REINFORCEMENT LEARNING WITH BAYESIAN CLASSIFIERS: EFFICIENT SKILL LEARNING FROM OUTCOME EXAMPLES

Anonymous authors
Paper under double-blind review

ABSTRACT

Exploration in reinforcement learning is, in general, an intractable problem. In this work, we study how a more tractable class of reinforcement learning problems can be defined with data that provides examples of successful outcome states. In this case, the reward function can be obtained automatically by training a classifier to classify states as successful or not. We argue that, with appropriate representation and regularization, such a classifier can guide a reinforcement learning algorithm to an effective solution. However, as we will show, this requires the classifier to make uncertainty-aware predictions that are very difficult with standard deep networks. To address this, we propose a novel mechanism for obtaining calibrated uncertainty based on an amortized technique for computing the normalized maximum likelihood distribution. We show that the resulting algorithm has a number of intriguing connections to both count-based exploration methods and prior algorithms for learning reward functions from data, while being able to guide algorithms towards the specified goal more effectively. We show how using amortized normalized maximum likelihood for reward inference is able to provide effective reward guidance for solving a number of challenging navigation and robotic manipulation tasks which prove difficult for other algorithms.

1 INTRODUCTION

While reinforcement learning (RL) has been shown to successfully solve certain types of problems with careful reward design (OpenAI et al., 2019; Rajeswaran et al., 2018), RL in its most general form, with no prior knowledge of the transitions dynamics or the reward function, is intractable. Since the most general RL problem experiences rewards only upon visiting certain states, the uninformed search problem for these states becomes increasingly difficult. Techniques which explicitly provide "reward-shaping" (Ng et al., 1999), or modify the reward function to guide learning, can help take some of the burden off of exploration, but shaped rewards can be difficult to obtain without domain knowledge.

In this paper, we aim to reformulate the reinforcement learning problem to make it easier for the user to specify the task and to provide a tractable reinforcement learning objective. Instead of requiring a reward function designed for an objective, our method instead assumes a user-provided set of *successful outcome examples*: states in which the desired task has been accomplished successfully. The algorithm aims to estimate the distribution over these states and maximize the probability of reaching states that are likely under the distribution. Prior work on learning from success examples (Fu et al., 2018b; Zhu et al., 2020) focused primarily on alleviating the need for manual reward design. In our work, we focus on the potential for this mode of task specification to produce more tractable RL problems and solve more challenging classes of tasks. Intuitively, when provided with examples of successful states, the RL algorithm should be able to direct its exploration, rather than simply hoping to randomly chance upon potentially high reward states.

The main challenge in instantiating this idea into a practical algorithm is to perform appropriate uncertainty quantification in estimating whether a given state corresponds to a successful outcome. Our approach trains a classifier to distinguish successful states, provided by the user, from those generated by the current policy, analogously to generative adversarial networks (Goodfellow et al.,

2014) and previously proposed methods for inverse reinforcement learning (Fu et al., 2018a). In general, such a classifier is not guaranteed to provide a good optimization landscape for learning the policy. We discuss how a particular form of uncertainty quantification based on the normalized maximum likelihood (NML) distribution produces better reward guidance for learning. We also connect our approach to count-based exploration methods, showing that a classifier with suitable uncertainty estimates reduces to a count-based exploration method in the absence of any generalization across states, while showing it improves over count-based exploration in the presence of good generalization. We then propose a practical algorithm to train success classifiers in a computationally efficient way with NML, and show how this form of reward inference allows us to solve difficult problems more efficiently, providing experimental results on a number of navigation and robotic manipulation domains.

2 Related Work

A number of techniques have been proposed to improve exploration.These techniques either add reward bonuses that encourage a policy to visit novel states in a task-agnostic manner (Wiering and Schmidhuber, 1998; Auer et al., 2002; Schaul et al., 2011; Houthooft et al., 2016; Pathak et al., 2017; Tang et al., 2017; Stadie et al., 2015; Bellemare et al., 2016a; Burda et al., 2018a; O'Donoghue, 2018) or perform Thompson sampling or approximate Thompson sampling based on a prior over value functions (Strens, 2000; Osband et al., 2013; 2016). While these techniques are uninformed about the actual task, we consider a constrained set of problems where examples of successes can allow for more task directed exploration. In real world problems, designing well-shaped reward functions makes exploration easier but often requires significant domain knowledge (Andrychowicz et al., 2020), access to privileged information about the environment (Levine et al., 2016) and/or a human in the loop providing rewards (Knox and Stone, 2009; Singh et al., 2019b). Prior work has considered specifying rewards by providing example demonstrations, and inferring rewards with inverse RL (Abbeel and Ng, 2004; Ziebart et al., 2008; Ho and Ermon, 2016; Fu et al., 2018a). This requires expensive expert demonstrations to be provided to the agent. In contrast our work has the minimal requirement of simply requiring successful outcome states, which can be provided cheaply. The subclass of problems is also related to goal conditioned RL (Kaelbling, 1993; Schaul et al., 2015; Zhu et al., 2017; Andrychowicz et al., 2017; Nair et al., 2018; Veeriah et al., 2018; Rauber et al., 2018; Warde-Farley et al., 2018; Colas et al., 2019; Ghosh et al., 2019; Pong et al., 2020) but is more general, since it allows for a more abstract notion of task success.

Bayesian inference with expressive models and high dimensional data can often be intractable, requiring assumptions on the form of the posterior (Hoffman et al., 2013; Blundell et al., 2015; Maddox et al., 2019). In this work, we build on the concept of normalized maximum likelihood (Rissanen, 1996; Shtar'kov, 1987) to learn Bayesian classifiers. Normalized maximum likelihood is typically considered from the perspective of optimal coding (Grünwald, 2007; Fogel and Feder, 2018). Here, we show how the normalized maximum likelihood distribution can be used to learn success classifiers, and discuss its connections to exploration and reward shaping.

3 Preliminaries

In this paper, we study a modified reinforcement learning problem, where instead of the standard reward function, the agent is instead provided with *successful outcome examples*. This reformulation not only provides a modality for task specification that may be more natural for users to provide in some settings (Fu et al., 2018b; Zhu et al., 2020; Singh et al., 2019a) but, as we will show, can also make learning easier. We also derive a meta-learned variant of the conditional normalized maximum likelihood (CNML) distribution for representing our reward function. We discuss background on successful outcome examples and CNML in this section.

3.1 Reinforcement Learning with Examples of Successful Outcomes

Formally, we follow the framework proposed by Fu et al. (2018b) and assume that we are provided with a Markov decision process (MDP) *without* a reward function, given by $\mathcal{M}$, where $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mu_0)$, as well as successful outcome examples $\mathcal{S}_+ = \{s_+^k\}_{k=1}^K$, which is a set of states in which the desired task has been accomplished. This formalism is easiest to describe in terms of the

control as inference framework (Levine, 2018), (Figure 7). This graphical model consists of states and actions, as well as binary *success* variables e_t which represent the occurrence of a particular event. The agent's goal is to cause this event to occur (e.g., a robot that is cleaning the floor must cause the "floor is clean" event to occur). Formally, we assume that the states in $\mathcal{S}_+$ are sampled from the distribution $p(s_t|e_t = \text{True})$ – that is, states where the desired event has taken place. In this work, we focus on efficient methods for solving this reformulation of the RL problem, by utilizing a novel uncertainty quantification method to represent the distribution $p(e_t|s_t)$.

In practice, prior methods that build on this and similar reformulations of the RL problem (Fu et al., 2018b) derive an algorithm where the reward function in RL is produced by a classifier that estimates $p(e_t = \text{True}|s_t)$. Following the adversarial inverse reinforcement learning (AIRL) derivation (Fu et al., 2018a; Finn et al., 2016), it is possible to show that the correct source of *negative* examples for training this classifier is the state distribution of the policy itself, $\pi(s)$. This insight results in a simple algorithm: at each iteration of the algorithm, the policy is updated to maximize the current reward, given by $\log p(e_t = \text{True}|s_t)$, then samples from the policy are added to the set of negative examples $\mathcal{S}_-$, and the classifier is retrained on the original positive set $\mathcal{S}_+$ and the updated negative set $\mathcal{S}_-$.

3.2 Conditional Normalized Maximum Likelihood

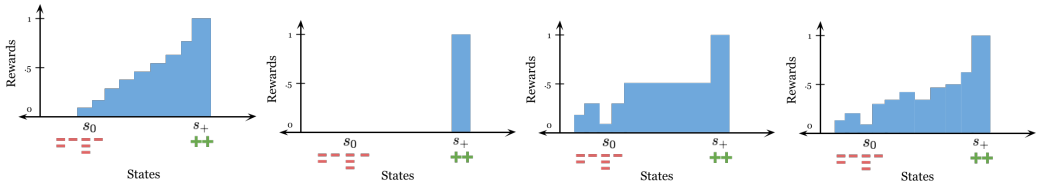
Our method builds on the principle of conditional normalized maximum likelihood (NML) (Rissanen and Roos, 2007; Grünwald, 2007; Fogel and Feder, 2018), which we review briefly. CNML is a method for performing k-way classification, given a model class Θ and a dataset $\mathcal{D} = \{(x_0, y_0), (x_1, y_1), ..., (x_n, y_n)\}$, and has been shown to provide better calibrated predictions and uncertainty estimates with minimax regret guarantees (Bibas et al., 2019). To predict the class of a query point x_q, CNML constructs k augmented datasets by adding x_q with a different label in each datasets, which we write as $\mathcal{D} \cup (x_q, y = i), i \in (1, 2, ..., k)$. CNML then defines the class distribution by solving the maximum likelihood estimation problem at query time for each of these augmented datasets to convergence, and normalize the likelihoods as follows:

$$p_{\text{CNML}}(y = i|x_q) = \frac{p_{\theta_i}(y = i|x_q)}{\sum_{j=1}^{k} p_{\theta_j}(y = j|x_q)}, \qquad \theta_i = \arg\max_{\theta \in \Theta} \mathbb{E}_{(x,y)\sim\mathcal{D}\cup(x_q,y=i)}[\log p_\theta(y|x)] \quad (1)$$

Intuitively, if x_q is close to other datapoints in $\mathcal{D}$, then the model will struggle to assign a high likelihood to labels that differ substantially from other nearby points. However, if x_q is far from all datapoints in $\mathcal{D}$, then the different augmented MLE problems can easily classify x_q as an arbitrary class, providing us with a likelihood closer to uniform. More formally, CNML is an application of the NML principle for obtaining minimum description length models, and we refer readers to Grünwald (2007) for an in-depth discussion. A major limitations of CNML is that it requires training an entire neural network to convergence on every augmented dataset every time we want to evaluate a test point's class probabilities. We will address this issue in Section 5.

4 Bayesian Success Classifiers for Reward Inference

Ideally, training a classifier with the policy samples as negative examples as described in Section 3.1 should yield a smooth decision boundary between the well-separated negative and positive examples.For example, Figure 1 depicts a simple 1-D example, where the agent starts at the left (s_0) and the positive outcomes are at the right (s_+) side of the environment. Since the positives are on the right and the negatives are on the left, one might expect a classifier to gradually increase its prediction of a success as we move to the right (Figure 1b), which would provide a dense reward signal for the policy to move to the right. However, this idealized scenario rarely happens in practice. Without suitable regularization, the decision boundary between the positive and negative examples may not be smooth. In fact, the decision boundary of an optimal classifier may take on the form of a sharp boundary *anywhere* between the positive and negative examples in the early stages of training (Figure 1b). As a result, the classifier might provide little to no reward signal for the policy, since it can assign arbitrarily small probabilities to the states sampled from the policy. We note that this issue is not pathological: our experiments in Section 6 show that this poor reward signal issue happens in practice and can greatly hinder learning. In this section, we will discuss how an appropriate classifier training method can avoid these uninformative rewards.



(a) Ideal rewards (b) MLE (c) Tabular CNML (d) CNML + generalization

Figure 1: An idealized illustration of a well-shaped reward and the solution that various classifier training might provide. Red bars represent visited states near the initial state s_0 and green pluses represent the example success examples s_+. (a) The ideal reward would provide learning signal to encourage the policy to move from the start states s_0 to the successful states s_+. (b) When training a success classifier with MLE, the classifier may output zero (or arbitrary) probabilities when evaluated at new states. (c) Tabular CNML will give a prior probability of 0.5 on new states. (d) When using function approximation, CNML with generalization will provide a degree of shaping towards the goal.

4.1 Regularized Success Classifiers via Normalized Maximum Likelihood

To avoid arbitrary likelihoods on novel states, we need to impose a prior on our classifier so that it provides a more informative reward when evaluated at novel states. A more informative reward function is one that assigns higher rewards to the *fringe* of the states visited by the policy, because this reward function will encourage the policy to explore and move towards the desired states. We can construct such a reward function by imposing a prior that the output of our classifier that novel states have a non-neglibile chance of being a success state. To construct such a classifier, we train a Bayesian classifier using conditional normalized maximum likelihood (CNML) (Shtar'kov, 1987), as we described in Section 3.

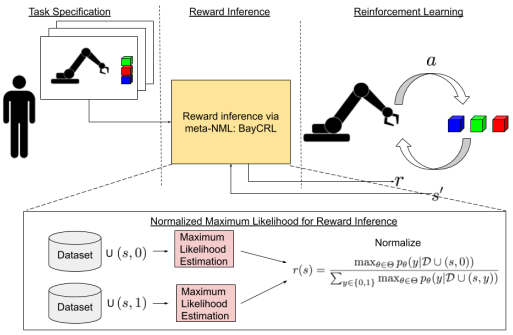


Figure 2: BayCRL: Illustration of how Bayesian classifiers learned with normalized maximum likelihood can be used to provide informative learning signal during learning. The human user provides examples of successful outcomes, which are then used in combination with on policy samples to iterate between training a classifier with NML and training RL with this reward.

To use normalized maximum likelihood classifiers for reward inference, the procedure is similar to the one described in Section 3. We construct a dataset using the provided successful outcomes as positives and the on-policy samples as negatives. However, for the classification we perform the CNML procedure described in Equation 1 to obtain the rewards $r(s) = p_{\text{CNML}}(e = 1|s)$. So how does this affect reward assignment during learning? To illustrate the impact of using CNML, we can visualize the rewards with a NML-regularized classifier on the problem described earlier. When the success classifier is evaluated with CNML instead of standard maximum likelihood, we see in Figure 1c that intermediate states provided non-zero rewards rather than simply having vanishing likelihoods like in Figure 1b. Interestingly, from Fig 1c, Fig 1d we see that the rewards obtained via NML directly incentivize exploration.

4.2 Relationship to Count-Based Exploration

In this section we formally relate the success likelihoods obtained via CNML to commonly used exploration methods based on counts. Formally, we prove that the success classifier trained with CNML is equivalent to a version of count-based exploration in the absence of any generalization across states (i.e., a fully tabular setting).

Theorem 4.1. *Suppose we are estimating success probabilities $p(e = 1|s)$ in the tabular setting, where we have a separate parameter independently for each state. Let $N(s)$ denote the number of times state s has been visited by the policy, and let $G(s)$ be the number of occurrences of state s in the successful outcomes. Then the CNML probability $p_{\text{CNML}}(e = 1|s)$ is equal to $\frac{G(s)+1}{N(s)+G(s)+2}$. For states that are never observed to be successful, we then recover inverse counts $\frac{1}{N(s)+2}$.*

Refer to Appendix Appendix A.4 for a full proof. However, in practice, the learned classifier attains non-negligible generalization. So long as this generalization is correlated with the dynamics of the environment, we expect this generalization to improve shaping, as we discuss next.

4.3 Reward Shaping with Bayesian Success Classifiers

While the analysis above suggests that a CNML classifier would give us something akin to an exploration bonus, the structure of the problem and the state space can actually provides us more information to guide us towards the goal. In most environments (Brockman et al., 2016; Yu et al., 2019) the state space does not consist of independent and uncorrelated categorical variables, but is provided in a representation that relates at least roughly to the dynamics structure in the environment. For instance, states close to the goal dynamically are also typically close to the goal in the metric space defined by the states. Indeed, this observation is the basis of many commonly used heuristic reward shaping methods, such as rewards given by Euclidean distance to target states.

In this case, the task specification can actually provide more information than simply performing uninformed count-based exploration. Since the uncertainty aware classifier described in Section 4.1 is built on top of features that are correlated with environment dynamics, it is able to incentivize task-aware directed exploration. A depiction of this phenomenon is illustrated in Fig 1d. As compared to CNML without generalization in Fig 1c, we expect the intermediate rewards to provide more shaping towards the goal. This is evident from our empirical results in Section 6 where BayCRL is able to significantly outperform methods for task agnostic exploration.

5 BayCRL: Training Bayesian Success Classifiers for Outcome Driven RL via Meta-Learning and CNML

In Section 4, we discussed how Bayesian success classifiers can incentivize exploration and provide reward shaping to guide RL. However, the reward inference technique via CNML described in Section 4.1 is computationally intractable, as it requires optimizing maximum likelihood estimation problems to convergence on every data point we want to query. In this section, we describe an approximation that allows us to instantiate this method in practice.

5.1 Meta-Learning for CNML

We adopt ideas from meta-learning to amortize the cost of obtaining the CNML distribution. As noted in Section 4.1, the computation of the CNML distribution involves repeatedly solving maximum likelihood problems that share a significant amount of common structure. We should use this commonality between problems to quickly obtain CNML estimates without redoing the slow training process repeatedly. One set of techniques that are directly applicable are ideas from meta-learning for few shot classification. Meta-learning uses a distribution of training problems to explicitly learn models that can quickly adapt to new problems.

To apply meta-learning to the CNML problem, we can formulate each of the maximum likelihood problems described in Equation 1 as a separate task for meta-learning, and apply any standard meta-learning technique to obtain a model capable of few-shot adaptation to the MLE problems required for CNML. While any meta-learning algorithm is applicable, we found model agnostic meta-learning (MAML) to be an effective choice of algorithm. In short, MAML tries to learn a model that can quickly adapt to new tasks via a few stepds of gradient descent. We refer readers to Finn et al. (2017) for a detailed overview.

This procedure is illustrated in Fig 8, and can be described as follows: given a dataset $\mathcal{D} = \{(x_0, y_0), (x_1, y_1), ..., (x_n, y_n)\}$, $2n$ different tasks τ_i can be constructed, each corresponding to performing maximum likelihood estimation on the dataset $\max_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}\cup(x_i,y=0)} \log p(y|x,\theta)$ or $\max_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}\cup(x_i,y=0)} \log p(y|x,\theta)$. Given these constructed tasks $\mathcal{S}(\tau)$, meta-training can be performed by performing the bi-level optimization described in Finn et al. (2017):

$$\max_\theta \ \mathbb{E}_{\tau\sim\mathcal{S}(\tau)}[\mathcal{L}(\tau,\theta')], \quad s.t \ \ \theta' = \theta - \alpha\nabla_\theta\mathcal{L}(\tau,\theta). \tag{2}$$

This training procedure gives us parameters θ that can then be quickly adapted to provide the CNML distribution simply by performing a step of gradient descent. The model can be queried for the CNML distribution by starting from θ and taking one step of gradient descent for the query point augmented dataset, each with a different potential label. These likelihoods are then normalized to provide the CNML distribution as follows:

$$p_{\text{meta-NML}}(y|x;\mathcal{D}) = \frac{p_{\theta_y}(y|x)}{\sum_{y\in\mathcal{Y}} p_{\theta_y}(y|x)}, \qquad \theta_y = \theta - \alpha\nabla_\theta \mathbb{E}_{(x_i,y_i)\sim\mathcal{D}\cup(x,y)}[\mathcal{L}(x_i,y_i,\theta)].$$

This algorithm, which we call *meta-NML*, allows us to obtain normalized likelihood estimates without having to retrain maximum likelihood to convergence at every single query point, since the model can now solve maximum likelihood problems of this form very quickly. A complete detailed description and pseudocode of this algorithm are provided in Appendix A.2.

5.2 Applying Meta-NML to Success Classification

We apply the meta-NML algorithm described above to learning Bayesian success classifiers for providing rewards for reinforcement learning, in our proposed algorithm, which we term BayCRL- Bayesian classifiers for reinforcement learning. Similarly to Fu et al. (2018b), we can train our Bayesian classifier by first constructing a dataset $\mathcal{D}$ for binary classification. This is done by using the provided examples of successful outcomes as positives, and on-policy examples collected by the policy as negatives. Given this dataset, the Bayesian classifier parameters $\theta_{\text{meta-NML}}$ can be trained via meta-NML as described in Equation 2. The classifier can then be used to directly and quickly assign rewards to a state s according to it's probabilities $r(s) = p_{\text{meta-NML}}(e=1|s)$ (via a step of gradient descent, as described in Equation 3), and perform standard reinforcement learning.

Algorithm 1 BayCRL: Bayesian Classifiers for RL

1: User provides success examples $\mathcal{S}_+$
2: Initialize dataset $\mathcal{D}$ with $\mathcal{S}_+$ as positives with label 1.
3: Initialize policy π randomly.
4: **for** iteration $i = 1, 2, ...$ **do**
5: Collect on-policy examples $\mathcal{S}_-$ from π.
6: Add on-policy examples $\mathcal{S}_-$ to the $\mathcal{D}$ as negatives with label 0.
7: **if** iteration $i\%$ k == 0 **then**
8: Meta-train via meta-NML using Equation 2
9: Assign rewards to states via Equation 5
10: Train π with RL algorithm

$$p_{\text{meta-NML}}(e=1|s) = \frac{p_{\theta_1}(e=1|s)}{\sum_{i\in\{0,1\}} p_{\theta_i}(e=i|s)} \tag{3}$$

$$\theta_0 = \theta_{\text{meta-NML}} - \alpha\nabla_\theta \mathbb{E}_{(s_j,e_j)\sim\mathcal{D}\cup(s,e=0)}[\mathcal{L}(e_j,s_j,\theta)] \tag{4}$$

$$\theta_1 = \theta_{\text{meta-NML}} - \alpha\nabla_\theta \mathbb{E}_{(s_j,e_j)\sim\mathcal{D}\cup(s,e=1)}[\mathcal{L}(e_j,s_j,\theta)] \tag{5}$$

An overview of this algorithm is provided in Algorithm 1.

6 Experimental Evaluation

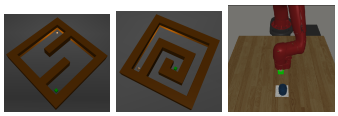
In our experimental evaluation we aim to answer the following questions: (1) Do the learning dynamics of prior classifier-based reward learning methods provide informative rewards for RL? (2) Does using BayCRL help address the exploration challenge when solving RL problems specified by successful outcomes? (3) Does using BayCRL help provide better reward shaping than simply performing naïvely uninformed exploration? To evaluate these questions, we evaluate our proposed algorithm BayCRL with the following setup.

6.1 Experimental Setup

We start off by understanding the algorithm behavior by evaluating it on maze navigation problems. Then, to evaluate our method in more complex domains, we consider three robotic manipulation tasks that were considered in Fu et al. (2018b) with a Sawyer robot arm: door opening, tabletop object pushing, and 3D object picking tasks. As we show in our results, exploration in these environments is

challenging and using naively chosen reward shaping does not solve the problem at hand. The tasks are specified by providing a set of 150 successful outcome examples.

We compare with a number of prior algorithms and ablations. To provide a comparison with a standard previous method based on success classifiers trained with an IRL-based adversarial method, we include the VICE algorithm Fu et al. (2018b). We also include a comparison with a recently proposed technique for learning dynamical distances Hartikainen et al. (2019). We additionally include comparisons to algorithms for uninformed exploration to show that BayCRL does a more directed form of exploration and reward shaping. To provide an apples-to-apples comparison, we use the same VICE method for training classifiers, but combine it with novelty-based exploration based on random network distillation (Burda et al., 2018b).



(a) Zigzag Maze (b) Spiral Maze (c) Sawyer Push

(d) Sawyer Pick (e) Sawyer Door

Figure 3: We evaluate on two maze environments and three robotics tasks: (a) the agent must navigate around an S-shaped corridor (b) the agent must navigate a spiral corridor, (c) the robot must push a puck to location, (d) the robot picks up a tennis ball from a table to a given location, (e) the robot opens the door to a specified angle.

6.2 Performance Comparisons with Prior Algorithms

We compare with prior algorithms across the domains described above, with the results shown in Fig 4. As we can see in Fig 4, BayCRL is able to very quickly learn how to solve these challenging exploration tasks, often reaching significantly better asymptotic performance than most prior methods, and doing so significantly more efficiently than methods like VICE (Fu et al., 2018b) or DDL (Hartikainen et al., 2019). This suggests that BayCRL is able to provide directed reward shaping and exploration to help RL escape local optima and achieve the specified task.

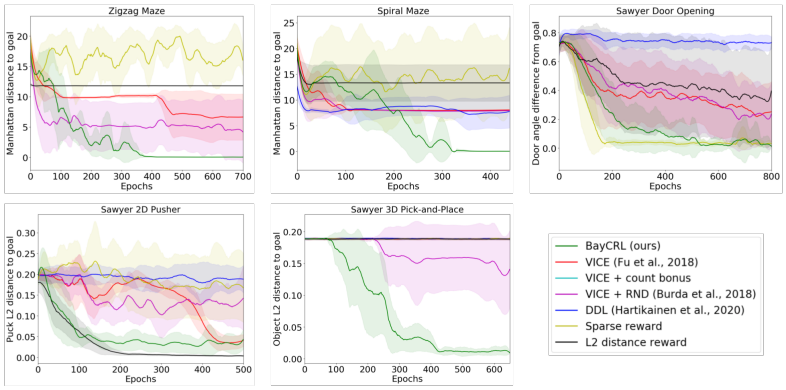


Figure 4: BayCRL performs better on a variety of environments when compared to other algorithms for goal reaching, standard success classifiers, or naively chosen uninformative reward functions (lower is better). Shading indicates a standard deviation across 5 seeds. We even see that we generally do better/comparable to using a naively shaped reward function.

To isolate whether the benefits purely come from exploration or are also resulting from task-aware reward shaping, we compare with methods that only perform uninformed, task-agnostic exploration. On the maze environments, where we can discretize the state space we can compute ground truth count based bonuses for exploration, while for the higher dimensional robotics tasks we must use a continuous space exploration method like RND (Burda et al., 2018b). From our comparisons with exploration algorithms such as count based exploration (Bellemare et al., 2016b), RND (Burda et al., 2018b), it is clear that BayCRL is able to significantly outperform methods that purely do exploration as well. This result in combination with our visualizations in Section 6.4 suggest that our proposed algorithm is providing useful task aware reward shaping, more effectively than state covering exploration like RND. We also compare the performance with learning progress using naively chosen, manually specified reward functions such as a sparse reward or the l_2 distance to goal as a reward. As shown in Fig 4, BayCRL generally outperforms these schemes in terms of sample

complexity and asymptotic performance. This indicates that the adaptive reward induced by BayCRL is able to solve more complex tasks than before, with simpler forms of task specification.

6.3 Ablations

We first consider the importance of performing meta-learning. If we simply compute NML by taking a few gradient steps from the MLE solution, without any meta-training, we see that the algorithm performance is significantly worse. Secondly, we analyze the importance of making the Bayesian classifier explicitly aware of the task being solved. To do so, we modify the training procedure so that the dataset $\mathcal{D}$ consists of only the on-policy negatives, and add the inferred reward from the Bayesian classifier to the reward obtained by a standard MLE classifier. We see that this performs poorly, showing that the classifier is doing more than just performing naive uninformed exploration. Tthe algorithm is slightly slower than regular MLE classifiers but only by a magnitude of 1.02.

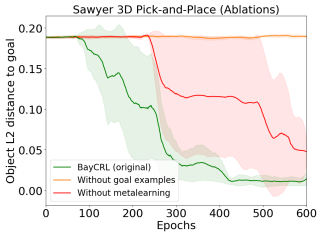
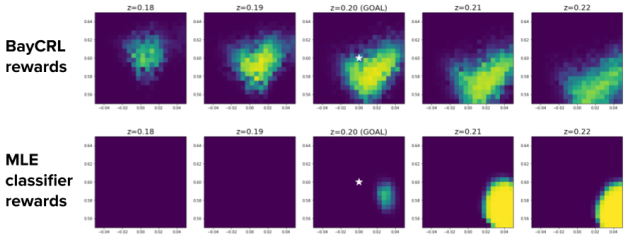


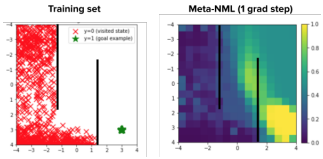
Figure 5: Ablative analysis of BayCRL. The amortization from meta-learning and access to goal examples are important components for performance.

6.4 Analysis of BayCRL

To better understand the source of empirical gains, we also run a number of analysis and visualization experiments on the reward functions inferred by BayCRL. Firstly, we visualize the shaped rewards on the Sawyer Pick task in Fig 6a. We can see that the rewards clearly correlate with distance to goal in the position of the object after the object is picked up, whereas the MLE classifier has a much more sharp and uninformative reward structure.



(a) Visualization of Reward Shaping for Picking. We see that BayCRL has shaped reward around the goal, guiding it to learn the task while MLE has uninformative sharp classifiers, making RL hard



(b) Visualization of meta-NML rewards on the Maze task. BayCRL is able to approximate a count bonus with just one gradient step, while also providing additional task-directed reward shaping.

Figure 6: Analysis of the learned reward function in terms of reward shaping and exploration

Next, to understand the connection between BayCRL and count based exploration, we visualize the reward landscape in an early iteration of the Zigzag Maze task with BayCRL in Fig 6b. We see here that the algorithm maintains a fringe of least visited states and rewards those states more, explicitly encouraging exploration. Furthermore, meta-NML is able to reasonably approximate the idealized NML rewards with just one gradient step, while also benefiting from additional shaping due to generalization of the classifier.

7 Discussion

In this work, we consider a sub-class of reinforcement learning problems where a task is specified by examples of successful outcomes. We discuss how solutions via standard success classifier suffer from shortcomings, and training Bayesian classifiers allows for better exploration to solve challenging problems. We discuss how the NML distribution can provide us a way to train such Bayesian classifiers, providing benefits of exploration and reward shaping. To make learning tractable, we propose a meta-learning approach to amortize the NML process.

While this work has shown the effectiveness of Bayesian classifiers for reward inference for tasks in simulation, it would be interesting to scale this solution to real world problems. Additionally, obtaining a theoretical understanding of how reward shaping interacts with learning dynamics would be illuminating in designing reward schemes.

REFERENCES

Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in neural information processing systems*, 2017.

Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 2002.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in neural information processing systems*, pages 1471–1479, 2016a.

Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Rémi Munos. Unifying count-based exploration and intrinsic motivation. *CoRR*, abs/1606.01868, 2016b. URL http://arxiv.org/abs/1606.01868.

Koby Bibas, Yaniv Fogel, and Meir Feder. Deep pnml: Predictive normalized maximum likelihood for deep neural networks. *CoRR*, abs/1904.12286, 2019. URL http://arxiv.org/abs/1904.12286.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622, 2015.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016. URL http://arxiv.org/abs/1606.01540.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2018a.

Yuri Burda, Harrison Edwards, Amos J. Storkey, and Oleg Klimov. Exploration by random network distillation. *CoRR*, abs/1810.12894, 2018b. URL http://arxiv.org/abs/1810.12894.

Cédric Colas, Pierre Fournier, Mohamed Chetouani, Olivier Sigaud, and Pierre-Yves Oudeyer. Curious: intrinsically motivated modular multi-goal reinforcement learning. In *International conference on machine learning*, pages 1331–1340, 2019.

C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*, 2017.

Chelsea Finn, Paul F. Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *CoRR*, abs/1611.03852, 2016. URL http://arxiv.org/abs/1611.03852.

Yaniv Fogel and Meir Feder. Universal batch learning with log-loss. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 21–25. IEEE, 2018.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adverserial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018a.

Justin Fu, Avi Singh, Dibya Ghosh, Larry Yang, and Sergey Levine. Variational inverse control with events: A general framework for data-driven reward definition. In *Advances in Neural Information Processing Systems*, pages 8547–8556, 2018b.

Dibya Ghosh, Abhishek Gupta, Justin Fu, Ashwin Reddy, Coline Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals without reinforcement learning. *arXiv preprint arXiv:1912.06088*, 2019.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial networks. *CoRR*, abs/1406.2661, 2014. URL http://arxiv.org/abs/1406.2661.

Peter D Grünwald. *The minimum description length principle*. MIT press, 2007.

Kristian Hartikainen, Xinyang Geng, Tuomas Haarnoja, and Sergey Levine. Dynamical distance learning for unsupervised and semi-supervised skill discovery. *CoRR*, abs/1907.08225, 2019. URL http://arxiv.org/abs/1907.08225.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in neural information processing systems*, pages 4565–4573, 2016.

Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pages 1109–1117, 2016.

Leslie Pack Kaelbling. Learning to achieve goals. In *IJCAI*, 1993.

W Bradley Knox and Peter Stone. Interactively shaping agents via human reinforcement: The tamer framework. In *Proceedings of the fifth international conference on Knowledge capture*, pages 9–16, 2009.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *CoRR*, abs/1805.00909, 2018. URL http://arxiv.org/abs/1805.00909.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *J. Mach. Learn. Res.*, 17:39:1–39:40, 2016. URL http://jmlr.org/papers/v17/15-522.html.

Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. In *Advances in Neural Information Processing Systems*, pages 13153–13164, 2019.

Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems*, pages 9191–9200, 2018.

Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In Ivan Bratko and Saso Dzeroski, editors, *Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999), Bled, Slovenia, June 27 - 30, 1999*, pages 278–287. Morgan Kaufmann, 1999.

Brendan O'Donoghue. Variational bayesian reinforcement learning with regret bounds. *arXiv preprint arXiv:1807.09647*, 2018.

OpenAI, :, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning, 2019.

Ian Osband, Daniel Russo, and Benjamin Van Roy. (more) efficient reinforcement learning via posterior sampling. In *Advances in Neural Information Processing Systems*, pages 3003–3011, 2013.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In *Advances in neural information processing systems*, 2016.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 16–17, 2017.

Vitchyr H Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2020.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations, 2018.

Paulo Rauber, Avinash Ummadisingu, Filipe Mutz, and Jürgen Schmidhuber. Hindsight policy gradients. In *International Conference on Learning Representations*, 2018.

Jorma Rissanen and Teemu Roos. Conditional NML universal models. In *2007 Information Theory and Applications Workshop*, pages 337–341, 2007.

Jorma J Rissanen. Fisher information and stochastic complexity. *IEEE transactions on information theory*, 42(1):40–47, 1996.

T. Schaul, Y. Sun, D. Wierstra, F. Gomez, and J. Schmidhuber. Curiosity-driven optimization. In *2011 IEEE Congress of Evolutionary Computation (CEC)*, 2011.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International conference on machine learning*, pages 1312–1320, 2015.

Yurii Mikhailovich Shtar'kov. Universal sequential coding of single messages. *Problemy Peredachi Informatsii*, 23(3):3–17, 1987.

Avi Singh, Larry Yang, Chelsea Finn, and Sergey Levine. End-to-end robotic reinforcement learning without reward engineering. In Antonio Bicchi, Hadas Kress-Gazit, and Seth Hutchinson, editors, *Robotics: Science and Systems XV, University of Freiburg, Freiburg im Breisgau, Germany, June 22-26, 2019*, 2019a. doi: 10.15607/RSS.2019.XV.073. URL https://doi.org/10.15607/RSS.2019.XV.073.

Avi Singh, Larry Yang, Kristian Hartikainen, Chelsea Finn, and Sergey Levine. End-to-end robotic reinforcement learning without reward engineering. In *Robotics: Science and Systems (RSS)*, 2019b.

Bradly C Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*, 2015.

Malcolm Strens. A bayesian framework for reinforcement learning. In *ICML*, 2000.

Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in neural information processing systems*, pages 2753–2762, 2017.

Vivek Veeriah, Junhyuk Oh, and Satinder Singh. Many-goals reinforcement learning. *arXiv preprint arXiv:1806.09605*, 2018.

David Warde-Farley, Tom Van de Wiele, Tejas Kulkarni, Catalin Ionescu, Steven Hansen, and Volodymyr Mnih. Unsupervised control through non-parametric discriminative rewards. In *International Conference on Learning Representations*, 2018.

Marco Wiering and Jürgen Schmidhuber. Efficient model-based exploration. In *Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior: From Animals to Animats*. MIT Press Cambridge, MA, 1998.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning, 2019.

Henry Zhu, Justin Yu, Abhishek Gupta, Dhruv Shah, Kristian Hartikainen, Avi Singh, Vikash Kumar, and Sergey Levine. The ingredients of real world robotic reinforcement learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=rJe2syrtvS.

Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

A APPENDIX

A.1 GRAPHICAL MODEL FOR CONTROL AS INFERENCE

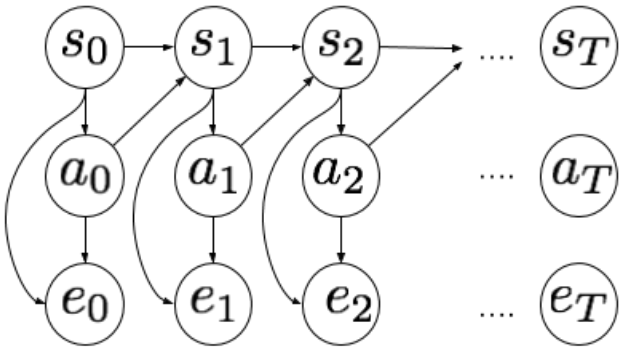


Figure 7: Graphical Model framework for Control as Inference. e_t correspond to auxiliary event variables correspond to accomplishing the task

A.2 DETAILED DESCRIPTION OF META-NML

We provide a detailed description of the meta-NML algorithm described in Section 5, and the details of the practical algorithm.

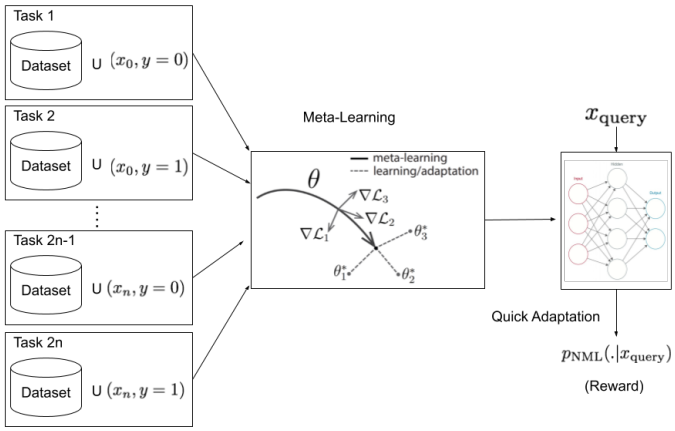


Figure 8: Figure illustrating the meta-training procedure for meta-NML (Maybe omit?)

Given a dataset $\mathcal{D} = \{(x_0, y_0), (x_1, y_1), .., (x_n, y_n)\}$, the meta-NML procedure proceeds by first constructing $k * n$ tasks from these data points, for a k shot classification problem. We will keep $k = 2$ for simplicity in this description. Each task τ_i is constructed by augmenting the dataset with a negative label $\mathcal{D} \cup (x_i, y = 0)$ or a positive label $\mathcal{D} \cup (x_i, y = 1)$. Now that each task consists of solving the maximum likelihood problem for it's augmented dataset, we can directly apply standard meta-learning algorithms to this setting. Building off the ideas in MAML (Finn et al., 2017), we can then train a set of model parameters θ such that after a single step of gradient descent it can quickly adapt to the optimal solution for the MLE problem on any of the augmented datasets. This is more formally written as

$$\max_{\theta} \ \mathbb{E}_{\tau \sim \mathcal{S}(\tau)}[\mathcal{L}(\tau, \theta')], \quad s.t \ \ \theta' = \theta - \alpha \nabla_{\theta} \mathcal{L}(\tau, \theta) \tag{6}$$

where $\mathcal{L}$ represents a standard classification loss function, α is the learning rate, and the distribution of tasks $p(\tau)$ is constructed as described above. For a new query point x, these initial parameters can then quickly be adapted to provide the CNML distribution by taking a gradient step on each augmented dataset to obtain the approximatley optimal MLE solution, and normalizing these as follows:

$$p_{\text{meta-NML}}(y|x; \mathcal{D}) = \frac{p_{\theta_y}(y|x)}{\sum_{y \in \mathcal{Y}} p_{\theta_y}(y|x)}, \qquad \theta_y = \theta - \alpha \nabla_{\theta} \mathbb{E}_{(x_i, y_i) \sim \mathcal{D} \cup (x, y)}[\mathcal{L}(x_i, y_i, \theta)]$$

This algorithm in principle can be optimized used any standard stochastic optimization method such as SGD, as described in Finn et al. (2017), backpropagating through the inner loop gradient update. For the specific problem setting that we consider, we have to employ some optimization tricks in order to enable learning:

A.2.1 Importance Weighting on Query Point

Since only one datapoint is augmented to the training set, it can get challenging for stochastic optimization to pay attention to this datapoint with increasing dataset sizes. In order to make sure the optimization considers the query point, we include the query point and proposed label (x_q, y) in every minibatch that is sampled, but downweight the loss computed on that point such that the overall objective remains unbiased. This is simply doing importance weighting, with the query point downweighted by a factor of $\frac{b}{N}$ where b is batchsize, N is the total number of points in the dataset.

A.2.2 Kernel Weighted Training Loss

The augmented dataset consists of points from the original dataset $\mathcal{D}$ and one augmented point (x_q, y). Given that we mostly care about having the proper likelihood on the query point, with an imperfect optimization process, the meta-training can yield solutions that are not very accurately representing true likelihoods on the query point. To counter this, we introduce a kernel weighting into the loss function in Equation [], Equation during meta-training and subsequently meta-testing. The kernel weighting modifies the training loss function as:

$$\max_{\theta} \ \mathbb{E}_{\tau \sim \mathcal{S}(\tau)}[\mathbb{E}_{(x,y) \sim \tau} \mathcal{K}(x, x_\tau) \mathcal{L}(x, y, \theta')], \quad s.t \ \ \theta' = \theta - \alpha \nabla_{\theta} \mathbb{E}_{(x,y) \sim \tau} \mathcal{K}(x, x_\tau) \mathcal{L}(x, y, \theta) \tag{7}$$

where x_τ is the query point for task τ and $\mathcal{K}$ is a choice of kernel. We typically choose Gaussian kernels centered around x_τ. Intuitively this allows the optimization to ensure that the gradient signal mainly considers the data-points that are copies of the query point in the dataset, or are similar to the query point and ensures that they have the correct likelihoods, instead of receiving interfering gradient signals from the many other points in the dataset.

A.2.3 Extra Gradient Steps for Meta-Testing

Given that we are using MAML for meta-training, we can continue training at meta-test time using standard gradient descent. Taking a few more gradient steps on the meta-test time MLE objective, allows the algorithm to be no worse than gradient descent in the worst case. When the meta-learner does not generalize as well, these extra gradient steps ensures that the estimated likelihoods are still well calibrated.

A.3 Hyperparameter and Implementation Details

We describe the hyperparmeter choices for our experiments here. We first list the general hyperparameters that were shared across runs, then provide a table of additional hyperparameters we tuned over for each domain and algorithm.

A.3.1 SAC

Below are the shared hyperparameters we used for Soft Actor-Critic:

Hyper-parameter	**Value**
Learning Rate	3×10^{-4}
Discount Factor γ	0.99
Policy Type	Gaussian
Policy Hidden Sizes	$(512, 512)$
Policy Hidden Activation	ReLU
RL Batch Size	1024
Reward Scaling	1
Replay Buffer Size	$500,000$
Q Hidden Sizes	$(512, 512)$
Q Hidden Activation	ReLU
Q Weight Decay	0
Q Learning Rate	3×10^{-4}
Target Network τ	5×10^{-3}

Table 1: Hyperparameters used for training with Soft Actor-Critic.

A.3.2 VICE

Below are the shared hyperparameters we used for VICE:

Hyper-parameter	**Value**
Classifier Learning Rate	1×10^{-4}
Classifier Batch Size	128
Classifier Optimizer	Adam
RL Algorithm	SAC

Table 2: Hyperparameters used for training with VICE.

A.3.3 RND

Below are the shared hyperparameters we used for RND:

Hyper-parameter	**Value**
Hidden Layer Sizes	$(256, 256)$
Output Units	512

Table 3: Hyperparameters used for training with RND.

A.3.4 Hyperparameter Tuning

Below are the hyperparameters we tuned over for each individual domain and algorithm.

Column 1	Column 2	Column 3
second row		
third row		

A.4 Proof of Theorem 1 connecting NML and inverse counts

We provide the proof of Theorem 1 here for completeness

Theorem A.1. *Suppose we are estimating success probabilities $p(e = 1|s)$ in the tabular setting, where we have a separate parameter independently for each state. Let $N(s)$ denote the number of times state s has been visited by the policy, and let $G(s)$ be the number of occurrences of state s in the successful outcomes. Then the CNML probability $p_{CNML}(e = 1|s)$ is equal to $\frac{G(s)+1}{N(s)+G(s)+2}$. For states that are never observed to be successful, we then recover inverse counts $\frac{1}{N(s)+2}$.*

Proof. In the fully tabular setting, our MLE estimates for $p(O|s)$ are simply given by finding the best parameter p_s for each state. The proof then proceeds by simple calculation.

For a state with $n = N(s)$ negative occurrences and $g = G(s)$ positive occurrences, the MLE estimate is simply given by $\frac{g}{n+g}$.

Now for evaluating CNML, we consider appending another instance for each class. The new parameter after appending a negative example is then $\frac{g}{n+g+1}$, which then assigns probability $\frac{n+1}{n+g+1}$ to the negative class. Similarly, after appending a positive example, the new parameter is $\frac{g+1}{n+g+1}$, so we try to assign probability $\frac{g+1}{n+g+1}$ to the positive class. Normalizing, we have

$$p_{\text{CNML}}(O = 1|s) = \frac{g+1}{n+g+2}. \tag{8}$$

When considering states that have only been visited on-policy, and are not included int the set of successful outcomes, then the likelihood reduces to

$$p_{\text{CNML}}(O = 1|s) = \frac{1}{n+2}. \tag{9}$$

□