Model-Based Active Exploration

Pranav Shyam [1] **Wojciech Jaśkowski** [1] **Faustino Gomez** [1]

Abstract

Efficient exploration is an unsolved problem in Reinforcement Learning which is usually addressed by reactively rewarding the agent for fortuitously encountering novel situations. This paper introduces an efficient *active* exploration algorithm, Model-Based Active eXploration (MAX), which uses an ensemble of forward models to plan to observe novel events. This is carried out by optimizing agent behaviour with respect to a measure of novelty derived from the Bayesian perspective of exploration, which is estimated using the disagreement between the futures predicted by the ensemble members. We show empirically that in semi-random discrete environments where directed exploration is critical to make progress, MAX is at least an order of magnitude more efficient than strong baselines. MAX scales to high-dimensional continuous environments where it builds task-agnostic models that can be used for any downstream task.

1. Introduction

Efficient exploration in large, high-dimensional environments 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 *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 probable models of the environment.

This paper introduces 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 constructs exploration policies to resolve those conflicts by visiting the relevant area. Unlearnable unknowns or *risk*, such as random noise in the environment does not interfere with the process since noise manifests as confusion among all models and not as a conflict.

In discrete environments, novelty can be evaluated using the Jensen-Shannon Divergence (JSD) between the predicted next state distributions of the models in the ensemble. In continuous environments, computing JSD is intractable, so MAX instead uses the functionally equivalent Jensen-Rényi Divergence based on Rényi quadratic entropy (Rényi, 1961).

While MAX can be used in conjunction with conventional policy learning to maximize external reward, this paper focuses on *pure exploration*: exploration disregarding or in the absence of external reward, followed by exploitation (Bubeck et al., 2009). This setup is more natural in situations where it is useful to do task-agnostic exploration and learn models that can later be exploited for multiple tasks, including those that are not known *a priori*.

Experiments in the discrete domain show that MAX is significantly more efficient than reactive exploration techniques which use exploration bonuses or posterior sampling, while also strongly suggesting that MAX copes with risk. In the high-dimensional continuous Ant Maze environment, MAX reaches the far end of a U-shaped maze in just 40 episodes (12k steps), while reactive baselines are only around the mid-way point after the same time. In the Half Cheetah

[1]NNAISENSE, Lugano, Switzerland. Correspondence to: Pranav Shyam <pranav@nnaisense.com>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

code: https://github.com/nnaisense/max

environment, the data collected by MAX leads to superior performance versus the data collected by reactive baselines when exploited using model-based RL.

2. Model-Based Active Exploration

The key idea behind our approach to active exploration in the environment, or the *external* Markov Decision Process (MDP), is to use a surrogate or *exploration* MDP where the novelty of 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.

2.1. Problem Setup

Consider the environment or the external MDP, represented as the tuple $(\mathcal{S}, \mathcal{A}, t^*, r, \rho_0)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, t^* is the unknown transition function, $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, \infty)$, specifying the probability density $p(s'|s, a, t^*)$ of the next state s' given the current state s and the action a, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\rho_0 : \mathcal{S} \to [0, \infty)$ is the probability density function of the initial state.

Let T be the space of all possible transition functions and $\mathcal{P}(T)$ be a probability distribution over transition functions that captures the current belief of how the environment works, with corresponding density function $p(T)$.

The objective of pure exploration is to efficiently accumulate information about the environment, irrespective of r. This is equivalent to learning an accurate model of the transition function, t^*, while minimizing the number of state transitions, ϕ, belonging to transition space Φ, required to do so, where $\phi = (s, a, s')$ and s' is the state resulting from action a being taken in state s.

Pure exploration can be defined as an iterative process, where in each iteration, an exploration policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, \infty)$, specifying a density $p(a|s, \pi)$, is used to collect information about areas of the environment that have not been explored up to that iteration. To learn such exploration policies, there needs to be a method to evaluate any given policy at each iteration.

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 r, i.e., the *return*. Because pure active exploration does not care about r, and t^* is unknown, the amount of new information conveyed about the environment by state transitions that *could* be caused by an exploration policy has to be used as the learning signal.

From the Bayesian perspective, this can be captured by the KL-divergence between $\mathcal{P}(T)$, the (prior) distribution over transition functions before a particular transition ϕ, and $\mathcal{P}(T|\phi)$, the posterior distribution after ϕ has occurred. This is commonly referred to as Information Gain, which is abbreviated as $\mathrm{IG}(\phi)$ for a transition ϕ:

$$\mathrm{IG}(s, a, s') = \mathrm{IG}(\phi) = D_{\mathrm{KL}}(\mathcal{P}(T|\phi) \parallel \mathcal{P}(T)). \quad (1)$$

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

$$\mathrm{IG}(\pi) = \mathbb{E}_{\phi \sim \mathcal{P}(\Phi|\pi)} \left[\mathrm{IG}(\phi) \right], \quad (2)$$

which can be expanded into (see Appendix A):

$$\mathrm{IG}(\pi) = \mathbb{E}_{t \sim \mathcal{P}(T)} \left[\mathbb{E}_{s,a \sim \mathcal{P}(\mathcal{S}, \mathcal{A}|\pi, t)} \left[u(s, a) \right] \right], \quad (3)$$

where

$$u(s, a) = \int_T \int_{\mathcal{S}} \mathrm{IG}(s, a, s') p(s'|a, s, t) p(t) \, ds' dt. \quad (4)$$

It turns out that (see Appendix B):

$$u(s, a) = \mathrm{JSD}\{\mathcal{P}(\mathcal{S}|s, a, t) \mid t \sim \mathcal{P}(T)\}, \quad (5)$$

where JSD is the Jensen-Shannon Divergence, which captures the amount of disagreement present in a space of distributions. Hence, the *utility* of the state-action pair $u(s, a)$ is the disagreement, in terms of JSD, among the next-state distributions given s and a of all possible transition functions weighted by their probability. Since u depends only on the prior $\mathcal{P}(T)$, the novelty of potential transitions can be calculated without having to actually effect them in the external MDP.

The "internal" exploration MDP can then be defined as $(\mathcal{S}, \mathcal{A}, \bar{t}, u, \delta(s_\tau))$, where the sets $\mathcal{S}$ and $\mathcal{A}$ are identical to those of the external MDP, and the transition function $\bar{t}$ is defined such that,

$$p(s'|s, a, \bar{t}) := \mathop{\mathbb{E}}_{t \sim \mathcal{P}(T)} \left[p(s'|s, a, t) \right], \quad (6)$$

which can be interpreted as a different sample of $\mathcal{P}(T)$ drawn at each state transition. u is to the exploration MDP what the r is to the external MDP, so that maximizing u results in the optimal exploration policy at that iteration, just as maximizing r results in the optimal policy for the corresponding task. Finally, the initial state distribution density is set to the Dirac delta function $\delta(s_\tau)$ such that the initial state of exploration MDP is always the current state s_τ in the environment. It is important to understand that the prior $\mathcal{P}(T)$ is used twice in the exploration MDP:

1. To specify the state-action joint distribution as per Equation 3. Each member t in the prior $\mathcal{P}(T)$ determines a distribution $\mathcal{P}(\mathcal{S}, \mathcal{A}|\pi, t)$ over the set of possible state-action pairs that can result by sequentially executing actions according to π starting in s_τ.

2. To obtain the utility for a particular transition as per Equation 5. Each state-action pair (s, a) in the above $\mathcal{P}(\mathcal{S}, \mathcal{A}|\pi, t)$, according to the transition functions from $\mathcal{P}(T)$, forms a set of predicted next-state distributions $\{\mathcal{P}(\mathcal{S}|s, a, t) \mid t \sim \mathcal{P}(T)\}$. The JSD of this set is $u(s, a)$.

2.3. Bootstrap Ensemble Approximation

The prior $\mathcal{P}(T)$ can be approximated using a bootstrap ensemble of N learned transition functions or models that are each trained independently using different subsets of the *history* D, consisting of state transitions experienced by the agent while exploring the external MDP (Efron, 2012). Therefore, while $\mathcal{P}(T)$ is uniform when the agent is initialized, thereafter it is conditioned on agent's history D so that the general form of the prior is $\mathcal{P}(T|D)$. 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 probability densities than a random model. Hence, even a relatively small ensemble can approximate the true distribution, $\mathcal{P}(T|D)$ (Lakshminarayanan et al., 2017). Recent work suggests that it is possible to do so even in high-dimensional state and action spaces (Kurutach et al., 2018; Chua et al., 2018). Using an N-model ensemble $\{t_1, t_2, \cdots, t_N\}$ approximating the prior and assuming that all models fit the data equally well, the dynamics of the exploration MDP can be approximated by randomly selecting one of the N models with equal probability at each transition.

Therefor, to approximate $u(s, a)$, the JSD in Equation 5 can be expanded as (see Appendix B):

$$\begin{aligned} u(s,a) &= \text{JSD}\{\mathcal{P}(\mathcal{S}|s,a,t) \mid t \sim \mathcal{P}(T)\} \\ &= H\left(\mathbb{E}_{t\sim\mathcal{P}(T)}\left[\mathcal{P}(\mathcal{S}|s,a,t)\right]\right) \\ &\quad - \mathbb{E}_{t\sim\mathcal{P}(T)}\left[H(\mathcal{P}(\mathcal{S}|s,a,t))\right], \end{aligned} \tag{7}$$

where $H(\cdot)$ denotes the entropy of the distribution. Equation 7 can be summarized as the difference between the entropy of the average and the average entropy and can be approximated by averaging samples from the ensemble:

$$\begin{aligned} u(s,a) \simeq H\left(\frac{1}{N}\sum_{i=1}^{N}\mathcal{P}(\mathcal{S}|s,a,t_i)\right) \\ - \frac{1}{N}\sum_{i=1}^{N} H(\mathcal{P}(\mathcal{S}|s,a,t_i)). \end{aligned} \tag{8}$$

2.4. Large Continuous State Spaces

For continuous state spaces, $\mathcal{S} \subset \mathbb{R}^d$, the next-state distribution $\mathcal{P}(\mathcal{S}|s, a, \tilde{t}_i)$ is generally parameterized, typically, using a multivariate Gaussian $\mathcal{N}_i(\mu_i, \Sigma_i)$ with mean vector μ_i and co-variance matrix Σ_i. With this, evaluating Equation 8 is intractable as it involves estimating the entropy of a mixture of Gaussians, which has no analytical solution. This problem can be circumvented by replacing the Shannon entropy with Rényi entropy (Rényi, 1961) and using the corresponding Jensen-Rényi Divergence (JRD).

The Rényi entropy of a random variable X is defined as

$$H_\alpha(X) = \frac{1}{1-\alpha}\ln\int p(x)^\alpha dx$$

for a given order $\alpha \geq 0$, of which Shannon entropy is a special case when α tends to 1. When $\alpha = 2$, the resulting quadratic Rényi entropy has a closed-form solution for a mixture of N Gaussians (Wang et al., 2009). Therefore, the Jensen-Rényi Divergence $\text{JRD}\{\mathcal{N}_i(\mu_i, \Sigma_i) \mid i = 1, \ldots, N\}$ is given by

$$H_2\left(\sum_i^N \frac{1}{N}\mathcal{N}_i\right) - \frac{1}{N}\sum_i^N H_2\left(\mathcal{N}_i\right),$$

can be calculated with

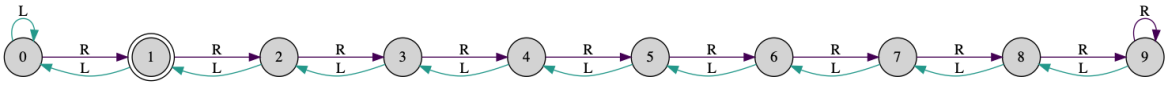
$$-\ln\left[\frac{1}{N^2}\sum_{i,j}^{N}\mathfrak{D}(\mathcal{N}_i,\mathcal{N}_j)\right] - \frac{1}{N}\sum_i^N\frac{\ln|\Sigma_i|}{2} - c, \tag{9}$$

where $c = d\ln(2)/2$ and

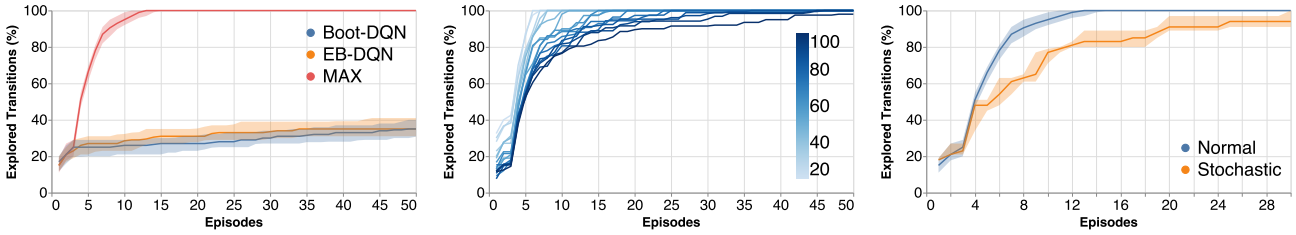
$$\mathfrak{D}(\mathcal{N}_i,\mathcal{N}_j) = \frac{1}{|\Omega|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}\Delta^T\Omega^{-1}\Delta\right), \tag{10}$$

with $\Omega = \Sigma_i + \Sigma_j$ and $\Delta = \mu_j - \mu_i$.

Equation 9 measures divergence among predictions of models based on some combination of their means and variances. However, when the models are learned, the parameters concentrate around their true values at different rates and the environments can greatly differ in the amount of noise they contain. On the one hand, if the environment is completely deterministic, exploration effort could be wasted in precisely matching the small predicted variances of all the models. On



(a) Chain environment of length 10.



(b) 50-state chain. (c) Chain lengths (d) Stochastic trap

Figure 1. **Algorithm performance on the randomized Chain environment**. For the first 3 episodes, marked by the vertical dotted line, actions were chosen at random (as warm-up). Each line corresponds to the median of 100 runs (seeds) in (b) and 5 runs in (c) and (d). The shaded area spans the 25th and 75th percentiles.

Algorithm 1 MODEL-BASED ACTIVE EXPLORATION

Initialize: Transitions dataset D, with random policy
Initialize: Model ensemble, $\tilde{T} = \{t_1, t_2, \cdots, t_N\}$
repeat
  while episode not complete **do**
    ExplorationMDP $\leftarrow (\mathcal{S}, \mathcal{A}, \text{Uniform}\{\tilde{T}\}, u, \delta(s_\tau))$
    $\pi \leftarrow$ SOLVE(ExplorationMDP)
    $a_\tau \sim \pi(s_\tau)$
    act in environment: $s_{\tau+1} \sim \mathcal{P}(\mathcal{S}|s_\tau, a_\tau, t^*)$
    $D \leftarrow D \cup \{(s_\tau, a_\tau, s_{\tau+1})\}$
    Train t_i on D for each t_i **in** $\tilde{T}$
  end while
until computation budget exhausted

the other hand, ignoring the variance in a noisy environment could result in poor exploration. To inject such prior knowledge into the system, an optional temperature parameter $\lambda \in [0, 1]$ that modulates the sensitivity of Equation 9 with respect to the variances was introduced. Since the outputs of parametric non-linear models, such as neural networks, are unbounded, it is common to use variance bounds for model and numerical stability. Using the upper bound Σ_U the variances can be re-scaled with λ:

$$\hat{\Sigma}_i = \Sigma_U - \lambda(\Sigma_U - \Sigma_i) \quad \forall\, i = 1, \ldots, N.$$

In this paper, λ was fixed to 0.1 for all continuous environments.

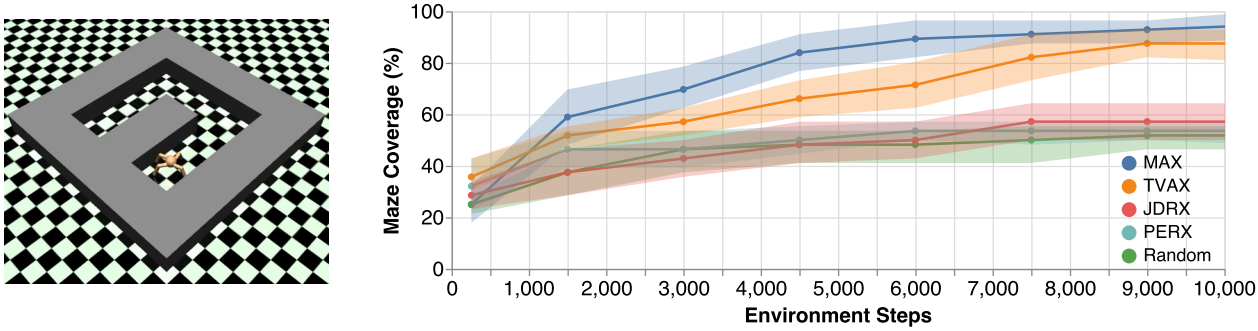
2.5. The MAX Algorithm

Algorithm 1 presents MAX in high-level pseudo-code. MAX is, essentially, a model-based RL algorithm with exploration as its objective. At each step, a fresh exploration policy is learned to maximise its return in the exploration MDP, a procedure which is generically specified as SOLVE(ExplorationMDP). The policy then acts in the external MDP to collect new data, which is used to train the ensemble yielding the approximate posterior. This posterior is then used as the approximate prior in the subsequent exploration step. Note that a transition function is drawn from $\tilde{T}$ for each transition in the exploration MDP. In practice, training the model ensemble and optimizing the policy can be performed at a fixed frequency to reduce the computational cost.

3. Experiments

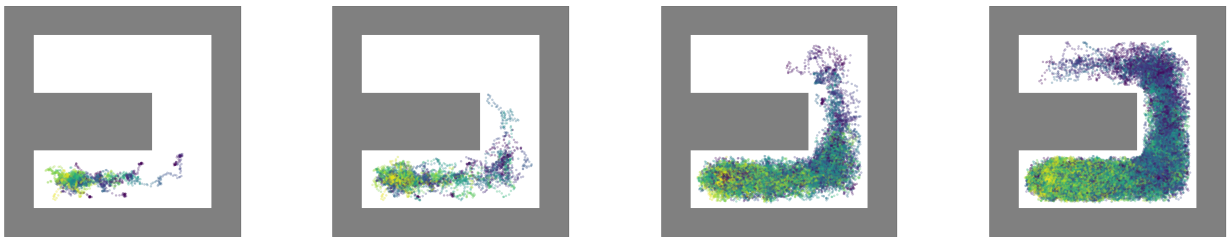
3.1. Discrete Environment

A randomized version of the Chain environment (Figure 1a), designed to be hard to explore proposed by Osband et al. (2016), which is a simplified generalization of RiverSwim (Strehl & Littman, 2005) was used to evaluate MAX. Starting in the second state (state 1) of an L-state chain, the agent can move either left or right at each step. An episode lasts $L + 9$ steps, after which the agent is reset to the start. The agent is first given 3 warm up episodes during which 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 (e.g. not solvable by the policy `always-go-right`), the effect of each action was randomly swapped so that in approximately half of the states, going RIGHT results in a leftward transition and vice-versa. Unless stated otherwise, $L = 50$ was used, so that exploring the environment fully and reaching the far right states is unlikely using random exploration. The probability of the latter decreases exponen-



(a) Ant Maze Environment

(b) Maze Exploration Performance



(c) 300 steps

(d) 600 steps

(e) 3600 steps

(f) 12000 steps

Figure 2. **Performance of MAX exploration on the Ant Maze task**. (a) shows the environment used. Results presented in (b) show that active methods (MAX and TVAX) are significantly quicker in exploring the maze compared to reactive methods (JDRX and PREX) with MAX being the quickest. (c)-(f) visualize the maze exploration by MAX across 8 runs. Chronological order of positions within an episode is encoded with the color spectrum, going from yellow (earlier in the episode) to blue (later in the episode).

tially with L. Therefore, in order to explore efficiently, an agent needs to exploit the structure of the environment.

MAX was compared to two exploration methods based on the *optimism in face of uncertainty* principle (Kaelbling et al., 1996): Exploration Bonus DQN (EB-DQN; Bellemare et al., 2016) and Bootstrapped DQN (Boot-DQN; Osband et al., 2016). Both algorithms employ 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. Note that these baselines are fundamentally "any-time-optimal" RL algorithms which minimize cumulative regret by trading-off exploration and exploitation in each action.

For the chain environment, MAX used Monte-Carlo Tree Search to find open-loop exploration policies (see Appendix C for details). The hyper-parameters for both of the baseline methods were tuned with grid search.

Figure 1b 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. Figure 1c shows the exploration progress curves for MAX when chain length was varied from 20 to 100 in intervals of 5.

To see if MAX can 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 (see Appendix C). Although MAX slowed down as a consequence, it still managed to explore the transitions as Figure 1d shows.

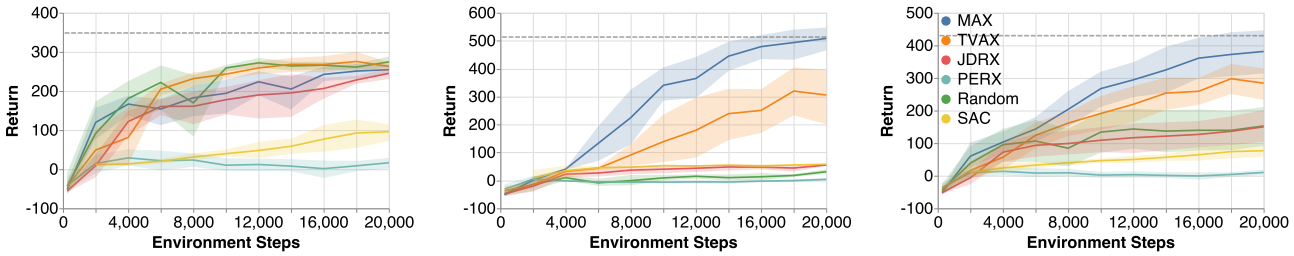
3.2. Continuous Environments

To evaluate MAX in the high-dimensional continuous setting, two environments based on MuJoCo (Todorov et al., 2012), Ant Maze and Half Cheetah, were considered. The exploration performance was measured directly for Ant Maze, and indirectly in the case of Half Cheetah.

MAX was compared to four other exploration methods that lack at least one feature of MAX:

1. **Trajectory Variance Active Exploration (TVAX):** an active exploration method that defines transition utilities as per-timestep variance in sampled trajectories in contrast to the per-state JSD between next state-predictions used in MAX.

2. **Jensen-Rényi Divergence Reactive Exploration**



(a) Running task performance (b) Flipping task performance (c) Average performance

Figure 3. **MAX on Half Cheetah tasks**. The grey dashed horizontal line shows the average performance of an oracle model-free policy trained for $200k$ (10x more) steps by SAC in the environment, directly using the corresponding task-specific reward function. Notice that exploring the dynamics for the flipping task is more difficult than the running task as evidenced by the performance of the random baseline. Overall, active methods are quicker and better explorers compared to the reactive ones in this task. Each curve is the mean of 8 runs.

(JDRX): a reactive counter-part of MAX, which learns the exploration policy directly from the experience collected so far without planning in the exploration MDP.

3. **Prediction Error Reactive Exploration (PERX):** a commonly used reactive exploration method (e.g. in Pathak et al. (2017)), which uses the mean prediction error of the ensemble as transition utility.

4. **Random** exploration policy.

In the Ant Maze (see Figure 2a), exploration performance was measured directly as the fraction of the U-shaped maze that the agent visited during exploration. In Half Cheetah, exploration performance was evaluated by measuring the usefulness of the learned model ensemble when exploiting it to perform two downstream tasks: running and flipping. For both environments, a Gaussian noise of $\mathcal{N}(0, 0.02)$ was added to the states to introduce stochasticity in the dynamics. Appendix D details the setup.

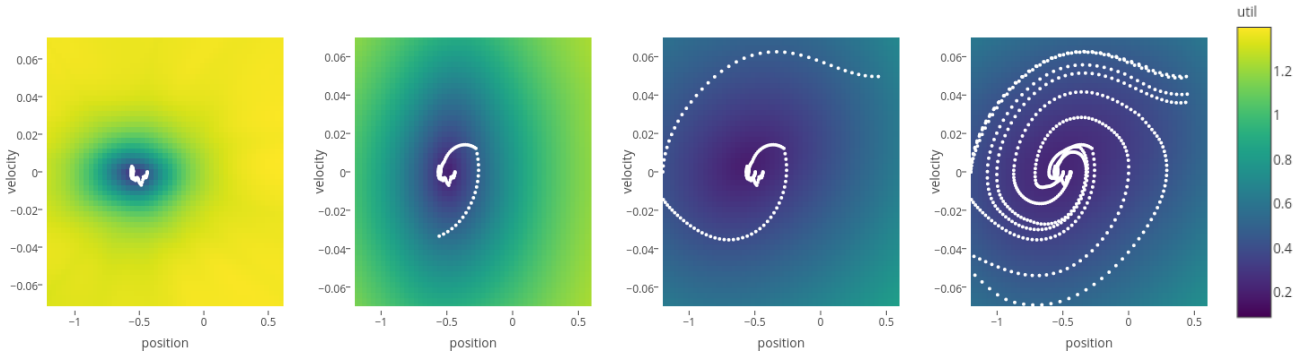
Models were probabilistic Deep Neural Networks trained with negative log-likelihood loss to predict the next state distributions in the form of multivariate Gaussians with diagonal covariance matrices. Soft-Actor Critic (SAC; Haarnoja et al., 2018) was used to learn both pure exploration and task-specific policies. The maximum entropy framework used in SAC is particularly well-suited to model-based RL as it uses an objective the both improves policy robustness, which hinders adversarial model exploitation, and yields multi-modal policies, which could mitigate the negative effects of planning with inaccurate models.

Exploration policies were regularly trained with SAC from scratch with the utilities re-calculated using the latest models to avoid over-commitment. The first phase of training involved only a fixed dataset containing the transitions experienced by the agent so-far (agent history D). For the active methods (MAX and TVAX), this was followed by an additional phase where the policies were updated by the data generated *exclusively* from the "imaginary" exploration MDP, which is the key feature distinguishing active from reactive exploration.

The results for Ant Maze and Half Cheetah are presented in Figures 2 and 3, respectively. For Half Cheetah, an additional baseline, obtained by training an agent with model-free SAC using the task-specific reward in the environment, is included. In both cases, the active exploration methods (MAX and TVAX) outperform the reactive ones (JDRX and PERX). Due to the noisy dynamics, PERX performs poorly. Among the two active methods, MAX is noticeably better as it uses a principled trajectory sampling and utility evaluation technique as opposed to the TVAX baseline which cannot distinguish the risk from the uncertainty. It is important to notice that the running task is easy since random exploration is sufficient which is not the case for flipping where good performance requires directed active exploration.

None of the methods were able to learn task-oriented planning in Ant Maze, even with larger models and longer training times than were used to obtain the reported results. The Ant Maze environment is more complex than Half Cheetah and obtaining good performance in downstream tasks using only the learned models is difficult due to other confounding factors such as compounding errors that arise in long-horizon planning. Hence, exploration performance was measured simply as the fraction of the maze the that agent explored. The inferior results of the baseline methods suggest that this task is non-trivial.

The evolution of the uncertainty landscape over the state-space of the environment when MAX is employed is visualized in Figure 4 for the Continuous Mountain Car environment. In the first exploration episode, the agent takes a spiral path through the state space (Figure 4c): MAX was able to drive the car up and down the sides of valley to develop enough velocity to reach the mountain top *without any* external reward. In subsequent episodes, it carves out more spiral paths in-between the previous ones (Figure 4d).



(a) 100 steps (b) 150 steps (c) 220 steps (d) 800 steps

Figure 4. **Illustration of MAX exploration in the Continuous Mountain Car environment**. Each plot shows the state space of the agent, discretized as a 2D grid. The color indicates the average uncertainty of a state over all actions. The dotted lines represent the trajectories of the agent.

4. Discussion

An agent is *meta-stable* if it is sub-optimal and, at the same time, has a policy that prevents it from gaining experience necessary to improve itself (Watkins, 1989). Simply put, 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) such as adding random noise to the actions of the policy might be sufficient to break out of meta-stability. If the environment is ergodic, then reactive 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.

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 & Rasmussen, 2011): in certain regions of the state space, the models could deviate significantly from the external MDP. Model-bias can have many causes such as improper generalization 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. If learning certain aspects of the environment is difficult, it will manifest itself as disagreement in the ensemble. MAX would collect more data about those aspects to improve the quality of models, thereby limiting adversarial exploitation by the policy. Since model-based RL does not have an inherent mechanism to explore, MAX could be considered as an important addition to the model-based RL framework rather than merely being an application of it.

Limitations. The derivation in Section 2 makes the assumption that the utility of a policy is the average utility of the probable transitions 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. This second-order effect was not considered in the 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 different uncertain states. MAX is also less computationally efficient in comparison to the baselines used in the paper as it trades off computational efficiency for data efficiency as is common in model-based algorithms.

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 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 the 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 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 RL. Optimal Bayesian experimental design (Chaloner & 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) proposed 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 the context of Active Learning, McCallum & Nigam (1998) proposed using Jensen-Shannon Divergence between predictions of a committee of classifiers to identify the most useful sample to be labelled next among a pool of unlabelled samples. (Singh et al., 2005) developed an intrinsic motivation framework inspired by neuroscience using prediction errors. (Itti & Baldi, 2009) presented the surprise formulation used in Equation 1 and demonstrated a strong correlation between 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). Schmidhuber (2009) suggested a general form of learning progress as compression progress which can be used as an extra intrinsic reward for curious RL systems.

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 & Stone (2012) and shown to outperform prediction error-based and other intrinsic motivation methods. In contrast to MAX, planning uses the mean prediction of a model ensemble to optimize a disagreement-based utility measure which is augmented with an additional state-distance bonus. Still & Precup (2012) derived a exploration and exploitation tradeoff in an attempt to maximize the predictive power of the agent. Mohamed & Rezende (2015) combined 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) presented a Bayesian approach to evaluate the value of experience taking a reactive approach. However, they also used Bayesian Neural Networks to maintain a belief over environment dynamics, and the information gain to bias the policy search with an intrinsic reward component. Variational Inference was used to approximate the prior-posterior KL divergence. Bellemare et al. (2016) derived a notion of pseudo-count for estimating state visitation frequency in high-dimensional spaces. They then transformed 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) used inverse models to avoid learning anything that the agent cannot control to reduce risk, and prediction error in the latent space to perform reactive exploration. A large-scale study of curiosity-driven exploration (Burda et al., 2019) found that curiosity is correlated with the actual objectives of many environments, and reported that using random features mitigates some of the non-stationarity implicit in methods based on curiosity. Eysenbach et al. (2018) demonstrated the power of optimizing policy diversity in the absence of a reward function in developing skills which could then be exploited.

Model-based RL has long been touted as the cure for sample inefficiency problems of modern RL (Schmidhuber, 1990; Sutton, 1991; Deisenroth & 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, this work was inspired by Chua et al. (2018) who combined probabilistic models with novel trajectory sampling techniques using particles to obtain better approximations of the returns in the environment. Concurrent with this work, Pathak et al. (2019) also showed the advantages of using an ensemble of models for exploring complex high-dimensional environments, including a real robot.

6. Conclusion

This paper introduced MAX, a model-based RL algorithm for pure exploration. It can distinguish between learnable and unlearnable unknowns and search for policies that actively seek learnable unknowns in the environment. MAX 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 actual interaction with the environment. Experiments in hard-to-explore discrete and high-dimensional continuous environments indicate that MAX is a powerful generic exploration method.

Acknowledgements

We would like to thank Jürgen Schmidhuber, Jan Koutník, Garrett Andersen, Christian Osendorfer, Timon Willi, Bas Steunebrink, Simone Pozzoli, Nihat Engin Toklu, Rupesh Kumar Srivastava and Mirek Strupl for their assistance and everyone at NNAISENSE for being part of a conducive research environment.

References

Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. Unifying Count-Based Exploration and Intrinsic Motivation. In *Advances in Neural Information Processing Systems*, pp. 1471–1479, 2016.

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

Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., and Efros, A. A. Large-Scale Study of Curiosity-Driven Learning. In *International Conference on Learning Representations*, 2019.

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

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

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

Deisenroth, M. and Rasmussen, C. E. PILCO: A Model-Based and Data-Efficient Approach to Policy Search. In *Proceedings of the 28th International Conference on Machine Learning*, pp. 465–472, 2011.

Efron, B. Bayesian Inference and the Parametric Bootstrap. *Annals of Applied Statistics*, 6(4):1971–1997, 2012. doi: 10.1214/12-AOAS571.

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is All You Need: Learning Skills without a Reward Function. *arXiv preprint arXiv:1802.06070*, 2018.

Fedorov, V. *Theory of Optimal Experiments Designs*. Academic Press, 01 1972.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *International Conference on Machine Learning (ICML)*, 2018.

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

Houthooft, R., Chen, X., Duan, Y., Schulman, J., De Turck, F., and Abbeel, P. VIME: Variational Information Maximizing Exploration. In *Advances in Neural Information Processing Systems*, pp. 1109–1117, 2016.

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

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

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

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

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

McCallum, A. K. and Nigam, K. Employing EM and Pool-Based Active Learning for Text Classification. In *Proceedings International Conference on Machine Learning (ICML)*, pp. 359–367. Citeseer, 1998.

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. Human-Level Control Through Deep Reinforcement Learning. *Nature*, 518 (7540):529, 2015.

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

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. Deep Exploration via Bootstrapped DQN. In *Advances in Neural Information Processing Systems*, pp. 4026–4034, 2016.

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

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

Pathak, D., Gandhi, D., and Gupta, A. Self-Supervised Exploration via Disagreement. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Rényi, A. On Measures of Entropy and Information. Technical report, Hungarian Academy Of Sciences, Budapest, Hungary, 1961.

Schmidhuber, J. 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, November 1990.

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

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

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

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

Schmidhuber, J. 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. (eds.), *Anticipatory Behavior in Adaptive Learning Systems. From Psychological Theories to Artificial Cognitive Systems*, volume 5499 of *LNCS*, pp. 48–76. Springer, 2009.

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

Singh, S. P., Barto, A. G., and Chentanez, N. Intrinsically Motivated Reinforcement Learning. In *Advances in Neural Information Processing Systems*, pp. 1281–1288, 2005.

Snoek, J., Larochelle, H., and Adams, R. P. Practical Bayesian Optimization Of Machine Learning Algorithms. In *Advances in Neural Information Processing Systems*, pp. 2951–2959, 2012.

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

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

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

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

Sutton, R. S. Reinforcement Learning Architectures for Animats. In *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior*, pp. 288–296, 1991.

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

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A Physics Engine for Model-Based Control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 5026–5033. IEEE, 2012.

Wang, F., Syeda-Mahmood, T., Vemuri, B. C., Beymer, D., and Rangarajan, A. Closed-form Jensen-Renyi Divergence for Mixture of Gaussians and Applications to Group-wise Shape Registration. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 648–655. Springer, 2009.

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

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

A. Policy Evaluation

Let $\mathrm{IG}(\pi)$ be the information gain of a policy π:

$$\begin{aligned}\mathrm{IG}(\pi) &= \mathbb{E}_{\phi\sim\mathcal{P}(\Phi|\pi)}\left[\mathrm{IG}(\phi)\right], \\ &= \int_{\Phi} \mathrm{IG}(\phi) p(\phi|\pi)\, d\phi, \end{aligned} \tag{11}$$

where $p(\phi|\pi) = p(s, a, s'|\pi)$ is the probability of transition (s, a, s') occurring given π is the behavioural policy in the external MDP.

(11) can be expanded as

$$\begin{aligned}\mathrm{IG}(\pi) &= \int_{\mathcal{S}}\int_{\mathcal{A}}\int_{\mathcal{S}} \mathrm{IG}(s, a, s') p(s, a, s'|\pi)\, ds'\, da\, ds \\ &= \int_{\mathcal{S}}\int_{\mathcal{A}}\int_{\mathcal{S}} \mathrm{IG}(s, a, s') p(s'|s, a) p(s, a|\pi)\, ds'\, da\, ds \\ &= \int_{\mathcal{S}}\int_{\mathcal{A}} u(s, a) p(s, a|\pi)\, da\, ds, \\ &= \int_{T}\int_{\mathcal{S}}\int_{\mathcal{A}} u(s, a) p(s, a|\pi, t) p(t)\, da\, ds\, dt \end{aligned} \tag{12}$$

where $u(s, a)$ is action utility function

$$u(s, a) = \int_{\mathcal{S}} \mathrm{IG}(s, a, s') p(s'|s, a)\, ds', \tag{13}$$

quantifing the net utility of taking action a from state s.

Equation 12 reduces then to the following expectation:

$$\mathrm{IG}(\pi) = \mathbb{E}_{t\sim\mathcal{P}(T)}\left[\mathbb{E}_{s,a\sim\mathcal{P}(\mathcal{S},\mathcal{A}|\pi,t)}\left[u(s, a)\right]\right] \tag{14}$$

B. Action Evaluation

To obtain a closed form expression for $u(s, a)$, Equation 13 can be expanded using Equation 1 to:

$$\begin{aligned}u(s, a) &= \int_{\mathcal{S}} \mathrm{IG}(s, a, s') p(s'|a, s)\, ds', \\ &= \int_{\mathcal{S}} D_{\mathrm{KL}}(\mathcal{P}(T|\phi) \parallel \mathcal{P}(T))\, p(s'|s, a)\, ds' \end{aligned}$$

Expanding from definition of KL-divergence:

$$D_{\mathrm{KL}}(\mathcal{P}_1 \parallel \mathcal{P}_2) = \int_{Z} p_1(z) \log\left(\frac{p_1(z)}{p_2(z)}\right) dz$$

we get

$$u(s, a) = \int_{\mathcal{S}}\int_{T} p(t|\phi) \log\left(\frac{p(t|\phi)}{p(t)}\right) p(s'|s, a)\, dt\, ds' \tag{15}$$

Using the Bayes rule

$$p(t|s', a, s) = \frac{p(s'|s, a, t) p(t|s, a)}{p(s'|s, a)} \tag{16}$$

and the observation that $p(t|s,a) = p(t)$ (Sun et al., 2011) leads to:

$$\begin{aligned} u(s,a) &= \int_{\mathcal{S}} \int_{T} \frac{p(s'|s,a,t)p(t)}{p(s'|s,a)} \log \left(\frac{p(s'|s,a,t)}{p(s'|s,a)} \right) p(s'|s,a)\, dt\, ds' \\ &= \int_{\mathcal{S}} \int_{T} p(s'|s,a,t)p(t) \log \left(\frac{p(s'|s,a,t)}{p(s'|s,a,\bar{t})} \right) dt\, ds'. \end{aligned} \tag{17}$$

Expanding and swapping integrals in the former term:

$$\begin{aligned} u(s,a) = &\int_{T} \int_{\mathcal{S}} p(s'|s,a,t) \log \left(p(s'|s,a,t)\right) p(t)\, ds'\, dt \\ &- \int_{\mathcal{S}} \int_{T} p(s'|s,a,t)p(t)\, dt \log \left(\int_{T} p(s'|s,a,t)p(t)\, dt \right) ds' \end{aligned} \tag{18}$$

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

$$u(s,a) = \mathfrak{H} \left(\int_{T} \mathcal{P}(\mathcal{S}|s,a,t)p(t)\, dt \right) - \int_{T} \mathfrak{H}(\mathcal{P}(\mathcal{S}|s,a,t))p(t)\, dt \tag{19}$$

$$= \mathfrak{H} \left(\mathbb{E}_{t \sim \mathcal{P}(T)} \left[\mathcal{P}(\mathcal{S}|s,a,t) \right] \right) - \mathbb{E}_{t \sim \mathcal{P}(T)} \left[\mathfrak{H}(\mathcal{P}(\mathcal{S}|s,a,t)) \right] \tag{20}$$

$\mathcal{P}(\mathcal{S}|s,a,t)$ 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:

$$\text{JSD}\{\mathcal{P}_{\bar{Z}} \mid \mathcal{P}_{\bar{Z}} \sim \mathcal{P}(\mathbf{Z})\} = \mathfrak{H} \left(\int_{\bar{Z}} \mathcal{P}_{\bar{Z}} p(\mathcal{P}_{\bar{Z}})\, d\bar{Z} \right) - \int_{\bar{Z}} \mathfrak{H}(\mathcal{P}_{\bar{Z}}) p(\mathcal{P}_{\bar{Z}})\, d\bar{Z} \tag{21}$$

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 13 can be expressed as:

$$u(s,a) = \text{JSD}\{\mathcal{P}(\mathcal{S}|s,a,t) \mid t \sim \mathcal{P}(T)\} \tag{22}$$

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

C. Chain Environment

C.1. Stochastic Environment

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 it compared to a far-off unexplored state. Figure 1d 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 were trained on different outputs for the same input. Hence the resulting output distribution of the models were 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.

C.2. MAX

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 enough to foster the diversity for the out-of-sample predictions and hence bootstrap sampling was not used. After 3-episode warm-up with random actions, the models were trained for 150 iterations. Searching for an exploration policy in the exploration MDP, which was an open-loop plan, was performed using 25 rounds of Monte Carlo Tree Search (MCTS), which used 5 random trajectories for each step of 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 $N = 50$.

Table 1. Hyper-Parameters for MAX. *Grid Size: 0.8k*

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

C.3. Baselines

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 can be computed using prediction errors of forward models (Pathak et al., 2017). However, for our simple environment, the bonuses were provided by an oracle, implemented as a transition look-up table. This is equivalent to having an ideal model that can learn about a transition in one-shot, therefore emulating the best-case scenario for the method.

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 or *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 some action a' in state s or 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.

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

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

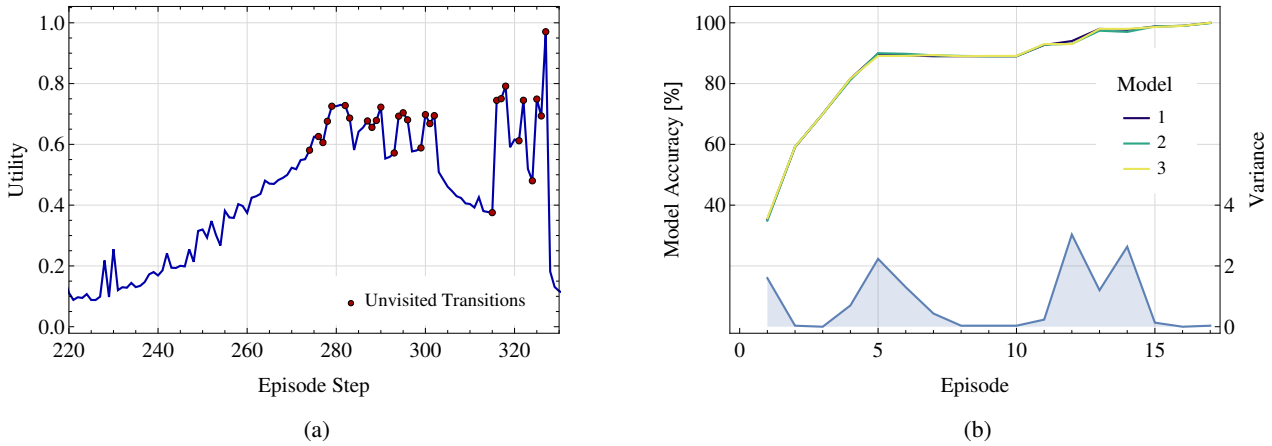
Commonly used ϵ-greedy exploration was turned off in both baselines, thereby making exploration reliant solely on the methods themselves. The Q-value functions were implemented with multi-layer fully connected neural networks. The state was encoded with thermometer encoding for Bootstrapped DQN as proposed 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, 10 Q-value network heads were trained *only after* each episode, with the exact number of iterations being tuned. 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.

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

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.

C.4. Supplementary Figures

Figures 5a, 5b and 6 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 were therefore indistinguishable from the environment. Notice also that the learning curves of the three models are 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.



(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.

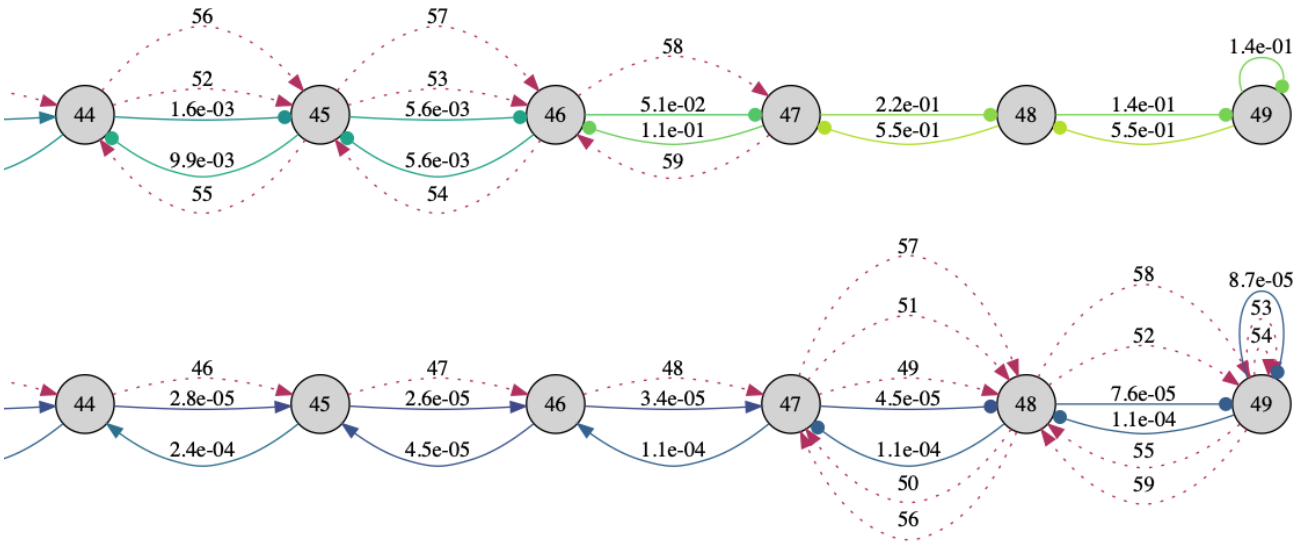


Figure 6. Instance of uncertainty minimization:MAX behaviour in the second-to-last (top) and the last (bottom) episodes. The solid arrows (environment transitions) and are annotated with the utilities of predicted information gains. The arrowheads are circular for the unvisited transitions and regular 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).

Robustness to Key Hyper-Parameters Number of models in the ensemble, trajectories per an MCTS iteration, and MCTS iterations were varied and 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.

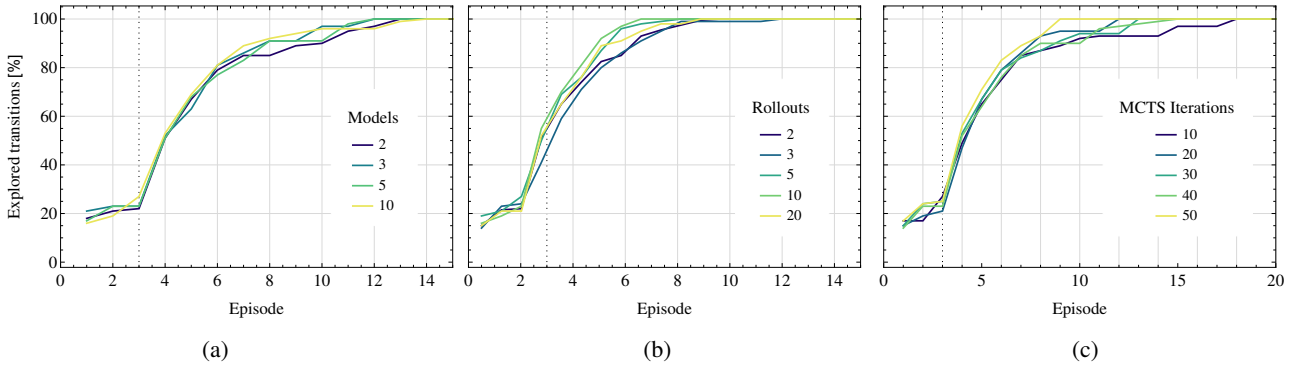


Figure 7. Algorithm Properties. Each learning curve is a median of 5 runs with different seeds. Vertical dotted line marks the ends of the warm-up phase. Sub-figures show how the percentage of explored transitions varies with respect to (a) ensemble size, (b) number of rollouts and (c) planning iterations.

D. Continuous Environments

D.1. Numerically Stable Jensen-Rényi Divergence Implementation

$$\mathfrak{D}(\mathcal{N}_i, \mathcal{N}_j) = \frac{1}{|\Sigma_i + \Sigma_j|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\left(\mu_j - \mu_i\right)^T \left(\Sigma_i + \Sigma_j\right)^{-1} \left(\mu_j - \mu_i\right)\right)$$

$$\begin{aligned} p &= \left(u_j - u_i\right)^T \left(\Sigma_i + \Sigma_j\right)^{-1} \left(\mu_j - \mu_i\right) \\ q &= \ln(|\Sigma_i + \Sigma_j|) \\ &= \operatorname{trace}(\ln(\Sigma_i + \Sigma_j)) \end{aligned}$$

$$\mathfrak{D}(\mathcal{N}_i, \mathcal{N}_j) = \exp\left(-\frac{1}{2}(p + q)\right)$$

Additionally, LOG-SUM-EXP trick is used to calculate the entropy of the mean.

D.2. Experimental Setup

256 steps of warm-up was used for all methods to train the models before exploration began. During exploration, for all methods, models were retrained from scratch for 50 epochs every 25 steps. SAC policies were also relearned from scratch every 25 steps to avoid over-commitment. The current exploration experience was transferred to SAC as its off-policy data and 100 steps of SAC off-policy training was done. For active methods, there was additional on-policy training done using data generated during 50 episodes, each of length 50, starting from current state using 128 actors and the model ensemble. The SAC entropy weight α was tuned separately for each method. It was found to be 0.02 for Rényi entropy based methods MAX and JDRX, and 0.2 for the TVAX and PERX.

Exploitation in Half Cheetah was done every 2000 steps of exploration. Using the exploration data, the model ensemble was trained for 200 epochs. For all methods, SAC policies were trained using both off-policy data from exploration and with on-policy data from the model ensemble. On-policy data was generated using 250 episodes of length 100 using 128 actors. Note that this is recursive prediction for 100 steps starting from the start state. As with exploration, the next states were sampled at each transition by randomly selecting a model from the ensemble and then sampling the next state from its output distribution. A policy was trained and evaluated 3 times and the average performance was reported.

Models were trained to predict the state deltas, instead of raw next states as is common. The states, actions and deltas were all normalized to have zero mean and unit variance. For all methods, the utility computation was done in normalized scales for stability.

Table 4. Hyper-Parameters for Models in Continuous Environments

Hyper-parameter	Value
Ensemble Size	32
Hidden Layers	4
Hidden Layer Size	512
Batch Size	256
Non-linearity	Swish
Learning Rate	0.001

Half Cheetah Reward Functions
Running: $r_t = v_t^x - 0.1\|a_t\|_2^2$; flipping: $r_t = \omega_t^y - 0.1\|a_t\|_2^2$,

where v_t^x is the velocity along the x axis at time t, and ω_t^y is the angular velocity around axis y at time time t.