśkowski
NNAISENSE
Lugano, Switzerland
wojciech@nnaisense.com

Faustino Gomez
NNAISENSE
Lugano, Switzerland
tino@nnaisense.com

Abstract

Efficient exploration is an unsolved problem in Reinforcement Learning. We introduce Model-Based Active eXploration (MAX), an algorithm that actively explores the environment. It minimizes data required to comprehensively model the environment by planning to observe novel events, instead of merely reacting to novelty encountered by chance. Non-stationarity induced by traditional exploration bonus techniques is avoided by constructing fresh exploration policies only at time of action. In semi-random toy environments where directed exploration is critical to make progress, our algorithm is at least an order of magnitude more efficient than strong baselines.

1 Introduction

Exploration in large, high-dimensional Markov Decision Processes (MDPs) is an unsolved problem in Reinforcement Learning (RL). Current exploration methods (Osband et al., 2016; Bellemare et al., 2016; Houthooft et al., 2016; Pathak et al., 2017) are mostly *reactive*: the agent accidentally observes something "novel" and then decides to obtain more information about it. Further exploration in the vicinity of the novel state is carried out typically through an exploration bonus or intrinsic motivation reward which have to be unlearned once the novelty has worn off, making exploration inefficient—a problem we refer to as *over-commitment*.

However, exploration can also be *active* where the agent seeks out novelty based on its own "internal" estimate of what action sequences will lead to interesting transitions. This approach is inherently more powerful than reactive exploration, but requires a method to predict the consequences of actions and their degree of novelty. This problem can be formulated optimally in the Bayesian setting where the novelty of a given state transition can be measured by the disagreement between the next-state predictions made by members of a distribution over models of the environment. Unfortunately, this inference problem is intractable for any realistic RL task. To overcome this, we introduce Model-Based Active eXploration (MAX), an efficient algorithm, based on this principle, that approximates the idealized distribution using an ensemble of learned forward dynamics models. The algorithm identifies learnable unknowns or *uncertainty* representing novelty in the environment by measuring the amount of conflict between the predictions of the constituent models. It then attempts to seek uncertainty by constructing exploration policies that would resolve these conflicts using a novel policy evaluation technique. Crucially, the search for exploration policies is done entirely with the ensemble. Since, such models are generally several orders of magnitude cheaper in both cost and time than real environments, MAX avoids over-commitment by regularly learning fresh exploration policies. Unlearnable unknowns or *risk*, such as environment noise does not interfere with the process since stochasticity manifests itself as confusion among all models and not as a conflict. Therefore, random data is not interesting, despite its high information content.

While MAX can be used in conjunction with conventional action learning to maximize external reward, this paper is restricted to *pure exploration*: exploration disregarding or in the absence of external reward (Bubeck et al., 2009). This is both to better study the properties of the algorithm in

isolation, irrespective of any particular external goal, and because it means that the learned models will be unbiased with respect to whatever downstream tasks they could potentially be used for.

Preliminary experiments show that MAX is significantly more efficient than reactive exploration techniques which use exploration bonuses or posterior sampling. They also strongly suggest that MAX is not sensitive to noise, scalable and robust to key-parameters.

The next section provides the theoretical foundation for this work and introduces the MAX algorithm.

2 Model-Based Active Exploration

The key idea behind our approach to true active exploration in the environment, or the *external* MDP, is to use a model, learned from the environment, as a surrogate or *exploration* MDP where the novelty of possible future transitions can be estimated *before* they have actually been encountered by the agent in the environment. The next section provides the formal context for the conceptual foundation of this work presented in Section 2.2, and Section 2.3 introduces an approximation to this idealized treatment that implements efficient model-based active exploration.

2.1 Problem Setup

Consider an environment represented by a Markov Decision Process (MDP) defined as the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \rho_0)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}^+$ is the probabilistic transition function, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\rho_0 : \mathcal{S} \to \mathbb{R}^+$ is the distribution of the initial states, s_0.

The objective of pure exploration is to learn a policy, $\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^+$, that selects actions probabilistically to efficiently accumulate information about the environment, irrespective of $\mathcal{R}$. This problem is equivalent to the problem of learning an accurate model, $\bar{\mathcal{T}}$, of the transition function, $\mathcal{T}$, while minimizing the number of state transitions, ζ, required to do so, where $\zeta = (s, a, s')$ and s' is the state resulting from action a being taken in state s. Let Γ be the space of all possible transition functions and $\mathcal{P}(\Gamma)$ be a probability distribution over transition functions that captures the current belief of how the environment behaves, with corresponding density function $p(\bar{\mathcal{T}})$.

Given an accurate $\bar{\mathcal{T}}$, the policy can take actions in an exploration MDP $(\mathcal{S}, \mathcal{A}, \bar{\mathcal{T}}, \mathcal{R}, \rho_0)$, having the same set of states and actions $(\mathcal{S}, \mathcal{A})$, as if it were the external MDP.

2.2 Utility of an Exploration Policy

In the standard RL setting, a policy would be learned to take actions that maximize some function of the external reward received from the environment according to $\mathcal{R}$, i.e. the *return*. Because pure active exploration cannot rely on $\mathcal{R}$, and $\mathcal{T}$ is unknown, there needs to be a way to compute the amount of information conveyed about the environment by unseen state transitions that *could* be caused by the policy, which requires using a model of $\mathcal{T}$, instead $\mathcal{T}$ itself to "internally" evaluate possible future transitions.

From a Bayesian perspective, this can be captured by the KL divergence between $\mathcal{P}(\Gamma)$, the (prior) distribution over transition functions before the transition, ζ, and $\mathcal{P}(\Gamma|\zeta)$, the posterior distribution after ζ has occurred. This is commonly referred to as Information Gain, which we abbreviate as the *utility*, $u(\zeta)$, of a transition ζ:

$$u(s, a, s') = u(\zeta) = D_{\mathrm{KL}}(\mathcal{P}(\Gamma|\zeta) \parallel \mathcal{P}(\Gamma)). \tag{1}$$

The utility can be understood as the number of extra bits needed to specify the posterior relative to the prior, roughly the number of bits of information that was gathered about the external MDP. Given, $u(\zeta)$, it is now possible to compute the utility of the exploration policy, $U(\pi)$, which is the expected utility over all possible transitions when π is used:

$$U(\pi) = \mathbb{E}_{\zeta \sim \mathcal{P}(\cdot|\pi)} [u(\zeta)], \tag{2}$$

which can be expanded into (see Appendix A):

$$U(\pi) = \mathbb{E}_{\bar{\mathcal{T}} \sim \mathcal{P}(\Gamma)} \left[\mathbb{E}_{s,a \sim \mathcal{P}(\mathcal{S},\mathcal{A}|\pi,\bar{\mathcal{T}})} [\mathcal{U}(s, a)] \right], \tag{3}$$

2

where

$$\mathcal{U}(s,a) = \int_{s'} u(s,a,s')p(s'|a,s)\,ds'. \tag{4}$$

It turns out that $\mathcal{U}(s,a)$ is the same as (see Appendix B):

$$\mathcal{U}(s,a) = \text{JSD}\{\mathcal{P}(\mathcal{S}|s,a,\bar{\mathcal{T}}) \mid \bar{\mathcal{T}} \sim \mathcal{P}(\Gamma)\} \tag{5}$$

where JSD is the Jensen–Shannon Divergence or Information Radius, which captures the amount of disagreement present in a space of distributions. Hence, the utility of the state-action pair, $\mathcal{U}(s,a)$, is the disagreement, in terms of JSD, among the next-state distributions of all possible transition functions, given s and a, weighted by their probability. This means that, $\mathcal{U}(s,a)$ can be used to assess the novelty of potential transitions using the prior, i.e. without having to actually execute the transition in external MDP.

Therefore, $\mathcal{U}$ can provide a signal with which to update the exploration policy in the same way that $\mathcal{R}$ is normally used to update an action policy. Stated another way, $\mathcal{U}$ is to the exploration MDP what the $\mathcal{R}$ is to the external MDP, so that maximizing $\mathcal{U}$ results in the optimal exploration policy for the exploration MDP just as maximizing $\mathcal{R}$ results in the optimal action policy.

The exploration MDP is then defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}^E, \mathcal{U}, \rho_0^E)$, where the sets $\mathcal{S}$ and $\mathcal{A}$ are identical to those of the external MDP being explored, and the transition function $\mathcal{T}^E$ is potentially a different member of Γ at each state transition. Finally, the initial state distribution ρ_0^E is set to $\rho_0^E(s_t) = 1$ such that start state of exploration MDP is always the current state in the environment, s_t.

It is important to understand for what follows in the next section that the prior, $\mathcal{P}(\Gamma)$, is used twice in Equation 5:

1. To specify the transition dynamics. The prior determines a distribution, $\mathcal{P}(\mathcal{S}, \mathcal{A}|\pi, \bar{\mathcal{T}}, \rho_0)$, over the set of possible state-action pairs that can result by sequentially executing an infinite number of actions according to π using all of the transition functions, starting in all states distributed according to ρ_0.

2. To obtain the utility for a particular transition. Each state-action pair in $\mathcal{P}(\mathcal{S}, \mathcal{A}|\pi, \bar{\mathcal{T}}, \rho_0)$, when input to all the transition functions in Γ form a set of predicted next state distributions $\{\mathcal{P}(\mathcal{S}|s,a,\bar{\mathcal{T}}) \mid \bar{\mathcal{T}} \sim \mathcal{P}(\Gamma)\}$, and $\mathcal{U}(s,a)$ is the JSD of this set.

Since, in general, performing these steps is intractable, the next section presents a practical method to implement exploration based on these principles.

2.3 Bootstrap Ensemble Approximation

The prior, $\mathcal{P}(\Gamma)$, can be approximated using a bootstrap ensemble of M learned transition functions or models that are each trained independently using different subsets of the *history*, H, consisting of state transitions experienced by the agent while exploring the external MDP (Efron, 2012). Therefore, while $\mathcal{P}(\Gamma)$ is uniform when the agent is initialized, thereafter it is conditioned on agent's history H and so that the the general form of the prior is $\mathcal{P}(\Gamma|H)$. For generalizations that are warranted by observed data, there is a good chance that the models make similar predictions. If the generalization is not warranted by observed data, then the models could disagree owing to their exposure to different parts of the data distribution.

Since the ensemble contains models that were trained to accurately approximate the data, these models have higher probabilities or weights in $p(\bar{\mathcal{T}})$. The probabilities of inaccurate models are negligible, so they do not affect the properties of the distribution significantly and can safely be ignored. Hence randomly sampling from the ensemble can approximate the true distribution, $\mathcal{P}(\Gamma|H)$, even with relatively small sample sizes (Lakshminarayanan et al., 2017). Recent work suggests that it possible to do so even in high-dimensional state and action spaces (Kurutach et al., 2018; Chua et al., 2018).

Assuming all models fit their corresponding data equally well, then they all have the same weights and can therefore be treated equally. Given an M-model ensemble approximating, the exploration MDP can be approximated by randomly selecting one of the M of the models with equal probability at each state transition.

Algorithm 1 MODEL-BASED ACTIVE EXPLORATION (MAX)
___
1: **Initialize:** Dataset H with transitions obtained using random policy
2: **Initialize:** Ensemble $\mathcal{T}^E = \{\bar{\mathcal{T}}^1, \bar{\mathcal{T}}^2, \ldots, \bar{\mathcal{T}}^M\}$ of probabilistic forward models
3: **repeat**
4:     **while** episode not complete **do**
5:         $\pi \leftarrow$ Solve EXPLORATIONMDP$(\mathcal{S}, \mathcal{A}, \mathcal{T}^E, \mathcal{U}, \rho_0^E)$ $\triangleright$ find best exploration policy
6:         $a_t \sim \pi(s_t)$ $\triangleright$ select action
7:         $s_{t+1} \sim \mathcal{P}(\mathcal{S}|, s_t, a_t, \mathcal{T})$ $\triangleright$ act in environment according to policy
8:         $H \leftarrow H \cup \{(s_t, a_t, s_{t+1})\}$ $\triangleright$ add observed transition to dataset
9:         **for each** $\bar{\mathcal{T}}^i$ **in** $\mathcal{T}^E$ **do**
10:            Train $\bar{\mathcal{T}}^i$ on H $\triangleright$ update all models with bootstrap sampling of data
11:         **end for**
12:     **end while**
13: **until** computation budget exhausted
___

To approximate $\mathcal{U}(s,a)$, the JSD in Equation 5 can be expanded as (see Appendix B):

$$\mathcal{U}(s,a) = \text{JSD}\{\mathcal{P}(\mathcal{S}|s,a,\bar{\mathcal{T}}) \mid \bar{\mathcal{T}} \sim \mathcal{P}(\Gamma)\} \tag{6}$$

$$= \mathfrak{H}\left(\mathbb{E}_{\bar{\mathcal{T}} \sim \mathcal{P}(\Gamma)}\left[\mathcal{P}(\mathcal{S}|s,a,\bar{\mathcal{T}})\right]\right) - \mathbb{E}_{\bar{\mathcal{T}} \sim \mathcal{P}(\Gamma)}\left[\mathfrak{H}(\mathcal{P}(\mathcal{S}|s,a,\bar{\mathcal{T}}))\right] \tag{7}$$

where $\mathfrak{H}(\cdot)$ denotes the entropy of the distribution. Equation 6 can be summarized as the difference between the entropy of the average and the average entropy and can be approximated by averaging over samples taken from the ensemble:

$$\mathcal{U}(s,a) \simeq \mathfrak{H}\left(\frac{1}{N}\sum_{i=1}^{N}\mathcal{P}(\mathcal{S}|s,a,\bar{\mathcal{T}}^i)\right) - \frac{1}{N}\sum_{i=1}^{N}\mathfrak{H}(\mathcal{P}(\mathcal{S}|s,a,\bar{\mathcal{T}}^i)) \tag{8}$$

The entropies in Equation 8 can also be evaluated in closed form since many models output the next state representations such as one-hot or Gaussian for which this is possible.

Algorithm 1 gives the high-level pseudocode of MAX. This process is essentially a model-based RL algorithm with exploration as its objective. At each step a fresh policy is initialized and improved to maximise its return in the exploration MDP. The solution policy thus obtained is used to act in the external MDP to collect data. Training the ensemble with the gathered data at each step gives us the approximate posterior. This posterior can then be used as approximate prior in the next exploration step.

The amount of environment experience needed to find exploration policies is substantially reduced by learning policies using the models in the ensemble. If the regions that the exploration policy visits have less disagreement among the predictions of models, then it is likely that the dynamics of the models are very close to environment. Hence the policy can rely on such dynamics to reach certain regions of the environment. If the policy visits areas where there is a lot of disagreement, then the models are likely to be wrong. Although their predicted future is incorrect and the policy cannot rely on it, the policy still directs the agent to regions with novelty.

MAX is also self-correcting to the errors of models based on function approximators. If learning some aspects of the environment is difficult, it will manifest itself as disagreement in the ensemble. Hence MAX would collect more data about those aspects to aid learning.

Ideally, both training the model ensemble and policy optimization can occur at each step in the environment. But in practice they can both just be updated from step to step, episode to episode or over multiple episodes to minimize cost. But as soon as the ensemble is trained, the policy has to be updated or discarded and retrained from scratch.

3 Experiments

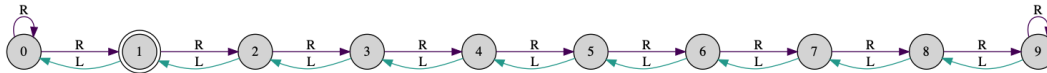## 3.1 Random Chain

3.1.1 Environment

Figure 1: Chain Environment of length 10.

We used a randomized version of the Chain Environment (Figure 1), designed to be hard to explore proposed by (Osband et al., 2016) which is a simplified generalization of RiverSwim (Strehl and Littman, 2005). The environment consists of N states in a series. Starting in the second state (state 1) of an N state chain, the agent can move either left or right at each step. An episode lasts for $N + 9$ steps, after which the agent is reset again to the second state. The agent is first given 3 episodes of warm up during which the actions are chosen randomly. Trying to move outside the chain results in staying in place. The agent is rewarded only for staying in the edge states: 0.001 and 1 for the leftmost and the rightmost state, respectively. To make the problem harder to exploit by a simple policy that always goes to the right, the effect of each action was randomly swapped so that in some states, going RIGHT results in a leftward transition and vice-versa. We used chain length of $N = 50$ unless stated otherwise. With random exploration, it is unlikely to explore the environment comprehensively and to reach the far right states, in particular. The probability of the latter is in $O(2^{-N})$. Thus, in order to explore efficiently, an agent needs to exploit the structure of the environment.

3.1.2 Evaluation Methodology

In the context of pure exploration the returns the methods obtain can be disregarded, shifting focus solely to how thoroughly they explore the MDP. A Random Chain Environment of length N has $2N$ transitions. A perfect exploration method should observe all transitions in the smallest number of episodes. Hence, the fraction of transitions observed by an agent so far was used as the metric. To asses both asymptotic performance and speed of the method, area under the curve over episodes was used.

3.1.3 Baseline Methods

We used two exploration methods based on the *optimism in face of uncertainty* (OFU) principle (Kaelbling et al., 1996): Exploration Bonus DQN (Bellemare et al., 2016) and Bootstrapped DQN (Osband et al., 2016). They are based on the sample efficient DQN algorithm (Mnih et al., 2015). Bootstrapped DQN is claimed to be better than "state of the art approaches to exploration via dithering (ϵ-greedy), optimism and posterior sampling"(Osband et al., 2016). Both of them are *reactive* since they do not explicitly seek new transitions, but upon finding one, prioritize frequenting it. It is important to note that these baselines are "any-time-optimal" RL algorithms which minimize cumulative regret by trading-off exploration and exploitation in each action. In contrast, the agent incurs no cost for exploration in the pure exploration problem considered in this paper. Hence, the baselines tackle a more general problem and do not take advantage of additional assumptions available in the pure exploration setup.

Appendix C contains a detailed description and implementation details of the baseline methods. The Hyper-Parameters for both the baseline methods were tuned with grid search (see Appendix D).

3.1.4 MAX Implementation

The ensemble consisted of 3 forward models implemented as fully-connected neural networks, each receiving one-hot-encoded state of the environment and the action of the agent as input. The outputs of the network were categorical distributions over the (next) states. The networks were independently and randomly initialized which was found to be sufficient to foster the diversity for the out-of-sample predictions and hence bootstrap sampling was not used. After warm-up the models were trained for 150 iterations. Searching for an exploration policy in the exploration MDP, which
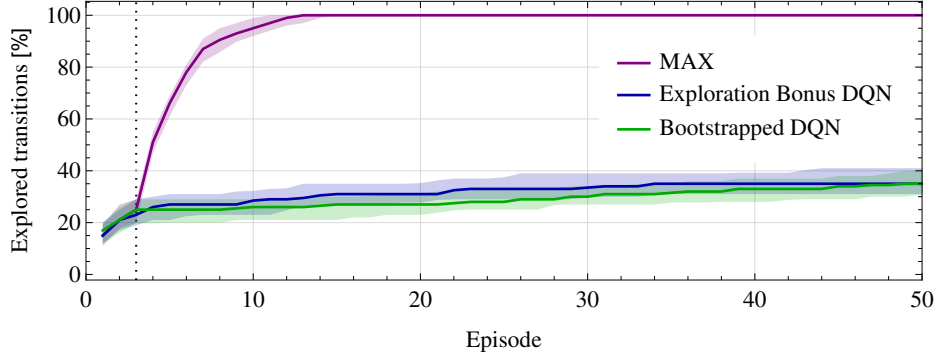
Figure 2: The percentage of explored transitions for the Random Chain Environment with the chain of length $N = 50$. For the first 3 episodes, marked by the vertical dotted line, actions were chosen at random. Each line corresponds to the median of 100 runs, and the shaded area spans the 25th and 75th percentiles.

was an open-loop plan, was performed using 25 rounds of Monte Carlo Tree Search (MCTS), which used 5 random trajectories for each tree expansion with Thompson Sampling as the selection policy. The best action was chosen based on the average value of the children. Models were trained after experiencing each transition in the environment and the MCTS tree was discarded after every step. The Hyper-Parameters of MAX were tuned with grid search for the chain length of $N = 50$ (see Appendix D).

3.1.5 Results

Figure 2 shows the percentage of explored transitions as a function of training episodes for all the methods. MAX explores 100% of the transitions in around 15 episodes while the baseline methods reach 40% in 60 episodes. Figures 5a, 5b and 6 in the appendix analyze the behaviour of the proposed algorithm. Fig. 5a plots the utility of the exploration policy during two learning episodes. Clearly, high utility correlates with experiencing novel transitions. Figure 5b, which shows the learning curves of individual models in an ensemble, indicates that less than 20 episodes is required to correctly learn all the transitions. The surrogate models are therefore indistinguishable from the environment. Notice also that the learning curves of the three models are very close to each other. This is because the models nearly always agree on the transitions from the training set and disagree on the others. The exploration is thus mainly driven by the divergence of predictions for the unobserved transitions. Figure 6 visualizes the transitions in the final two episodes of a run showing how MAX plans for uncertain (unvisited) transitions, which results in minimizing their corresponding uncertainties.

Scalability We varied the chain length from 20 to 100 in intervals of 5. Figure 3a shows the experience curves obtained with MAX. Exploring longer chains require more episodes but the figure clearly indicates that the same algorithm that works well for the chain length of 50 works well for other lengths, without any extra hyper-parameter turning.

Sensitivity to Noise To see if MAX can actually distinguish between the environment risk and uncertainty, the left-most state (state 0) of the Chain Environment was modified to be a *stochastic trap* state. If the agent is in the trap state, regardless of the chosen action, it is equally likely to remain in state 0 or end up in state 1. A method that relies only on prediction errors cannot separate risk from uncertainty. Hence it would repeatedly visit the trap state as it is easier to reach than some far-off uncertain state. Figure 3b compares the performance of our method on both the normal Chain Environment and the one with a stochastic trap state. Although MAX is slowed down, it still manages to explore the transitions. The stochastic trap state increases the variance in the output of the models. This is because the models are trained on different outputs for the same input. Hence the resulting output distribution of the models are sensitive to the training samples the models have seen. This can cause disagreements and hence result in a higher utility right next to the initial state. The true uncertainty on the right, which is fainter, is no longer the dominant source of uncertainty. This causes even our method to struggle, despite being able to separate the two forms of uncertainty.
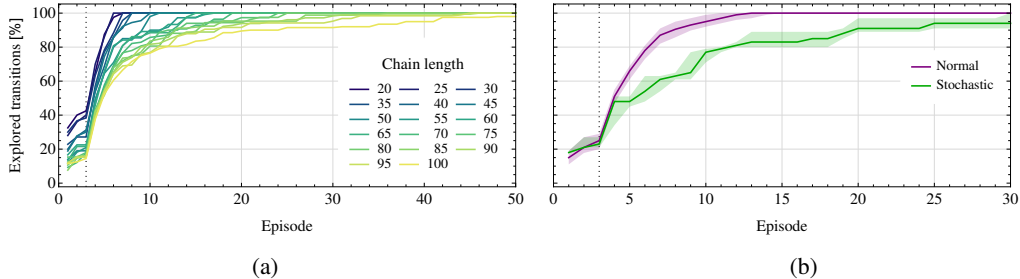
6

Figure 3: The MAX algorithm performance for (a) different chain lengths and (b) in the presence of a stochastic trap. Each learning curve is a median of 5 runs with different seeds. The vertical dotted line marks the ends of the warm-up phase.

Robustness to Key Hyper-Parameters   We varied the number of models in the ensemble, the number of trajectories per an MCTS iteration, and the number of MCTS iterations. The results are shown in Figure 7. In general, more trajectories, more models in the ensemble and more planning iterations lead to better performance. Notice, however, that the algorithms works efficiently also for minimal settings.

3.2   Continuous Mountain Car

MAX can also be applied to environments with continuous state and action spaces. Figure 4 shows the result when MAX was used for the simple Continuous Mountain Car problem which has a 2D state space and a 1D action space. MAX was able to drive the car up and down the valley to develop the required resonance energy and reach the mountain top quickly *without* any external reward.
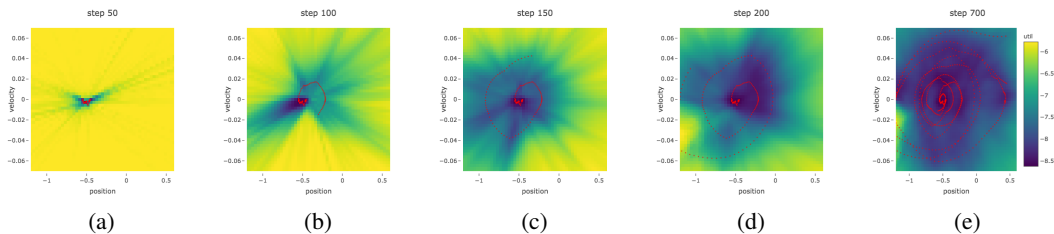

Figure 4: Exploration in Continuous Mountain Car Environment. Each plot shows a 2D grid representing the state space of the agent. The average uncertainty of a state over all actions determines the color of its corresponding point in the grid. The red dots mark the current experience of the agent.

4   Discussion

Meta-Stability and Active Exploration   A policy is meta-stable if it is sub-optimal and, at the same time, prevents the agent from gaining experience necessary to improve it (Watkins, 1989). That is, a policy can get stuck in a local optimum and not be able to get out of it. In the simple cases, undirected exploration techniques (Thrun, 1992) like adding random noise to the actions of the policy might be sufficient to break away of meta-stability. If the environment is ergodic, then reactive exploration strategies that use exploration bonuses can solve meta-stability. But active exploration of the form presented in this paper, can in principle break free of any type of meta-stability.

The Uncertainty Gradient   While exploring an environment completely can be challenging, finding the best exploration policy in the exploration MDP requires exploration itself, which could also be hard. For the Chain Environment, within 25 MCTS planning rounds, the MCTS planner can typically look-ahead 5 states which is minuscule compared to the number of look-ahead steps required to plan to reach the right end of the chain starting from the left end. However in practice, our sample efficiency is dramatically greater due to the gradual uncertainty increase from left to right or the *uncertainty gradient*. Due to more visitation, states near the initial state have a higher representation

in the dataset. There are less samples for transitions of states farther away from initial state hence there is more uncertainty. The uncertainty gradient is highly informative and the planner can get away acting nearly greedily.

Without the uncertainty gradient, MAX cannot do better than random search at exploring the exploration MDP. However, models are faster, often by several orders of magnitude and can be searched in parallel. The uncertainty gradient could naturally occur in many environments as it did in the Chain Environment. The uncertainty gradient could also be extracted by differentiating through the ensemble or by probing it by some other means.

Implications to Model Based RL   MAX can also been seen as a natural exploration mechanism in model-based RL. Model-based RL promises to be significantly more efficient and more general compared to model-free RL methods. However, it suffers from model-bias (Deisenroth and Rasmussen, 2011): in certain regions of the state space, the models could deviate significantly from the external MDP. Model-bias could occur due to many reasons such as improper generalization and/or poor exploration. A strong policy search method could then exploit such degeneracy resulting in over-optimistic policies that fail in the environment. Thorough exploration is one way to potentially mitigate this issue. Since model-based RL does not have any inherent mechanism to explore, we consider our method to be an important addition to the model-based RL framework rather than merely being an application of it.

Limitations   Our derivation makes the assumption that the utility of a policy is the average utility of the transitions possible when the policy is used. However, encountering a subset of those transitions and training the models can change the utility of the remaining transitions, thereby affecting the utility of the policy. We do not consider this second order effect in our derivation. In the Chain Environment for example, this effect leads to the agent planning to loop between pairs of uncertain states, rather than visiting many uncertain states. However ignoring this effect is not necessarily problematic as information gain is additive under expectation Sun et al. (2011).

5   Related Work

Our work is inspired by the framework developed in Schmidhuber (1997, 2002), in which two adversarial reward-maximizing modules called the *left brain* and the *right brain* bet on outcomes of experimental protocols or algorithms they have collectively generated and agreed upon. Each brain is intrinsically motivated to outwit or surprise the other by proposing an experiment such that the other *agrees* on the experimental protocol but *disagrees* on the predicted outcome. After having executed the action sequence protocol approved by both brains, the surprised loser pays a reward to the winner in a zero sum game. MAX greatly simplifies this previous active exploration framework, distilling certain essential aspects. Two or more predictive models that may compute different hypotheses about the consequences of the actions of the agent, given observations are still used. However, there is only one single reward maximizer or RL machine, which is separate from the predictive models.

The information provided by an experiment was first analytically measured by Lindley (1956) in the form of expected information gain in Shannon sense (Shannon, 1948). Fedorov (1972) also proposed a theory of optimal resource allocation during experimentation. By the 1990s, information gain was used as an intrinsic reward for reinforcement learning (RL) systems (Storck et al., 1995). Even earlier, intrinsic reward signals were based on prediction errors of a predictive model (Schmidhuber, 1991a) and on the learning progress of a predictive model (Schmidhuber, 1991b). Thrun (1992) introduced notions of directed and undirected exploration in Reinforcement Learning. Optimal Bayesian experimental design (Chaloner and Verdinelli, 1995) is a framework for efficiently performing sequential experiments that uncover a phenomenon. However, usually the approach is restricted to linear models with Gaussian assumptions. Busetto et al. (2009) propose an optimal experimental design framework for model selection of nonlinear biochemical systems using expected information gain where they solve for the posterior using the Fokker-Planck equation. In Model Based Interval Estimation (Wiering, 1999) the uncertainty in the transition function captured by a surrogate model is used to boost Q-values of actions. In context of Active Learning, McCallumzy and Nigamy (1998) proposed to use Jensen Shannon Divergence among predictions of a committee of classifiers to identify the most useful sample to be labelled next among a pool of unlabelled samples. Itti and Baldi (2009) presented the surprise formulation used in Equation 1 and demonstrated a strong

correlation surprise and human attention. At a high-level, MAX can be seen as a form of Bayesian Optimization (Snoek et al., 2012) adopted for exploration in RL which employs an inner search based optimization during planning. Curiosity has also been studied extensively from the perspective of developmental robotics (Oudeyer, 2018). A very general form of learning progress is compression progress used as an extra intrinsic reward for curious RL systems (Schmidhuber, 2009).

Following these, Sun et al. (2011) developed an optimal Bayesian framework for curiosity-driven exploration using learning progress. After proving that Information Gain is additive in expectation, a dynamic programming based algorithm was proposed to maximize Information Gain. Experiments however were limited to small tabular MDPs with a Dirichlet prior on transition probabilities. A similar Bayesian inspired, hypothesis-resolving, model-based RL exploration algorithm was proposed in Hester and Stone (2012) and shown to outperform prediction error based and other intrinsic motivation methods. In contrast to MAX, planning was performed by taking the mean prediction of an ensemble of models in order to optimize a disagreement-based utility measure which was also augmented with an additional state-distance bonus. Still and Precup (2012) derive a trade-off between exploration and exploitation in an attempt to maximize the predictive power of the agent. Mohamed and Rezende (2015) combine Variational Inference and Deep Learning to form an objective based on mutual information to approximate agent empowerment. In comparison to our method, Houthooft et al. (2016) present a Bayesian approach to evaluate the value of experience taking a reactive approach. However, they also use Bayesian Neural Networks to maintain a belief over environment dynamics and use the information gain to bias the policy search with an intrinsic motivation reward component. Variational Inference is used to approximate the prior-posterior KL divergence. Bellemare et al. (2016) derive a notion of pseudo-count for estimating state visitation frequency in high-dimensional spaces. They then transform this into a form of exploration bonus that is maximized using DQN. Osband et al. (2016) propose Bootstrapped DQN which was used as a baseline. Pathak et al. (2017) use inverse models to avoid exploring learning anything that the agent cannot control to reduce risk. But they are still restricted to reactive exploration. A large scale study of curiosity driven exploration Burda et al. (2018) found that curiosity is correlated with the actual objectives of many environments, and report that using random features mitigates some of the non-stationarity implicit in methods based on curiosity.

Model-based RL has long been touted as the cure for sample inefficiency problems of modern RL (Schmidhuber, 1990; Deisenroth and Rasmussen, 2011). Yet, learning accurate models of high-dimensional environments and exploiting them appropriately in downstream tasks is still an active area of research. Recently Kurutach et al. (2018) and Chua et al. (2018) have shown the potential of model-based RL when combined with Deep Learning in high-dimensional environments. In particular, we were inspired by Chua et al. (2018) who use probabilistic models and combine them with novel trajectory sampling techniques using particles to obtain better approximations of the returns in the environment.

6 Conclusion

We introduced MAX, a model-based RL algorithm for pure exploration. It can separate learnable and unlearnable unknowns and actively search for policies that seek learnable unknowns from the environment. The algorithm derived in this paper provides the means to use an ensemble of models for simulation and evaluation of an exploration policy. The quality of the exploration policy can therefore be directly optimized without real environment experience. MAX does not utilize non-stationary objectives or suffer from over-commitment as it is specified as a model-based RL algorithm. Preliminary experiments in an environment intractable for traditional exploration methods indicate that MAX is generic and could scale to large environments.

References

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.

Bubeck, S., Munos, R., and Stoltz, G. (2009). Pure Exploration In Multi-armed Bandits Problems. In *International Conference On Algorithmic Learning Theory*, pages 23–37. Springer.

Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., and Efros, A. A. (2018). Large-Scale Study of Curiosity-Driven Learning. *arXiv preprint arXiv:1808.04355*.

Busetto, A. G., Ong, C. S., and Buhmann, J. M. (2009). Optimized Expected Information Gain for Nonlinear Dynamical Systems. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 97–104. ACM.

Chaloner, K. and Verdinelli, I. (1995). Bayesian Experimental Design: A Review. *Statistical Science*, pages 273–304.

Chua, K., Calandra, R., McAllister, R., and Levine, S. (2018). Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models. *arXiv preprint arXiv:1805.12114*.

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*, pages 465–472.

Efron, B. (2012). Bayesian Inference and the Parametric Bootstrap. *Annals of Applied Statistics*, 6(4):1971–1997.

Fedorov, V. (1972). *Theory of Optimal Experiments Designs*. Academic Press.

Hester, T. and Stone, P. (2012). Intrinsically Motivated Model Learning for Developing Curious Robots. In *Development and Learning and Epigenetic Robotics (ICDL), 2012 IEEE International Conference on*, pages 1–6. IEEE.

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.

Itti, L. and Baldi, P. (2009). Bayesian Surprise Attracts Human Attention. *Vision Research*, 49(10):1295–1306.

Kaelbling, L. P., Littman, M. L., and Moore, a. W. (1996). Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4:237–285.

Kurutach, T., Clavera, I., Duan, Y., Tamar, A., and Abbeel, P. (2018). Model-Ensemble Trust-Region Policy Optimization. *arXiv preprint arXiv:1802.10592*.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413.

Lindley, D. V. (1956). On a Measure of the Information Provided by an Experiment. *The Annals of Mathematical Statistics*, pages 986–1005.

McCallumzy, a. K. and Nigamy, K. (1998). Employing EM and Pool-Based Active Learning for Text Classification. In *Proceedings International Conference on Machine Learning (ICML)*, pages 359–367. Citeseer.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, a. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, a. K., Ostrovski, G., et al. (2015). Human-Level Control Through Deep Reinforcement Learning. *Nature*, 518(7540):529.

Mohamed, S. and Rezende, D. J. (2015). Variational Information Maximisation For Intrinsically Motivated Reinforcement Learning. In *Advances In Neural Information Processing Systems*, pages 2125–2133.

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.

Oudeyer, P.-Y. (2018). Computational Theories of Curiosity-Driven Learning. *arXiv preprint arXiv:1802.10546*.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). Curiosity-Driven Exploration by Self-Supervised Prediction. In *Proceedings of The 35th International Conference On Machine Learning*.

Schmidhuber, J. (1990). Making the World Differentiable: On Using Fully Recurrent Self-Supervised Neural Networks for Dynamic Reinforcement Learning and Planning in Non-Stationary Environments. Technical Report FKI-126-90 (revised), Institut für Informatik, Technische Universität München.

Schmidhuber, J. (1991a). A Possibility for Implementing Curiosity and Boredom in Model-Building Neural Controllers. In Meyer, J. A. and Wilson, S. W., editors, *Proceedings of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats*, pages 222–227. MIT Press/Bradford Books.

Schmidhuber, J. (1991b). Curious Model-Building Control Systems. In *Proceedings of the International Joint Conference on Neural Networks, Singapore*, volume 2, pages 1458–1463. IEEE press.

Schmidhuber, J. (1997). What's Interesting? Technical Report IDSIA-35-97, IDSIA.

Schmidhuber, J. (2002). Exploring the Predictable. In Ghosh, A. and Tsuitsui, S., editors, *Advances in Evolutionary Computing*, pages 579–612. Springer.

Schmidhuber, J. (2009). Driven by Compression Progress: A Simple Principle Explains Essential Aspects of Subjective Beauty, Novelty, Surprise, Interestingness, Attention, Curiosity, Creativity, Art, Science, Music, Jokes. In Pezzulo, G., Butz, M. V., Sigaud, O., and Baldassarre, G., editors, *Anticipatory Behavior in Adaptive Learning Systems. From Psychological Theories to Artificial Cognitive Systems*, volume 5499 of *LNCS*, pages 48–76. Springer.

Shannon, C. E. (1948). A Mathematical Theory of Communication (Parts I and II). *Bell System Technical Journal*, XXVII:379–423.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical Bayesian Optimization Of Machine Learning Algorithms. In *Advances in neural information processing systems*, pages 2951–2959.

Still, S. and Precup, D. (2012). An Information-Theoretic Approach To Curiosity-Driven Reinforcement Learning. *Theory in Biosciences*, 131(3):139–148.

Storck, J., Hochreiter, S., and Schmidhuber, J. (1995). Reinforcement Driven Information Acquisition in Non-Deterministic Environments. In *Proceedings of the International Conference on Artificial Neural Networks, Paris*, volume 2, pages 159–164. Citeseer.

Strehl, A. L. and Littman, M. L. (2005). A Theoretical Analysis of Model-Based Interval Estimation. In *Proceedings of The 22nd International Conference On Machine Learning*, pages 856–863. ACM.

Sun, Y., Gomez, F., and Schmidhuber, J. (2011). Planning to Be Surprised: Optimal Bayesian Exploration in Dynamic Environments. In *International Conference on Artificial General Intelligence*, pages 41–51. Springer.

Thrun, S. B. (1992). Efficient Exploration In Reinforcement Learning. *Technical Report*.

Watkins, C. J. C. H. (1989). *Learning From Delayed Rewards*. PhD thesis, King's College, Cambridge.

Wiering, M. A. (1999). *Explorations in Efficient Reinforcement Learning*. PhD thesis, University of Amsterdam.

A   Policy Evaluation

$U(\pi)$ can be rewritten as:

$$U(\pi) = \int_s \int_a \int_{s'} u(s,a,s')p(s,a,s'|\pi)\, ds'\, da\, ds$$

where $p(s,a,s'|\pi)$ is the probability of transition (s,a,s') occurring given π is the acting policy.

$$U(\pi) = \int_s \int_a \int_{s'} u(s,a,s')p(s'|s,a)p(s,a|\pi)\, ds'\, da\, ds \tag{9}$$

Define action utility function $\mathcal{U}(s,a)$ (Equation 4) which quantifies the net utility of taking action a from state s as:

$$\mathcal{U}(s,a) = \int_{s'} u(s,a,s')p(s'|a,s)\, ds' \tag{10}$$

Rewriting Equation 9 using 4

$$U(\pi) = \int_s \int_a \mathcal{U}(s,a)p(s,a|\pi)\, da\, ds$$

Since the true transition function is unknown, the utility of a policy has to be marginalized over our belief of it:

$$U(\pi) = \int_{\bar{\mathcal{T}}} \int_s \int_a \mathcal{U}(s,a)p(s,a|\pi,\bar{\mathcal{T}})p(\bar{\mathcal{T}})\, da\, ds\, d\bar{\mathcal{T}}$$

which reduces to the following expectation (Equation 11):

$$U(\pi) = \mathbb{E}_{\bar{\mathcal{T}} \sim \mathcal{P}(\Gamma)} \left[\mathbb{E}_{s,a \sim \mathcal{P}(\mathcal{S},\mathcal{A}|\pi,\bar{\mathcal{T}})} \left[\mathcal{U}(s,a) \right] \right] \tag{11}$$

B   Action Evaluation

To obtain a closed form expression for $\mathcal{U}(s,a)$, Equation 4 can be expanded using Equation 1 as:

$$\mathcal{U}(s,a) = \int_{s'} u(s,a,s')p(s'|a,s)\, ds'$$

$$\mathcal{U}(s,a) = \int_{s'} D_{\mathrm{KL}}(\mathcal{P}(\Gamma|\zeta) \,\|\, \mathcal{P}(\Gamma))\, p(s'|a,s)\, ds'$$

Expanding the equation from definition of KL divergence:

$$D_{\mathrm{KL}}(\mathcal{P}_1 \,\|\, \mathcal{P}_2) = \int_z p_1(z) \log \left(\frac{p_1(z)}{p_2(z)} \right) dz$$

$$\mathcal{U}(s,a) = \int_{s'} \int_{\bar{\mathcal{T}}} p(\bar{\mathcal{T}}|\zeta) \log \left(\frac{p(\bar{\mathcal{T}}|\zeta)}{p(\bar{\mathcal{T}})} \right) p(s'|a,s)\, d\bar{\mathcal{T}}\, ds' \tag{12}$$

The prior $p(\Gamma)$ and the posterior $p(\Gamma|\zeta)$ are related by Bayes rule:

$$p(\Gamma|\zeta) = \frac{p(\zeta|\Gamma)p(\Gamma)}{p(\zeta)} \tag{13}$$

Equation 12 can be now rewritten as:

$$\mathcal{U}(s,a) = \int_{s'} \int_{\bar{\mathcal{T}}} \frac{p(\zeta|\bar{\mathcal{T}})p(\bar{\mathcal{T}})}{p(\zeta)} \log \left(\frac{p(\zeta|\bar{\mathcal{T}})}{p(\zeta)} \right) p(s'|a,s)\, d\bar{\mathcal{T}}\, ds'$$

Substituting $\zeta = (s, a, s')$:

$$\mathcal{U}(s, a) = \int_{s'} \int_{\bar{\mathcal{T}}} \left[\frac{p(s, a, s'|\bar{\mathcal{T}})p(\bar{\mathcal{T}})}{p(s, a, s')} \log \left(\frac{p(s, a, s'|\bar{\mathcal{T}})}{p(s, a, s')} \right) \right] p(s'|a, s) \, d\bar{\mathcal{T}} \, ds'$$

$$\mathcal{U}(s, a) = \int_{s'} \int_{\bar{\mathcal{T}}} \left[p(s'|s, a, \bar{\mathcal{T}})p(\bar{\mathcal{T}}) \log \left(\frac{p(s'|s, a, \bar{\mathcal{T}})}{p(s'|s, a)} \right) \right] d\bar{\mathcal{T}} \, ds' \tag{14}$$

However:

$$p(s'|s, a) = \int_{\bar{\mathcal{T}}} p(s'|s, a, \bar{\mathcal{T}})p(\bar{\mathcal{T}}) \, d\bar{\mathcal{T}} \tag{15}$$

Using Equation 15 in 14, expanding and swapping integrals in the former term:

$$\mathcal{U}(s, a) = \int_{\bar{\mathcal{T}}} \int_{s'} p(s'|s, a, \bar{\mathcal{T}}) \log \left(p(s'|s, a, \bar{\mathcal{T}}) \right) p(\bar{\mathcal{T}}) \, ds' \, d\bar{\mathcal{T}}$$
$$- \int_{s'} \int_{\bar{\mathcal{T}}} p(s'|s, a, \bar{\mathcal{T}})p(\bar{\mathcal{T}}) \log \left(\int_{\bar{\mathcal{T}}} p(s'|s, a, \bar{\mathcal{T}})p(\bar{\mathcal{T}}) \, d\bar{\mathcal{T}} \right) d\bar{\mathcal{T}} \, ds' \tag{16}$$

$- \int_z p(z) \log \left(p(z) \right) dz$ is just entropy $\mathfrak{H}(\mathcal{P}(z))$.

$$\mathcal{U}(s, a) = \mathfrak{H} \left(\int_{\bar{\mathcal{T}}} \mathcal{P}(\mathcal{S}|s, a, \bar{\mathcal{T}})p(\bar{\mathcal{T}}) \, d\bar{\mathcal{T}} \right) - \int_{\bar{\mathcal{T}}} \mathfrak{H}(\mathcal{P}(\mathcal{S}|s, a, \bar{\mathcal{T}}))p(\bar{\mathcal{T}}) \, d\bar{\mathcal{T}} \tag{17}$$

$$= \mathfrak{H} \left(\mathbb{E}_{\bar{\mathcal{T}} \sim \mathcal{P}(\Gamma)} \left[\mathcal{P}(\mathcal{S}|s, a, \bar{\mathcal{T}}) \right] \right) - \mathbb{E}_{\bar{\mathcal{T}} \sim \mathcal{P}(\Gamma)} \left[\mathfrak{H}(\mathcal{P}(\mathcal{S}|s, a, \bar{\mathcal{T}})) \right] \tag{18}$$

$\mathcal{P}(\mathcal{S}|s, a, \bar{\mathcal{T}})$ represents a probability distribution over the next state with removal of the integrals over s'. Given a space of distributions, the entropy of the average distribution minus the average entropy of distributions is the Jensen Shannon Divergence (JSD) of the space of distributions. It is also termed as the Information Radius and is defined as:

$$\mathrm{JSD}\{\mathcal{P}_{\bar{Z}} \mid \mathcal{P}_{\bar{Z}} \sim \mathcal{P}(\mathbf{Z})\} = \mathfrak{H} \left(\int_{\bar{Z}} \mathcal{P}_{\bar{Z}} p(\mathcal{P}_{\bar{Z}}) \, d\bar{Z} \right) - \int_{\bar{Z}} \mathfrak{H}(\mathcal{P}_{\bar{Z}}) p(\mathcal{P}_{\bar{Z}}) \, d\bar{Z} \tag{19}$$

where $\mathbf{Z}$ is the space of distributions $\mathcal{P}_{\bar{Z}}$. $\mathcal{P}(\mathbf{Z})$ is a compound probability distribution with $p(\mathcal{P}_{\bar{Z}})$ as its density function.

Therefore, Equation 10 can be expressed as:

$$\mathcal{U}(s, a) = \mathrm{JSD}\{\mathcal{P}(\mathcal{S}|s, a, \bar{\mathcal{T}}) \mid \bar{\mathcal{T}} \sim \mathcal{P}(\Gamma)\} \tag{20}$$

where Γ is the space of transition functions with probabilistic outputs.

C   Baselines

C.1   Description

Exploration Bonus DQN is an extension of the DQN algorithm (Mnih et al., 2015), in which the transitions that are visited for the first time carry an extra bonus reward. This causes the value of those transitions to be temporarily over-estimated and results in more visits to novel regions of the environment. In general, the exploration bonuses are computed using prediction errors of forward models (Pathak et al., 2017). However, in our simple environment, the bonuses are provided by an oracle, implemented as a transition look-up table. This is equivalent to having a model that can learn about a transition in one-shot.

Bootstrapped DQN by Osband et al. (2016) also extends the DQN algorithm and it is based on the same underlying principle as Exploration Bonus DQN. Instead of introducing arbitrary extra rewards, it relies on stochastic over-estimation of values in novel states. Multiple Q-value networks, referred to as *heads*, are maintained and trained on a shared replay buffer. Before every episode, a head is randomly selected and the agent acts greedily with respect to it. If a novel transition (s, a, s')

is added to the replay buffer, the hope is that at least one of the heads over-estimates the value of s', causing the policy resulting from TD-bootstrapping to prefer this state. This leads to that particular state being further explored when that head is used to act.

Commonly used ϵ-greedy exploration was turned off in both cases, thereby making exploration reliant solely on the methods themselves.

C.2 Implementation Details

The Q-value functions were implemented with multi-layer fully connected neural networks. The state was encoded with thermometer encoding for Bootstrapped DQN as proposed by Osband et al. (2016).

The Q-value network in Exploration Bonus DQN was trained with one batch of data after each step in the environment. For Bootstrapped DQN, the Q-value network was trained for a number of iterations *only after* each episode, with the exact number of iterations being tuned. The number of Bootstrap heads was fixed to 10. Bootstrap sampling was not employed and all heads were trained on all transitions collected so far. Target networks and replays were used for both methods as in the standard DQN.

RMSprop optimizer with a momentum of 0.9 was used to minimize the Huber Loss with gradient clipping of 5.

Both baselines were given 3 episodes of warm up to populate the buffers, during which the actions were chosen randomly.

D Hyper-parameter Tuning

In the experiments (see Section 3) Each hyper-parameter configuration was tested using 5 different random seeds. Hyper-Parameters were ranked according to the median of the area under the exploration curve.

All neural networks consisted of Glorot-initialized, *tanh*-activated fully connected layers with their depth and width being tuned.

Table 1: Hyper-Parameters for DQN with Exploration Bonus. *Grid Size: 4.3k*

Hyper-parameter	Values		
Exploration Bonus	5×10^{-3}	10^{-2}	10^{-1}
Replay Size	256	10^{5}	
Number of Hidden Layers	1	2	3
Hidden Layer Size	32	64	128
Learning Rate	10^{-2}	10^{-3}	10^{-4}
Batch Size	16	32	64
Discount Factor	0.95	0.975	0.99
Target Network Update Frequency	16	64	256

Table 2: Hyper-Parameters for Bootstrapped DQN. *Grid Size: 2.1k*

Hyper-parameter	Values		
Number of Hidden Layers	1	2	3
Hidden Layer Size	32	64	128
Learning Rate	10^{-2}	10^{-3}	10^{-4}
Batch Size	16	32	64
Number of Training Iterations per Episode	16	32	64
Discount Factor	0.95	0.975	0.99
Target Network Update Frequency	16	64	256

Table 3: Hyper-Parameters for MAX. *Grid Size: 0.8k*

Hyper-parameter	Values			
Number of Hidden Layers	2	3	4	
Hidden Layer Size	64	128	256	
Learning Rate	10^{-3}	10^{-4}		
Batch Size	16	64	256	
Number of Training Iterations per Episode	16	32	64	128
Weight Decay	0	10^{-5}	10^{-6}	10^{-7}

E   Supplementary Figures

(a)   (b)
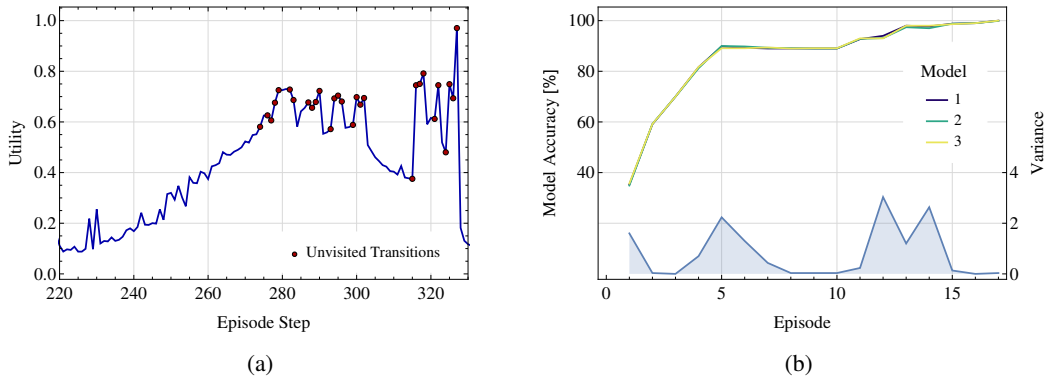
Figure 5: Exploration with MAX. (a) Utility of the exploration policy (in normalized log scale) during an exemplary episode. The red points mark encounters with novel transitions. (b) The accuracy of the models in the ensemble at each episode and the variance of model accuracy.
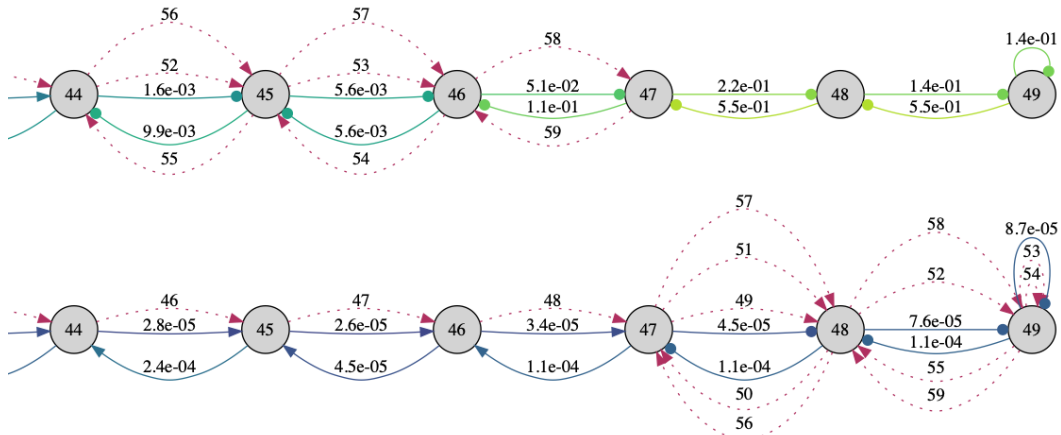

Figure 6: Instance of uncertainty minimization:MAX behaviour in the second-to-last (top) and the last (bottom) episodes. The solid arrows (environment transitions) and are annotated with the utilities of predicted information gains. The arrowheads are circular for the unvisited transitions and regular for the visited ones (before an episode). The dotted arrows with step numbers visualize the path the agent was following during an episode (only the final steps are shown).
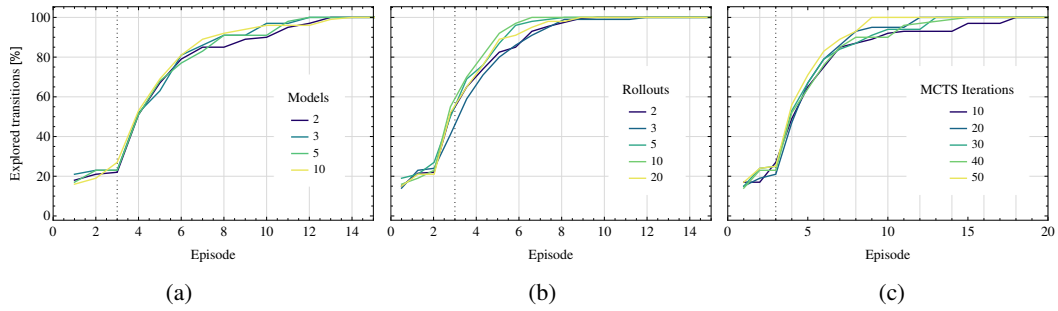
15

Figure 7: Algorithm Properties. Each learning curve is a median of 5 runs with different seeds. Vertical dotted line marks the ends of the warm-up phase. Sub-figures show how the percentage of explored transitions varies with respect to (a) ensemble size, (b) number of rollouts and (c) planning iterations.