Residual Bayesian Q-Learning for Meta-Reinforcement Learning with Experts

Gilwoo Lee, Sanjiban Choudhury, Brian Hou, Siddhartha S. Srinivasa
Paul G. Allen School of Computer Science and Engineering
University of Washington
Seattle, WA
Email: gilwoo, sanjibac, bhou, siddh@cs.washington.edu

***Abstract*—We examine the problem of Bayesian meta-reinforcement learning for Markov Decision Processes. Meta-reinforcement learning is a process of training a learning system to solve an unknown MDP that shares commonalities with MDPs observed in the past. We build upon existing meta-reinforcement learning approaches by leveraging a Bayes filter over underlying latent MDPs. We utilize the fact that training an expert policy for single MDP is much easier than training a universal Bayes-optimal policy, and that the expected Q-value of the experts upper-bounds the Bayes-optimal Q-value.**

We propose a Q-learning network architecture and an algorithm that jointly trains the experts and the residual network while keeping experts to specialize per MDP. Our key insight that we can keep the experts distinctive by crediting only the expert corresponding to the latent MDP that ran the episode, while always crediting residual network across all episodes. This strikes a balance between learning for an MDP via expert and learning across MDPs via residual network. Our framework encompasses several alternate architectures and techniques for Bayesian Q-learning. Empirical evaluations on established baselines reveal insightful details and bear promise.

I. Introduction

We focus on the problem of meta-reinforcement learning for a distribution of Markov Decision Processes (MDP). The goal of meta-reinforcement learning is to train a system that, at test time, is able to solve a new MDP in a data-efficient fashion. Meta-reinforcement learning can be used for many practical robotics problems where the dynamics function or the task are unknown but share a common structure. Model-Agnostic Meta Learning approaches [5] have successfully produced robust, generalizable learners by exposing the agent to a distribution of MDPs during training time. Recent works have observed that by exposing the agent to a history of observations using a Bayes filter or a recurrent neural network, we can aim for the Bayes-optimal objective, the best theoretical performance that a meta-learner can achieve [13, 17]. We refer to this as Bayesian meta-reinforcement learning.

Unfortunately, achieving the Bayes-optimal objective is difficult because the agent must learn multiple things, all of which are challenging: 1) a mapping from the history of observation to belief over MDPs (a Bayes Filter) 2) a Bayes-optimal policy that balances a set of distinctive, exploitative policies with an explorative, risk-sensitive policy. For example, consider the problem of learning to fly a glider when the agent does not know the weather conditions (Figure 1). Flying in

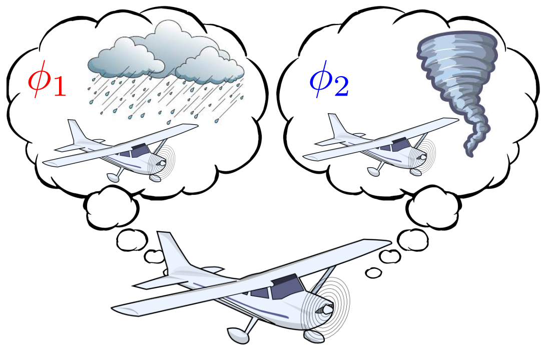


Fig. 1: A glider flying in unknown weather. A Bayes-optimal agent should be able to know how to both fly in different weather situations and distinguish between the two.

icy showers is very different from flying in strong winds. The agent must not only figure out distinct, exploitative policies for flying in each of these conditions individually, but also discern in a risk-free manner which situation it is in.

In this paper, we focus on the problem of learning a Bayes-optimal policy and assume that the Bayes filter is either given or can be approximated from a recurrent neural network. Existing approaches solve this by directly training a universal policy given a belief [13], sampling from the belief [29, 18], or combining multiple expert policies via imitation learning [15] without maintaining belief.

We leverage the fact that training an expert policy for each MDP is much easier than training a single Bayes-optimal agent across multiple MDPs. Given an MDP, such an expert policy could come from optimal control, demonstration, or another RL algorithm. Once these experts are provided, we can use their expected value under the belief as a baseline, which is a well-known upper-bound to the Bayes-optimal value function [14]. Since the expected value is optimal when the belief collapses to single MDP, the agent can focus learning in regions where the entropy is high. Formally, we decompose the Bayes-optimal value function as the following:

$$\begin{aligned} Q^*(s,b,a) = (1-w(b))\sum_k b(k)Q^{(k)}(s,a) \\ + w(b)Q^r(s,b,a) \end{aligned} \tag{1}$$

where $w(b)$ is the entropy, $b(k)$ is the belief of the kth MDP, and $Q^{(k)}$ is the value function of the corresponding expert and Q^r is the *residual*. We assume there is a known finite number of MDPs or that they have been clustered into a finite number of clusters.

Our algorithm is a Bayesian Q-learning method which simultaneously learns the residual and improves the expert values. Our key insight is that we can maintain distinct experts by only exposing each to the MDP that it is responsible for; because each expert is responsible only for a single MDP, the experts remain belief-agnostic. Meanwhile, the residual Q-network (Q^r) is belief-aware and is trained across all episodes.

We make the following contributions:

- An architecture for a Bayesian meta-learner with distinct experts and a residual Q-network
- A family of algorithms that jointly train expert and residual Q-networks, warm-start learned experts, or use fixed experts
- Empirical evaluation that demonstrates the network is able to learn Bayes-optimal behaviors in a data-efficient manner

II. Preliminary: Bayesian Meta-Learning

Bayesian meta-reinforcement learning is an alternate term for the more traditional Bayesian Reinforcement Learning (BRL).[1]. In this problem, the agent at test time does not know the reward and transition functions but knows that they are determined by a latent variable $\phi \in \Phi$. The goal is to do as well as it can given uncertainty over ϕ.

Formally, the problem is defined by a tuple $\langle S, \Phi, A, T, R, P_0, \gamma \rangle$, where S is the observable state space of the underlying MDP, Φ is the latent space, and A is the action space. T and R are the parameterized transition and reward functions, respectively. The transition function is defined as: $T(s, \phi, a', s', \phi') = P(s', \phi' | s, \phi, a') = P(s'|s, \phi, a')P(\phi'|s, \phi, a', s')$. The initial distribution over (s, ϕ) is given by $P_0 : S \times \Phi \rightarrow \mathbb{R}^+$, and γ is the discount.

BRL considers the long-term expected reward with respect to the uncertainty over ϕ rather than the true (unknown) value of ϕ. The uncertainty is represented as a *belief distribution* $b \in B$ over latent variables ϕ. The Bayes-optimal action value function is given by the following Bellman equations

$$Q(s, b, a) = R(s, b, a) + \gamma \sum_{s', b'} P(s', b'|s, b, a') \max_{a'} Q(s', b', a') \tag{2}$$

where the Bayesian reward and transition functions are defined in expectation with respect to ϕ: where the reward function is the expected reward $R(s, b, a') = \sum_{\phi \in \Phi} b(\phi) R(s, \phi, a')$, the state transition function is $P(s'|s, b, a') = \sum_{\phi \in \Phi} b(\phi) P(s'|s, \phi, a')$ and the belief transition function is computed using a Bayes update.

[1]Our choice of nomenclature is to draw connections to meta-learning which also reasons about learning on a distribution of MDPs

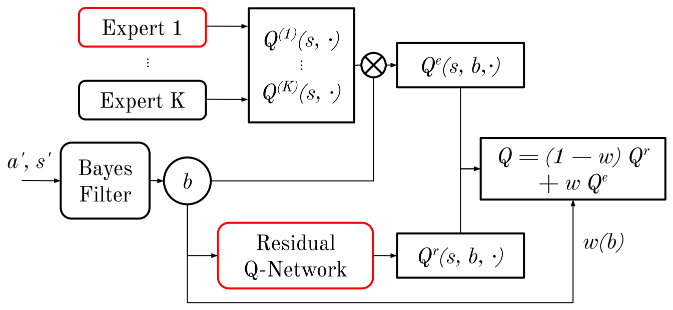


Fig. 2: Network overview. State s is an input to every module, and s', a' are the state and action from the previous step. Red is one example of trained modules per sample batch. Samples from $\mathcal{M}^k$ are only used to train expert k, while the Residual Q-network is trained with samples from all MDPs.

III. Related Work

Meta-reinforcement learning (MRL) approaches train sample-efficient learners by exploiting structure common to a distribution of MDPs. For example, MAML [5] trains gradient-based learners while RL2 [4] trains memory based learners. While meta-supervised learning has well established Bayesian roots [1, 2], it wasn't until recently that meta-reinforcement learning was strongly tied to Bayesian Reinforcement Learning (BRL) [17, 20]. Nevertheless, even non-Bayesian MRL approaches address problems pertinent to BRL. MAESN [10] learns structured noise for exploration in MAML. E-MAML [25] adds an explicit exploration bonus to MAML objective. GMPS [16] exploit availability of MDP experts to partially reduce BRL to IL. Our work is more closely related to Bayesian MRL approaches. MAML-HB [8] casts MAML as hierarchical Bayes and improves posterior estimates. BMAML [28] uses non-parametric variational inference to improve posterior estimates. PLATIPUS [6] learns a parameter distribution instead of a fixed parameter. PEARL [21] learns a data-driven Bayes filter across tasks. In contrast to these approaches, we focus on the structure of Bayes-optimal value function and try to learn it directly.

Bayesian reinforcement learning aims to train learners that are both safe and sample-efficient by leveraging a prior distribution [7, 22]. However, computing Bayes-optimal policy for most practical problems is intractable and requires approximation. Point-based solvers, like SARSOP [12] and PBVI [19], exploit the piecewise-linear-convex structure of value functions to approximate the value of a belief state. Sampling-based approaches, such as BAMCP [9] and POMCP [23], combine Monte Carlo sampling and simple rollout policies to approximate values at the root node in a search tree. POMCPOW [26] extend POMCP to continuous spaces using double progressive widening.

IV. Residual Bayesian Q-Network

We propose a Q-learning algorithm which combines a set of expert value functions in a Bayes-optimal manner by simultaneously learning a residual value function and expert

Algorithm 1 Residual Bayesian Q-Network

Require: MDP distribution P_0, Bayes filter $\psi(\cdot)$, initial belief b_0, θ_0, horizon H, N, Replay memory $\mathcal{D}$

```
1:  for episode= $1, \cdots, N$ do
2:      Sample an MDP $\mathcal{M}_i \sim P_0$
3:      Initialize belief $b_0$
4:      for $t = 1, \cdots, H$ do
5:          With probability $\epsilon$ select a random action $a_t$, or
6:          $a_t = \arg\max_a Q_\theta(s_t, b_t, a)$
7:          Execute $a_t$ and observe $r_t, s_{t+1}$
8:          Update $b_{t+1} = \psi(b_t, a_t, s_{t+1})$
9:          Store $(i, s_t, b_t, a_t, r_t, s_{t+1}, b_{t+1})$ in $\mathcal{D}$
10:     Sample $\mathcal{M}_k \sim P_0$
11:     Sample a random minibatch from $\mathcal{D}$ with $k$, i.e. $(k, s_j, b_j, a_j, r_j, s_{j+1}, b_{j+1})$
12:     Set $y_j^{(k)} = r_j + \gamma \max_{a'} Q_\theta^k(s_j, a')$
13:     Set $y_j = r_j + \gamma \max_{a'} Q_\theta(s_j, b_j, a')$
14:     Perform a gradient descent step on $(y_j^{(k)} - Q_\theta^k(s_j, a_j))^2$, update only $\theta_k$ for $Q_\theta^k$
15:     Perform a gradient descent step on $(y_j - Q_\theta(s_j, b_j, a_j))^2$, update only $\theta_r$ for $Q_\theta^r$
```

value functions. We start from the idea that each expert can be responsible for one of the latent MDPs, and combine the expert value functions with a residual value function (Equation 1). Assuming that the experts are indeed good experts for each MDP, we use entropy as w, i.e. $w(b) = -\alpha \sum_k b(k) \log(b(k))$ with α as a tuning parameter so that the residual function disappears when the entropy is low.

Now we need to make sure that the experts are actually trained to achieve high performance on their assigned MDP. The key insight is to expose expert k only to the episodes where $\mathcal{M}^k$ was the latent MDP, and train it in a belief-agnostic manner.

The algorithm is provided in Algorithm 1, which is a modified version of Deep Q Network (DQN) [27]. Figure 2 shows the network structure. During training, we keep track of the latent MDPs and maintain target values for the final value function and expert value functions separately. Given a replay sample $(k, s_j, b_j, a_j, r_j, s_{j+1}, b_{j+1})$ generated from $\mathcal{M}^k$, we have

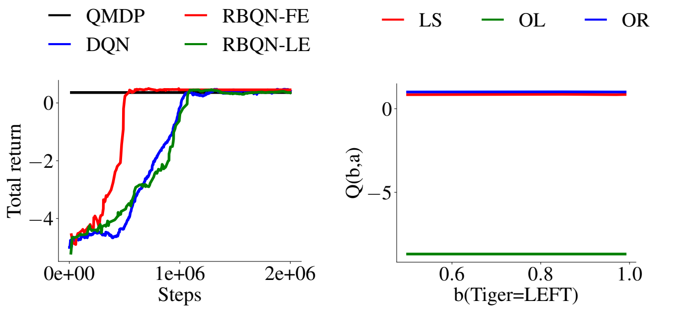
$$y_j^{(k)} = r_j + \gamma \max_{a'} Q_\theta^{(k)}(s_j, a') \tag{3}$$

corresponding to the target for the expert Q_θ^k and

$$y_j = r_j + \gamma \max_{a'} Q_\theta(s_j, b_j, a') \tag{4}$$

corresponding to the target for the weight-balanced sum of Q^r and $Q^{(1),\ldots,(k)}$ in Equation 1.

In certain cases, we may have access to near-optimal experts even during test time. In this case we propose to use a simpler version of our algorithm, which we refer to as RBQN-FixedExperts (RBQN-FE). In RBQN-FE, we perform the gradient descent steps from Equation 4 and update only θ_r.



(a) Total rewards (b) RBQN-LE expert

Fig. 3: `Tiger`. (a) RBQN-FE converges quickly to the optimum. (b) RBQN-LE produces experts specialized in one MDP, which opens one of the doors deterministically.

Through experiments we verify that this indeed results in a data-efficient, fast convergence to the optimal value. We refer to our original algorithm which jointly trains the experts as RBQN-LearnedExperts (RBQN-LE). If the experts are accessible only during training, we can consider an intermediate approach where we warm-start the expert networks in RBQN-LE with expert value functions and jointly train them.

V. Experimental Results

A. Toy Example: Tiger

We first use a toy problem to verify our hypothesis that our algorithm produces experts that specialize in each MDP. We look into the `Tiger` problem, originally proposed by Kaelbling et al. [11]. In this problem, a tiger is hiding behind one of two doors. An agent must choose among three actions: listen (LS), or open one of the two doors (OL, OR). When the agent listens, it receives a noisy observation of the tiger's position. If the agent opens the door and reveals the tiger, it receives a penalty of -100. Opening the door without the tiger results in a reward of 10. Listening incurs a penalty of -1. In this problem, a Bayes-optimal agent would listen until its belief about which door the tiger is behind is substantially higher for one door than the other. Chen et al. [3] frame `Tiger` as a BRL problem with two latent states.

Figure 3a shows two versions of our algorithm, RBQN-FixedExperts, RBQN-LearnedExperts, compared with a vanilla DQN trained on the belief MDP which provides belief of Tiger location as an input. We can see that the RBQN-FE has a much faster convergence as the value is near-optimal for one-hot beliefs.

Figure 3b shows that RBQN-LE is able to train each expert on one MDP, without mixing other MDPs. It shows the value function of one of the experts trained by RBQN-LE. The expert trained with the tiger behind the left door learns precisely that it can open the right door for all beliefs. Nonetheless, the final value function produced by RBQN-LearnedExperts is Bayes-optimal, as can be seen in Figure 3a.

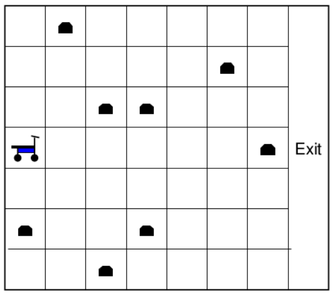


Fig. 4: `RockSample`. Figure from Smith and Simmons [24].

RBQN-FE	DQN	BPO	QMDP	SARSOP
19.34± 0.33	0.0 ± 0.0	7.35 ± 0.0	16.58 ± 0.99	21.47 ± 0.04

TABLE I: `RockSample` with 95% confidence interval.

B. *RockSample*

Next, we turn to the `RockSample` POMDP problem, which has also been shown to to be a BRL problem by Chen et al. [3], to demonstrate how much our algorithm can improve beyond the experts. In this problem, the agent is in a grid world where rocks are positioned at a set of predetermined locations (Figure 4). The rocks are either GOOD or BAD, and the agent must sense the rock and use the noisy sensor information to determine which rock to sample. Sampling GOOD rocks provides +10 reward, while sampling BAD rocks result in -10 penalty. This problem is very challenging because the agent has to associate the sparse rewards on the rocks with sensing actions which do not have immediate rewards. We use the setting with 8 rocks in a 7×7 grid.

Table I shows the comparison of RBQN-FE with other algorithms. Vanilla DQN was not able to learn this relationship, resulting in zero rewards. BPO, a PPO-based algorithm on the belief MDP [13], learns to go straight to the goal.

In RBQN-FE, there are $2^8 = 256$ experts. Each expert takes the most value-maximizing path given a fixed rock configuration of GOOD and BAD. RBQN-FE combines the experts' value functions with the output of the residual network with a weight parameter of 0.01 given to the residual network. The weight parameter was found by offline tuning. Our algorithm in fact learns to significantly outperform the QMDP agent, getting closer to the near-optimal value given by SARSOP, an offline appoximate POMDP solver [12].

However, we were not able to get RBQN-LE to produce as good as performance as RBQN-FE, as RBQN-LE struggled to get each of the experts to the level comparable to optimal experts.

We believe that this shows our algorithm as a potential way to combine optimal control methods with Bayesian RL approaches to produce an agent which performs beyond what each optimal control expert can suggest, thus getting the best of both worlds.

VI. Discussion and Future Work

We have proposed an algorithm for Bayesian meta-reinforcement learning which simultaneously trains experts and a residual Q-network. Our algorithm trains each expert solely on the MDP it is responsible for while training the residual network to learn how to combine the experts Bayes optimally.

Our results show both limitations of and promising directions for our algorithm. Through the experiment on `Tiger` we have verified that the experts in the joint training scenario indeed learn to be single-MDP experts. From the experiment on `RockSample`, we show that the residual network is indeed capable of learning to perform better what the experts propose even when the experts are optimal with respect to each of the latent MDPs. However, jointly training the experts and the residual for this large-scale problem turned out to be quite challenging, even when we warm-started the Q functions with approximated Q values from the experts. This is in fact a phenomenon commonly observed in Imitation Learning algorithms, where even a small discrepancy between the learned Q function and the expert Q function results in the agent exploring unknown domains and therefore failing to learn. On the other hand, RBQN-FE sheds new light on how to combine experts which can be acquired from much simpler settings, to get an agent that can handle complex Bayesian RL problems.

Acknowledgments

Gilwoo Lee is partially supported by Kwanjeong Educational Foundation, and Brian Hou is partially supported by NASA Space Technology Research Fellowships (NSTRF). This work was partially funded by the National Institute of Health R01 (#R01EB019335), National Science Foundation CPS (#1544797), National Science Foundation NRI (#1637748), the Office of Naval Research, the RCTA, Amazon, and Honda.

References

[1] Jonathan Baxter. Theoretical models of learning to learn. In *Learning to learn*, pages 71–94. Springer, 1998.
[2] Jonathan Baxter. A model of inductive bias learning. *Journal of artificial intelligence research*, 12:149–198, 2000.
[3] Min Chen, Emilio Frazzoli, David Hsu, and Wee Sun Lee. POMDP-lite for robust robot planning under uncertainty. In *IEEE International Conference on Robotics and Automation*, 2016.
[4] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.
[5] Chelsea Finn, Pieter Abbeel, and Sergey 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.

[6] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. *arXiv preprint arXiv:1806.02817*, 2018.

[7] Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, Aviv Tamar, et al. Bayesian reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 8(5-6):359–483, 2015.

[8] Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. *arXiv preprint arXiv:1801.08930*, 2018.

[9] Arthur Guez, David Silver, and Peter Dayan. Efficient Bayes-adaptive reinforcement learning using sample-based search. In *Advances in Neural Information Processing Systems*, 2012.

[10] Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Meta-reinforcement learning of structured exploration strategies. In *Advances in Neural Information Processing Systems*, pages 5302–5311, 2018.

[11] Leslie Pack Kaelbling, Michael Littman, and Anthony Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, 1998.

[12] Hanna Kurniawati, David Hsu, and Wee Sun Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Robotics: Science and Systems*, 2008.

[13] Gilwoo Lee, Brian Hou, Aditya Mandalika, Jeongseok Lee, Sanjiban Choudhury, and Siddhartha S. Srinivasa. Bayesian policy optimization for model uncertainty. In *International Conference on Learning Representations*, 2019.

[14] Michael Littman, Anthony Cassandra, and Leslie Pack Kaelbling. Learning policies for partially observable environments: Scaling up. In *International Conference on Machine Learning*, 1995.

[15] Russell Mendonca, Abhishek Gupta, Rosen Kralev, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Guided meta-policy search. *CoRR*, abs/1904.00956, 2019.

[16] Russell Mendonca, Abhishek Gupta, Rosen Kralev, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Guided meta-policy search. *arXiv preprint arXiv:1904.00956*, 2019.

[17] Pedro A. Ortega, Jane X. Wang, Mark Rowland, Tim Genewein, Zeb Kurth-Nelson, Razvan Pascanu, Nicolas Heess, Joel Veness, Alexander Pritzel, Pablo Sprechmann, Siddhant M. Jayakumar, Tom McGrath, Kevin Miller, Mohammad Gheshlaghi Azar, Ian Osband, Neil C. Rabinowitz, András György, Silvia Chiappa, Simon Osindero, Yee Whye Teh, Hado van Hasselt, Nando de Freitas, Matthew Botvinick, and Shane Legg. Meta-learning of sequential strategies. *arXiv preprint arXiv:1905.03030*, 2019.

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

[19] Joelle Pineau, Geoff Gordon, Sebastian Thrun, et al. Point-based value iteration: An anytime algorithm for POMDPs. In *International Joint Conference on Artificial Intelligence*, 2003.

[20] Neil C. Rabinowitz. Meta-learners' learning dynamics are unlike learners'. *arXiv preprint arXiv:1905.01320*, 2019.

[21] Kate Rakelly, Aurick Zhou, Deirdre Quillen, Chelsea Finn, and Sergey Levine. Efficient off-policy meta-reinforcement learning via probabilistic context variables. *arXiv preprint arXiv:1903.08254*, 2019.

[22] Guy Shani, Joelle Pineau, and Robert Kaplow. A survey of point-based POMDP solvers. *Journal on Autonomous Agents and Multiagent Systems*, 27(1):1–51, 2013.

[23] David Silver and Joel Veness. Monte-carlo planning in large POMDPs. In *Advances in Neural Information Processing Systems*, 2010.

[24] Trey Smith and Reid Simmons. Heuristic search value iteration for pomdps. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 520–527. AUAI Press, 2004.

[25] Bradly C. Stadie, Ge Yang, Rein Houthooft, Xi Chen, Yan Duan, Yuhuai Wu, Pieter Abbeel, and Ilya Sutskever. Some considerations on learning to explore via meta-reinforcement learning. *arXiv preprint arXiv:1803.01118*, 2018.

[26] Zachary Sunberg and Mykel Kochenderfer. Online algorithms for POMDPs with continuous state, action, and observation spaces. In *International Conference on Automated Planning and Scheduling*, 2018.

[27] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI Conference on Artificial Intelligence*, 2016.

[28] Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, pages 7332–7342, 2018.

[29] Wenhao Yu, Jie Tan, C. Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification. In *Robotics: Science and Systems*, 2017.