Offline Meta Reinforcement Learning

Ron Dorfman  
Department of Electrical Engineering  
Technion  
rdorfman@campus.technion.ac.il

Aviv Tamar  
Department of Electrical Engineering  
Technion  
avivt@technion.ac.il

Abstract

Consider the following problem, which we term Offline Meta Reinforcement Learning (OMRL): given the complete training histories of $N$ conventional RL agents, trained on $N$ different tasks, design a learning agent that can quickly maximize reward in a new, unseen task from the same task distribution. In particular, while each conventional RL agent explored and exploited its own different task, the OMRL agent must identify regularities in the data that lead to effective exploration/exploitation in the unseen task. To solve OMRL, we take a Bayesian RL (BRL) view, and seek to learn a Bayes-optimal policy from the offline data. We extend the recently proposed VariBAD BRL algorithm to the off-policy setting, and demonstrate learning of Bayes-optimal exploration strategies from offline data using deep neural networks. Furthermore, when applied to the online meta-RL setting (agent simultaneously collects data and improves its meta-RL policy), our method is significantly more sample efficient than the conventional VariBAD.

1 Introduction

A central question in reinforcement learning (RL) is how to learn quickly (i.e., with few samples) in a new environment. Meta-RL addresses this issue by assuming a distribution over possible environments, and having access to a large set of environments from this distribution during training [6, 12]. Intuitively, the meta-RL agent can learn regularities in the environments, which allow quick learning in any environment that shares a similar structure. Indeed, recent work demonstrated this by training memory-based controllers that can ‘identify’ the domain [6, 30, 22], or by learning a parameter initialization that can lead to good performance with only a few gradient updates [12].

Another approach to quick reinforcement learning is Bayesian RL [15] (BRL). In BRL, the environment parameters are treated as unobserved variables, with a known prior distribution. Consequentially, the standard problem of maximizing expected returns (taken with respect to the posterior distribution) explicitly accounts for the environment uncertainty, and its solution is a Bayes-optimal policy, wherein actions optimally balance exploration and exploitation. Recently, Zintgraf et al. [39] showed that meta-RL is in fact an instance of BRL, where the meta-RL environment distribution is simply the BRL prior. Furthermore, a Bayes-optimal policy can be trained using standard policy gradient methods, simply by adding to the state the posterior belief over the environment parameters. The VariBAD algorithm [39] is an implementation of this approach that uses a variational autoencoder (VAE) for parameter estimation and deep neural network policies.

Most meta-RL studies, including VariBAD, have focused on the online setting, where, during training, the meta-RL policy is continually updated using data collected from running it in the training environments. In domains where data collection is expensive, such as robotics and healthcare to name a few, online training is a limiting factor. For standard RL, offline (a.k.a. batch) RL mitigates this problem by learning from data collected beforehand by an arbitrary policy [10, 25]. In this work we investigate the offline approach to meta-RL (OMRL). In OMRL, we assume that data has been collected by running standard RL agents on a set of environments from the environment distribution.

Preprint. Under review.
Figure 1: Illustration of offline meta-RL: the task is to navigate to a goal position that can be anywhere on the semi-circle. The reward is sparse (light-blue), and the offline data (left) contains training histories of conventional RL agents trained to find individual goals. The meta-RL agent (right) needs to find a policy that quickly finds the unknown goal, here, by searching across the semi-circle. Note that this search behavior is completely different than the behaviors seen in the data.

Importantly - we do not allow any modification to the RL algorithms used for data collection, and the meta-RL learner must make use of data that was not specifically collected for the meta-RL task. Nevertheless, we hypothesize that regularities between the training domains can still be learned, to provide faster learning in new environments. Figure 1 illustrates our problem: in this navigation task, each RL agent in the data learned to find its own goal, and converged to a behavior that quickly navigates toward it. The meta-RL agent, on the other hand, needs to learn a completely different behavior that effectively searches for the unknown goal position, by leveraging knowledge about the distribution of goal positions that can be gleaned from the data.

Our key idea to solving OMRL is an off-policy variant of the VariBAD algorithm, based on replacing the on-policy policy gradient optimization in VariBAD with an off-policy Q-learning based method. This, however, requires some care, as Q-learning applies to states of fully observed systems. We show that the VariBAD approach of augmenting states with the belief in the data applies to the off-policy setting as well, leading to an effective algorithm we term Off-Policy VariBAD. The offline setting, however, brings about another challenge – when the agent visits different parts of the state space in different environments, it becomes challenging to obtain an accurate belief estimate, a problem we term MDP ambiguity. We propose a simple solution, based on a reward relabelling trick that significantly improves the performance of the VariBAD VAE trained on offline data.

We show that our method can effectively solve OMRL on both discrete and continuous control problems with deep neural network policies. To our knowledge, this is the first demonstration of learning deep RL policies that are approximately Bayes-optimal in the offline setting. Furthermore, our method can also be applied in the online setting, and in this case we demonstrate significantly improved sample efficiency compared to conventional VariBAD, and learning more effective exploration policies than PEARL [30], a state-of-the-art off-policy meta-RL algorithm that is not Bayes-optimal.

2 Background

Our work leverages ideas from meta-RL, BRL and the VariBAD algorithm, as we now survey.

Meta-RL. In meta-RL [12], a distribution over tasks is assumed. A task $T_i$ is described by a Markov Decision Process (MDP) $M_i = (S, A, R_i, P_i)$, where the state space $S$ and the action space $A$ are shared across tasks, and $R_i$ and $P_i$ are task specific reward and transition functions. Thus, we write the task distribution as $p(R, P)$. For simplicity, we assume throughout that the initial state distribution $P_{init}(s_0)$ is the same for all MDPs. The goal in meta-RL is to train an agent that can quickly maximize reward in new, unseen tasks, drawn from $p(R, P)$. To do so, the agent must leverage any shared structure among tasks, which can typically be learned from a set of training tasks.
Bayesian Reinforcement Learning: The goal in BRL is to find the optimal policy $\pi$ in an MDP, when the MDP transitions and rewards are not known in advance. Similar to meta-RL, we assume a prior distribution over the MDP parameters $p(R, P)$, and seek to maximize the expected discounted return.\footnote{For ease of presentation, we consider the infinite horizon discounted return. Our formulation easily extends to the episodic and finite horizon settings.}

$$\mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right],$$

where the expectation is taken with respect to both the uncertainty in state-action transitions $s_{t+1} \sim P(\cdot | s_t, a_t)$, $a_t \sim \pi$, and the uncertainty in the MDP parameters $R, P \sim p(R, P)$. A key observation is that this formulation naturally accounts for the exploration/exploitation tradeoff – an optimal agent must plan its actions to reduce uncertainty in the MDP parameters, if such leads to higher rewards.

One way to approach the BRL problem is to model $R, P$ as unobserved state variables in a partially observed MDP (POMDP), reducing the problem to solving a particular POMDP instance where the unobserved variables cannot change in time ($\mathcal{P}$ and $\mathcal{R}$ are assumed to be stationary). The belief at time $t$, $b_t$, denotes the posterior probability over $\mathcal{R}, \mathcal{P}$ given the history of state transitions and rewards observed until this time $b_t = \mathcal{P}( \mathcal{R}, \mathcal{P} | h_t)$, where $h_t = \{s_0, a_0, r_1, s_1, \ldots, r_t, s_t\}$ (note that we denote the reward after observing the state and action at time $t$ as $r_{t+1} = r(s_t, a_t)$). The belief can be updated iteratively according to Bayes rule, where $b_0(\mathcal{R}, \mathcal{P}) = p(\mathcal{R}, \mathcal{P})$, and:

$$b_{t+1}(\mathcal{R}, \mathcal{P}) = \mathcal{P}(\mathcal{R}, \mathcal{P} | h_{t+1}) \propto \mathcal{P}(s_{t+1}, r_{t+1} | h_t, \mathcal{R}, \mathcal{P}) b_t(\mathcal{R}, \mathcal{P}).$$

Similar to the idea of solving a POMDP by representing it as an MDP over belief states, the state in BRL can be augmented with the belief to result in the Bayes-Adaptive MDP (BAMDP) model. Denote the augmented state $s_t^+ = (s_t, b_t)$ and the augmented state space $S^+ = S \times B$, where $B$ denotes the belief space. The transitions in the BAMDP are given by: $P^+(s_{t+1}^+ | s_t^+, a_t) = \mathbb{E}_{b_t} [P(s_{t+1} | s_t, a_t) \delta(b_{t+1} = \mathcal{P}(\mathcal{R}, \mathcal{P} | h_{t+1}))],$ and the reward in the BAMDP is the expected reward with respect to the belief: $R^+(s_t^+, a_t) = \mathbb{E}_{b_t} [R(s_t, a_t)].$ The Bayes-optimal agent seeks to maximize the expected discounted return in the BAMDP, and the optimal solution of the BAMDP gives the optimal BRL policy.

As in standard MDPs, the optimal action-value function in the BAMDP satisfies the Bellman equation:

$$Q(s^+, a) = R^+(s^+, a) + \gamma \sum_{s^+ \in S^+} P^+(s^+ | s^+, a) \max_{a'} Q(s^{+'}, a'), \quad \forall s^+ \in S^+, a \in \mathcal{A}. \tag{2}$$

In principle, computing a Bayes-optimal agent amounts to solving the BAMDP, and the optimal policy can therefore be represented as a function of the augmented state. However, for most problems this is intractable, as the state space is continuous and high-dimensional (the space of beliefs), and the posterior update is also intractable in general.

The VariBAD Algorithm: VariBAD\footnote{\cite{39}} approximates the Bayes-optimal solution by combining a model for the MDP parameter uncertainty, and an optimization method for the corresponding BAMDP. The MDP parameters are represented by a vector $m \in \mathbb{R}^d$, corresponding to the latent variables in a parametric generative model for the state-reward trajectory distribution conditioned on the actions $P(s_0, r_1, s_1, \ldots, r_H, s_H | a_0, \ldots, a_{H-1}) = \int p_\theta(m) \mathcal{P}(s_0, r_1, s_1, \ldots, r_H, s_H | m, a_0, \ldots, a_{H-1}) dm$. The model parameters $\theta$ are learned by a variational approximation to the maximum likelihood objective, where the variational approximation to the posterior $P(m | s_0, r_1, s_1, \ldots, r_H, s_H, a_0, \ldots, a_{H-1})$ is chosen to have the structure $q_\phi(m | s_0, a_0, r_1, s_1, \ldots, r_t, s_t) = q_\phi(m | h_t)$. That is, the approximate posterior is conditioned on the history up to time $t$. The evidence lower bound (ELBO) in this case is

$$ELBO_t = \mathbb{E}_{m \sim q_\phi(m | h_t)} [\log p_\theta(s_0, r_1, s_1, \ldots, r_H, s_H | m, a_0, \ldots, a_{H-1})] - D_{KL}(q_\phi(m | h_t) || p_\theta(m)).$$

The main claim in\footnote{\cite{39}} is that $q_\phi(m | h_t)$ can be taken as an approximation of the belief $b_t$. In practice, $q_\phi(m | h_t)$ is represented as a Gaussian distribution $q(m | h_t) = N(\mu(h_t), \Sigma(h_t))$, where $\mu$ and $\Sigma$ are learned recurrent neural networks.

To approximately solve the BAMDP,\footnote{\cite{39}} exploit the fact that an optimal BAMDP policy is a function of the state and belief, and therefore consider neural network policies that take the augmented...
We start with an observation about the use of the BAMDP formulation in VariBAD, which will
follow the Meta-RL and BRL formulation described above, with a prior distribution over MDP
parameters $\pi(a_t|s_t, q_\phi(m|h_{i,j}))$, where the posterior is practically represented by the
distribution parameters $\mu(h_{i,j}), \Sigma(h_{i,j})$. To train such policies, maximize the BRL objective,

$$J(\pi) = \mathbb{E}_{R, \mathcal{P}} \mathbb{E}_\pi \left[ \sum_{t=0}^H \gamma^t r(s_t, a_t) \right], \quad (3)$$

using policy gradient based methods. The expectation over MDP parameters in (3) is approximated
by averaging over training environments, and the RL agent is trained online, alongside the VAE.

3 OMRL with Off-Policy VariBAD

In this section, we derive an off-policy variant of the VariBAD algorithm, and apply it to the OMRL
problem. We begin by describing our OMRL problem setting, and then present our algorithm.

3.1 OMRL Problem Definition

We follow the Meta-RL and BRL formulation described above, with a prior distribution over MDP
parameters $p(R, P)$. We are provided training data of an agent interacting with $N$ different MDPs,
$\{R_i, P_i\}_{i=1}^N$, sampled from the prior. We assume that each interaction is organized as $M$ trajectories
of length $H$, $\pi^{i,j} = s^{i,j}_0, a^{i,j}_0, r^{i,j}_1, s^{i,j}_1, \ldots, r^{i,j}_H, s^{i,j}_H$, $i \in 1, \ldots, N, j \in 1, \ldots, M$, where the
rewards satisfy $r^{i,j}_t = R_i(s^{i,j}_t, a^{i,j}_t)$, the transitions satisfy $s^{i,j}_{t+1} \sim P_i(s^{i,j}_t, a^{i,j}_t)$, and the actions
are chosen from an arbitrary data collection policy. To ground our work in a specific context, we
further assume that the different trajectories are obtained from running a conventional RL agent in
each one of the training MDPs, which implicitly specifies the data collection policy. We emphasize,
however, that this is merely an illustration, and our approach does not place any such constraint –
the trajectories can be collected using any other method. Our goal is to use the data for learning a
Bayes-optimal policy, i.e., a policy $\pi$ that maximizes Eq. (1).

3.2 Off-Policy VariBAD

The VariBAD algorithm in [39] builds on policy gradient optimization, and as such, it is inherently
online: policy gradient algorithms update the policy using data sampled from the current policy, and
thus cannot be applied to our offline setting. Our first step is to modify VariBAD to work off-policy.

We start with an observation about the use of the BAMDP formulation in VariBAD, which will
motivate our subsequent development.

Does VariBAD really optimize the BAMDP? It is well known that the optimal policy in a POMDP is
in general history dependent, while for MDPs, an optimal policy that is Markov (i.e., depends only
on the current state) exists [2]. Recall that a BAMDP is in fact a reduction of a POMDP to
an MDP over augmented states $s^+ = (s, b)$, and with the rewards and transitions given by $R^+$
and $P^+$. Thus, an optimal policy for the BAMDP exists in the form of $\pi(s^+)$, the VariBAD policy, as
described above, similarly takes as input the augmented state, and is thus capable of representing an
optimal BAMDP policy. However, VariBAD’s policy optimization in Eq. [3] does not make use of the
BAMDP parameters $R^+$ and $P^+$! While at first this may seem counterintuitive, Eq. [3] is in fact a
sound objective for the BAMDP, as we now show.

**Proposition 1.** Let $\tau = s_0, a_0, r_1, s_1, \ldots, r_H, s_H$ denote a random trajectory from a fixed history
dependent policy $\pi$, generated according to the following process. First, MDP parameters $R, P$
are drawn from the prior $p(R, P)$. Then, the state trajectory is generated according to $s_0 \sim P_{\text{init}},
a_t \sim \pi(\cdot|s_0, a_{t-1}, r_t, s_t)$. Let $b_t$ denote the posterior belief at time $t$, $b_t = P(R, P|s_0, a_0, r_1, \ldots, s_t)$.

Then

$$P(s_{t+1}|s_0, a_0, r_1, \ldots, r_t, s_t, a_t) = \mathbb{E}_{R, P \sim b_t} P(s_{t+1}|s_t, a_t), \text{ and}$$

$$P(r_{t+1}|s_0, a_0, r_1, \ldots, s_t, a_t) = \mathbb{E}_{R, P \sim b_t} R(r_{t+1}|s_t, a_t).$$

This result is closely related to the discussion in [28], here applied to our particular setting.
We make the following assumptions:

1. The transition and reward uncertainties are decoupled, i.e., the prior can be written as $p(R, P) = p_R(R) \times p_P(P)$.

2. For each MDP $i$ in the training data, we know the reward function $R_i$.

These assumptions are largely satisfied in most meta-RL studies to date. The decoupled uncertainty assumption can be hard to verify in practice. However, we remark that assuming a decoupled prior when the true prior is coupled can only enlarge the support of the prior distribution, since if
Figure 2: Reward ambiguity: from the two trajectories, it is impossible to know if there are two MDPs with different rewards (blue and yellow circles), or one MDP with rewards at both locations.

$p(R, P) > 0$, then the marginal distributions satisfy $p_R(R) > 0, p_P(P) > 0$. Thus, this assumption may lead to less effective estimation (as there are more MDP ‘possibilities’), but should not prohibit the agent from estimating the true posterior with enough data.

We next propose reward relabelling, a simple solution to the MDP ambiguity problem. From the illustration in Figure 2 it is clear that the fundamental cause for MDP ambiguity is that in each MDP, the agent visits a different part of the state space. We thus propose to make the state distribution in the offline data approximately uniform across all MDPs. We do this by replacing the rewards in a trajectory from some MDP $i$ in the data with rewards from another randomly chosen MDP $i' \neq i$. That is, for each $i \in 1, \ldots, N$, we add to the data $M$ trajectories $\hat{\tau}_{i,j}$, $j \in 1, \ldots, M$, where $\hat{\tau}_{i,j} = (s_{i,j}^0, a_{i,j}^0, \hat{r}_{i,j}^1, a_{i,j}^1, s_{i,j}^1, \ldots, \hat{r}_{i,j}^H, s_{i,j}^H)$, where the relabelled rewards $\hat{r}$ satisfy $\hat{r}_{i,j}^{t+1} = R_{i'}(s_{i,j}^t, a_{i,j}^t)$.

Note that our relabelling effectively samples data from an MDP with transitions $P_i$ and reward $R_{i'}$, which has non-zero prior probability mass under the decoupled prior assumption. We only use the relabelled data for training the VariBAD VAE. While in principle it could also be used for training the off-policy RL algorithm, we did not find it useful in practice.

4 Related Work

Meta-learning considers training agents that quickly solve a new learning problem, by exploiting structure in the problem distribution [21, 35]. In this work we focus on meta-RL – quickly learning to solve RL problems; a comprehensive recent survey of general meta-learning can be found in [13].

Gradient based approaches to meta-RL seek policy parameters that can be updated to the current task with a few gradient steps [12, 16, 31, 4], allowing for quick learning. These are essentially online methods, and several studies investigated learning of structured exploration strategies in this setting [18, 31, 33]. Memory-based meta-RL, on the other hand, trains recurrent neural network models that map the observed history in a task $h_t$ to an action $\sigma h_t$. These methods effectively treat the problem as a POMDP, and learn a memory based controller for it.

The connection between meta-learning and Bayesian methods, and between meta-RL and Bayesian RL in particular, has been investigated in a series of recent papers [24, 22, 39, 28], and our work closely follows these ideas. In particular, these works elucidate the difference between Thompson-sampling based strategies, such as in PEARL [30], and Bayes-optimal policies, such as in VariBAD [39], and suggest to estimate the BAMDP belief using the latent state of deep generative models.

Our contribution is an extension of these ideas to the offline RL setting, which to the best of our knowledge is novel. Technically, the VariBAD algorithm in [39] is limited to on-policy RL, and the off-policy method in [22] requires specific task descriptors during learning, while VariBAD, which our work is based on, does not [39].

Close to our work is Vuong et al. [37], who propose an offline meta-RL algorithm with two components: the first component learns value functions that are parameterized by a task context vector, by using the offline data for multiple tasks. The second component learns to identify the context vector of a task from observed transitions and rewards. At test time, the identified context vector is input to the value function to decide actions. Most importantly, since the two components are trained independently, and value functions are learned for individual tasks without explicitly accounting for the task uncertainty, this method cannot learn Bayes-optimal behavior. Related to this approach are contextual-MDPs [20, 23]. We also mention that recent work on meta Q-learning [11] also does not incorporate task uncertainty, and per our discussion above is not Bayes-optimal.
Classical works on BRL are mostly model based, and include offline methods that are suitable for small state and actions spaces, such as Duff’s work on finite state controllers [7], or the BEETLE algorithm [29], which is based on point-based POMDP planning. Online methods include, among others, Guez et al. [17], based on MCTS, and Strens [34], based on Thompson sampling. Model free methods include value function estimation based on Gaussian processes [9] and Kalman temporal differences [5]. We refer the reader to [15] for a comprehensive survey.

Finally, there is growing interest in offline deep RL [14, 32, 25]. Most recent work focus on how to avoid actions that were not sampled enough in the data. These works are tangential to OMRL, and in our experiments we did not require such techniques. We foresee that future work in OMRL will benefit from offline RL developments.

5 Experiments

In our experiments, we aim to answer the following questions: (1) can we learn approximately Bayes-optimal policies in the offline setting? and (2) does our off-policy method improve meta-RL performance in the online setting as well?

We emphasize that online methods are not limited by the quality of the offline data, and thus have a natural advantage. Thus, to answer (1) fairly, we compare our offline results with online methods based on Thompson sampling, which are not Bayes-optimal, and aim to show that the performance improvement due to being approximately Bayes-optimal gives an advantage even with the offline data restriction. To answer (2), we compare the online version of our method with online VariBAD, and also with our offline results. In our experiments, we investigate performance both on illustrative toy domains, and also on high-dimensional continuous control tasks in MuJoCo [36].

Our offline meta-RL agent training procedure is comprised of 3 steps: (1) Data collection by training RL agents to solve single tasks, sampled from the task distribution; (2) VAE training after applying reward relabelling to the collected data; and (3) Offline training of a meta-RL agent, using the collected data after applying state relabelling with the trained VAE encoder.

Data collection and organization: For data collection, we used off-the-shelf DQN (discrete domains) and SAC (continuous domains) implementations. We diversified the offline dataset by modifying the initial state distribution $P_{\text{init}}$ in the training tasks to be uniform over a large region (while at meta test time, the agent starts from a fixed position). The tasks are episodic, but we want a meta-RL agent that can maintain its belief between episodes, so that it can continually improve performance at test time (see Figure 1). We follow [39], and aggregate $k$ consecutive episodes of length $H$ to form a long trajectory of length $k \times H$ (denoted as $H^+$ in [39]), and we do not reset the hidden state in the VAE recurrent neural network after every episode termination. For reward relabelling, we replace the first $k/2$ trajectories with trajectories from a randomly chosen MDP, and relabel their rewards.

In our experiments we roughly follow the architecture and parameters used in VariBAD [39], as detailed in the supplementary material.

5.1 Offline Results

Illustrative Domains: We begin with two toy domains, in which we demonstrate our method on both discrete and continuous state spaces, and visualize the learned belief states.

1. Gridworld: A $5 \times 5$ gridworld environment as in [39]. The task distribution is defined by the location of a goal, which is unobserved and can be anywhere but around the starting state at the bottom-left cell. For each task, the agent receives a reward of $-0.1$ on non-goal cells and $+1$ at the goal. Given a new task, a Bayes-optimal agent would search for the goal without visiting states more than once, and stop moving as soon as the goal cell is found. In subsequent episodes, it will move directly to the found goal and stay there.

2. Semi-circle: A continuous 2D environment as in Figure 1, where the agent must navigate to an unknown goal, randomly chosen on a semi-circle of radius 1 [30]. For each task, the

---

*We note that the single agents’ training can be done in parallel.*
agent receives a reward of $+1$ if it is within a small radius of the goal, and $0$ otherwise. Due to the sparse reward, solving a new task efficiently requires a structured exploration strategy, e.g., quickly reaching the semi-circle and then searching along it. After the reward has been found, an optimal agent should directly move toward it in subsequent episodes.

Our results on the toy tasks are summarized in Figure 3, where we compare our offline algorithm with online Thompson sampling based methods, and also with an ablation of the reward relabelling method. For Gridworld, the Thompson sampling method can be computed exactly, while for Semi-circle, we use PEARL [30]. Note that we outperform online methods based on Thompson sampling, demonstrating our main claim – learning of approximately Bayes-optimal policies offline. Figure 1 shows a trajectory of the learned agent in Semi-circle – note the searching behavior, which is very different from behaviors in the data. Without reward relabelling, our method fails to learn a good meta-RL policy, even with a diverse dataset, demonstrating the severity of the MDP ambiguity problem.

In Figure 4, we plot the reward belief (obtained from the VAE decoder; see [39]) at different steps during the agent’s interaction in the Semi-circle domain. Note how the belief starts as uniform over the semi-circle, and narrows in on the target as more evidence is collected. Also note that without reward relabelling, the agent fails to find the goal. In this instance of the MDP ambiguity problem, the training data for the meta-RL agent consists of trajectories that mostly reach the goal, and as a result, the agent believes that the reward is located at the first point it reaches on the semi-circle.

MuJoCo Domains: To show that our method can scale to more complex domains, we next consider more challenging high-dimensional locomotion tasks based on MuJoCo [36]. We experimented with the following two domains:

1. **Half-Cheetah-Vel**: A commonly used environment in the meta-RL literature [12, 30, 39], in which a half-cheetah agent must run at a fixed velocity, randomly drawn from some interval. The rewards in this environment are dense, and given by the sum of the negative absolute value between the agent’s current velocity and the goal velocity, and an additional control cost on action magnitude.

2. **Ant-Semi-circle**: We modified the popular Ant-Goal task [11, 30, 39] to a sparse reward setting similar to the Semi-circle task above. Here, the ant needs to navigate to an unknown goal, randomly chosen on a semi-circle of radius $1$. The agent receives a reward of $+1$ if its center of mass is within a small radius of the goal, and $0$ otherwise. In addition, a control cost on action magnitude is applied at every step. This task involves both high-dimensional control and a non-trivial search for the goal.

---

4 A similar task was considered by Humplik et al. [22], but with several technical differences: the semi-circle had a smaller radius, actions were repeated for 10 time steps, and the algorithm in [22] requires a task description vector, which in this case was given by the angle in which the reward was located. In our approach, we chose the most simple modification of Ant-Goal to sparse rewards described above, and did not require any additional modifications of the task.
Figure 4: Semi-circle belief visualization. The plots show the reward belief over the 2-dimensional state space (obtained from the VAE) at different stages of interacting with the system. The red line marks the agent trajectory, and the light blue circle marks the true reward location. **Top:** Once the agent finds the true goal, it reduces the belief over other possible goals from the task distribution. **Middle:** As long as the agent doesn’t find the goal, it explores efficiently, reducing the uncertainty until the goal is found. **Bottom:** Without reward relabelling, the agent doesn’t learn to differentiate between different MDPs, and therefore fails to identify the goal (see text for further details).

When collecting data for Ant-Semi-circle, we found that the standard SAC algorithm was not able to solve the task effectively due to the sparse reward, and did not produce trajectories that reached the goal. We thus modified the reward **only during data collection** to be dense, and inversely proportional to the distance from the goal. After collecting the data trajectories, we replaced all the dense rewards in the data with sparse rewards, as outlined above, and proceeded with our OMRL training. We note that in [30], a similar approach was used to cope with sparse rewards in the online setting.

Figure 5: Ant-Semi-circle: trajectories from trained policy on a new goal. **Left:** Trajectory of the center of mass. **Right:** Visualization of the ant at different steps along the trajectories. Note that in the first episode, the ant searches for the goal, and in the second one it directly moves toward the goal it has previously found. This search behavior is different from the goal-reaching behaviors that dominate the training data.

Our results, presented in Figure 3, show that our offline approach effectively scales to high-dimensional control tasks, and outperforms the Thompson sampling based online PEARL. In the sparse Ant-Semi-circle domain, reward relabelling significantly improved the performance of our method. In the Half-Cheetah-Vel environment, on the other hand, the rewards are dense and MDP ambiguity is not a problem. In this domain we therefore did not require reward relabelling.
In Figure 5, we visualize the trajectories of a trained agent in the Ant-Semi-circle domain. Note the approximately Bayes-optimal behavior: in the first episode, the agent searches for the goal along the semi circle, and in the second episode, the agent maximizes reward by moving directly towards the already found goal. We emphasize that this behavior is very different from the training data, which mostly consists of trajectories that directly reach the goal.

5.2 Online Setting

Our method can also be applied to the online setting. In this case, it is simply a modification of VariBAD, where the policy gradient optimization is replaced with an off-policy RL algorithm. As shown in Figure 6, by exploiting the efficiency of off-policy RL, our method significantly improves sample-efficiency, without sacrificing final performance.

Figure 6: Online performance comparison. The off-policy optimization significantly improved VariBAD performance.

When comparing Figure 6 and Figure 3, the reader may notice that the online algorithm’s final performance outperforms the final performance in the offline setting. We emphasize that this phenomenon largely depends on the quality of the offline data, and not on the algorithm itself.

6 Conclusion

We presented the first offline meta-RL algorithm that is approximately Bayes-optimal. Key to our approach is the connection between Bayesian RL and meta learning, which in principle allows to reduce the problem to standard offline RL. In practice, however, we showed that the MDP ambiguity problem prohibits learning, and proposed a simple solution based on mixing trajectories in the data and relabelling their rewards. Our results show that this solution is effective on several domains.

Offline learning is appealing for domains where data collection is costly, such as robotics and healthcare, and there is growing interest in applying deep RL to this setting [25]. Key advances in this field will likely play a role in improving OMRL as well – an exciting direction for future research.

Acknowledgments

This work is partly funded by the Israel Science Foundation (ISF-759/19). The authors would like to thank Luisa Zintgraf for sharing the VariBAD code.
References

[1] J. Baxter and P. L. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

[2] D. P. Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena Scientific, 1995.

[3] A. R. Cassandra, L. P. Kaelbling, and M. L. Littman. Acting optimally in partially observable stochastic domains. In *AAAI*, volume 94, pages 1023–1028, 1994.

[4] I. Clavera, J. Rothfuss, J. Schulman, Y. Fujita, T. Asfour, and P. Abbeel. Model-based reinforcement learning via meta-policy optimization. In *Conference on Robot Learning*, pages 617–629, 2018.

[5] S. Di-Castro Shashua and S. Mannor. Kalman meets bellman: Improving policy evaluation through value tracking. *arXiv*, pages arXiv–2002, 2020.

[6] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel. RL2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

[7] M. O. Duff. Monte-carlo algorithms for the improvement of finite-state stochastic controllers: Application to bayes-adaptive markov decision processes. In *AISTATS*, 2001.

[8] M. O. Duff and A. Barto. *Optimal Learning: Computational procedures for Bayes-adaptive Markov decision processes*. PhD thesis, University of Massachusetts at Amherst, 2002.

[9] Y. Engel, S. Mannor, and R. Meir. Reinforcement learning with gaussian processes. In *Proceedings of the 22nd international conference on Machine learning*, pages 201–208, 2005.

[10] D. Ernst, P. Geurts, and L. Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(Apr):503–556, 2005.

[11] R. Fakoor, P. Chaudhari, S. Soatto, and A. J. Smola. Meta-q-learning. *arXiv preprint arXiv:1910.00125*, 2019.

[12] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[13] C. Finn and S. Levine. ICML tutorial on meta-learning: from few-shot learning to rapid reinforcement learning. [https://sites.google.com/view/icml19metalearning](https://sites.google.com/view/icml19metalearning).

[14] S. Fujimoto, D. Meger, and D. Precup. Off-policy deep reinforcement learning without exploration. *arXiv preprint arXiv:1812.02900*, 2018.

[15] M. Ghavamzadeh, S. Mannor, J. Pineau, and A. Tamar. Bayesian reinforcement learning: A survey. *arXiv preprint arXiv:1609.04436*, 2016.

[16] E. Grant, C. Finn, S. Levine, T. Darrell, and T. Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. *arXiv preprint arXiv:1801.08930*, 2018.

[17] A. Guez, D. Silver, and P. Dayan. Efficient bayes-adaptive reinforcement learning using sample-based search. In *Advances in neural information processing systems*, pages 1025–1033, 2012.

[18] A. Gupta, R. Mendonca, Y. Liu, P. Abbeel, and S. Levine. Meta-reinforcement learning of structured exploration strategies. In *Advances in Neural Information Processing Systems*, pages 5302–5311, 2018.

[19] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[20] A. Hallak, D. Di Castro, and S. Mannor. Contextual markov decision processes. *arXiv preprint arXiv:1502.02259*, 2015.

[21] S. Hochreiter, A. S. Younger, and P. R. Conwell. Learning to learn using gradient descent. In *International Conference on Artificial Neural Networks*, pages 87–94. Springer, 2001.

[22] J. Humplik, A. Galashov, L. Hasenclever, P. A. Ortega, Y. W. Teh, and N. Heess. Meta reinforcement learning as task inference. *arXiv preprint arXiv:1905.06424*, 2019.

[23] N. Jiang, A. Krishnamurthy, A. Agarwal, J. Langford, and R. E. Schapire. Contextual decision processes with low bellman rank are pac-learnable. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1704–1713. JMLR. org, 2017.
[24] G. Lee, B. Hou, A. Mandalika, J. Lee, S. Choudhury, and S. S. Srinivasa. Bayesian policy optimization for model uncertainty. arXiv preprint arXiv:1810.01014, 2018.

[25] S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. arXiv preprint arXiv:2005.01643, 2020.

[26] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971, 2015.

[27] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. Nature, 518(7540):529–533, 2015.

[28] P. A. Ortega, J. X. Wang, M. Rowland, T. Genewein, Z. Kurth-Nelson, R. Pascanu, N. Heess, J. Veness, A. Pritzel, P. Sprechmann, et al. Meta-learning of sequential strategies. arXiv preprint arXiv:1905.03030, 2019.

[29] P. Poupart, N. Vlassis, J. Hoey, and K. Regan. An analytic solution to discrete bayesian reinforcement learning. In Proceedings of the 23rd international conference on Machine learning, pages 697–704, 2006.

[30] K. Rakelly, A. Zhou, D. Quillen, C. Finn, and S. Levine. Efficient off-policy meta-reinforcement learning via probabilistic context variables. arXiv preprint arXiv:1903.08254, 2019.

[31] J. Rothfuss, D. Lee, I. Clavera, T. Asfour, and P. Abbeel. Prompt: Proximal meta-policy search. arXiv preprint arXiv:1810.06784, 2018.

[32] E. Sarafian, A. Tamar, and S. Kraus. Safe policy learning from observations. arXiv preprint arXiv:1805.07805, 2018.

[33] B. C. Stadie, G. Yang, R. Houthooft, X. Chen, Y. Duan, Y. Wu, P. Abbeel, and I. Sutskever. Some considerations on learning to explore via meta-reinforcement learning. arXiv preprint arXiv:1803.01118, 2018.

[34] M. Strens. A bayesian framework for reinforcement learning. In ICML, volume 2000, pages 943–950, 2000.

[35] S. Thrun and L. Pratt. Learning to learn: Introduction and overview. In Learning to learn, pages 3–17. Springer, 1998.

[36] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 5026–5033. IEEE, 2012.

[37] Q. Vuong, S. Liu, M. Liu, K. Ciosek, H. Su, and H. I. Christensen. Meta reinforcement learning from observational data, 2019.

[38] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick. Learning to reinforcement learn. arXiv preprint arXiv:1611.05763, 2016.

[39] L. Zintgraf, K. Shiariis, M. Igl, S. Schulze, Y. Gal, K. Hofmann, and S. Whiteson. Varibad: A very good method for bayes-adaptive deep rl via meta-learning. In International Conference on Learning Representation (ICLR), 2020.
As in [39], we trained only a reward decoder.

We now describe the rewards used for the MuJoCo experiments:

The overall training objective of the VAE is to maximize the sum of ELBO terms for different time steps,

\[ \log P(s_0, r_1, s_1, \ldots, s_H | a_0, \ldots, a_{H-1}) = \log \int P(s_0, r_1, s_1, \ldots, s_H, m | a_0, \ldots, a_{H-1}) \, dm \]

\[ = \log \int P(s_0, r_1, s_1, \ldots, s_H, m | a_0, \ldots, a_{H-1}) \frac{q_\phi(m|h,t)}{q_\phi(m|h,t)} \, dm \]

\[ = \log E_{m \sim q_\phi(\cdot|h,t)} \left[ P(s_0, r_1, s_1, \ldots, s_H, m | a_0, \ldots, a_{H-1}) \right] \frac{q_\phi(m|h,t)}{q_\phi(m|h,t)} \]

\[ \geq E_{m \sim q_\phi(\cdot|h,t)} \left[ \log p_\theta(s_0, r_1, s_1, \ldots, s_H | m, a_0, \ldots, a_{H-1}) \right] \]

\[ + \log p_\theta(m) - \log q_\phi(m|h,t) \]

\[ = E_{m \sim q_\phi(\cdot|h,t)} \left[ \log p_\theta(s_0, r_1, s_1, \ldots, s_H | m, a_0, \ldots, a_{H-1}) \right] \]

\[ - D_{KL}(q_\phi(m|h,t) || p_\theta(m)) \]

\[ = ELBO_t(\theta, \phi). \]

The prior \( p_\theta(m) \) is set to be the previous posterior \( q_\phi(m|h,t-1) \), with initial prior chosen to be standard normal \( p_\theta(m) = \mathcal{N}(0, I) \). The decoder \( p_\theta(s_0, r_1, s_1, \ldots, s_H | m, a_0, \ldots, a_{H-1}) \) factorizes to reward and next state models \( p_\theta(s'|s, a, m) \) and \( p_\theta(r|s, a, m) \), according to:

\[ \log p_\theta(s_0, r_1, s_1, \ldots, s_H | m, a_0, \ldots, a_{H-1}) = \log p_\theta(s_0|m) \]

\[ + \sum_{t=0}^{H-1} \left[ \log p_\theta(s_{t+1}|s_t, a_t, m) + \log p_\theta(r_{t+1}|s_t, a_t, m) \right]. \]

As in [39], we trained only a reward decoder.

The overall training objective of the VAE is to maximize the sum of ELBO terms for different time steps,

\[ \max_{\theta, \phi} \sum_{t=0}^{H} ELBO_t(\theta, \phi). \quad (4) \]

### B Experiment Details

In this section we describe the details of the domains we experimented with and outline our training hyperparameters. In all our experiments we average the performance over 3 random seeds and present the mean and standard deviation.

**Gridworld:** Similarly to [39], the horizon in the Gridworld domain is 15 and we aggregate \( k = 4 \) consecutive episodes to form a trajectory of length 60. We used DQN [27] in our experiments with soft target network updates, as proposed in [26], which has shown to improve the stability of learning.

**Semi-circle:** We set the horizon in the Semi-circle domain to 60 and aggregate \( k = 2 \) consecutive episodes to form a trajectory of length 120. We used SAC [19] with the architectures of the actor and critic chosen similarly, and with a fixed entropy coefficient.

**MuJoCo:** The horizon is set to \( H = 200 \) and we consider \( k = 2 \) episodes. As in the Semi-circle domain, we used SAC for continuous control with the same architecture for both actor and critic and we set a fixed entropy coefficient. We now describe the rewards used for the MuJoCo experiments:

1. **Half-Cheetah-Vel:** Following recent works in meta-RL [12, 30, 39], we consider velocities drawn uniformly between 0.0 and 3.0. The reward in this environment is given by

\[ r_t = -|v_t - v_{goal}| - 0.05 \cdot ||a_t||^2 \]

where \( v_t \) is the current velocity, and \( a_t \) is the current action.
2. **Ant-Semi-circle**: For training data collection RL agents, we used the following dense reward:

\[
r_t^{\text{dense}} = -\|x_t - x_{\text{goal}}\|_1 - 0.1 \cdot \|a_t\|_2^2
\]

where \(x_t\) is the current 2D location and \(a_t\) is the current action. Then, we replaced the dense rewards in the collected data with sparse rewards that are given by

\[
r_t^{\text{sparse}} = -0.1 \cdot \|a_t\|_2^2 + \begin{cases} 
1, & \|x_t - x_{\text{goal}}\|_2 \leq 0.2 \\
0, & \text{else}
\end{cases}
\]

In all our experiments we set the soft target update parameter to 0.005.

### B.1 Offline Results

Our offline training procedure is comprised of 3 separate training steps. First is the training of the data collection RL agents. Each agent is trained on a different task from the task distribution.

For the Gridworld domain, we train 21 agents. We note that this covers the entire task distribution, as goals can be anywhere but around the starting state at the bottom-left cell. For the Semi-circle and Ant-Semi-circle domains, we train 80 data collection agents, and for the Half-Cheetah-Vel environment we used 100 agents.

For all tasks, we used a similar architecture of 2 fully-connected (FC) hidden layers of size that depends on the domain with ReLU activations, and set the batch size to 256. The rest of the hyperparameters used for training the data collection RL agents are summarized in the following table:

| Hyperparameter                  | Gridworld (DQN) | Semi-circle (SAC) | MuJoCo (SAC) |
|---------------------------------|-----------------|------------------|--------------|
| Hidden layers size              | 16              | 32               | 128          |
| Num. iterations                 | 200             | 300              | 1000         |
| RL updates per iter.            | 500             | 500              | 2000         |
| Exploration/entropy coeff.      | \(\epsilon\)-greedy, linear annealing from 1 to 0.1 over 100 iterations | 0.01          | 0.2          |
| Collected episodes per iter.    | 5               | 2                | 2            |
| Learning rate/s                 | \(3 \cdot 10^{-4}\) | \(3 \cdot 10^{-4}\) | \(3 \cdot 10^{-4}\) |
| Discount factor (\(\gamma\))    | 0.99            | 0.9              | 0.99         |

The second training step is the VAE training after applying reward relabelling to the collected data.

The VAE consists of a recurrent encoder, which at time step \(t\) takes as input the tuple \((a_t, r_{t+1}, s_{t+1})\). The state and reward are passed each through a different fully-connected (FC) layer. The state FC layer is of size 32 and the reward FC layer is of size 8 for the Gridworld and 16 for the rest of the domains, all with ReLU activations. For the MuJoCo environments, we also pass the action through a FC layer of size 16 with ReLU. Then, the state and reward layers’ outputs are concatenated along the action (or with the output of the action layer in the case of MuJoCo) and passed to a GRU of the latent vector \(m\), whose dimensionality is 5 in all our experiments.

The VAE reward decoder takes as input a latent sample \(m \sim \mathcal{N}(\mu(h_{\lambda}), \Sigma(h_{\lambda}))\) and the states along the trajectory \(s_1, \ldots, s_H\), each state at a time, and outputs (for every timestep \(t = 1, \ldots, H\)) the entire reconstructed/predicted rewards \(r_1, \ldots, r_H\) along the trajectory. In the MuJoCo domains the reward decoder also takes as input the actions along the trajectory \(a_1, \ldots, a_H\) and the previous states \(s_0, \ldots, s_{H-1}\) as the reward \(r_t\) in these environments generally depends on \(s_{t-1}, a_t, s_t\). The reward decoder is comprised of 2 FC layers, each of size 32.

The VAE is trained to optimize \([4]\), but similarly to \([39]\), we weight the KL term in each of the ELBO terms with some parameter \(\beta\), which is not necessarily 1. In our experiments we used \(\beta = 0.05\).

After the VAE is trained, we apply state relabelling to the data collected by the RL agents, to create a large offline dataset that effectively comes from the BAMDP. Then, we train an off-policy RL algorithm, which is our meta-RL agent, using the offline data.

For the offline meta-RL agents training, we used similar hyperparameters to those used for the data collection RL agents training. We only enlarge the size of the hidden layers in our models from 16,
32 and 128 to 64, 128 and 256 for the Gridworld, Semi-circle and MuJoCo domains, respectively. In every iteration we perform 1000 parameter updates for all environments except the Ant-Semi-circle, in which case we perform 2000 updates per iteration.

**B.2 Online Results**

In the online setting we didn’t apply reward relabelling to the data, since, as we explained, MDP ambiguity doesn’t concern online meta-RL. The hyperparameters used in the online setting are as follows:

| RL parameters                          | Gridworld (DQN) | Semi-circle (SAC) | Cheetah-Vel (SAC) |
|----------------------------------------|-----------------|-------------------|------------------|
| Architecture/s                         | 2 FC layers     | 2 FC layers       | 3 FC layers      |
|                                        | of size 100.    | of size 128.      | of size 128.     |
| Num. updates per iter.                 | 250             | 1000              | 2000             |
| Exploration/entropy coeff.             | ϵ-greedy, linear annealing from 1 to 0.1 over 1000 iterations. | 0.01             | 0.2              |
| Collected episodes per iter.           | 25              | 25                | 25               |
| Learning rate/s                        | 7 · 10⁻⁵        | 7 · 10⁻⁵          | 3 · 10⁻⁴         |
| Discount factor (γ)                    | 0.99            | 0.9               | 0.99             |
| VAE parameters                         |                 |                   |                   |
| Encoder architecture                   | state/reward FC layer of size 32/8. GRU of size 64. | state/reward FC layer of size 32/8. GRU of size 128. | state/action/reward FC layer of size 32/16/16. GRU of size 128. |
| Reward decoder architecture            | 2 FC layers of size 32. | 2 FC layers of sizes 64 and 32. | 2 FC layers of sizes 64 and 32. |
| Num. updates per iter.                 | 20              | 25                | 20               |
| Learning rate                          | 3 · 10⁻⁴        | 10⁻³              | 3 · 10⁻⁴         |
| Weight of KL term (β)                  | 1.0             | 0.1               | 1.0              |