

Towards data-efficient deployment of reinforcement learning systems

Sebastian Schulze

St. Anne’s College
University of Oxford

*A thesis submitted for the degree of
Doctor of Philosophy*

Hillary 2021

Abstract

A fundamental concern in the deployment of artificial agents in real-life is their capacity to quickly adapt to their surroundings. Traditional reinforcement learning (RL) struggles with this requirement in two ways. Firstly, iterative exploration of unconstrained environment dynamics yields numerous uninformative updates and consequently slow adaptation. Secondly, final policies have no capacity to adapt to future observations and have to either slowly learn indefinitely or retrain entirely as observations occur.

This thesis explores two formulations aimed at addressing these issues. The consideration of entire task distributions in meta-RL evolves policies quickly adapting to specific instances on their own. By forcing agents to specifically request feedback, Active RL enforces selective observations and updates. Both of these formulations reduce to a Bayes-Adaptive setting in which a probabilistic belief over possible environments is maintained. Many existing solutions only provide asymptotic guarantees that are of limited use in practical contexts. We develop a variational approach to approximate belief management and support its validity empirically through a broad range of ablations. We then consider recently successful planning approaches but uncover and discuss obstacles in their application to the discussed settings.

An important factor influencing the data requirements and stability of RL systems is the choice of appropriate hyperparameters. We develop a Bayesian optimisation approach exploiting the iterative structure of training processes whose empiric performance exceeds that of existing baselines.

A final contribution of this thesis concerns increasing the scalability and expressiveness of Gaussian Processes (GPs). While we make no direct use of the presented framework, GPs have been used to model probabilistic beliefs in closely related settings.

Towards data-efficient deployment of reinforcement learning systems



Sebastian Schulze
St. Anne's College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

Hillary 2021

Acknowledgements

First and foremost, I would like to express my sincerest gratitude to my supervisor Prof. Michael A. Osborne. He has been a constant source of support and feedback without which this thesis would not have been possible. It is safe to say that our stimulating discussions not only inspired my past and future research but have left me a (more) Bayesian person.

I would equally like to thank my second advisor Prof. Shimon Whiteson for jumping in to take over my supervision during Michael's absence and welcoming me into his research group. His introductions and subsequent guidance were invaluable in shaping and focusing my work during this time.

I am grateful to all members of the MLRG & BXL research groups as well as my various collaborators for the thought-provoking discussions, feedback, and creating such a great and friendly research environment. Equally, I thank my friends in the Jiu Jitsu and Judo societies for providing a distraction and outlet in any stressful situation and preserving my insanity.

Finally and most of all, I would like to thank my family for their continuing love and support.

Abstract

A fundamental concern in the deployment of artificial agents in real-life is their capacity to quickly adapt to their surroundings. Traditional reinforcement learning (RL) struggles with this requirement in two ways. Firstly, iterative exploration of unconstrained environment dynamics yields numerous uninformative updates and consequently slow adaptation. Secondly, final policies have no capacity to adapt to future observations and have to either slowly learn indefinitely or retrain entirely as observations occur.

This thesis explores two formulations aimed at addressing these issues. The consideration of entire task distributions in meta-RL evolves policies quickly adapting to specific instances on their own. By forcing agents to specifically request feedback, Active RL enforces selective observations and updates. Both of these formulations reduce to a Bayes-Adaptive setting in which a probabilistic belief over possible environments is maintained. Many existing solutions only provide asymptotic guarantees that are of limited use in practical contexts. We develop a variational approach to approximate belief management and support its validity empirically through a broad range of ablations. We then consider recently successful planning approaches but uncover and discuss obstacles in their application to the discussed settings.

An important factor influencing the data requirements and stability of RL systems is the choice of appropriate hyperparameters. We develop a Bayesian optimisation approach exploiting the iterative structure of training processes whose empiric performance exceeds that of existing baselines.

A final contribution of this thesis concerns increasing the scalability and expressiveness of Gaussian Processes (GPs). While we make no direct use of the presented framework, GPs have been used to model probabilistic beliefs in closely related settings.

Contents

List of Figures	viii
List of Tables	x
List of Common Abbreviations	xi
1 Introduction	1
1.1 Efficient adaptation through Bayes-adaptivity	2
1.2 Thesis outline	5
2 Background	6
2.1 Markov Decision Processes	6
2.2 Value-based Algorithms	7
2.3 Policy-based Algorithms	8
2.4 Model-based Algorithms	9
2.5 Generalisation	9
2.6 Exploration-Exploitation Trade-off	10
3 A Very Good Method for Bayes-Adaptive Deep RL via Meta-Learning	11
3.1 Abstract	14
3.2 Introduction	14
3.3 Background	18
3.3.1 Training Setup	18
3.3.2 Bayesian Reinforcement Learning	19
3.4 Bayes-Adaptive Deep RL via Meta-Learning	21
3.4.1 Approximate Inference	22
3.4.2 Training Objective	23
3.4.3 Meta Training	24
3.4.4 Meta testing	26
3.5 Related Work	26
3.6 Empirical Evaluation	31
3.6.1 Gridworld	31

3.6.2	Sparse 2D Navigation	33
3.6.3	Mujoco Continuous Control Meta-Learning Tasks	34
3.6.4	Meta-World	36
3.7	Empirical Analysis	38
3.7.1	Modelling Horizon	38
3.7.2	KL Regularisation	40
3.7.3	Belief Dimensionality	41
3.8	Conclusion & Future Work	42
4	Active Reinforcement Learning	45
4.1	Abstract	48
4.2	Introduction	48
4.3	ARL: Definition and Illustration	49
4.4	Related work	50
4.4.1	Bayesian RL	52
4.4.2	BAMCP: MCTS for Bayesian RL	53
4.4.3	AlphaZero: Planning with function approximation	54
4.5	Reduction to BAMDPs	55
4.5.1	Discussion of existing BRL approaches	56
4.6	A planning approach to ARL	57
4.6.1	BAMCP fails in ARL problems	59
4.7	Alpha-style planning for ARL	61
4.7.1	Empirical observations and analysis	61
4.8	Discussion	64
5	Bayesian Optimisation for Iterative Learning	67
5.1	Abstract	70
5.2	Introduction	70
5.3	Related Work in Iteration-Efficient Bayesian Optimisation	72
5.4	Bayesian Optimisation for Iterative Learning (BOIL)	73
5.4.1	Selecting points using iteration-efficient modelling	74
5.4.2	Training curve compression and estimating the transformation function	75
5.4.3	Augmenting the training data	77
5.5	Experiments	79
5.5.1	Model illustration	80
5.5.2	Ablation study: curve compression	81
5.5.3	Tuning deep reinforcement learning and CNN	82
5.6	Conclusion and Future work	83

6	Marginalising Stationary Kernels across Spectral Densities	86
6.1	Abstract	89
6.2	Introduction	89
6.3	Background	91
6.3.1	Gaussian Processes	91
6.3.2	Bayesian Quadrature	92
6.4	Our Method: MASKERADE	93
6.4.1	The generative model	93
6.4.2	Posterior Inference	95
6.4.3	Bayesian Quadrature	96
6.4.4	Computational Complexity	99
6.5	Related Work	100
6.5.1	Kernel Learning	100
6.5.2	Bayesian Quadrature	101
6.5.3	Gaussian Processes on Spaces of Measures	102
6.6	Results	102
6.6.1	Experiment setup	103
6.6.2	Medium scale data sets	103
6.6.3	Large scale data sets	105
6.6.4	Ablation Study	105
6.7	Discussion	106
7	Concluding remarks	110

Appendices

A	Supplementary Material to Chapter 3	113
A.1	Full ELBO derivation	113
A.2	Experiments: Gridworld	113
A.2.1	Additional Remarks	113
A.2.2	Comparison to RL ²	114
A.3	Experiments: MuJoCo	114
A.3.1	Learning Curves	116
A.3.2	CheetahDir Test Time Behaviour	116
A.3.3	Latent Space Visualisation	117
A.3.4	Runtime Comparison	118
A.3.5	Ablation Study: Backpropagating the RL loss Through the Encoder	119
A.4	Experiments: Meta-World	120
A.5	Hyperparameters	120

B	Supplementary Material to Chapter 5	124
B.1	GP kernels and treatment of GP hyperparameters	124
B.2	Experiment settings	125
B.3	Algorithm Illustration and Further Experiments	128
B.3.1	Ablation Study using a Freeze-Thaw Time Kernel	130
B.3.2	Additional Experiments for Tuning DRL and CNN	130
C	Supplementary Material to Chapter 6	132
C.1	Summary of WSABI	132
C.2	A Remark on Posterior Inference (6.4)	134
C.2.1	Derivation of Information Theoretic Sample Acquisition . . .	134
C.2.2	Monte Carlo Approximation to the Kernel Integrals	135
C.3	Full Algorithm Description	135
C.4	Qualitative Analysis	137
	References	139

List of Figures

3.1	Illustration of different exploration strategies	16
3.2	VariBAD architecture	23
3.3	Behaviour of VariBAD in the gridworld environment	32
3.4	Meta-test performance on the Sparse 2D Navigation environment	33
3.5	Meta-test performance on MuJoCo environments	35
3.6	Ablation: Decoding targets – performance comparison	38
3.7	Ablation: Decoding targets – visualisation	39
3.8	Comparison of VAE regularisation penalties	40
3.9	Ablation: VAE regularisation - performance comparison	40
3.10	Ablation: VAE regularisation - visualisation	40
3.11	Ablation: Number of latent features	42
4.1	Didactic environments for ARL	51
4.2	Visualisation of MCTS search trees	58
4.3	Empirical investigation of BAMCP search trees	60
4.4	Performance of alpha-style planning in ARL	62
5.1	BOIL training curve scoring	75
5.2	GP posteriors of dense and sparsified data	77
5.3	Deterioration of covariance conditioning development over time	77
5.4	Ablation: Data inclusion over time	80
5.5	Illustration of elicited training progressions	81
5.6	Comparison of algorithm on hyperparameter tuning tasks	82
6.1	Bayesian Network of MASKERADE	94
A.1	Meta-test performance on gridworld	115
A.2	Learning curves for MuJoCo domains	115
A.3	Learning curves for ablation study – number of latent features	116
A.4	Test-time behaviour on HalfCheetahDir task	117
A.5	Test-time latent space visualisation for the HalfCheetahDir tasks	118
A.6	Learning curves for ablation study – loss backpropagation	120
A.7	Learning curves for the different ML1 Meta-World tasks	121

B.1	Illustration of BOIL point selection	126
B.2	Illustration of BOIL point selection without augmented observations	127
B.3	Learned hyperparameter scoring functions	129
B.4	Additional performance comparisons	129
B.5	Comparison to a FreezeThaw time kernel.	130
C.1	Schematic representation of inference in MASKERADE	136
C.2	Visualisation of kernel recovery	137
C.3	Visualisation of learned posteriors	138

List of Tables

3.1	Meta-test success rates on the ML1 Meta-World benchmark	37
6.1	Test set RMSE for the Solar Irradiance Data Set	104
6.2	Test set log likelihood and RMSE for UCI MLR datasets	104
6.3	Test set MSE on two UCI data sets	105
6.4	Test set log likelihood and RMSE on large-scale datasets	106
6.5	Ablation performance on Airline Passenger and Mauna Loa datasets	107
A.1	VariBAD hyperparameters for MuJoCo experiments	122
A.2	VariBAD hyperparameters for MuJoCo experiments (cont'd)	123
B.1	Training results for DDQN on CartPole	125
B.2	Training results for A2C on Reacher and InvertedPendulum	128
B.3	Training results for for CNN tuning	128
B.4	Further DRL agent specifications	128
C.1	Sensitivity of Monte Carlo sampling approximations	135

List of Common Abbreviations

ARL	Active Reinforcement learning
BAMDP	Bayes-Adaptive Markov Decision Process
BRL	Bayesian Reinforcement learning
BO	Bayesian Optimisation
BQ	Bayesian Quadrature
CNN	Convolutional Neural Network
DL	Deep Learning
DRL	Deep Reinforcement Learning
GMM	Gaussian Mixture Model
GP	Gaussian Process
MCMC	Markov Chain Monte Carlo
MCTS	Monte Carlo Tree Search
MDP	Markov Decision Process
ML	Machine Learning
MLE	Maximum Likelihood estimate
RL	Reinforcement Learning
RNN	Recurrent Neural Network
SM	Spectral Mixture
VAE	Variational Auto-Encoder

1

Introduction

Contents

1.1	Efficient adaptation through Bayes-adaptivity	2
1.2	Thesis outline	5

A fundamental goal of artificial intelligence research is the design of agents¹ independently working towards the resolution of tasks allocated to them. Not only need such agents be capable of reliably discharging their duties but they should also integrate as seamlessly as possible into existing workflows and routines. A key requirement for this is an ability to efficiently explore their surroundings and adapt their behaviour accordingly – all the while minimising requirements for external guidance and supervision.

The problem of learning from interaction is studied in the field of reinforcement learning (RL). In its most basic incarnation, RL considers an agent manipulating the state of the environment it is situated in. This agent’s behaviour is informed by two sources of feedback: its perception of state changes and an external reward signal locally characterising the agent’s task. Due to the absence of further guidance, successful RL agents have to identify globally optimal policies through

¹An agent in this context refers to any autonomous decision-making entity such as autonomous cars or robots but also abstract computer programs like recommendation systems.

careful exploration of untested courses of action and exploitation of the observed dynamics. In recent years, much attention has been paid to extensions building on advances in deep learning. These have led to a significant increase in the complexity of learned behaviours and astounding progress on challenging tasks in high-dimensional environments such as game play [126, 171, 172] and simulated robotic control [77, 114].

Despite their increased flexibility and expressiveness, deep reinforcement learning (DRL) approaches are not without problems of their own. Their complexity often makes them brittle with respect to hyperparameter choices; suitable settings being hard to identify and unfavourable ones producing unstable policies or even divergent training. Additionally, deep learning (DL) models typically make only minute updates per observation. Combined with common heuristic or random exploration strategies, this requires agents to interact with problem instances for prolonged periods, until salient points are encountered sufficiently often for desired behaviours to emerge. In the standard RL setup this induces high demands for (potentially human) supervision in the form of reward labels. Consequently, there is an on-going research interest in how to best overcome these problems and transfer DRL advances from idealised research setups (e.g. simulations) to practical problems of interest.

A promising way of dealing with the above issues is a Bayes-Adaptive formulation which explicitly models uncertainty about environment dynamics. This enables agents to reason about different aspects of their environment and their importance for the task at hand. Consequently, it has been shown to enable agents to optimally trade off further exploration and exploitation of the current best policy estimate. In the next section we formalise this approach and contrast it to the prevalent MDP formulation before detailing our own contributions in Section 1.2.

1.1 Efficient adaptation through Bayes-adaptivity

To better illustrate some of the limitations popular RL algorithms face in practical settings, we will sketch some of the standard formalisms, as well as the Bayes-

Adaptive formulation. A more detailed review of RL concepts, notation and methodology is provided in Chapter 2.

The central object of study in RL are Markov Decision Processes[149]. These naturally describe an agent’s environment via a set of states and admissible actions and time-independent transition and reward functions. At any given time step, the latter generate the successive state and reward solely dependent on the current state and agent’s choice of action. Given a full MDP description, it is possible to (approximately) compute the optimal policy via Dynamic Programming[18]. The challenge arises from the agent’s knowledge of the dynamics being incomplete, creating uncertainty about the optimal policy.

Instead of explicitly quantifying this (epistemic) uncertainty, many (deep) RL solutions focus on incrementally improving a “bootstrap” – a point estimate of the best policy, action values or environment dynamics. In doing so they lose access to the main measure of training progress and resign themselves to the use of universally applicable heuristics that control the speed of adaptation to new observation. They can also no longer distinguish between visited and unknown parts of the state space and are unable to formulate future goals for targeted exploration. The randomised exploration strategies they are employing instead follow a more “blanket” style approach by assuring that all possible trajectories will eventually be visited. Consequently, they can take a long time before visiting all states relevant to the optimal policy.

This laxness stems from the assumption that no or minimal prior knowledge about the environment is available. Given the independence assumptions underlying the MDP formulation even a full probabilistic update could only draw limited conclusions regarding the globally optimal policy and require numerous interactions as well. Thus, the decreased sample-efficiency becomes an acceptable price to pay in exchange for the computational savings of not having to maintain and update expensive probabilistic models.

This information void however is hardly reflective of many practical settings in which we may want to deploy machine learning systems. In many conceivable

applications, ranging from domestic robotics to dialogue and recommender systems, agents will have a fairly specific idea of the task they are to solve and exploration should focus on the particularities of specific task instances (e.g. relevant locations in a flat most likely to need cleaning or key preferences of the person to be advised).

When such prior information is available or can potentially be acquired in a pre-deployment phase, the Bayes-Adaptive formulation can mitigate the problems traditional (deep) RL agents would face. It considers a fully probabilistic model of environment dynamics in which uncertainty directly relates to the agent’s uncertainty regarding optimal behaviour. In the past, the Bayes-adaptive formulation has largely been studied as a means of achieving optimal exploration. By reasoning about the expected effect of possible observations on the model estimate and thereby policy uncertainty, Bayesian reinforcement learning (BRL) agents can identify worthwhile exploration goals whilst dismissing unknown but ultimately irrelevant targets. In fact, this approach has been shown to theoretically provide the decision-theoretically-optimal exploration strategy for any given prior.

Beyond this, we note that the Bayesian approach provides a natural way of equipping the agent with available information about environment dynamics (via specification of appropriate priors) and existing correlations between transitions and rewards (via specification of corresponding likelihood updates). Particularly the latter is of great interest, as it potentially increases the informativeness of individual observations and allows us to circumvent certain theoretical results of the basic MDP formulation that relate the worst-case sample complexity of any RL algorithm to the size of state and action spaces.

In this thesis, we investigate ways of leveraging and extending the potential of Bayesian Reinforcement Learning for designing agents that can adapt quickly and efficiently after their deployment on a new task.

1.2 Thesis outline

The remainder of this thesis is structured as follows:

- Chapter 2 provides an overview of core concepts in reinforcement learning relevant to the following discussions.
- Chapter 3 shows how the expressiveness of neural models can be harnessed to learn (approximate) Bayes-adaptive behaviour in a pre-training phase.
- Chapter 4 discusses Active Reinforcement Learning, an extension of BRL which forces the agents to select interactions of interest for which reward labels are specifically requested.
- Chapter 5 presents a Bayesian Optimisation framework for efficient and reliable hyperparameter tuning often crucial to the success of modern RL systems. The proposed approach exploits the iterative nature of reinforcement learning training but is more broadly applicable beyond it.
- Chapter 6 concerns a separate contribution aiming to expand the capacities of Gaussian Process models.

2

Background

Contents

2.1	Markov Decision Processes	6
2.2	Value-based Algorithms	7
2.3	Policy-based Algorithms	8
2.4	Model-based Algorithms	9
2.5	Generalisation	9
2.6	Exploration-Exploitation Trade-off	10

In this chapter, we provide some general background on several topics related to reinforcement learning. This is by no means intended as an exhaustive summary but merely provides some of the required models and additional context for the works discussed throughout this thesis. The remaining chapters are self-contained and include additional reviews pertaining to the respective problem settings discussed. We recommend the surveys by Sutton and Barto [187] and Ghavamzadeh et al. [65] as excellent starting points for an in-depth study of reinforcement learning and its Bayesian formulation respectively.

2.1 Markov Decision Processes

The central focus of reinforcement learning is an agent interacting with a Markov Decision Process [149] defined by a tuple $(S, A, \mathcal{T}, \mathcal{R}, \rho_0, H)$. Here, S denotes a set

of environment states, A a set of actions, $\mathcal{T}$ describes the environment dynamics $P(s_{t+1}|s_t, a_t)$ and $\mathcal{R}$ the agent's reward function $P(r_t|s_t, a_t)$. Starting at $s_0 \sim \rho_0$ the agent repeatedly selects an action according to its policy $\pi(a_t|s_t)$ and observes the reward r_t and new environment state s_{t+1} for a total of H steps. The objective is to identify the optimal policy π^* maximising one of several criteria, the most common one being the (discounted) cumulative reward or return:

$$J(\pi) = \mathbb{E}_{r_t, s_{t+1}, a_{t+1} \sim \mathcal{T}, \mathcal{R}, \pi|s_t, a_t} \left[\sum_{t=0}^H \gamma^t r_t \right] \quad (2.1)$$

Here, γ denotes a discount factor of future rewards which ensures bounded returns for $H = \infty$. Numerous variations of this formalisation exist, particular w.r.t. the interdependence of $\mathcal{T}$ and $\mathcal{R}$ as well as the optimisation criterion considered. However, excluding minor adjustments the character of the underlying solution remains unchanged. A common restriction is the consideration of policies stationary in time.

We generally assume an online learning setting in which the agent observes and updates on the data one interaction at a time, in contrast to offline settings in which the agent learns from a predetermined set of sampled trajectories.

Algorithms for reinforcement learning are commonly distinguished based on the quantities they explicitly represent. The most common classifications distinguish between value-, policy- and model-based approaches.

2.2 Value-based Algorithms

This group of algorithms is centred around the Q-function of a policy π which describes the expected discounted return of taking action a_t in state s_t and thereafter adhering to policy π :

$$Q^\pi(s_t, a_t) = \mathbb{E}_{r_i, s_{i+1}, a_{i+1} \sim \mathcal{T}, \mathcal{R}, \pi|s_t, a_t} \left[\sum_{i=t}^{\infty} \gamma^{i-t} r_i \right] \quad (2.2)$$

Recursive application of this definition gives rise to the Bellman equation for general policies and the Bellman optimality equations [18]:

$$\forall_{s \in S, a \in A} Q^\pi(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim \mathcal{R}, \mathcal{T} | s, a} [r + \gamma Q(s_{t+1}, \pi(s_{t+1}))] \quad (2.3)$$

$$\forall_{s \in S, a \in A} Q^{\pi^*}(s, a) = \mathbb{E}_{r_t, s_{t+1} \sim \mathcal{R}, \mathcal{T} | s, a} [r + \gamma \max_{a'} Q(s_{t+1}, a')] \quad (2.4)$$

Reinterpretation of these as update rules gives rise to so-called “Temporal Difference” (TD)-methods [186]. These continuously improve a “bootstrapped” estimate towards the true Q-function of a fixed target policy or the optimal policy. The former is known as an algorithm called SARSA [160] which performs *policy evaluation*. The latter is known as Q-Learning [206] and also leads to *policy improvement*. The data a Q-Learning agent updates on is not necessarily generated by the policy it evaluates which is known as *off-policy* learning.

2.3 Policy-based Algorithms

Policy-based methods directly parametrise an agent’s policy and optimise for the desired objective. This allows their application in continuous domains and can be done using standard black-box optimisers such as CMAES [75] or cross-entropy search [39] based on Monte Carlo sampling of returns.

More RL-specific are so-called policy-gradient methods centred around the principle of stochastic gradient descent. By using observed rewards to estimate a gradient of the return, the current policy is gradually improved. The most basic implementation of this idea is called REINFORCE [208].

To stabilise the approximate computation of gradients a model of the Q-function (or related quantities) can be useful. This gives rise to a union of value-based and policy-based methods called the actor-critic architecture. These implement a step-wise policy iteration in which the critic evaluates the current policy and the actor improves it based on the latest evaluation. Various formulations based on differing value targets and stabilisation methods following this principle have been proposed [188, 146, 170].

2.4 Model-based Algorithms

The final class of solution approaches explicitly models the environment dynamics $\mathcal{T}$ and $\mathcal{R}$. This often induces a two-step process in which sampled observations are first used to train a model before a policy is derived.

The former typically reduces to a standard supervised learning task and past research has investigated both questions of suitable model choices and learning objectives. The second step can, in principle, be solved by adapting any model-free algorithm and training a policy based on simulated experiences [185]. Alternatively, the ability to arbitrarily reset a model also allows the application of classical and heuristic planning approaches [33].

The freedom to almost arbitrarily combine model choices, learning objectives and approaches to policy computation gives rise to a rich array of methods which has been applied to a broad set of domains [200, 129].

2.5 Generalisation

Generalisation refers to an RL agent’s ability to share information across different states. This reduces the effective size of the space the agent needs to explore and significantly accelerates learning. Good generalisation capabilities are a prerequisite for dealing with large or continuous state domains in which exhaustive visitation of all states is impractical or infeasible.

In reinforcement learning this is achieved by describing quantities of interest (value functions, policies or environment dynamic) via appropriate function approximators. Their inclusion in standard RL methods is often non-trivial, as convergence towards optimal policies can no longer be guaranteed and different types of learning instabilities may occur. This is particularly pronounced in the “deadly triad” [15, 198] of function approximation, bootstrapping and off-policy learning and different remedies have been proposed [115, 197]

In recent years particular attention has been paid to the use of deep neural networks as they exhibit high degrees of flexibility, good generalisation capabilities

and (comparatively) low computational demands. Their adaption in standard reinforcement learning algorithms led to state-of-the-art results [126, 125, 165] and sparked a wave of rapid innovation of new RL algorithms [8, 58] and benchmarks [24, 195, 16].

2.6 Exploration-Exploitation Trade-off

A key difficulty of reinforcement learning is that the agent’s choice of policy is directly responsible for collecting data that forms the basis of future learning. At each step the agent has to decide whether or not to focus on actions known to lead towards (more) optimal returns or explore parts of the state space with unknown outcomes. Successful management of this “exploration-exploitation trade-off” is crucial to an algorithm’s performance as exploration will often include sub-optimal behaviour, but is necessary to make informed decisions at later stages.

Many contemporary RL algorithms “solve” this dilemma by noise-corrupting the current estimate of the optimal policy prior to action selection (ϵ -greedy and Boltzmann exploration). Other methods select exploratory actions based on their uncertainty regarding value estimates either by random sampling (Thompson sampling [192]) or via the addition exploration bonuses (e.g. upper confidence bounds [11]). A third alternative is given by novelty-seeking mechanisms like curiosity [144].

Arguably the most principled approach, however, is provided by adopting a Bayesian perspective which is a central point of discussion in Chapters 3 and 4.

3

VariBAD: A Very Good Method for Bayes-Adaptive Deep RL via Meta-Learning

Contents

3.1	Abstract	14
3.2	Introduction	14
3.3	Background	18
3.3.1	Training Setup	18
3.3.2	Bayesian Reinforcement Learning	19
3.4	Bayes-Adaptive Deep RL via Meta-Learning	21
3.4.1	Approximate Inference	22
3.4.2	Training Objective	23
3.4.3	Meta Training	24
3.4.4	Meta testing	26
3.5	Related Work	26
3.6	Empirical Evaluation	31
3.6.1	Gridworld	31
3.6.2	Sparse 2D Navigation	33
3.6.3	Mujoco Continuous Control Meta-Learning Tasks	34
3.6.4	Meta-World	36
3.7	Empirical Analysis	38
3.7.1	Modelling Horizon	38
3.7.2	KL Regularisation	40
3.7.3	Belief Dimensionality	41
3.8	Conclusion & Future Work	42

Preface

The following chapter proposes a new model for (deep) meta-reinforcement learning – a framework proposed in direct response to criticism regarding the learning efficiency and restricted flexibility of “standard” deep RL solutions [199]. Within it, agents are required to solve instances from a set of related problems, rather than only a single, fixed task. To optimally meet this challenge, it is necessary to quickly identify and adapt to peculiarities of any presented task whilst exploiting facets shared across the entire set. These commonalities can be learned during a training stage in which the agent has access to example instances. Adaptation, however, can only occur in an online fashion at test time and therefore has to be a feature of learned policies. This represents a significant increase in flexibility and complexity compared to traditional stationary RL policies.

Consequently, finding successful approaches to meta-RL represents a major step towards deployment of RL agents in practical scenarios. As mentioned previously, many applications of interest (e.g. domestic or industrial robotics) concern relatively specific duties which differ only in key situational and individual elements. Here, the meta-RL framework provides a natural division of an agent’s life-cycle into pre- and post-deployment learning phases with objectives matching our understanding of general conceptual design and user adaptation. It is worthwhile emphasising that nearly all task profiles that artificial agents may be assigned – regardless how seemingly diverse and uncorrelated – contain an underlying shared structure by virtue of the fact that they are set in the real-world. As humans, we like to pride ourselves in our capacity to learn quickly, but forget that this in large part builds on our subconscious understanding of the laws of physics, social concepts etc. which artificial agents initially have no access to.

A key observation in the design of meta-RL agents is that they are effectively facing a second sequential decision problem at test time – by probing their surroundings repeatedly and learning about the task identity, they are able to adjust their behaviour towards long-term optimality. Any (near-)optimal policy will therefore have to solve all issues encountered in conventional RL agent design –

the central one being the exploration-exploitation trade-off. Since meta-RL agents have the opportunity to derive a belief structure about task dynamics, we can interpret meta-RL and BRL as two sides of the same coin. Both aim to identify the optimal policy for a distribution of tasks (described as MDPs) – what differs is the interpretation and specification of said distribution. While BRL traditionally considers a subjective belief explicitly described as a probabilistic model, meta-RL provides implicit characterisation of a true initial task distribution via samples.

Our framework aims to draw on BRL results, to inform the design of meta-RL agents. Recognising that the optimal policy is stationary once conditioned on the updated belief, our framework enables the application of any of a number of well-established (deep) RL algorithms to train the final policy. To be able to condition on beliefs, however, it is necessary to derive a sufficient statistic for our prior as well as a mechanism for updating it in response to observations. For this we fall back on recurrent, variational models from the deep unsupervised learning literature which allow us to compress observation sequences into distributions over a succinct set of features.

The resulting variBad algorithm opens a path to tractable learning of approximate Bayes-optimal exploration in meta-learning scenarios.

In contrast to approaches from the BRL literature, the combination of variational belief learning and model-free reinforcement learning is no longer bound by pre-specified parametric belief families. Instead a setup-specific belief mechanism is learned in a fully data-driven manner that does not require additional privileged training information (e.g. task labels). This mechanism can be applied in an online fashion, enabling variBad to adapt much quicker to new task instances at test time than previous meta-learning algorithms.

Subsequent work has further increased training efficiency of the proposed framework by equipping it with more targeted meta-exploration capabilities of belief spaces[222] as well as the ability to learn from pre-collected data in an offline fashion [43].

3.1 Abstract

Trading off exploration and exploitation in an unknown environment is key to maximising expected online return during learning. A Bayes-optimal policy, which does so optimally, conditions its actions not only on the environment state but also on the agent’s uncertainty about the environment. Computing a Bayes-optimal policy is however intractable for all but the smallest tasks. In this paper, we introduce variational Bayes-Adaptive Deep RL (variBAD), a way to meta-learn approximately Bayes-optimal policies for complex tasks. variBAD jointly meta-learns a variational auto-encoder to perform approximate inference, and a policy that incorporates task uncertainty directly during action selection by conditioning on both the environment state and the approximate belief. In two toy domains, we illustrate how variBAD performs structured online exploration as a function of task uncertainty. We further evaluate variBAD on MuJoCo tasks widely used in meta-RL and show that it achieves higher online return than existing methods. On the recently proposed Meta-World ML1 benchmark, variBAD achieves state of the art results by a large margin, fully solving two out of the three ML1 tasks for the first time.

3.2 Introduction

Reinforcement learning (RL) is typically concerned with finding an *optimal policy* that maximises expected return for a given Markov decision process (MDP) with an unknown reward and transition function. If these were known, the optimal policy could in theory be computed without environment interactions. By contrast, learning in an *unknown* environment usually requires trading off exploration (learning about the environment) and exploitation (taking promising actions). Balancing this trade-off is key to maximising expected return *during learning*, which is desirable in many settings, particularly in high-stakes real-world applications like healthcare and education [116, 217]. A *Bayes-optimal policy*, which does this trade-off optimally, conditions actions not only on the environment state but on the agent’s own uncertainty about the current MDP.

In principle, a Bayes-optimal policy can be computed using the framework of Bayes-adaptive Markov decision processes (BAMDPs) [121, 49], in which the agent maintains a belief distribution over possible environments. Augmenting the state space of the underlying MDP with this belief yields a BAMDP, a special case of a belief MDP [92]. A Bayes-optimal agent maximises expected return in the BAMDP by systematically seeking out the data needed to quickly reduce uncertainty, but only insofar as doing so helps maximise expected return. Its performance is bounded from above by the optimal policy for the given MDP, which does not need to take exploratory actions but requires prior knowledge about the MDP to compute.

Unfortunately, planning in a BAMDP, i.e., computing a Bayes-optimal policy that conditions on the augmented state, is intractable for all but the smallest tasks. A common shortcut is to rely instead on *posterior sampling* [192, 181, 139]. Here, the agent periodically samples a single hypothesis MDP (e.g., at the beginning of an episode) from its posterior, and the policy that is optimal *for the sampled MDP* is followed until the next sample is drawn. Planning is far more tractable since it is done on a regular MDP, not a BAMDP. However, posterior sampling’s exploration can be highly inefficient and far from Bayes-optimal.

Consider the example of a gridworld in Figure 3.1, where the agent must navigate to an *unknown* goal located in the grey area (3.1a). To maintain a posterior, the agent can uniformly assign non-zero probability to cells where the goal could be, and zero to all other cells. A Bayes-optimal strategy strategically searches the set of goal positions that the posterior considers possible, until the goal is found (3.1b). Posterior sampling by contrast samples a possible goal position, takes the shortest route there, and then resamples a different goal position from the updated posterior (3.1c). Doing so is much less efficient since the agent’s uncertainty is not reduced optimally (e.g., states are revisited unnecessarily).

As this example illustrates, Bayes-optimal policies can explore much more efficiently than posterior sampling. A key challenge is to learn approximately Bayes-optimal policies while retaining the tractability of posterior sampling. In

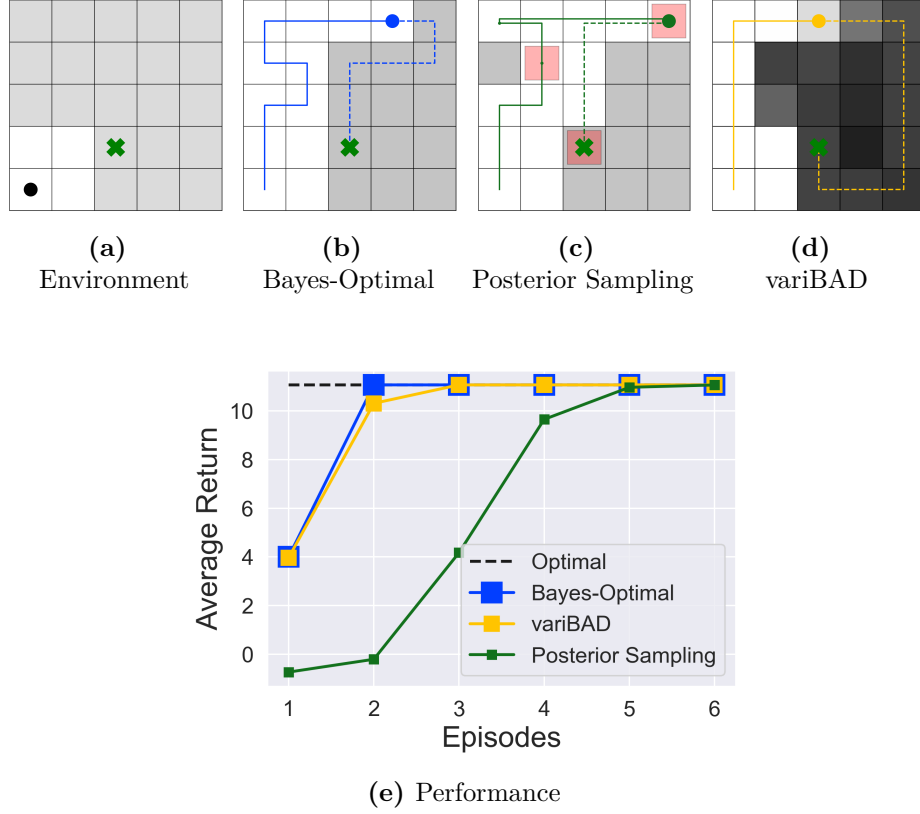


Figure 3.1: Illustration of different exploration strategies. **(a)** Environment: The agent starts at the bottom left and has to navigate to an unknown goal, located in the grey area. **(b)** A Bayes-optimal exploration strategy that systematically searches possible grid cells to find the goal, shown in solid (past actions) and dashed (future actions) blue lines. A simplified posterior over possible goal locations is shown in the background in grey ($p = 1/(\text{number of possible goal positions left})$) of containing the goal) and white ($p = 0$). **(c)** Posterior sampling, which repeatedly samples a possible goal position (red squares) from the current posterior, takes the shortest route there, and then updates its posterior. **(d)** Exploration strategy learned by variBAD. The grey background represents the approximate posterior the agent has learned. The predictions are not perfectly calibrated, and the model generally predicts slightly higher probabilities, shown by the darker shade compared to (b) **(e)** Average return over all possible environments, over six episodes with 15 steps each (after which the agent is reset to the starting position). VariBAD results are averaged across 20 random seeds. The performance of any exploration strategy is bounded above by the optimal behaviour (of a policy with access to the true goal position). The Bayes-optimal agent matches this behaviour from the second episode, whereas posterior sampling needs six rollouts. VariBAD closely approximates Bayes-optimal behaviour in this environment.

addition, the inference involved in maintaining a posterior belief, needed even for posterior sampling, may itself be intractable.

In this paper, we combine ideas from Bayesian RL, approximate variational inference, and meta-learning to tackle these challenges, and equip an agent with the ability to strategically explore unseen (but related) environments for a given distribution, in order to maximise its expected online return.

More specifically, we propose *variational Bayes-Adaptive Deep RL* (variBAD), a way to meta-learn to perform approximate inference on an unknown task,¹ and incorporate task uncertainty directly during action selection. Given a distribution over MDPs $p(M)$, we represent a single MDP M using a learned, low-dimensional stochastic latent variable m and jointly meta-train:

1. A variational auto-encoder that can infer the posterior distribution over m in a new task, given the agent’s experience, while interacting with the environment, and
2. A policy that conditions on this posterior belief over MDP embeddings, and thus learns how to trade off exploration and exploitation when selecting actions *under task uncertainty*.

Figure 3.1e shows the performance of our method versus the hard-coded optimal (with privileged goal information), Bayes-optimal, and posterior sampling exploration strategies. VariBAD’s performance closely matches the Bayes-optimal one, matching optimal performance from the third rollout.

We evaluate our approach on two toy domains, the GridWorld shown above and a 2D PointRobot Navigation task, which illustrate how variBAD maximises expected online return using the meta-learned approximate belief. We further evaluate variBAD on the widely used MuJoCo meta-RL benchmarks, and show that variBAD exhibits superior exploratory behaviour at test time compared to existing methods, achieving higher returns during learning. Lastly, on the recently proposed challenging Meta-World ML1 benchmark, variBAD achieves state of the art performance with a

¹We use the terms environment, task, and MDP, interchangeably.

large margin compared to existing methods, fully solving two out of the three ML1 benchmark tasks for the first time. As such, variBAD opens a path to tractable approximate Bayes-optimal exploration for deep reinforcement learning.

3.3 Background

We define a Markov decision process (MDP) as a tuple $M = (\mathcal{S}, \mathcal{A}, R, T, T_0, \gamma, H)$ with $\mathcal{S}$ a set of states, $\mathcal{A}$ a set of actions, $R(r_{t+1}|s_t, a_t, s_{t+1})$ a reward function, $T(s_{t+1}|s_t, a_t)$ a transition function, $T_0(s_0)$ an initial state distribution, γ a discount factor, and H the horizon. In the standard RL setting, we want to learn a policy π that maximises $\mathcal{J}(\pi) = \mathbb{E}_{T_0, T, \pi} \left[\sum_{t=0}^{H-1} \gamma^t R(r_{t+1}|s_t, a_t, s_{t+1}) \right]$, the expected return. Here, we consider a multi-task meta-learning setting, which we introduce next.

3.3.1 Training Setup

We adopt the standard meta-learning setting where we have a distribution $p(M)$ over MDPs from which we can sample during meta-training, with an MDP $M_i \sim p(M)$ defined by a tuple $M_i = (\mathcal{S}, \mathcal{A}, R_i, T_i, T_{i,0}, \gamma, H)$. Across tasks, the reward and transition functions vary but share some structure. The index i represents an unknown task description (e.g., a goal position or natural language instruction) or task ID. Sampling an MDP from $p(M)$ is typically done by sampling a reward and transition function from a distribution $p(R, T)$. During meta-training, batches of tasks are repeatedly sampled, and a small training procedure is performed on each of them, with the goal of learning to learn. At meta-test time, the agent is evaluated based on the average return it achieves *during learning*, for tasks drawn from p . Doing this well requires at least two things: (1) incorporating prior knowledge obtained in related tasks, and (2) reasoning about task uncertainty when selecting actions to trade off exploration and exploitation. In the following, we combine ideas from meta-learning and Bayesian RL to tackle these challenges.

3.3.2 Bayesian Reinforcement Learning

When the MDP is unknown, optimal decision making has to trade off exploration and exploitation when selecting actions. In principle, this can be done by taking a Bayesian approach to reinforcement learning formalised as a Bayes-Adaptive MDP (BAMDP), the solution to which is a Bayes-optimal policy [17, 49, 65].

In the Bayesian formulation of RL, we assume that the transition and reward functions are distributed according to a prior $b_0 = p(R, T)$. Since the agent does not have access to the true reward and transition function, it can maintain a belief

$$b_t(R, T) = p(R, T | \tau_{:t}), \quad (3.1)$$

which is the posterior over the MDP given the agent's experience up until the current timestep $\tau_{:t} = \{s_0, a_0, r_1, s_1, a_1, \dots, s_t\}$. This is often done by maintaining a distribution over the model parameters.

To allow the agent to incorporate the task uncertainty into its decision-making, this belief can be augmented to the state, resulting in hyper-states $s_t^+ \in \mathcal{S}^+ = \mathcal{S} \times \mathcal{B}$, where $\mathcal{B}$ is the belief space. These transition according to

$$\begin{aligned} T^+(s_{t+1}^+ | s_t^+, a_t, r_t) &= T^+(s_{t+1}, b_{t+1} | s_t, a_t, r_t, b_t) \\ &= T^+(s_{t+1} | s_t, a_t, b_t) T^+(b_{t+1} | s_t, a_t, r_t, b_t, s_{t+1}) \\ &= \mathbb{E}_{b_t} [T(s_{t+1} | s_t, a_t)] \delta(b_{t+1} = p(R, T | \tau_{:t+1})), \end{aligned} \quad (3.2)$$

i.e., the new environment state s_t is the expected new state w.r.t. the current posterior distribution of the transition function, and the belief is updated deterministically according to Bayes rule. The reward function on hyper-states is defined as the expected reward under the current posterior (after the state transition) over reward functions,

$$R^+(s_t^+, a_t, s_{t+1}^+) = R^+(s_t, b_t, a_t, s_{t+1}, b_{t+1}) = \mathbb{E}_{b_{t+1}} [R(s_t, a_t, s_{t+1})]. \quad (3.3)$$

This results in a BAMDP $M^+ = (\mathcal{S}^+, \mathcal{A}, R^+, T^+, T_0^+, \gamma, H^+)$ [49], which is a special case of a belief MDP, i.e., the MDP formed by taking the posterior beliefs maintained by an agent in a partially observable MDP and reinterpreting them as Markov

states [27]. In an arbitrary belief MDP, the belief is over a hidden state that can change over time. In a BAMDP, the belief is over the transition and reward functions, which are constant for a given task. The agent’s objective is now to maximise the expected return in the BAMDP,

$$\mathcal{J}^+(\pi) = \mathbb{E}_{b_0, T_0^+, T^+, \pi} \left[\sum_{t=0}^{H^+-1} \gamma^t R^+(r_{t+1} | s_t^+, a_t, s_{t+1}^+) \right], \quad (3.4)$$

i.e., maximise the expected return in an initially unknown environment, while learning, within the horizon H^+ . The MDP horizon H is distinct from the BAMDP horizon H^+ . Although they often coincide, we might instead want the agent to act Bayes-optimal within the first N MDP episodes, so $H^+ = N \times H$. Trading off exploration and exploitation optimally depends heavily on how much time the agent has left (e.g., to decide whether information-seeking actions are worth it).

The objective in (3.4) is maximised by the Bayes-optimal policy, which automatically trades off exploration and exploitation: it takes exploratory actions to reduce its task uncertainty *only insofar as it helps to maximise the expected return within the horizon*. The BAMDP framework is powerful because it provides a principled way of formulating Bayes-optimal behaviour. However, solving the BAMDP is hopelessly intractable for most interesting problems. The main challenges are as follows.

- We typically do not know how to parameterise the environment model,
- The belief update (computing the posterior $p(R, T | \tau_{:t})$) is often intractable,
- Even with the correct posterior, planning in belief space is typically intractable.

In the following, we propose a method that simultaneously meta-learns the reward and transition functions, how to perform inference in an unknown MDP, and how to use the belief to maximise expected online return. Since the Bayes-adaptive policy is learned end-to-end with the inference framework, no planning is necessary at test time. We make minimal assumptions (no privileged task information is required during training), resulting in a highly flexible and scalable approach to Bayes-adaptive Deep RL.

3.4 Bayes-Adaptive Deep RL via Meta-Learning

In this section, we present variBAD and describe how we tackle the challenges outlined above. We start by describing how to represent reward and transition functions, and (posterior) distributions over these. We then consider how to meta-learn to perform approximate variational inference in a given task, and finally put all the pieces together to form our training objective.

In the typical meta-learning setting, the reward and transition functions that are unique to each MDP are unknown, but also share some structure across the MDPs M_i in $p(M)$. We know that there exists a true i which represents either a task description or task ID, but we do not have access to this information. We therefore represent this value using a learned stochastic latent variable m_i . For a given MDP M_i we can then write

$$R_i(r_{t+1}|s_t, a_t, s_{t+1}) \approx R(r_{t+1}|s_t, a_t, s_{t+1}; m_i), \quad (3.5)$$

$$T_i(s_{t+1}|s_t, a_t) \approx T(s_{t+1}|s_t, a_t; m_i), \quad (3.6)$$

where R and T are shared across tasks. Since we do not have access to the true task description or ID, we need to *infer* m_i given the agent’s experience up to time step t collected in M_i ,

$$\tau_{:t}^{(i)} = (s_0, a_0, r_1, s_1, a_1, r_2, \dots, s_{t-1}, a_{t-1}, r_t, s_t), \quad (3.7)$$

i.e., we want to infer the posterior distribution $p(m_i|\tau_{:t}^{(i)})$ over m_i given $\tau_{:t}^{(i)}$ (from now on, we drop the sub- and superscript i for ease of notation).

Recall that our goal is to *learn a distribution over the MDPs, and given a posteriori knowledge of the environment compute the optimal action*. Given the above reformulation, it is now sufficient to reason about the embedding m , instead of the transition and reward dynamics. This is particularly useful when deploying deep learning strategies, where the reward and transition function can consist of millions of parameters, but the embedding m can be a small vector.

3.4.1 Approximate Inference

Computing the exact posterior is typically not possible: we do not have access to the MDP (and hence the transition and reward function), and marginalising over tasks is computationally infeasible. Consequently, we need to learn a model of the environment $p_\theta(\tau_{:H+}|a_{:H+-1})$, parameterised by θ , together with an amortised inference network $q_\phi(m|\tau_{:t})$, parameterised by ϕ , which allows fast inference at runtime *at each timestep* t . The action-selection policy is not part of the MDP, so an environmental model can only give rise to a distribution of trajectories when conditioned on actions, which we typically draw from our current policy, $a \sim \pi$. At any given time step t , our model learning objective is thus to maximise

$$\mathbb{E}_{\rho(M, \tau_{:H+})} [\log p_\theta(\tau_{:H+}|a_{:H+-1})], \quad (3.8)$$

where $\rho(M, \tau_{:H+})$ is the trajectory distribution induced by our policy and we slightly abuse notation by denoting by τ the state-reward trajectories, excluding the actions. In the following, we drop the conditioning on $a_{:H+-1}$ to simplify notation.

Instead of optimising (3.8), which is intractable, we can optimise a tractable lower bound, defined with a learned approximate posterior $q_\phi(m|\tau_{:t})$ which can be estimated by Monte Carlo sampling (for the full derivation see Appendix A.1):

$$\begin{aligned} \mathbb{E}_{\rho(M, \tau_{:H+})} [\log p_\theta(\tau_{:H+})] &\geq \mathbb{E}_\rho \left[\mathbb{E}_{q_\phi(m|\tau_{:t})} [\log p_\theta(\tau_{:H+}|m)] - KL(q_\phi(m|\tau_{:t})||p_\theta(m)) \right] \\ &= ELBO_t. \end{aligned} \quad (3.9)$$

The term $\mathbb{E}_q[\log p(\tau_{:H+}|m)]$ is often referred to as the reconstruction loss, and $p(\tau_{:t}|m)$ as the decoder. The term $KL(q(m|\tau_{:t})||p_\theta(m))$ is the KL-divergence between our variational posterior q_ϕ and the prior over the embeddings $p_\theta(m)$.

Instead of using the same fixed prior for each timestep and ELBO t , we set the prior to our previous posterior, $q_\phi(m|\tau_{:t-1})$, with initial prior $q_\phi(m) = \mathcal{N}(0, I)$. This is akin to a Bayesian filtering update, and works better empirically, as we confirm in Section 3.7.2. We take the gradient w.r.t. the entire term (i.e., through the current and previous approximate posterior).

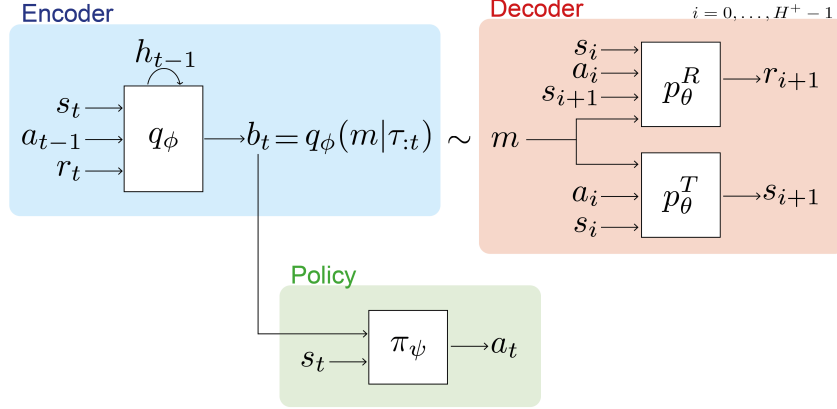


Figure 3.2: variBAD architecture: A trajectory of states, actions and rewards is processed online using an RNN to produce the posterior over task embeddings, $q_\phi(m|\tau_{:t})$. The posterior is trained using a decoder which predicts past and future states and rewards from current states and actions. The policy conditions on the posterior in order to act in the environment and is trained using RL.

As can be seen in Equation (3.9) and Figure 3.2, when the agent is at timestep t , we encode the *past trajectory* $\tau_{:t}$ to get the current posterior $q(m|\tau_{:t})$ since this is all the information available to perform inference about the current task. We then decode the entire trajectory $\tau_{:H^+}$ *including the future*, i.e., model $\mathbb{E}_q[p(\tau_{:H^+}|m)]$. This is different from the conventional VAE setup (and possible since we have access to this information during training). Decoding not only the past but also the future is important because this way, variBAD learns to perform inference about unseen states given the past. The reconstruction term $\log p(\tau_{:H^+}|m)$ factorises as

$$\begin{aligned} \log p(\tau_{:H^+}|m, a_{:H^+-1}) &= \log p((s_0, r_0, \dots, s_{t-1}, r_{t-1}, s_t)|m, a_{:H^+-1}) \\ &= \log p(s_0|m) + \sum_{i=0}^{H^+-1} [\log p(s_{i+1}|s_i, a_i, m) + \log p(r_{i+1}|s_i, a_i, s_{i+1}, m)]. \end{aligned} \quad (3.10)$$

Here, $p(s_0|m)$ is the initial state distribution T'_0 , $p(s_{i+1}|s_i, a_i; m)$ the transition function T' , and $p(r_{i+1}|s_t, a_t, s_{i+1}; m)$ the reward function R' . From now on, we include T'_0 in T' for ease of notation.

3.4.2 Training Objective

We can now formulate a training objective for learning the approximate posterior distribution over task embeddings, the policy, and the generalised reward and

transition functions R' and T' . We use deep neural networks to represent the individual components:

1. The encoder $q_\phi(m|\tau_{:t})$, parameterised by ϕ ;
2. An approximate transition function $T' = p_\theta^T(s_{i+1}|s_i, a_i; m)$ and an approximate reward function $R' = p_\theta^R(r_{i+1}|s_t, a_t, s_{i+1}; m)$ which are jointly parameterised by θ ; and
3. A policy $\pi_\psi(a_t|s_t, q_\phi(m|\tau_{:t}))$ parameterised by ψ and dependent on ϕ .

The policy $\pi_\psi(a_t|s_t, q_\phi(m|\tau_{:t}))$ is conditioned on both the environment state and the posterior over m . This is similar to the formulation of BAMDPs introduced in Section 3.3.2, with the difference that we learn a unifying distribution over MDP embeddings, instead of the transition/reward function directly. This makes learning easier since there are fewer parameters to perform inference over, and we can use data from all tasks to learn the shared reward and transition function. The posterior can be represented by the distribution’s parameters (e.g., mean and standard deviation if q is Gaussian).

Our overall objective is to maximise

$$\mathcal{L}(\phi, \theta, \psi) = \mathbb{E}_{p(M)} \left[\mathcal{J}(\psi, \phi) + \lambda \sum_{t=0}^{H^+} ELBO_t(\phi, \theta) \right]. \quad (3.11)$$

3.4.3 Meta Training

During meta-training, we train the policy and the VAE using Equation (3.11).

In Equation (3.11), λ weights the supervised model learning objective against the RL loss. This is necessary since parameters ϕ are shared between the VAE and the policy. However, we found that backpropagating the RL loss through the encoder is typically unnecessary in practice. Not doing so speeds up training considerably, avoids the need to trade off these losses, and prevents interference between gradients of opposing losses. In ablation studies (see Appendix A.3.5) we found that backpropagating both losses through the encoder can marginally improve performance in some cases, but can be detrimental in others, and depends strongly

on the relative weighting between the VAE and the RL loss. In our experiments, we therefore optimise the policy and VAE alternately, with separate optimisers.

Meta-Training the VAE. The VAE is trained by approximating expectations in Equation (3.11) using Monte Carlo samples, and using the reparameterisation trick [97]. For $t = 0$, we use the prior $q_\phi(m) = N(0, I)$. We encode past trajectories using a recurrent network as in Duan et al. [48] and Wang et al. [199], but other types of encoders could be considered like the ones used in Zaheer et al. [220], Garnelo et al. [62], and Rakelly et al. [152]. The encoder outputs an approximate belief over the current task m , by predicting the mean and variance of a Gaussian distribution. The decoder is trained using samples from this approximate posterior, and by separately predicting the rewards and state transitions of the entire current trajectory (i.e., including future steps to which we have access during meta-training).

Equation (3.11) shows that the ELBO appears for all possible context lengths t . This way, variBAD can learn to perform inference online (while the agent is interacting with an environment), and decrease its uncertainty over time given more data. In practice, we may subsample a fixed number of ELBO terms for computational efficiency if H^+ is large.

Meta-Training the Policy. We meta-train the policy using proximal policy optimisation [164], but other methods can in principle be used, including off-policy methods [43]. The policy receives the approximate belief as an input in addition to the state, allowing it to meta-learn how to act approximately Bayes-optimal. This approximate belief, $q_\phi(m|\tau_{:t})$, is represented by two vectors: the mean and variance of a Gaussian distribution with diagonal covariance matrix. These two vectors are predicted by the encoder, and are concatenated with the environment state before being passed to the policy. In the default case, this is treated as part of the observation, and the RL loss is not backpropagated through the encoder.

We train the RL agent and the VAE using different data buffers: the policy is only trained with the most recent data since we use an on-policy algorithm

in our experiments; and for the VAE we maintain a separate, larger buffer of observed trajectories.

3.4.4 Meta testing

At meta-test time, we roll out the policy in randomly sampled test tasks to evaluate performance, by performing forward passes through the encoder (to compute the approximate belief) and the policy (to choose how to act given the state and approximate belief). The belief is updated after every step by feeding in the new (state, action, reward) tuple, allowing the agent to adapt online. The decoder is not used at test time, and no gradient updates are performed on the encoder or policy network: the agent has learned to act approximately Bayes-optimal during meta-training.

3.5 Related Work

Bayesian Reinforcement Learning. Bayesian methods for RL can be used to quantify uncertainty to support action-selection, and provide a way to incorporate prior knowledge into the algorithms (see [65] for a review). A Bayes-optimal policy is one that optimally trades off exploration and exploitation, and thus maximises expected return during learning. While such a policy can in principle be computed using the BAMDP framework, it is hopelessly intractable for all but the smallest tasks. Existing methods are therefore restricted to small and discrete state and action spaces [9, 69, 70], or a discrete set of tasks [25, 148]. VariBAD opens a path to tractable approximate Bayes-optimal exploration for deep RL by leveraging ideas from meta-learning and approximate variational inference, with the only assumption that we can meta-train on a set of related tasks. Existing approximate Bayesian RL methods often require us to define a prior or belief update on the reward and transition functions, and rely on (possibly expensive) sample-based planning procedures. Due to the use of deep neural networks however, variBAD lacks the formal guarantees enjoyed by some of the methods mentioned above.

Posterior sampling [181, 139], which extends Thompson sampling [192] from bandits to MDPs, estimates a posterior distribution over MDPs (i.e., model

and reward functions), in the same spirit as variBAD. This posterior is used to periodically sample a single hypothesis MDP (e.g., at the beginning of an episode), and the policy that is optimal *for the sampled MDP* is followed subsequently. This approach is less efficient than Bayes-optimal behaviour and therefore typically has lower expected return during learning.

A related approach for inter-task transfer of abstract knowledge is to pose policy search with priors as Markov Chain Monte Carlo inference [210]. Similarly Guez, Silver, and Dayan [70] propose a Monte Carlo Tree Search method for Bayesian planning for tractable, sample-based approximately Bayes-optimal behaviour. Osband, Aslanides, and Cassirer [138] note that non-Bayesian treatment for decision making can be arbitrarily suboptimal and propose a simple randomised prior based approach for structured exploration. Some recent deep RL methods use stochastic latent variables for structured exploration [72, 152], which gives rise to behaviour similar to posterior sampling. Other ways to use the posterior for exploration are, e.g., certain reward bonuses [100, 178] and methods based on optimism in the face of uncertainty [95, 23]. Non-Bayesian methods for exploration are often used in practice, such as other exploration bonuses (e.g., via state-visitation counts) or using uninformed sampling of actions (e.g., ϵ -greedy action selection). Such methods are prone to wasteful exploration that does not help maximise expected reward.

Related to BAMDPs are *contextual MDPs*, where the task description is referred to as a context, on which the environment dynamics and rewards depend [74, 89, 38, 128]. Research in this area is often concerned with developing tight bounds by putting assumptions on the context, such as having a small known number of contexts, or that there is a linear relationship between the contexts and dynamics/rewards. Similarly, the framework of hidden parameter (HiP-)MDPs assumes that there is a set of low-dimensional latent factors which define a family of related dynamical systems (with shared reward structure), similar to the assumption we make in Equation (3.6) [44, 96, 216]. These methods however do not directly learn Bayes-optimal behaviour but allow for a longer training period in new environments to infer the latents and train the policy.

Meta-Learning Bayes-Adaptive Policies It has been shown theoretically [137] and empirically [124] that meta-trained agents approximate Bayes-optimal agents on the given task distribution.

A prominent model-free meta-RL approach to this problem is to utilise the dynamics of recurrent networks for fast adaptation (RL², Wang et al. [199] and Duan et al. [48]). At every time step, the network receives an auxiliary input consisting of the preceding action and reward. This allows learning within a task to happen online, entirely in the dynamics of the recurrent network. If we remove the decoder (Fig 3.2) and the VAE objective (Eq (3.8)), variBAD reduces to this setting, i.e., the main differences are that we use a stochastic latent variable (an inductive bias for representing uncertainty) together with a decoder to reconstruct previous *and future* transitions and rewards (which acts as an auxiliary loss [86] to encode the task in latent space and deduce information about unseen states). [137] discuss meta-learning sequential strategies in depth as well as how to recast memory-based meta-learning within a Bayesian framework. While model-free meta-learning methods (RL², [199, 48]) can also meta-learn approximately Bayes-optimal policies [137, 124], but as we show empirically do not scale as well in terms of performance as variBAD. In addition, variBAD provides explicit belief representations, which can be used in downstream tasks or for auxiliary tasks [222].

Closely related to our approach is the work of Humplik et al. [84]. Like variBAD, they condition the policy on a posterior distribution over the MDP, which is meta-trained using privileged information such as a task description. In comparison, variBAD meta-learns to represent the belief in an unsupervised way, and does not rely on privileged task information during training.

Other Meta Reinforcement Learning Settings. Another popular approach to meta RL is to learn an initialisation of the model, such that at test time, only a few gradient steps are necessary to achieve good performance [57, 133]. These methods do not directly account for the fact that the initial policy needs to explore, a problem addressed, a.o., by Stadie et al. [180] (E-MAML) and Rothfuss et al. [158] (ProMP).

In terms of model complexity, MAML and ProMP are relatively lightweight, since they typically consist of a feed-forward policy. RL² and variBAD use recurrent modules, which increases model complexity but allows online adaptation. Other methods that perform gradient adaptation at test time are, e.g., Houthoofd et al. [83] who meta-learn a loss function conditioned on the agent’s experience that is used at test time so learn a policy (from scratch); and Sung et al. [184] who learn a meta-critic that can criticise any actor for any task, and is used at test time to train a policy. Compared to variBAD, these methods usually separate exploration (before gradient adaptation) and exploitation (after gradient adaptation) at test time by design, making them less sample efficient.

Skill and Task Embeddings. Learning (variational) task or skill embeddings for meta or transfer reinforcement learning is used in a variety of approaches. Hausman et al. [76] use approximate variational inference learn an embedding space of skills (with a different lower bound than variBAD). At test time the policy is fixed, and a new embedder is learned that interpolates between already learned skills. Arnekvist, Kragic, and Stork [7] learn a stochastic embedding of optimal Q -functions for different skills, and condition the policy on (samples of) this embedding. Adaptation at test time is done in latent space. Co-Reyes et al. [156] learn a latent space of low-level skills that can be controlled by a higher-level policy, framed within the setting of hierarchical RL. This embedding is learned using a VAE to encode state trajectories and decode states and actions. Zintgraf et al. [223] learn a deterministic task embedding trained similarly to MAML [57]. Similar to variBAD, [221] use learned dynamics and reward modules to learn a latent representation which the policy conditions on and show that transferring the (fixed) encoder to new environments helps learning. Perez, Such, and Karaletsos [145] learn dynamic models with auxiliary latent variables, and use them for model-predictive control. Lan et al. [107] learn a task embedding with an optimisation procedure similar to MAML, where the encoder is updated at test time, and the policy is fixed. [162] explicitly learn an embedding of the environment model, which

is subsequently used for model predictive control (and not, like in variBAD, for exploration). In the field of imitation learning, some approaches embed expert demonstrations to represent the task; e.g., Wang et al. [204] use variational methods and Duan et al. [47] learn deterministic embeddings.

variBAD differs from the above methods mainly in what the embedding represents (i.e., task uncertainty) and how it is used: the policy conditions on the posterior *distribution* over MDPs, allowing it to reason about task uncertainty and trade off exploration and exploitation online. Our objective (3.9) explicitly optimises for Bayes-optimal behaviour. Unlike some of the above methods, we do not use the model at test time, but model-based planning is a natural extension for future work.

Variational Inference and Meta-Learning. A main difference of variBAD to many existing Bayesian RL methods is that we meta-learn the inference procedure, i.e., how to do a posterior update. Apart from (RL) methods mentioned above, related work in this direction can be found, a.o., in Garnelo et al. [62], Gordon et al. [66], and Wu et al. [214]. By comparison, variBAD has an inference procedure tailored to the setting of Bayes-optimal RL, where the objective is to model trajectories in the environment.

POMDPs. Several deep learning approaches to model-free reinforcement learning [85] and model learning for planning [196] in partially observable Markov decision processes have recently been proposed and utilise approximate variational inference methods. variBAD by contrast focuses on BAMDPs [121, 49, 65], a special case of POMDPs where the transition and reward functions constitute the hidden state and the agent must maintain a belief over them. While in general the hidden state in a POMDP can change at each time-step, in a BAMDP the underlying task, and therefore the hidden state, is fixed per task. We exploit this property by learning an embedding that is *fixed* over time, unlike approaches like Igl et al. [85] which use filtering to track the changing hidden state. While we utilise the power of deep approximate variational inference, other approaches for BAMDPs often use

more accurate but less scalable methods, e.g., Lee et al. [111] discretise the latent distribution and use Bayesian filtering for the posterior update.

3.6 Empirical Evaluation

In this section we first investigate the properties of variBAD on the didactic gridworld domain from Figure 3.1. These results show that variBAD performs *structured* and *online* exploration as it infers the task.

We then evaluate variBAD on several more challenging domains: a sparse 2D navigation task, a range of tasks from the MuJoCo benchmark, and the Meta-World ML1 benchmark. On these, variBAD achieves state of the art performance. In Section 3.7, we show the results of several ablation studies to motivate our design choices and analyse how robust variBAD is to the size of the latent space.

The two main baselines we consider are RL² [48, 199] and PEARL [152]. RL² can be seen as a model-free version of variBAD. In principle, it can learn to act approximately Bayes-optimal by performing task inference in the recurrent state of the RNN. RL² only consists of an encoder and a policy, and trains both end-to-end using an RL loss only. PEARL is akin to posterior sampling, which is a sub-optimal exploration strategy in theory compared to Bayes-optimal exploration (see Figure 3.1), but has the advantage that computing the optimal policy for a single MDP (sample from the approximate posterior) is more tractable than doing so in a BAMDP.

Experimental details, hyperparameters, as well as additional results, can be found in the appendix. The source code is available at <https://github.com/lmzintgraf/varibad>.

3.6.1 Gridworld

To gain insight into variBAD’s properties, we start with a didactic gridworld environment. The task is to reach a goal (selected uniformly at random) in a 5×5 gridworld. The goal is unobserved by the agent, inducing task uncertainty and necessitating exploration. The goal can be anywhere except around the starting

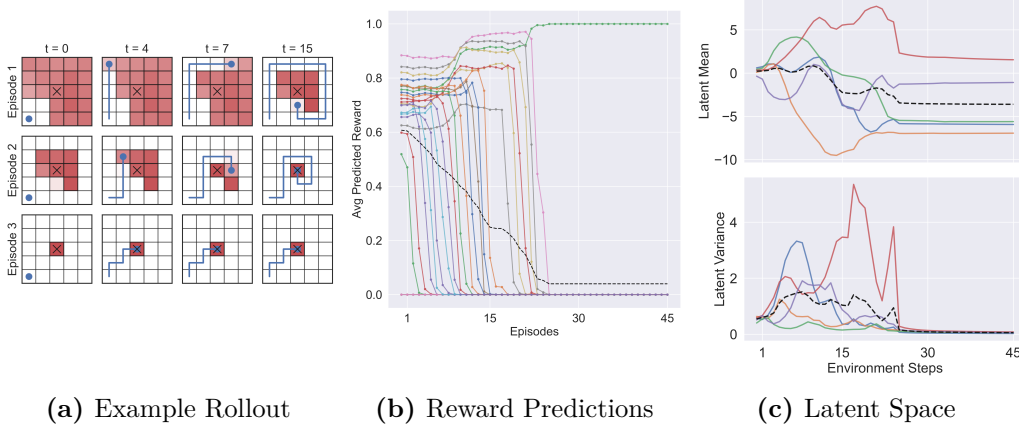
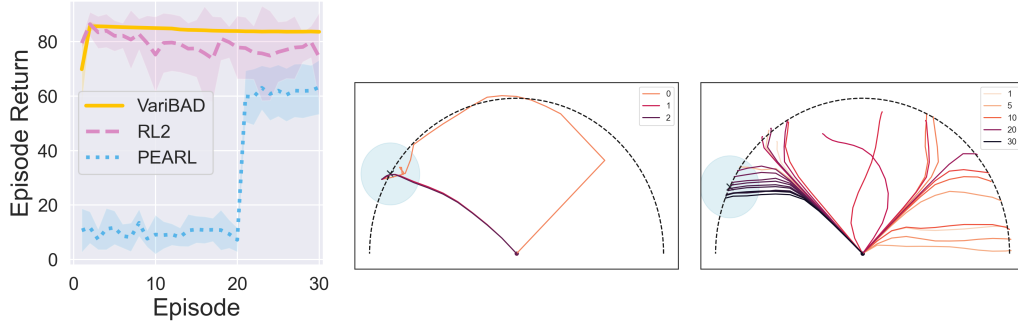


Figure 3.3: Behaviour of variBAD in the gridworld environment. **(a)** Hand-picked but representative example test rollout. The red background indicates the posterior probability of receiving a reward at that cell. **(b)** Probability of receiving a reward for each cell, as predicted by the decoder, over the course of interacting with the environment (average in black, goal state in green). **(c)** Visualisation of the latent space; each line is one latent dimension, the black line is the average.

cell, which is at the bottom left. Actions are: *up*, *right*, *down*, *left*, *stay* (executed deterministically), and after 15 steps the agent is reset. The horizon within the MDP is $H = 15$, but we choose a horizon of $H^+ = 4 \times H = 60$ for the BAMDP, i.e., we train our agent to maximise performance for 4 MDP episodes. The agent gets a sparse reward signal: -0.1 on non-goal cells, and $+1$ on the goal cell. The best strategy is to explore until the goal is found, and stay at the goal or return to it when reset to the initial position. We use a latent dimensionality of 5.

Figure 3.3 illustrates how variBAD behaves at test time with deterministic actions (i.e., all exploration is done by the policy). In Figure 3.3a we see how the agent interacts with the environment, with the red background visualising the posterior belief by using the learned reward function. variBAD learns the correct prior and adjusts its belief correctly over time. It predicts no reward for cells it has visited, and explores the remaining cells until it finds the goal. As the agent gathers more data, more and more cells are excluded ($p(\text{rew} = 1) = 0$, white cells), until eventually the agent finds the goal.

Figure 3.3b shows the reward predictions: each line represents a grid cell and its value the probability of receiving a reward at that cell. As the agent gathers more data, more and more cells are excluded ($p(\text{rew} = 1) = 0$), until



(a) Meta-Test Performance (b) VariBAD example rollout (c) PEARL example rollout

Figure 3.4: Meta-test performance on the Sparse 2D Navigation environment. (a): Performance averaged over the task distribution at the end of training. (b) and (c): Example rollouts of meta-trained variBAD (a) and PEARL (b) agents. PEARL does not optimise for optimal exploration, and thus it requires many more episodes to find the goal. On the other hand, variBAD optimises for optimal exploration, efficiently covering possible goal locations, and is able to quickly find the goal.

eventually the agent finds the goal. At this point, the predictions correspond to the true reward function. In Figure 3.3c we visualise the 5-dimensional latent space (mean and variance in separate sub-plots). We see that once the agent finds the goal, the posterior concentrates: the variance drops close to zero, and the mean converges to a fixed value.

As we saw in Figure 3.1e, the behaviour of variBAD closely matches that of the Bayes-optimal policy, which optimally trades off exploration and exploitation in an unknown environment, and outperforms posterior sampling. Our results on this gridworld indicate that variBAD is an effective way to approximate Bayes-optimal control, and has the additional benefit of giving insight into the task belief of the policy.

3.6.2 Sparse 2D Navigation

We evaluate on a Point Robot 2D navigation task used by Gupta et al. [72], Rakelly et al. [152], and Humplik et al. [84], to further illustrate how variBAD performs online adaptation. The agent must navigate to an unknown goal sampled along the border of a semicircle of radius 1.0, and receives a reward relative to its proximity to the goal when it is within a goal radius of 0.2.

Figure 3.4a shows the average performance of PEARL, RL^2 , and variBAD at test time, when rolling out the agent for 30 episodes in a single task. VariBAD adapts much faster to the task compared to PEARL. RL^2 also adapts rapidly but is less stable compared to variBAD when rolled out for a much longer time than during training: both variBAD and RL^2 were trained to perform well over three consecutive episodes.

To shed light on the performance difference between variBAD and PEARL, Figures 3.4b and 3.4c visualise representative example rollouts for meta-trained variBAD and PEARL agents. We picked examples where the target goals are at the end of the semi-circle, which we found are most difficult for the agent. PEARL performs posterior sampling for exploration which means it is restricted to a fixed hypothesised goal position during each rollout. On the other hand, variBAD (Figure 3.4b) strategically explores the space within an episode to find the goal, which is more efficient. Once the goal is found, both variBAD and PEARL are able to quickly return to it.

The two toy experiments, GridWorld and PointRobot, illustrate how variBAD makes decisions: it adapts to the task *online* while updating the approximate belief, which allows it to rapidly adapt to new tasks. In the following sections, we test variBAD on more challenging meta-RL benchmarks, MuJoCo and Meta-World ML1.

3.6.3 Mujoco Continuous Control Meta-Learning Tasks

We show that variBAD can scale to more complex meta learning settings by employing it on MuJoCo [195] locomotion tasks commonly used in the meta-RL literature.² We consider the AntDir, HalfCheetahDir and Humanoid environments where the agent has to run either forwards or backwards (i.e., there are only two tasks), the HalfCheetahVel environment where the agent has to run at different velocities, the Ant-Goal environment where the agent has to navigate to an initially unknown goal position; and the Walker environment where the system parameters are randomised. The rewards in these environments are dense, so that in principle

²Environments taken from <https://github.com/katerakelly/oyster>.

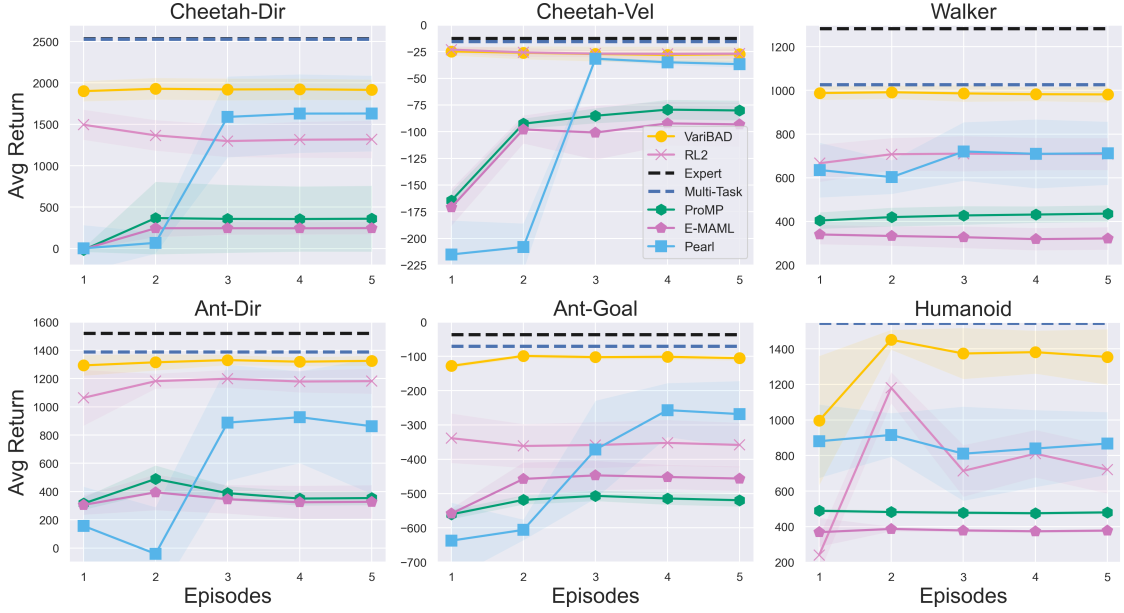


Figure 3.5: Average test performance on six different MuJoCo environments, trained separately with 10 seeds per MuJoCo environment per method. The meta-trained policies are rolled out for 5 episodes to show how they adapt to the task. Values shown are averages across tasks (95% confidence intervals shaded). Being an online adaptation method, variBAD adapts within the first episode. It outperforms other methods, even when these are given longer than one episode to adapt, and even though the first episode includes exploratory actions. RL^2 is also an online adaptation method and can adapt to the task within the first episode. On most environments, variBAD outperforms RL^2 [48, 199] significantly. The other methods, PEARL [152], E-MAML [180], and ProMP [158], need at least one episode to adapt by design.

the agent only needs a few exploratory actions to infer the task by observing the rewards it receives.

VariBAD and RL^2 were trained to maximise performance within two episodes (mainly so that we can roll them out for multiple episodes at test time; we are primarily interested in their adaptation behaviour within the first episode). PEARL [152], was trained using the open sourced code³, to maximise performance after 3-5 episodes, depending on the environment. E-MAML [180] and ProMP [158] were trained using the open sourced code by Rothfuss et al. [158]⁴, to maximise performance after 1 gradient step on 10-20 rollouts. To get an approximate upper bound on performance, we train a multi-task agent which conditions on a task

³The implementation which we used for our PEARL experiments was published by Rakelly et al. [152] and can be found at <https://github.com/katerakelly/oyster>.

⁴The implementation which we used for our E-MAML and ProMP experiments was published by Rothfuss et al. [158] and can be found at <https://github.com/jonasrothfuss/ProMP>.

descriptor, and an ensemble of expert agents (one per task) whose performances are averaged, using PPO.

Figure 3.5 shows the average performance at test time across different tasks. While we show the return of the agent during the first five rollouts, we emphasise that our primary interest lies in the agent’s performance during the first episode, while learning about the environment, which tells us how well the agent can trade off exploration and exploitation.

Only variBAD and RL² are able to adapt to the task at hand within a single episode. VariBAD outperforms RL² in all environments except HalfCheetah-Vel where they are on par, and is close to the multi-task agent’s performance in several environments. We generally found that learning with RL² is slower and less stable (see learning curves and runtime comparisons in Appendix refappendix:mujoco). We hypothesise that this is because the reinforcement learning loss is backpropagated through an RNN. In variBAD on the other hand, we train the encoder-RNN with a supervised loss only. Even though the first rollout includes exploratory steps, this matches the optimal oracle policy (which is conditioned on the true task description) up to a small margin.

The methods PEARL, E-MAML, and ProMP are not designed to maximise reward during a single rollout, and perform poorly in the first episode. They all require substantially more environment interactions in each new task to achieve good performance. PEARL, which is akin to posterior sampling, only starts performing well starting from the third episode. We evaluated E-MAML and ProMP by performing gradient steps after every episode; however, they are typically updated after collecting data for 20 episodes.

3.6.4 Meta-World

Finally, we evaluate variBAD on the challenging Meta-World ML1 benchmark [219], which has emerged as a key challenge for the meta-learning community. In the Meta-World environment, a robot arm has to perform different tasks like pushing objects to an (initially unknown) target location. There are three variants of the

Method	Episode	Reach	Push	Pick-Place
MAML*	10	48	74	12
PEARL*	10	38	71	28
RL2*	10	45	87	24
variBAD	1	100	100	29 (6/20 seeds)
variBAD	2	100	100	29

Table 3.1: Meta-test success rates on the ML1 Meta-World benchmark.

*Results taken from Yu et al. [219]. variBAD was trained to maximise expected online return within 2 episodes (20 random seeds per setting). The first (few) episodes often *includes exploratory actions*, yet have higher success rate than existing methods that maximise final episodic return. For the Pick-Place environment, in brackets we report the number of seeds that learned something.

ML1 benchmark: Reach (where the robot has to reach different initially unknown goal positions), Push (where the robot arm has to push objects to initially unknown goal positions), and Pick-Place (where the robot arm has to pick up an object, and place it near a target goal in 3D space).

Table 3.1 shows the results for variBAD and several baselines on the ML1 benchmark. VariBAD achieves state of the art results. On Reach and Push, variBAD outperforms the previous state of the art results by a significant margin, and is the first to fully solve these tasks. On the harder task Pick & Place, variBAD performs on par with PEARL.

Though the benchmark allows for up to 10 episodes for adaptation, we train variBAD to optimise expected online return within two episodes. As these results show, variBAD can adapt rapidly even within the first episode, which includes exploratory actions. On the Pick-Place task, variBAD either learns to solve the task (6 seeds), or it does not meta-learn at all (14 seeds). We hypothesise that variBAD (and the other methods) face a meta-exploration challenge: the agent does not explore enough during meta-training, which prevents it from learning how to adapt to the task. We found that increasing the episode horizon from 150 to 200 helps mitigate this problem to some extent (results not shown).

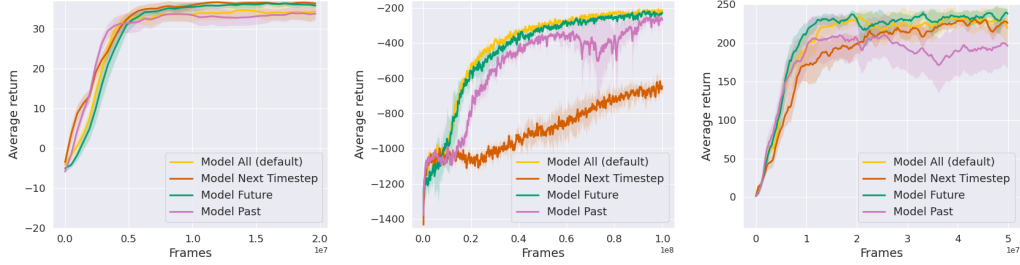


Figure 3.6: Performance of VAE decoding targets for (from left): GridWorld, AntGoal, and Pointrobot (15/5/20 seeds). VariBAD’s default settings (modelling the past and future) perform well, and behave as expected in terms of predicted beliefs (Fig 3.7a). Other choices either underperform (modelling only the past or only the next step), or have undesired effects on the beliefs (modelling only the future, Fig 3.7c).

3.7 Empirical Analysis

In this section, we first examine the impact of different VAE loss formulations on the performance and approximate beliefs to shed light on our design choices. We further analyse how variBAD’s performance is impacted by different choices for the latent dimension.

3.7.1 Modelling Horizon

When formulating our objective in Equation (3.8), we argue that at any time t in its trajectory, an agent should be able to model entire trajectories. In Equation (3.10) we see that this amounts to reconstructing transitions and rewards from the past $\leq t$ and the future $\geq t$.

The intuition is that the agent has to learn to represent the task information in its belief, which encompasses a sufficient statistic about transitions the agent has already observed, and a belief about transitions it can observe in the future.

To analyse this choice empirically, we test how different “modelling horizons” affect performance: modelling only the past ($\leq t$), modelling only the future ($> t$), and modelling only one step into the future ($t + 1$). Figure 3.6 shows the resulting learning curves for the GridWorld, Mujoco AntGoal and PointRobot tasks.

The traditional VAE approach of reconstructing only the embedded part of the trajectory (past) tends to produce latent codes not sufficiently informative to

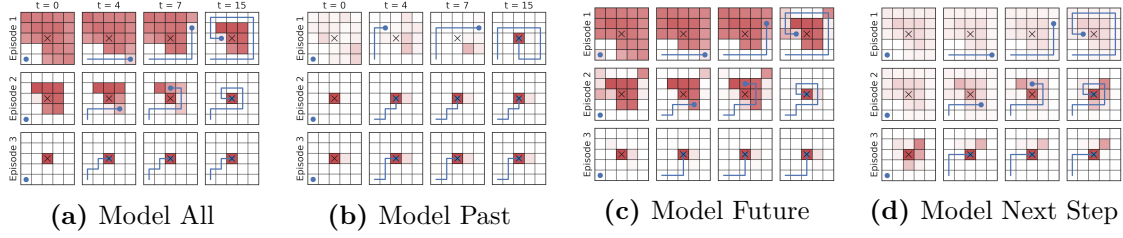


Figure 3.7: Belief visualisation in the Gridworld environment for different decoding targets. Compared to the default variBAD objective which models both the past and the future (a), the approximate belief is less accurate when decoding only the past(b), only the future (c), or only one step into the future (d). In (b) the agent only learns to decode the past, and assigns low probability to all cells (seen and unseen), when it has not found the goal yet. This strategy minimises the loss: for seen cells it correctly predicts that they do not contain the goal; for unseen cells it predicts the same (even though one of them actually contains the goal), which however is not penalised through the loss. In (c) the agent only learns to decode the future, and assigns high probability to all unseen cells, and also to some seen cells, when it has not found the goal yet. Again, this minimises the loss: for unseen cells it correctly predicts that they could contain the goal; for seen cells the loss does not incentivise it to predict the correct thing and the predictions are noisy. We believe some predictions are zero because there is a chance of the agent re-visiting states. In (d) the agent only learns to predict the immediate-next step, and assigns non-zero (but low) probability to these, when it has not found the goal yet. Only when decoding both the entire past and future (default setting, (a)), does the agent correctly predict that seen cells do not contain the goal, and unseen cells could all contain the goal. All methods (a)-(d) learn to correctly predict the goal position with high certainty once they have seen it once.

predict future transitions and therefore leads to suboptimal performance. Similarly, targeting only the subsequent transition (next) does not encode sufficient information to reliably inform the policy. While decoding only the entire remainder of the observed training trajectories (future) performs well, we find that this has an undesired effect on the learned beliefs, discussed below.

Figure 3.7 visualises the approximate belief for different decoding targets in the Gridworld environment. We do so by plotting the reward predictions of the decoder from the VAE latent for each cell in the grid. Decoding only the past or only the next step leads to embeddings that predict no or only spurious rewards until the goal has been found (Figures 3.7b and 3.7d). Only decoding the future enables learning about the rewards prior to actually encountering them, but leads to spurious predictions for visited states (Figure 3.7c): predictions of non-zero rewards at visited non-goal states are not penalised, as these are unlikely to be revisited.

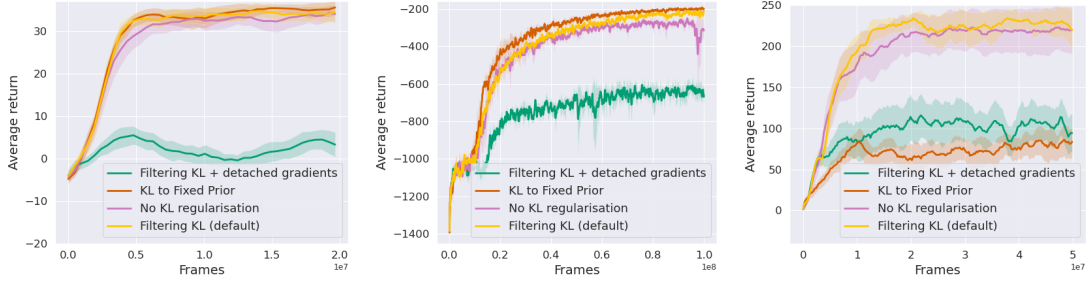


Figure 3.8: Comparison of VAE regularisation penalties

Figure 3.9: Performance of different KL regularisations in the VAE. Domains from left to right are GridWorld (15 seed average), Ant-Goal (5 seed average) and Pointrobot (20 seed average). Using the filtering KL objective performs well consistently across these environments. Using the KL to a fixed prior works well except in the PointRobot environment. We hypothesise this is due to the sparsity of the environment, and the fact that “excluding” certain regions without rewards becomes more challenging when the prior is fixed.

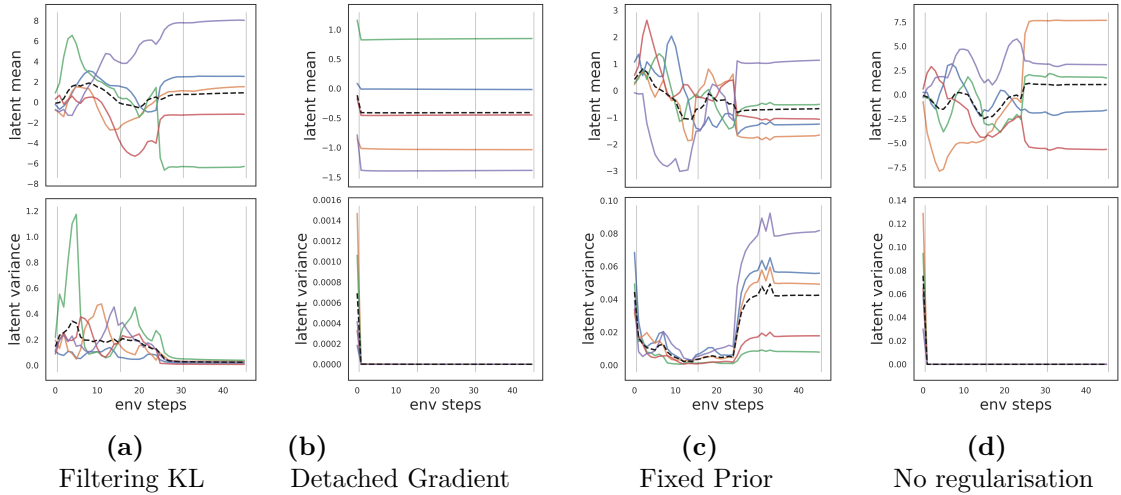


Figure 3.10: Visualisation of latent estimates for different decoding targets, for a representative GridWorld rollout. The agent finds the goal after ~ 25 steps. When using a filtering KL (variBAD’s default setting), the variance collapses once the goal is found (a). When using a fixed Gaussian prior (c), the posterior behaves unexpectedly: the variance increases once the goal is found.

These artefacts are cleared up by decoding full trajectories (Figure 3.7a), which is the default setting we chose for the variBAD objective.

3.7.2 KL Regularisation

The second component of the ELBO, the KL term between the posterior and prior, acts as a regulariser and prior for the latent codes. In the variBAD objective, the

prior is set to the previous approximate posterior, i.e., the KL term is defined as $KL(q(m|\tau_t)||q(m|\tau_{t-1}))$, which is akin to Bayesian filtering. The gradient is taken w.r.t. both terms in the KL. In this subsection, we empirically motivate this choice, by comparing to using a fixed standard Normal distribution as a prior, $N(0, I)$, across all times steps (“fixed prior”), detaching the gradient of the approximate previous posterior, or not doing KL regularisation. The learning curves are shown in Figure 3.8 for the GridWorld, AntGoal, and Pointrobot environment, and visualisations of the learned approximate beliefs in the GridWorld are shown in Figure 3.10. Using a fixed prior is not sufficiently flexible to allow the learning of good latent codes in all cases: the performance is significantly worse in PointRobot (Fig 3.8, right), and when rolling out the meta-learned policy the variance increases sharply after the agent finds the goal in the GridWorld (Fig 3.10c). This is undesirable—the variance should collapse once the agent is certain about the task. By forcing the VAE to adhere to a fixed prior we artificially increase uncertainty as the embedding is encouraged to regress towards the prior even once the goal has been identified.

When training with the default variBAD objective we get reasonable variance estimates that decay with progressing exploration and collapse when the goal is found (Fig 3.10a). Using no KL regularisation at all slightly reduces performance (Fig 3.8) and a total collapse in variance (Fig 3.10c). A lack of regularisation allows the VAE to encode observed trajectories as point masses with no variance, indicating it overfits by producing a unique code for each of them. Detaching the gradient on the previous approximate posterior significantly harms performance in all cases (Fig 3.8) and yields an inflexible latent space (Fig 3.10b).

3.7.3 Belief Dimensionality

Finally, we examine how the size of the latent belief impacts performance in Figure 3.11, for the GridWorld and Ant-Goal tasks. As expected, if the latent dimensionality is too small, then not all relevant information can be retained and the policy is unable to adapt to different tasks. This only happens when using a dimensionality of size 1 or 2, and any choice larger than this leads to decent

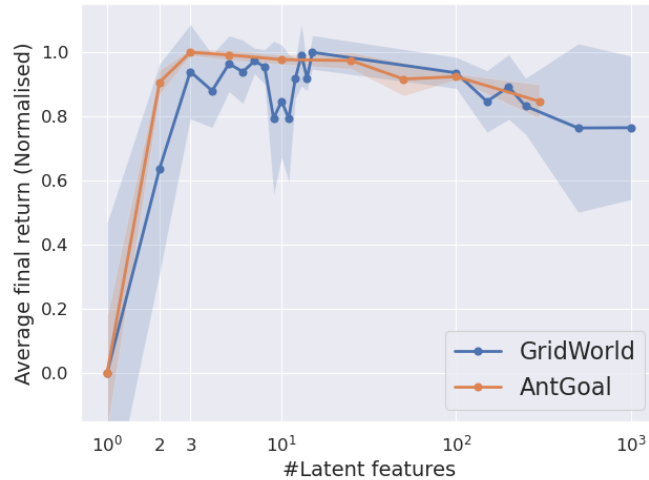


Figure 3.11: Normalised performance for different number of latent features, for the Grid-world and Ant environment. As long as the latent is not underparameterised, variBAD achieves good performance. Only when vastly overparameterising the latent we see a small performance decrease.

performance. Interestingly, very large parameterisations (1000 for GridWorld and 300 for AntGoal which was the maximum we could fit into the memory of a single GPU) have a comparatively minute impact despite artificially increasing the size of the state space on which the policy acts. In practice this means that as long as we do not underparameterise the latent dimension, we achieve good performance.

3.8 Conclusion & Future Work

We presented variBAD, a novel deep RL method to learn approximately Bayes-optimal behaviour, which uses meta-learning to exploit knowledge obtained in related tasks and perform approximate inference in unknown environments. In a didactic gridworld environment, our agent closely matches Bayes-optimal behaviour, and in more challenging MuJoCo tasks, variBAD outperforms existing methods in terms of achieved reward during a single episode. On the recently proposed Meta-World ML1 benchmark, variBAD outperforms existing methods by a large margin and fully solves two out of the three benchmark tasks for the first time. In summary, we believe variBAD opens a path to tractable approximate Bayes-optimal exploration for deep reinforcement learning.

There are several interesting directions of future work based on variBAD. For example, we currently do not use the decoder at test time. One could instead use the decoder for online planning, or to get a sense for how wrong the predictions are (which might indicate we are out of distribution, and further training is necessary). Another exciting direction for future research is considering settings where the training and test distribution of environments are different. Generalising to out-of-distribution tasks poses additional challenges and in particular for variBAD two problems are likely to arise: the inference procedure will be wrong (the prior and/or posterior update) and the policy will not be able to interpret a changed posterior. In this case, further training of both the encoder/decoder might be necessary, together with updates to the policy and/or explicit planning.

Statement of authorship

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor (**only required where there isn't already a statement of contribution within the paper itself**).

Title of Paper	VaribAD: A Very Good Method for Bayes-Adaptive Deep RL via Meta-Learning
Publication Status	☐ Published ☒ Accepted for Publication ☐ Submitted for Publication ☒ Unpublished and unsubmitted work written in a manuscript style
Publication Details	Zintgraf, L., Shiarlis, K., Igl, M., Schulze, S., Gal, Y., Hofmann, K., & Whiteson, S. (2020). Varibad: a very good method for bayes-adaptive deep rl via meta-learning. <i>Proceedings of ICLR 2020</i>. The thesis will include a substantially extended content in preparation of a submission with the same title to the "Journal of Machine Learning Research" with a changed authorship list. The new authorship list will be: Luisa Zintgraf, Sebastian Schulze, Cong Lu, Leo Feng, Maximilian Igl, Kyriacos Shiarlis, Yarin Gal, Katja Hofmann, Shimon Whiteson

Student Confirmation

Student Name:	Sebastian Schulze		
Contribution to the Paper	• joint discussion of theoretical aspects • initial implementation of MAML/ProMP baseline • review and extension of initial code base • joint design, execution; analysis and write-up of experiments in section 6		
Signature:		Date	13/04/2021

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Prof Michael A. Osborne			
Supervisor comments			
All confirmed.			
Signature		Date	14 April 2021

This completed form should be included in the thesis, at the end of the relevant chapter.

4

Active Reinforcement Learning with Monte-Carlo Tree Search

Contents

4.1	Abstract	48
4.2	Introduction	48
4.3	ARL: Definition and Illustration	49
4.4	Related work	50
4.4.1	Bayesian RL	52
4.4.2	BAMCP: MCTS for Bayesian RL	53
4.4.3	AlphaZero: Planning with function approximation	54
4.5	Reduction to BAMDPs	55
4.5.1	Discussion of existing BRL approaches	56
4.6	A planning approach to ARL	57
4.6.1	BAMCP fails in ARL problems	59
4.7	Alpha-style planning for ARL	61
4.7.1	Empirical observations and analysis	61
4.8	Discussion	64

Preface

As seen in the previous chapter, we can in principle equip artificial agents with the ability to quickly adapt to their surroundings in a pre-training stage. Conceptually, this works by sharpening the agent's beliefs from making no prior assumptions to

the point at which it is able to link the optimal policy to a few salient environment features it has to achieve clarity about.

While representing a major stepping stone towards real-world deployment of RL agents, a severe practical hindrance is the remaining in-built assumption and requirement for continuous feedback – even in parts of the environment the agent has no uncertainty about or where uncertainty is of no consequence to future decisions. Many of these observations will be superfluous post-deployment and trigger “empty updates”, as much information has already been acquired during pre-training.

Even though there are scenarios in which we may be able to pre-specify rewards offline prior to deployment (e.g. in simulation), feedback in many tasks of interest has to be generated situationally and online. Obvious examples for this are found in any setting in which agent performance is linked to a degree of human satisfaction. For training purposes we might be able to provide the agent with general population statistics, however, post-deployment adaptation will have to fit to the individual user and thus enquire about their satisfaction regarding exhibited behaviour.

Particularly under such circumstances, constant supervision requests have the potential to not just be a nuisance but to represent a considerable invasion of privacy. Whilst training on population data, the agent has full access to the preferences of the “average human” and can determine which information is relevant to its duties. At test time, however, users may not actually wish to divulge all of these aspects and require a way of withholding feedback.

Even in scenarios in which we are willing to provide constant supervision it may be impractical to do so. Artificial systems typically operate on a much higher-resolution time scale than humans. Consequently, our ability to generate feedback may be far outpaced by the rate at which it is requested and we once more have to find ways to only provide it selectively.

A promising solution is to shift the burden of requesting knowledge to the agent. Given a cost quantifying the effort of providing supervision in different scenarios, the agent can deliberate about the potential gains prior to requesting feedback selectively and even avoid exploratory actions for which feedback cannot be obtained

altogether. In this setting the above mentioned privacy issues could be resolved by imposing an “infinitely high” penalty on the information humans do not want to share and even small penalties would deter the agent from requesting information it ultimately doesn’t need. Finally, this approach could help shape the agent’s learning behaviour, e.g. by increasing the cost over time to artificially limit the time window in which the agent can expect to obtain feedback and adapt. It bears pointing out that providing the agent with a supervision cost is generally easier than providing the feedback itself. Typically the feedback mechanism will remain the same (e.g. human scoring via an app) and require constant amounts of effort whereas the evaluation outcome (i.e. the actual score) varies drastically.

In the following we discuss a variant of the RL problem which formalises this approach as “Active Reinforcement Learning”(ARL). It was originally introduced in [105] but discussion focused on long-term behaviour and asymptotic performance. By contrast we are interested in immediate near-optimal performance over a short post-deployment adaptation period and examine short-horizon settings.

We once more realise that optimal behaviour is a function of the agents beliefs and formally reduce ARL problems to Bayes-Adaptive reinforcement learning settings (BAMDPs). This allows us to adopt the connected notions of optimality under an initial belief prior. Since the neural models employed for belief learning in the previous section make for a poor object of study, we fall back on beliefs specified as probabilistic models in standard BRL fashion.

We further discuss why various existing BRL methods fail pathologically in ARL before proposing and implementing two planning based algorithms. Such approaches have seen tremendous advances in the recent past, but even sophisticated frameworks struggle to meet the challenges inherent in ARL. We empirically investigate and analyse these causes and outline potential avenues for overcoming them.

4.1 Abstract

Active Reinforcement Learning (ARL) is a twist on RL where the agent observes reward information only if it pays a cost. This subtle change profoundly complicates solving the exploration-exploitation trade-off. We relate ARL in tabular environments to Bayes-Adaptive MDPs and discuss why powerful principles like optimism, Thompson sampling, and random exploration do not help with ARL. In search of an (approximately) optimal approach to ARL we turn to recently successful planning approaches and provide an algorithm that is asymptotically optimal. In practice, a naive implementation fails due to poor allocation of planning resources. While more powerful frameworks can identify optimal solutions in toy domains, they struggle to scale to even moderately-sized settings. We discuss this failure and its implications for future research. ARL is most relevant to settings where reward signals are provided online and incrementally (e.g. a human providing interactive feedback). This is important in complex environments where offline hand-engineering of reward functions is infeasible.

4.2 Introduction

Imagine two treatments are being tested in a medical trial. The treatments are cheap but having doctors evaluate whether they worked costs £5,000 for each additional patient. Treatments are assigned using a Bandit design[106] and after 200 trials the difference in mean evaluation between the two treatments is tiny. Should the trial continue?

At some point an additional trial is not worth another £5,000. This cost of evaluating outcomes is not incorporated into standard Bandits. When playing Bandits, deciding whether to explore depends only on the estimated differences in expected (discounted) return between arms. The same is true for Reinforcement Learning in MDPs: the cost of providing a reward for a state-action pair is not a parameter of the learning problem. This makes sense when the reward function is created all at once and *offline*, as when it is hand-engineered. But if the rewards

are created incrementally *online*, as in the medical trial, then an important feature of the decision problem has been left out.

Online construction of rewards is common in real-world Bandit problems. Customers may be part of an A-B test and be paid to give feedback on new products [166]. Recent research, spurred by the difficulty of hand-engineering rewards, has formalised more general approaches to online reward construction. In *Reward Learning*, a reward function is learned online from human evaluations of the agent’s behaviour [205, 32, 163]. In *Inverse Reinforcement Learning* (IRL) and *Imitation Learning* [1, 82, 53], humans provide demonstrations that are used to infer the reward function or the optimal policy. These demonstrations can be provided offline or online but the reward function is always specified *incrementally*, as a set of human actions or trajectories. Reward Learning and IRL are approaches to problems in AI Safety [4, 53]. If RL is applied to real-world problems with direct impact on human users and on society, then constructing reward functions offline (by hand) may be infeasibly difficult. Online training by humans is a more scalable approach to learning robustly safe policies that incorporate human preferences [163, 73].

When RL agents are trained via online interaction with humans the human labour required to construct rewards is a significant cost. How can this cost be reduced? Intuitively, if the RL agent can predict an action’s reward then a human need not provide it. In *Active Reinforcement Learning* (ARL), this choice of whether to pay for reward construction is given to the RL agent [105]. It is the analogue of Active Learning, where an algorithm decides online whether to have the next data point labelled in a classification or regression task [168].

4.3 ARL: Definition and Illustration

To fix intuition, we define the ARL problem here and elaborate on this definition in later sections. An active reinforcement learning problem is a tuple $(S, A, \mathcal{T}, \mathcal{R}, \tau, c)$. The components $(S, A, \mathcal{T}, \mathcal{R}, \tau)$ define a regular Markov Decision Process (MDP), with state space S , action space A , transition function $\mathcal{T}$, reward function $\mathcal{R}$, and time horizon τ . The component $c > 0$ is a scalar “query cost” of observing rewards.

While we consider constant c , it could more generally be a function of (s, a) and the planning approaches discussed readily accommodate this. All components except $\mathcal{T}$ and $\mathcal{R}$ are initially known to the agent.

ARL proceeds as follows. At time step t , the agent takes an action pair (i_t, a_t) , where $i_t \in \{0, 1\}$ and $a_t \in A$, which determines the next state $s_{t+1} \sim \mathcal{T}(s_t, a_t)$ and reward $r_t \sim \mathcal{R}(s_t, a_t)$ ¹. If $i_t = 1$, the agent pays to observe the reward r_t , and so receives a total reward of $r_t - c$. If $i_t = 0$ the agent does not observe the reward; so if the agent did something bad it will not be *knowingly* punished. The agent’s total return after H time steps is defined as:

$$\text{Return}(H) := \sum_{t=0}^H r_t - i_t c.$$

We emphasise that actions for which the agent did not observe the reward still count towards the return.

An ARL problem depends crucially on how the query cost c compares to the expected reduction of the agent’s uncertainty about the optimal policy and its expected returns. When c is large relative to expected gains, the agent should never query and rely on prior knowledge about $\mathcal{R}$. When c is very small, the agent can use a regular RL algorithm and always query. In between these two extremes, the agent must carefully select a subset of state-actions to query and so RL algorithms are not readily applicable to ARL. Figure 4.1 shows three MDPs that illustrate the challenge of deciding which actions to query. Even seemingly simple examples can exhibit complex correlations between (individual) reward estimates, their expected posterior distributions and the resulting expected returns for different policies under consideration.

4.4 Related work

How does ARL relate to regular RL? In regular RL there is no cost for deciding to observe a reward. Yet regular RL does involve “active learning” in the more general

¹Different dependencies between $\mathcal{T}$ and $\mathcal{R}$ can be considered. In particular, we often assume the reward to depend on the next state, i.e. $r_t \sim \mathcal{R}(s_{t+1})$.

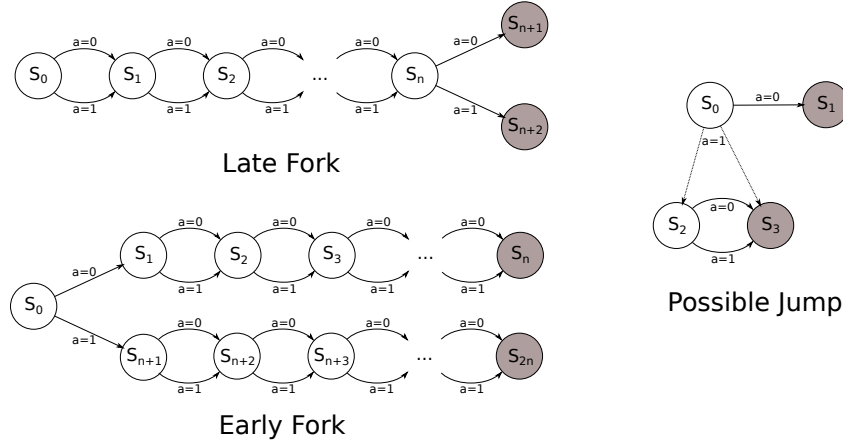


Figure 4.1: Three didactic, tabular MDPs illustrating the active reinforcement learning problem in which rewards are a function of visited states only. For known transitions and suitable reward uncertainty, an agent should only query the right-most states in “Late Fork” as all other actions lead to unavoidable states. Under equivalent conditions, an agent should query all actions in “Early Fork” as observed rewards influence the policy for state 0. In “Possible Jump” it may be optimal to delay querying uncertain rewards until the transition outcome of action 1 in state 0 is known.

sense: the agent decides which actions to explore instead of passively receiving them. So techniques for exploration in regular RL might carry over to ARL.

Unfortunately, most practical algorithms for regular RL use heuristics for exploration such as ϵ -greedy, optimism [11, 100], and Thompson sampling [139]. While these heuristics achieve near-optimal exploration for certain classes of RL problem [26, 12], they are not directly applicable to ARL, as explained in Section 4.5. There are RL algorithms that try to explore in ways closer to the decision-theoretic optimum. Various algorithms use an approximation to the Bayesian value of information [179, 40] and so relate to our Section 4.5. An alternative non-Bayesian approach is to have the agent learn about the transitions to which the optimal policy is most sensitive [52].

There is a substantial literature on active learning of rewards provided online by humans [211, 45]. Daniel et al. learn a reward function on *trajectories* (not actions) from human feedback and use Bayesian optimisation techniques to select which trajectories to have labelled. Sadigh et al. learn a reward function on state-action pairs and their agent optimises actions to be informative about this function. These reward-learning techniques are aimed at continuous-state environments and do

not straightforwardly transfer to our tabular ARL setting. Our work also differs from Sadigh et al. in that we optimise for informativeness about the optimal policy and not the true reward function. As Figure 4.1 illustrates, if some states are unavoidable then their reward is irrelevant to the optimal policy.

There is also work applying active learning to tabular RL with human teachers but where human input is quite different than in the ARL model [183, 91].

4.4.1 Bayesian RL

An MDP is specified by $\mathcal{M} = (S, A, \mathcal{T}, \mathcal{R}, \tau)$, with components defined in Section 4.3. A Bayesian RL problem [49, 65, 68] is specified by an MDP $\mathcal{M}$ and an agent’s prior distribution $b_0(\theta_T)$ on the transition function parameters θ_T^2 in place of or jointly with θ_T . The agent’s posterior at time step t is then given by $b_t(\theta_T) = b_t(\theta_T|h_t) \propto \mathcal{L}(h_t|\theta_T)b_0(\theta_T)$, where $\mathcal{L}(h_t|\theta_T)$ is the likelihood of history h_t given the transition function with parameters θ_T .

The Bayesian RL problem can be transformed into an MDP planning problem by augmenting the state space with the agent’s belief and the transition function with the agent’s belief update. The resulting MDP is defined by $\mathcal{M}^+ = \langle S^+, A, \mathcal{T}^+, \mathcal{R}, \tau \rangle$ and is called a *Bayes-Adaptive MDP* (BAMDP), where:

- S^+ is the set of hyperstates $S \times \theta_T$;
- P^+ is the combined transition function between states and beliefs:

$$S^+ \times A \times \mathcal{R} \times S^+ \rightarrow [0, 1]; \text{ and}$$
- The initial hyperstate is given by the initial real-world state s_0 and prior b_0 .

²The standard formulation only considers uncertainty about the transition function. Uncertainty regarding rewards is treated analogously by considering the reward function parameters θ_R .

4.4.2 BAMCP: MCTS for Bayesian RL

BAMCP is a Monte Carlo Tree Search (MCTS) algorithm for Bayesian RL [69]. In the limit of infinitely many Monte Carlo simulations it provably converges to a Bayes-optimal policy (i.e. the optimal policy for the corresponding BAMDP). In experiments, it has achieved near state-of-the-art performance in a range of environments [28, 68].

At any given time step BAMCP attempts to compute the Bayes-optimal action for the current state under the agent’s posterior b_t online. In typical MCTS fashion [33] it iteratively constructs a search tree mapping possible future trajectories to value-function estimates (see Fig. 4.2). Each tree node corresponds to a hyperstate-action (b_t, s_t, a_t) , and for each node the algorithm maintains a value estimate $Q(\{s_t, b_t\}, a_t)$ and visit count $N(\{s_t, b_t\}, a_t)$. Tree construction proceeds in four steps:

1. Node selection: The algorithm descends from the tree root to a leaf L . When at node $n = \{s_t, b_t\}$, the algorithm follows the UCT policy given by:

$$\operatorname{argmax}_a Q(n, a) + u \sqrt{\frac{\log(\sum_a N(n, a))}{N(n, a)}}$$

where u is an exploration parameter. Environment transitions are sampled according to the posterior belief b_t .

2. Expansion: The tree is expanded by a single leaf node L^+ corresponding to taking an unexplored action from L .
3. Rollouts: The value of the new node $V(L^+)$ is estimated as the rewards accumulated by a random policy during a single MC simulation of the remaining trajectory.
4. Backup: Node statistics of tree nodes along the selected trajectory are updated with the sampled return.

This procedure repeats until a predefined budget of simulations is exhausted, before selecting an action based on the statistics accumulated at the root node³.

To avoid derivation of posterior updates throughout the sampling procedure, BAMCP fixes a single MDP drawn from the posterior at the root node per simulation to determine environment response (“root sampling”). This provably generates the correct distribution of transitions at each node whilst being computationally much cheaper [68]. To further increase computational efficiency, environment parameters are sampled in a lazy fashion where possible.

4.4.3 AlphaZero: Planning with function approximation

AlphaZero [172] is a representative of a range of related algorithms [171, 172, 5] that recently achieved superhuman performance in game play. Their main innovation is the combination of (deep) supervised learning methods and MCTS into a continuous learning cycle of knowledge distillation and policy improvement.

The agent’s estimate of the best policy and corresponding values are represented by two neural networks, which are trained on a labelled set of environment states. Labels are provided by extracting node information from a number of fast MCTS searches with limited budgets. After each supervised learning step, the updated networks are used to bias a new round of MCTS runs towards exploring and evaluating high performing policies, producing higher quality learning targets and creating a closed loop of continuous improvement.

Individual tree searches follow the four-step procedure outlined above with minor but vital alterations. To include policy information and bias trees towards promising policies, AlphaZero makes use of the PUCT [157] formula for node selection which weights exploration bonuses according to the current policy estimate. To promote the exploration of “local” policy alterations, the bias at the root node is altered to be more uniform across actions. Initial value estimates for newly added leaf nodes are provided directly by the value network which can give more accurate estimates than random rollouts.

³Typical selection criteria are the maximum value or visit counts of children nodes corresponding to available actions

The starting points for MCTS searches, which serve as training locations later on, are created by following a simulated trajectory whose policy is determined by the online MCTS estimates. Different methods of converting the created search trees into learning targets can be considered [207].

4.5 Reduction to BAMDPs

We begin by observing that the agent’s optimal (querying) behaviour is a function of its beliefs regarding the environment and the resulting uncertainty about the optimal policy (or equivalently the associated returns). Consequently, we follow Bayesian methodology and place prior distributions over the reward and transition functions. This admits a reduction to an augmented MDP planning problem, mirroring that of the Bayes-adaptive formulation.

Actions in ARL reduce to regular RL actions by crossing each regular action with an indicator variable. But unlike in RL, an ARL agent does not always observe a scalar reward. To accommodate this, we introduce the null reward “ $\perp$ ”. If the agent takes an action without querying, it receives a reward $\perp$. The definition of the agent’s belief update is modified to not update on $\perp$. We emphasise that in the episodic settings under consideration, time is part of the original state-space and the hyperstate representation of holding the same belief at different times differs.

We note that the Bayesian perspective allows for substantial extension beyond the original ARL formulation as given in Section 4.3. We briefly discuss them in the following and leave a more in-depth investigation as future work.

From an agent’s perspective, query and non-query of any action manifest themselves as transitions to different points in hyperstate-space (more specifically in the features denoting model parameters). These transitions are governed by the likelihood update of the probabilistic environment model. Consequently, we can – at least in theory – consider different query types by defining the matching likelihood updates for the transition function. Which query types are available depends on the type of feedback any particular application admits and the functional shape of our probabilistic reward model. As an example, consider an agent representing

rewards with normal distributions. This readily admits likelihood updates for reward observations at any location (including but not limited to the most recent state-action) as well as return observations for entire or partial trajectories (sums of rewards corresponding to integral observations).

A second consideration is that we can remove the assumption of immediate feedback by further augmentation of the hyperspace to encode “pending” feedback for queried state-action pairs. This, however, is conditional on identifying an appropriate feature representation, as a naïve “count-down” to the likelihood update is likely to result in an explosion of the size of the state space.

4.5.1 Discussion of existing BRL approaches

Can we apply Bayesian RL algorithms to ARL? Many such algorithms can be straightforwardly adapted to deal with the null reward and produce well-typed outputs for ARL. Yet naïve adaptations often fail pathologically. For instance, they might never choose to query and hence learn nothing. In the following we discuss a set of principles listed in a recent BRL survey [65] and the pathologies they lead to in ARL.

Bayesian model-free methods

Bayesian model-free approaches have no principled way of accounting for the selective nature of observations made in ARL. This class of algorithms aims to use the agent’s observations to iteratively improve a bootstrapped estimate of either estimated long-term utilities of actions or the agent’s policy parameters. Either way, the bootstrap and its uncertainty bounds are only indirectly related to individual rewards (e.g. via the unmodelled transition function). As a consequence, the actual source of uncertainty cannot be determined and the agent would have to query indiscriminately across all visited state-actions to reduce it.

Optimism in the face of uncertainty

Optimism means adding bonuses to more uncertain rewards and taking optimal actions in the resulting optimistic MDP [100, 6, 11]. An optimal agent in a *known*

MDP never queries. Optimism treats the optimistic MDP as (temporarily) known and hence optimism applied to ARL never queries.

More specifically, these approaches solve the augmented Bellman equations

$$V(s_t, b_t) = \max_{a_t} \left[\hat{R}(s_t, a_t, b_t) + \int_S \hat{T}(s_t, a_t, s_{t+1}, b_t) V(s_{t+1}, b_t) ds_{t+1} \right],$$

where $\hat{R}$ and $\hat{T}$ are optimistic environment dynamics. As no likelihood update is applied, the belief b_t remains fixed throughout the recursion. Accordingly, the values of querying and non-querying a particular action differ exactly by the query cost deducted from $\hat{R}(s_t, a_t, b_t)$ and the non-querying value always dominates.

Sampling based exploration

Thompson Sampling and related approaches [10, 139, 181] act optimally with respect to a number of MDPs sampled from the current posterior belief. Exploration is driven by this sampling procedure, as uncertain actions will eventually be sampled with above-average returns and consequently executed. During planning, sampled MPDs are treated as ground truth and such approaches fail to recognise the value of querying for the same reason as optimism.

Myopic Value-of-Information

Value-of-information means estimating the Bayesian expected return of exploratory actions by explicitly modelling their information gain [40, 65]. Myopic (one-step) approaches assume the agent gains information only from the current action and thereafter exploits. This can lead to highly sub-optimal behaviour in ARL because it stops exploring when the agent can not learn enough from a single query to change their policy.

4.6 A planning approach to ARL

BAMCP is simple to adapt to Bayesian ARL and, unlike the principles in Sec 4.5, does not lead to obvious pathologies. In fact, in the limit of infinite computation it converges to the Bayes-optimal policy representing the ideal solution to a given ARL problem.

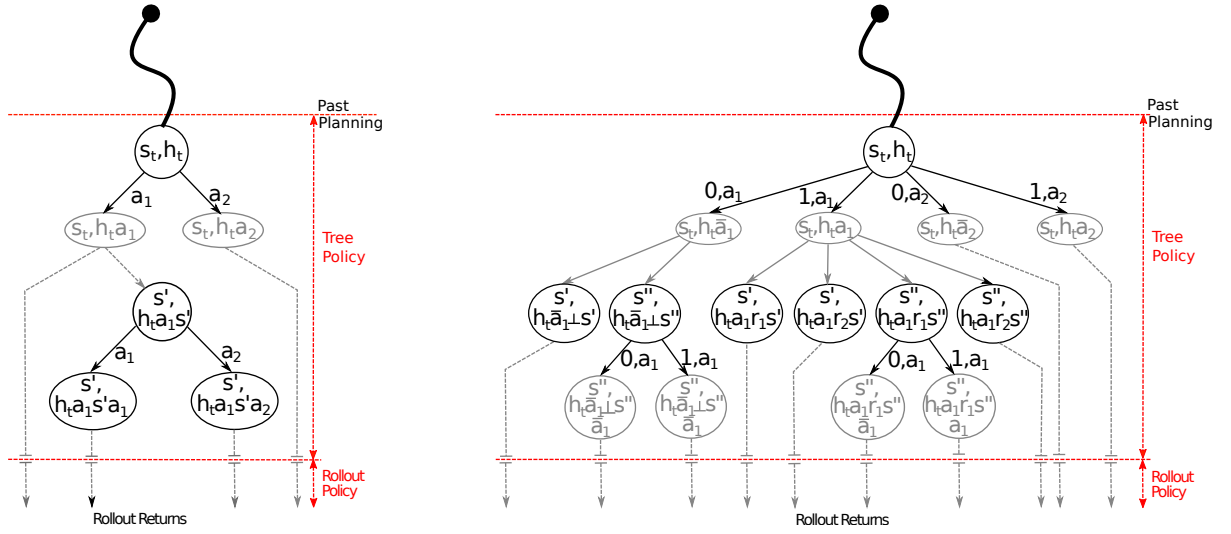


Figure 4.2: Search tree for BAMCP applied to regular RL (**Left**) and applied to ARL (**Right**). Nodes in black correspond to a history h_t and current state s_t . In the ARL tree, query actions have form $(1, a_t)$ and cause extra branching because they result in multiple possible observed rewards as well as multiple possible state transitions. At leaf nodes, actions are selected by a rollout policy.

Adapting BAMCP to ARL requires a few modifications which are depicted in Figure 4.2. First, Guez [68] considered the standard BRL setting which only places uncertainty on transitions. A key feature of ARL is that learning about the reward function is much more expensive than usual and we explicitly model uncertainty over both the reward function $\mathcal{R}$ and transition function $\mathcal{T}$. We thereby assume a factored representation that admits separate updates on transition and reward observations. Second, querying is incorporated into Monte-Carlo simulations. When simulating a trajectory, each action $a \in A$ can be queried or not queried, as represented by indicator i . If the action is not queried, the search tree may not branch (since there is no reward observation) but the reward backup is still performed. If the action is queried, its reward is observed and reduced by the query cost c . Finally, random rollouts make no use of the additional information gained by querying. We therefore remove this option during rollouts to avoid underestimating state values due to random queries that cannot be exploited.

4.6.1 BAMCP fails in ARL problems

Despite the theoretical guarantees of asymptotic correctness, BAMCP performs poorly even in simple settings. We tested the algorithm on small instances of the “Late Fork” in Figure 4.1 with $n = 2..5$ and $H = 2..10$ episodes. To have a clear intuition of the optimal querying behaviour, we assume transitions to be known and place $\text{Beta}(0.0001, 0.0001)$ priors on the probability of state rewards being either 0 or 1 (this sharply peaked prior corresponds to a binary belief over deterministic rewards).

Under these conditions we observe that the agent makes the decision to query inconsistently and in particular can arbitrarily choose to request reward observations at unavoidable states. The frequency with which the agent queries is influenced by the query cost and planning horizon, with higher penalties and smaller horizons having deterring effects across all states.

Analysis of the constructed search trees yields a simple explanation for this behaviour (see Fig. 4.3). Firstly, initial value estimates have high variances. This is due to the random nature of the rollout policy and variability in MDPs drawn from the agent’s belief. Longer search horizons accumulate more rewards and amplify this effect. Secondly, UCT preferentially allocates simulations to promising subtrees. As a consequence, the tree search almost exclusively concentrates on a small, quasi-random set of actions with high initial value estimates from which the final decision is picked. The querying cost hereby acts as a bias against querying actions which makes their selection less likely.

To address the above, we experimented with delaying the expansion of leaf nodes to stabilise their value estimates at the cost of reducing the size of the search tree. While this indeed leads to a more even allocation of search efforts, it does not induce optimal querying behaviour. This is due to the ignorance of the rollout policy to information obtained by prior queries. In essence, it forces the agent to construct entire subtrees in which exploitative trajectories can systematically be chosen and backed-up before the value of a query action can be realised. When the

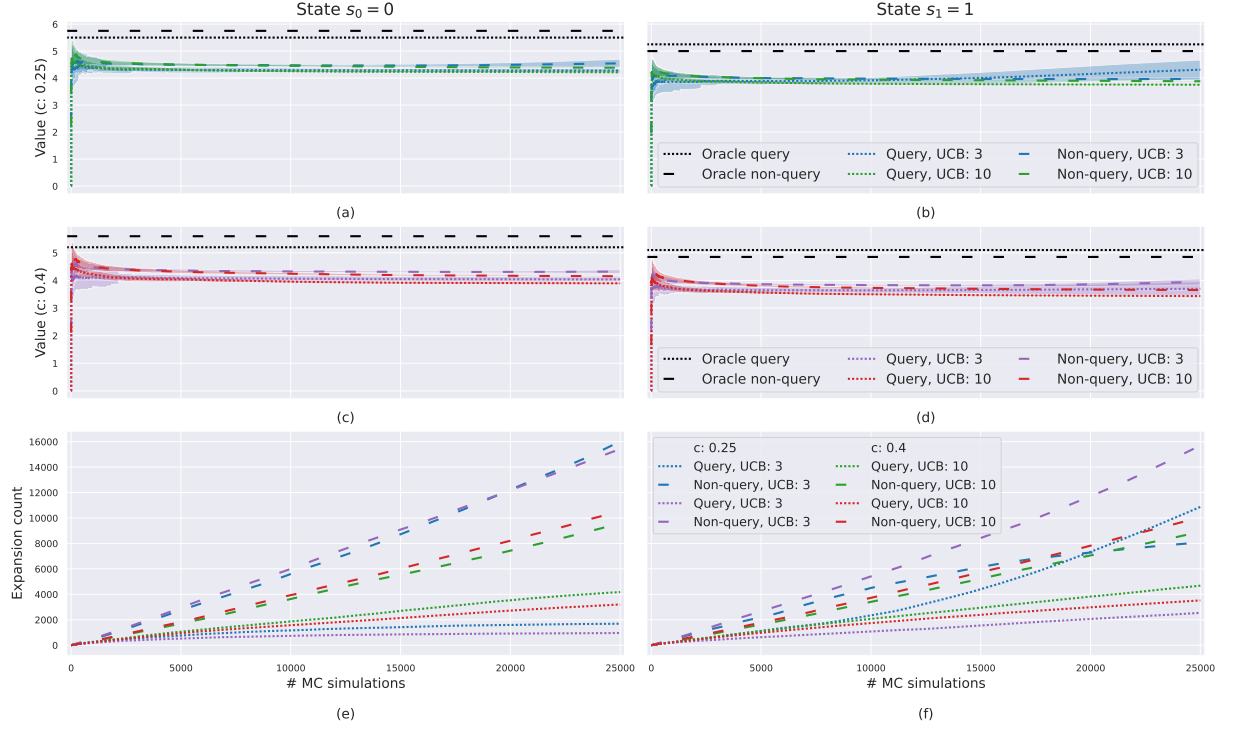


Figure 4.3: Statistics of BAMCP tree searches in Late Fork ($n = 2$; $H = 5$).

Value estimates and expansion behaviour at the root nodes are shown for the two states in the first episode (100 seed average) and varying query cost and UCB parameters. In both states two actions with equal prior are available. At s_0 the outcome is unavoidable and the agent should not query. At s_1 the agent transitions into differing states and should query the result for later exploitation. Note the unintuitively low value difference between query and non-query stemming from the possibility to query at the next visit, missing only a single exploitation opportunity.

Figures (a)-(d) show the highest value estimate for querying (respectively non-querying) as a function of the number of MCTS rollouts and for different query-costs c and expansion parameters (UCB). Shaded regions indicate one standard deviation.

For low simulation budgets value estimates fluctuate, inducing quasi-random behaviour. Subsequent rollouts reduce this variance. However, continuous expansion of querying action and back-up of associated costs that cannot be recouped by random rollouts lead to a systematic underestimate. Only once the search tree grows to include exploitative paths do estimates improve as seen in Figure (b). However, this requires a prohibitive number of MC samples.

As seen in figures (e) and (f) low UCB values typically lead to an extreme disparity in rollout allocations. While this can focus the search on tree branches of interest, it typically favours non-queries and delays the recognition of query values. Higher UCB values lead to a more even allocation, but this dispersion also deteriorates value estimates overall.

rate of expansion is reduced and/or larger numbers of simulations are allocated to non-exploitative paths, this does not happen and the agent almost never queries.

4.7 Alpha-style planning for ARL

To overcome the issues described in the previous section, we consider an alpha-style planning approach to ARL. Analogously to AlphaZero, we use two neural networks with shared initial layers to represent a policy and value function over hyperstates. This allows us to bootstrap the tree search process to continuously evaluate and improve the policy in a closed learning cycle as per Section 4.4.3. As the agent is enabled to optimise its behaviour at later time steps (and no longer relies on random initialisation), we expect subsequent tree searches to better identify relevant paths and recognise the value of querying.

We follow existing work and determine the root state for successive MCTS iterations using a tempered Boltzman policy based on the visitation counts of the last search tree [172]. The corresponding transitions are drawn from the prior. We note that analogously to root sampling, we can sample simulation transition parameters $\theta_{\mathcal{T}}$ at the start of each simulation to reduce computational effort without modifying the resulting visitation distribution in the data buffer⁴. Learning targets are based on the greedy approach introduced in [207]. A high-level description of the resulting framework is presented in algorithm 1.

4.7.1 Empirical observations and analysis

We examine the presented framework on the same problem instances that were used to analyse the behaviour of BAMCP in ARL in Section 4.6.1. Somewhat surprisingly, the described approach struggles to reliably solve even these small domains. Instead we observe that performance is highly dependent on a set of key hyperparameters – for ill-suited configurations, the agent settles on strategies that never query. While

⁴A proof for the correctness of root sampling can be found in Section 3.4.1 of [68]

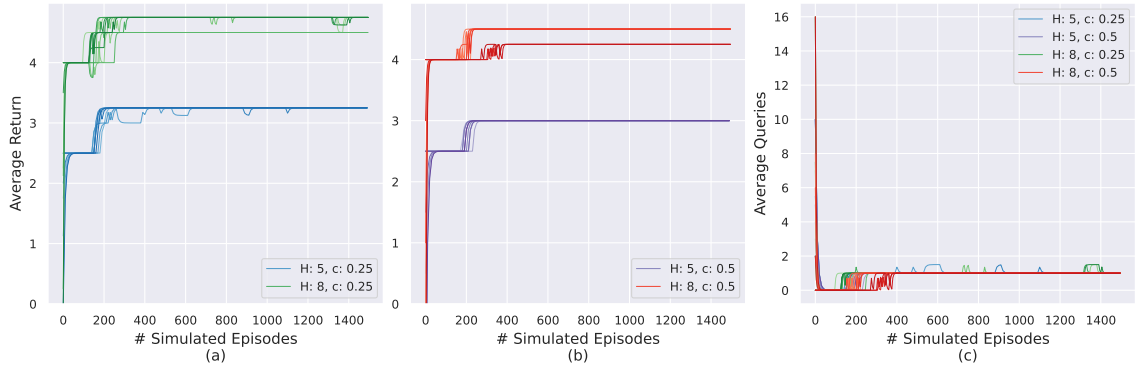
Algorithm 1 Alpha style planning in Bayes-Adaptive (ARL) MDPs**Input:** Training budget, initial hyperstate $\langle s_0, b_0 \rangle$.**Output:** Trained policy and value network π, V Initialise data buffer D of fixed capacity and neural networks π, V **while** Budget remaining **do** $s_{sim} = \langle s_0, b_0 \rangle, (\mathcal{T}_{sim}, \mathcal{R}_{sim}) \sim b_0$ **while** not s_{sim} terminal **do**Run BAMCP starting from s_{sim} with bootstrap π, V Extract supervised data $\{s_{sim}, V(s_{sim}), \pi(s_{sim})\}$ and insert into D Draw $a_{sim} \sim \pi_{sim}(s_{sim}), s_n \sim \mathcal{T}_{sim}(s_{sim}, a_{sim}), r \sim \mathcal{R}_{sim}(s_{sim}, a_{sim})$ Update $s_{sim} = \langle s_n, \text{Belief-update}(s_{sim}, a_{sim}, s_n, r) \rangle$ **end while**Perform gradient descent on D to update $\pi(s), V(s)$ **end while**

Figure 4.4: Performance of a fixed set of hyperparameters on the Late Fork environment ($n = 2$) with differing query costs and planning horizons. Each setting is repeated across 10 different seeds. Figures (a) and (b) show the performance averaged across possible environments under the initial prior as a function of simulated episodes. For $H = 5$ the algorithm learns to correctly query in the initial episode to obtain maximal possible rewards (3.25 and 3.0 depending on the query cost). For $H = 8$ the algorithm still learns to query once, but only in later episodes and falls short of the optimal return (5.5 and 5.25 respectively).

we were able to consider individual problem instances and identify favourable settings for each of them⁵, no setting fit all problem instances at once (see Figure 4.4).

The tolerance regarding those hyperparameters seems to shrink with increasing problem difficulty, i.e. narrowing gap between query cost and value of information or increasing planning horizons. Hyperparameters that matter the most are those governing different exploratory processes (number of rollouts, UCB constant, noise

⁵Parameters were tuned via gridsearch on a single random seed. Settings that led to successful learning for one seed typically also did so for held out random seeds.

injection at root node, temperature of Boltzman distribution for policy targets and simulation policy) and those governing supervised learning (size of data buffer, size of batches, number of sweeps across buffer per update step). While a range of other parameters (network architecture, initial learning rate etc.) were considered, their settings are less impactful across problem instances.

From this we conclude that our framework can – in principle – indeed construct Bayes-optimal policies and successfully solve the ARL problem. However, hyper-parameters are essential in striking a balance between knowledge consolidation and innovation.

The former consists of generating learning targets along trajectories likely under the current policy. This is necessary to ensure the quality of bootstrap estimates for future tree searches. Unfortunately, this process is relatively slow, as estimates have to be incorporated into the network before further simulations can generate more accurate learning targets. Additional care has to be taken to avoid high degrees of non-stationarity and “catastrophic forgetting”[154, 122] of acquired knowledge, which would further slow down training.

The latter, however, requires repeated deviation from the same trajectories to discover better performing alternatives. Similar to knowledge consolidation, this has to occur in several stages, as bootstrap estimates in “off-policy regions” have to be improved before the agent is able to realise the value of seeking them out.

These two learning objectives stand in direct competition to one another. If the agent explores slowly it will gather more accurate estimates but be unable to identify improvements consistently enough to incorporate them into the bootstrap. Too fast rates of innovation, however, cause learning instabilities.

In the ARL setting it is particularly difficult to find the correct balance. The need to average across the agent’s beliefs introduces a higher variance into our learning targets delaying knowledge consolidation. Innovation is also hindered as it is necessary to discover relatively specific action sequences that are “hidden” behind initially poor performance. This stands in stark contrast to the domains in which AlphaZero has been previously tested on (mainly game settings). Here,

rewards only occur at the end of trajectories (giving no early indicators that might bias our policy) and transitions are deterministic.

Finally, the representation of hyperstates may not be particularly amenable to generalisation by function approximation. Under the chosen count-based representation of interaction histories, similar feature vectors may encode beliefs with drastically different optimal policies and value estimates. Remedying this, by developing more appropriate representations, or even learning them, has the potential to speed up learning significantly and at least partially reduce the sensitivity to hyperparameters.

Another way of easing the trade-off between learning objectives would be the computation of uncertainty bounds for model predictions as a measure of learning progress. This would allow the agent to allocate simulations more gainfully to either stabilise value estimates or explore policies with high associated uncertainties. Provided they are well-calibrated, such bounds would also remove the need to “blindly” recompute the same information to combat forgetting, as knowledge loss should be indicated by an increase in uncertainty. A significant challenge in maintaining such uncertainty bounds, however, is that uncertainties about values and policies correlate – even small changes in policy may have significant ramifications for associated values and vice versa.

4.8 Discussion

Active RL is a twist on standard RL in which the cost of evaluating the reward of actions is incorporated into the agent’s objective. It is motivated by settings where rewards are constructed incrementally online, as when humans provide feedback to a learning agent. In this paper we reduced the formal problem description to a Bayes-adaptive formulation and discussed why existing approaches in large parts fail pathologically in ARL. Due to their empirical success in the past, we paid closer attention to planning based solutions and provided an adaptation of AlphaZero for BAMDPs. Both standard tree search and our proposed framework

fall short of expectations in practice and we discussed possible causes and avenues for future research.

Statement of authorship

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor (**only required where there isn't already a statement of contribution within the paper itself**).

Title of Paper	Active Reinforcement Learning with Monte-Carlo Tree Search
Publication Status	☐ Published ☐ Accepted for Publication ☐ Submitted for Publication ☒ Unpublished and unsubmitted work written in a manuscript style
Publication Details	The content published in the thesis will be an extension of work made publicly available in: Sebastian Schulze, and Owain Evans. "Active Reinforcement Learning with Monte-Carlo Tree Search." arXiv preprint arXiv:1803.04926, 2018.

Student Confirmation

Student Name:	Sebastian Schulze		
Contribution to the Paper	Contributions to the original manuscript were: • joint discussion of theoretical approaches / research impulses • implementation of framework and experiments • joint experiment design and result analysis • joint write-up of manuscript The extended version includes additions to the theoretical part of the manuscript as well as additional experiments and their discussion solely designed and executed by myself.		
Signature:		Date	13/04/2021

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Prof Michael A. Osborne			
Supervisor comments			
All above confirmed.			
Signature		Date	14 April 2021

This completed form should be included in the thesis, at the end of the relevant chapter.

5

Bayesian Optimisation for Iterative Learning

Contents

5.1	Abstract	70
5.2	Introduction	70
5.3	Related Work in Iteration-Efficient Bayesian Optimisation	72
5.4	Bayesian Optimisation for Iterative Learning (BOIL) .	73
5.4.1	Selecting points using iteration-efficient modelling . . .	74
5.4.2	Training curve compression and estimating the transformation function	75
5.4.3	Augmenting the training data	77
5.5	Experiments	79
5.5.1	Model illustration	80
5.5.2	Ablation study: curve compression	81
5.5.3	Tuning deep reinforcement learning and CNN	82
5.6	Conclusion and Future work	83

Preface

Previous chapters solely focused on the agent’s post-deployment ability in terms of overall performance and adaptability. However, even the best agent design is of little practical use if pre-deployment training for a specific application consumes excessive

and unjustifiable resource quantities (e.g. time or computation). This particularly gains importance in light of recent concerns regarding the energy consumption and environmental impact of modern machine learning [182, 3].

A large concern here is that most ML systems can not be deployed “out-of-the-box” but need to be properly configured to the task at hand. Since unfavourable starting conditions can delay training termination, produce unstable solutions or even result in complete training failure, it is often necessary to repeatedly train the agent until the desired training outcome is obtained. As machine learning systems increase in complexity, the cost of individual training trials grows, adding to the above-mentioned resource consumption. At the same time, estimating training outcomes based on initial conditions becomes non-trivial rendering manual configuration nigh impossible. Consequently, interest in principled and automated solutions to this problem has grown in recent years.

Apart from the increase in training efficiency, such approaches promise a range of secondary benefits. As intimate knowledge of ML systems is no longer a prerequisite for their deployment, they become more easily accessible, allowing a broader audience to directly profit from their development both in industry and potentially even (parts of) the broader public.

For the scientific community, automated hyperparameter tuning also provides a means of establishing that benchmark models are configured properly for the test scenarios at hand. This in turn ensures a fair comparison and easier reproducibility – a lack of which the machine learning community has been criticised for in the past [78].

Finally such systems automatically rank training outcomes when assessing different configurations. Thereby, users are free to specify (almost) any criteria enabling them to combine or trade-off various desiderata such as performance, robustness or interpretability of the final trained system.

The following paper aims to further the state-of-the-art in developing such systems that efficiently identify stable configurations of hyperparameters. We thereby follow a Bayesian Optimisation approach which tries to deduce the utility at untested setups from previous training runs. We note that many contemporary

ML systems are trained in an iterative fashion and show how to exploit the accruing intermediate results. An empiric assessment on several different scenarios shows that the proposed approach is able to compete with existing methods both in terms of the final performance of trained models as well as the time needed to identify them.

We believe that the presented approach represents a step towards the construction of reliable and resource-efficient AutoML-systems.

Bayesian Optimisation has enjoyed much attention in this domain and our framework further increases its efficiency. Specifically, our surrogate explicitly models the progression through different learning phases allowing for highly targeted acquisition of knowledge. Whereas previous approaches typically focus on the final performance of a parameter setup at the end of this progression, our novel objective function also promotes stable learning behaviours before that point. We leave the choice of desirable learning curve shapes to the practitioner and believe there is open future work in identifying and encoding other criteria for assessing machine learning pipelines.

Contributions in the area of AutoML are often orthogonal to one another and an interesting question lies in how to make use of domain knowledge to select the most appropriate selection of components from the ever-growing toolbox of (Bayesian Optimisation) methods.

5.1 Abstract

The performance of deep (reinforcement) learning systems crucially depends on the choice of hyperparameters. Their tuning is notoriously expensive, typically requiring an iterative training process to run for numerous steps to convergence. Traditional tuning algorithms only consider the final performance of hyperparameters acquired after many expensive iterations and ignore intermediate information from earlier training steps. In this paper, we present a Bayesian optimisation (BO) approach which exploits the iterative structure of learning algorithms for efficient hyperparameter tuning. We propose to learn an evaluation function compressing learning progress at any stage of the training process into a single numeric score according to both training success and stability. Our BO framework is then balancing the benefit of assessing a hyperparameter setting over additional training steps against their computation cost. We further increase model efficiency by selectively including scores from different training steps for any evaluated hyperparameter set. We demonstrate the efficiency of our algorithm by tuning hyperparameters for the training of deep reinforcement learning agents and convolutional neural networks. Our algorithm outperforms all existing baselines in identifying optimal hyperparameters in minimal time.

5.2 Introduction

Deep learning (DL) and deep reinforcement learning (DRL) have led to impressive breakthroughs in a broad range of applications such as game play [127, 171], motor control [195], and image recognition [104]. To maintain general applicability, these algorithms expose sets of hyperparameters to adapt their behaviour to any particular task at hand. This flexibility comes at the price of having to tune an additional set of parameters – poor settings lead to drastic performance losses [78, 142, 174]. On top of being notoriously sensitive to these choices, deep (reinforcement) learning systems often have high training costs, in computational resources and time. For example, a single training run on the Atari Breakout game took approximately 75

hours on a GPU cluster [127]. Tuning DRL parameters is further complicated as only noisy evaluations of an agent’s final performance are obtainable.

Bayesian optimisation (BO) [79, 132, 169] has recently achieved considerable success in optimising these hyperparameters. This approach casts the tuning process as a global optimisation problem based on noisy evaluations of a black-box function f . BO constructs a surrogate model typically using a Gaussian process (GP) [153], over this unknown function. This GP surrogate is used to build an acquisition function [81, 201] which suggests the next hyperparameter to evaluate.

In modern machine learning (ML) algorithms [90], the training process is often conducted in an iterative manner. A natural example is given by deep learning where training is usually based on stochastic gradient descent and other iterative procedures. Similarly, the training of reinforcement learning agents is mostly carried out using multiple episodes. The knowledge accumulated during these training iterations can be useful to inform BO. However, most existing BO approaches [169] define the objective function as the average performance over the final training iterations. In doing so, they ignore the information contained in the preceding training steps.

In this paper, we present a Bayesian optimisation approach for tuning algorithms where iterative learning is available – the cases of deep learning and deep reinforcement learning. First, we consider the joint space of input hyperparameters and number of training iterations to capture the learning progress at different time steps in the training process. We then propose to transform the whole training curve into a numeric score according to user preference. To learn across the joint space efficiently, we introduce a data augmentation technique leveraging intermediate information from the iterative process. By exploiting the iterative structure of training procedures, we encourage our algorithm to consider running a larger number of cheap (but high-utility) experiments, when cost-ignorant algorithms would only be able to run a few expensive ones. We demonstrate the efficiency of our algorithm on training DRL agents on several well-known benchmarks as well as the training of convolutional neural networks. In particular, our algorithm

outperforms existing baselines in finding the best hyperparameter in terms of wall-clock time. Our main contributions are:

- an algorithm to optimise training progressions of ML algorithms based on training curve convolution, instead of averaged final performance;
- an approach to learn the compression function from data and a data augmentation technique for increased sample-efficiency;
- an empirical assessment of our framework.

5.3 Related Work in Iteration-Efficient Bayesian Optimisation

Approaches to efficient (Bayesian) optimisation of black-box functions can roughly be divided into two categories.

The first employs stopping criteria to terminate some training runs early and allocate resources towards more promising settings. These criteria typically involve projecting towards a final score from early training stages. Freeze-thaw BO [189] models the training loss over time using a GP regressor under the assumption that the training loss roughly follows an exponential decay. Based on this projection, training resources are allocated to the most promising settings. Hyperband [54, 113] dynamically allocates computational resources (e.g. training epochs or dataset size) through random sampling and eliminates under-performing hyperparameter settings by successive halving.

Attempts have also been made to improve the epoch efficiency of other hyperparameter optimisation algorithms in [36, 42, 99] which predict the final learning outcome based on partially trained learning curves to identify hyperparameter settings that are expected to under-perform and early-stop them. In the context of DRL, however, these stopping criteria, including the exponential decay assumed in Freeze-thaw BO [189], may not be applicable, due to the unpredictable fluctuations of DRL reward curves.

The second category [93, 98, 113, 190, 213] aims to reduce the resource consumption of BO by utilising low-fidelity functions which can be obtained by using a subset of the training data or by training the ML model for a small number of iterations. Multi-task BO [190] requires the user to define a division of the dataset into predefined and discrete subtasks. Multi-fidelity BO with continuous approximation (BOCA) [93] and hierarchical partition [167] extend this idea to continuous settings. Specifically, BOCA first selects the hyperparameter input and then the corresponding fidelity to be evaluated at. The fidelity in this context refers to the use of different number of learning iterations. Analogous to BOCA’s consideration of continuous fidelities, Fabolas [98] proposes to model the combined space of input hyperparameter and dataset size and then select the optimal input and dataset size jointly.

The above approaches typically identify performance of hyperparameters via the average (either training or validation) loss of the last learning iterations. Thereby, they do not account for potential noise in the learning process (e.g., they might select unstable settings that jumped to high performance in the last couple of iterations).

5.4 Bayesian Optimisation for Iterative Learning (BOIL)

Problem setting. We consider training a machine learning algorithm given a d -dimensional hyperparameter $\mathbf{x} \in \mathcal{X} \subset \mathcal{R}^d$ for t iterations. This process has a training time cost $c(\mathbf{x}, t)$ and produces training evaluations $r(\cdot \mid \mathbf{x}, t)$ for t iterations, $t \in [T_{\min}, T_{\max}]$. These could be episode rewards in DRL or training accuracies in DL. An important property of iterative training is that we observe intermediate performances at preceding steps $r(t' \mid \mathbf{x}, t)$, $\forall t' \leq t$.

We assume that performances are noise-corrupted observations of an underlying smooth black-box function f . Formally, we aim to find $\mathbf{x}^* = \arg \max_{\mathbf{x} \in X} f(\mathbf{x}, T_{\max})$; at the same time, we want to keep the overall training time, $\sum_{i=1}^N c(\mathbf{x}_i, t_i)$, of evaluated settings $[\mathbf{x}_i, t_i]$ as low as possible.

5.4.1 Selecting points using iteration-efficient modelling

We follow popular designs in [98, 103, 177, 190] and model the cost-sensitive black-box function as $f(\mathbf{x}, t) \sim \text{GP}(0, k([\mathbf{x}, t], [\mathbf{x}', t']))$, where k is an appropriate covariance function and $[\mathbf{x}, t] \in \mathcal{R}^{d+1}$. For simplicity and robustness, the cost function $c(\mathbf{x}, t)$ is approximated by a linear regressor. Depending on the setting, it may be more appropriate to employ a second GP or different parametric model if the cost has a more complex dependence on hyperparameters $\mathbf{x}$ and iterations t . We regularly (re-)optimise both kernel and cost function parameters in between point acquisitions.

More specifically, we choose the covariance function as a product $k([\mathbf{x}, t], [\mathbf{x}', t']) = k(\mathbf{x}, \mathbf{x}') \times k(t, t')$ to induce joint similarities over parameter and iteration space. We estimate predictive mean and uncertainty of a GP [153] at any input $\mathbf{z}_* = [\mathbf{x}_*, t_*]$ as

$$\mu(\mathbf{z}_*) = \mathbf{k}_* [\mathbf{K} + \sigma_y^2 \mathbf{I}]^{-1} \mathbf{y} \quad (5.1) \quad \sigma^2(\mathbf{z}_*) = k_{**} - \mathbf{k}_* [\mathbf{K} + \sigma_y^2 \mathbf{I}]^{-1} \mathbf{k}_*^T \quad (5.2)$$

where $\mathbf{y} = [y_i]_{\forall i}$, $\mathbf{k}_* = [k(\mathbf{z}_*, \mathbf{z}_i)]_{\forall i}$, $\mathbf{K} = [k(\mathbf{z}_i, \mathbf{z}_j)]_{\forall i, j}$, $k_{**} = k(\mathbf{z}_*, \mathbf{z}_*)$, and σ_y^2 is the noise variance of f . Cost predictions at any particular parameter $\mathbf{x}$ and time t are given by $\mu_c([\mathbf{x}_*, t_*]) = \beta^T [\mathbf{x}, t]$, where β is directly computed from data $\{Z = [\mathbf{x}_i, t_i], \mathbf{c} = [c_i]\}_{\forall i}$ as $\beta = (Z^T Z)^{-1} Z \mathbf{c}$ [20].

Our goal is to select a point with high function value (exploitation), high uncertainty (exploration) and low cost (expense). At each iteration n , we query the input parameter $\mathbf{x}_n$ and the number of iteration t_n [176, 213]:

$$\mathbf{z}_n = [\mathbf{x}_n, t_n] = \arg \max_{\mathbf{x} \in \mathcal{X}, t \in [T_{\min}, T_{\max}]} \alpha(\mathbf{x}, t) / \mu_c(\mathbf{x}, t). \quad (5.3)$$

Although our framework is available for any acquisition choices [81, 112, 212], to cope with output noise, we follow [202] and slightly modify the expected improvement criterion using the maximum mean GP prediction $\mu_n^{\max}$. Let $\lambda = \frac{\mu_n(\mathbf{z}) - \mu_n^{\max}}{\sigma_n(\mathbf{z})}$, we then have a closed-form for the new expected improvement (EI) as $\alpha_n^{\text{EI}}(\mathbf{z}) = \sigma_n(\mathbf{z}) \phi(\lambda) + [\mu_n(\mathbf{z}) - \mu_n^{\max}] \Phi(\lambda)$ where ϕ is the standard normal p.d.f., Φ is the c.d.f., μ_n and σ_n are the GP predictive mean and variance defined in Eq. (5.1) and Eq. (5.2), respectively.

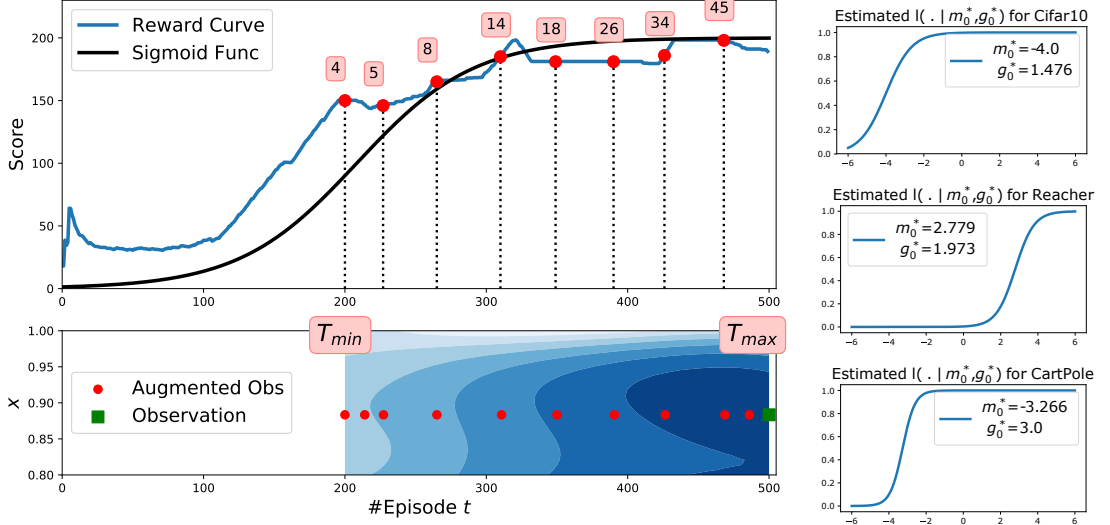


Figure 5.1: Left: the score in pink box is a convolution of the reward curve $r(\cdot \mid \mathbf{x} = 0.9, t = 500)$ and a Sigmoid function $l(u \mid g_0, m_0) = \frac{1}{1 + \exp(-g_0[u - m_0])}$ up to time step t . Bottom: observations are selected to augment the dataset (red dots). The heatmap indicates the GP predictive mean μ for f across the number of episodes t used to train an agent. $T_{\min}$ and $T_{\max}$ are two user-defined thresholds for the number of training episodes. x is a hyperparameter to be tuned. Right: learned compression parameters g_0^* and m_0^* for different experiments.

5.4.2 Training curve compression and estimating the transformation function

Existing BO approaches [31, 113] typically define the objective function as an average loss over the final learning episodes. However, this does not take into consideration how stable performance is or the training stage at which it has been achieved. We argue that averaging learning losses is likely misleading due to the noise and fluctuations of our observations (learning curves) – particularly during the early stages of training. We propose to compress the whole learning curve into a numeric score via a preference function representing the user’s desired training curve. In the following, we use the Sigmoid function (specifically the Logistic function) to compute the utility score as

$$y = \hat{y}(r, m_0, g_0) = r(\cdot \mid \mathbf{x}, t) \bullet l(\cdot \mid m_0, g_0) = \sum_{u=1}^t \frac{r(u \mid \mathbf{x}, t)}{1 + \exp(-g_0[u - m_0])} \quad (5.4)$$

where $\bullet$ is a dot product, a Logistic function $l(\cdot \mid m_0, g_0)$ is parametrised by a growth parameter g_0 defining a slope and the middle point of the curve m_0 . The

optimal parameters g_0 and m_0 are estimated directly from the data. We illustrate different shapes of l parametrised by g_0 and m_0 in the appendix. The Sigmoid preference has a number of desirable properties. As early weights are small, less credit is given to fluctuations at the initial stages, making it less likely for our surrogate to be biased towards randomly well performing settings. However, as weights monotonically increase, hyperparameters with improving performance are preferred. As weights saturate over time, stable, high performing configurations are preferred over short “performance spikes” characteristic of unstable training. Lastly, this utility score assigns higher values to the same performance if it is being maintained over more episodes.

Learning the transformation function from data. Different compression functions l , parametrised by different choices of g_0 and m_0 in Eq. (5.4), produce different utilities y and may thus affect predictive performance. Optimal values g_0^* and m_0^* are unknown in advance and we propose to learn them as GP hyperparameters from raw observation data. Our intuition is that the ‘optimal’ compression curve $l(m_0^*, g_0^*)$ devalues performances in overly noisy training periods to allow a better fit of the GP, resulting in more accurate predictions as well as optimisation performance.

We parametrise the GP log marginal likelihood L as the function of m_0 and g_0 :

$$L(m_0, g_0) = \frac{1}{2} \hat{\mathbf{y}}^T (\mathbf{K} + \sigma_y^2 \mathbf{I})^{-1} \hat{\mathbf{y}} - \frac{1}{2} \ln |\mathbf{K} + \sigma_y^2 \mathbf{I}| + \text{const} \quad (5.5)$$

where σ_y^2 is the output noise variance, $\hat{\mathbf{y}}$ is the function of m_0 and g_0 defined in Eq. (5.4). We optimise m_0 and g_0 (jointly with other GP hyperparameters) using multi-start gradient descent. We derive the derivative $\frac{\partial L}{\partial m_0} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial m_0}$ and $\frac{\partial L}{\partial g_0} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial g_0}$ which can be computed analytically as:

$$\frac{\partial L}{\partial \hat{y}} = (\mathbf{K} + \sigma_y^2 \mathbf{I}_N)^{-1} \hat{\mathbf{y}}; \quad \frac{\partial \hat{y}}{\partial m_0} = \frac{-g_0 \times e^{-g_0(u-m_0)}}{[1 + e^{-g_0(u-m_0)}]^2}; \quad \frac{\partial \hat{y}}{\partial g_0} = \frac{-m_0 \times e^{-g_0(u-m_0)}}{[1 + e^{-g_0(u-m_0)}]^2}.$$

The estimated compression curves are illustrated in Fig. 5.1 and Sec. 5.5.1.

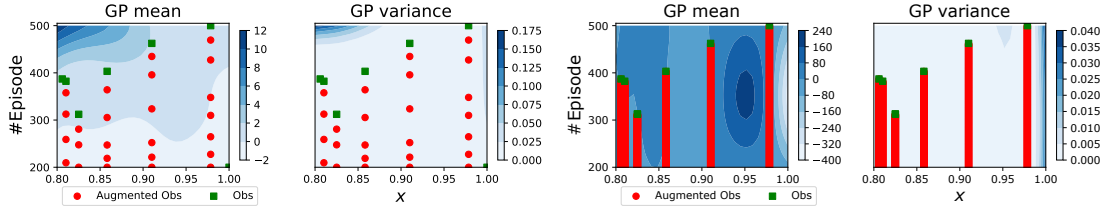


Figure 5.2: Two data augmentation strategies. Left: Our sparsification. Right: Full data inclusion. If too many observations are added, the GP covariance matrix becomes ill-conditioned. On the right, the GP fit is poor with a large mean estimate range of $[-400, 240]$ even though the output is standardised $\mathcal{N}(0, 1)$. All x-axis are over x , a hyperparameter to be tuned.

5.4.3 Augmenting the training data

When evaluating a parameter $\mathbf{x}$ over t iterations, we obtain not only a final score but also all reward sequences $r(t' | \mathbf{x}, t), \forall t' = 1, \dots, t$. The intermediate information from the learning curve can be useful for BO. Therefore, we propose to augment the sample set of our GP model with some of these additional observations.

A naïve approach to augmentation would add a full curve of points $\{[\mathbf{x}, j], y_j\}_{j=1}^t$ where y_j is computed using Eq. (5.4). However, this approach includes redundant data and may impose serious issues in the conditioning of the GP covariance matrix. As we cluster more evaluations closely, the conditioning of the GP covariance degrades further, as discussed in [123]. This conditioning issue is especially serious in our noisy DRL settings.

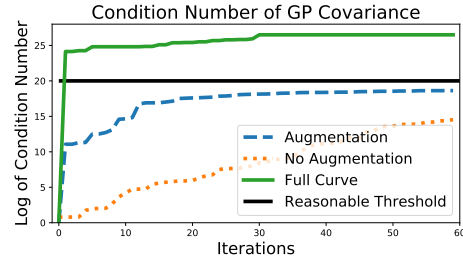


Figure 5.3: The condition number of our GP covariance matrix deteriorates if we add the whole curve of points into a GP. Large condition numbers indicate proximity to singularity.

We highlight this effect on GP estimation in Fig. 5.2 wherein the GP mean varies erratically when the *natural log* of the condition number of the GP covariance matrix goes above 25 (see Fig. 5.3) as we include the whole curve.

Selecting subsets of points from the curve. Different solutions, such as the addition of artificial noise or altering the kernel’s length-scales, have been proposed.

We decide to use an active learning approach [59, 140] as sampled data points are expected to contain a lot of redundant information. As a consequence, the loss of information from sub-sampling the data should be minimal and information-eroding modification of the kernel matrix itself can be avoided. As a side benefit, the reduced number of sampled points speeds up inference in our GP models.

In particular, we select samples at the maximum of the GP predictive uncertainty. Formally, we sequentially select a set $Z = [z_1, \dots, z_M]$, $z_m = [\mathbf{x}, t_m]$, by varying t_m while keeping $\mathbf{x}$ fixed as

$$z_m = \arg \max_{\forall t' \leq t} \sigma([\mathbf{x}, t'] \mid D'), \forall m \leq M \text{ s.t. } \ln \text{ of cond}(K) \leq \delta \quad (5.6)$$

where $D' = D \cup \{z_j = [\mathbf{x}, t_j]\}_{j=1}^{m-1}$. This sub-optimisation problem is done in a one-dimensional space of $t' \in \{T_{\min}, \dots, t\}$, thus it is *cheap* to optimise using (multi-start) gradient descent (the derivative of GP predictive variance is available [153]). Alternatively, a fixed-size grid could be considered, but this could cause conditioning issues when a point in the grid $[\mathbf{x}, t_{\text{grid}}]$ is placed near another existing point $[\mathbf{x}', t_{\text{grid}}]$, i.e., $\|\mathbf{x} - \mathbf{x}'\|_2 \leq \epsilon$ for some small ϵ .

These generated points Z are used to calculate the output $r(z_m)$ and augmented into the observation set (X, Y) for fitting the GP. The number of samples M is adaptively chosen such that the natural log of the condition number of the covariance matrix is less than a threshold. This is to ensure that the GP covariance matrix condition number behaves well by reducing the number of unnecessary points added to the GP at later stages. We compute the utility score y_m given z_m for each augmented point using Eq. (5.4). In addition, we can estimate the running time c_m using the predictive mean $\mu_c(z_m)$. We illustrate the augmented observations and estimated scores in Fig. 5.1.

We summarise the overall algorithm in Alg. 2. To enforce non-negativity and numerical stability, we make use of the transformations $\alpha \leftarrow \log[1 + \exp(\alpha)]$ and $\mu_c \leftarrow \log[1 + \exp(\mu_c)]$.

Algorithm 2 Bayesian Optimization with Iterative Learning (BOIL)

Input: #iter N , initial data D_0 , $\mathbf{z} = [\mathbf{x}, t]$.**Output:** optimal $\mathbf{x}^*$ and $y^* = \max_{y \in D_N} y$ **for** $n = 1 \dots N$ **do**Fit GP to estimate μ_f, σ_f from Eqs. (5.1, 5.2) and LR for cost μ_c Select $\mathbf{z}_n = \arg \max_{\mathbf{x}, t} \alpha(\mathbf{x}, t) / \mu_c(\mathbf{x}, t)$ and observe curve and cost of $f(\mathbf{z}_n)$ Compress learning curve $r(\mathbf{z}_n)$ into numeric scores using Eq. (5.4).Sample intermediate points $\mathbf{z}_{n,m}, y_{n,m}, c_{n,m}, \forall m \leq M$ according to Eq. (5.6)Augment D_n and estimate hyperparameters m_0 and g_0 .**end for**

5.5 Experiments

We assess our model by tuning hyperparameters for two DRL agents on three environments and a CNN on two datasets. We provide additional illustrations and experiments in the appendix.

Experimental setup. All experimental results are averaged over 20 independent runs with different random seeds. Final performance is estimated by evaluating the chosen hyperparameter over the maximum number of iterations. All experiments are executed on a NVIDIA 1080 GTX GPU using the tensorflow-gpu Python package. The DRL environments are available through the OpenAI gym [24] and Mujoco [195]. Our DRL implementations are based on the open source from Open AI Baselines [41].

We use square-exponential kernels for the GP in our model and estimate their parameters by maximising the marginal likelihood [153]. We set the maximum number of augmented points to be $M = 15$ and a threshold for a natural log of GP condition number $\delta = 20$. We note that the optimisation overhead is much less than the black-box function evaluation time.

Baselines. We compare with Hyperband [113] which demonstrated empirical successes in tuning deep learning applications in an iteration-efficient manner. We extend the discrete multi-task BO [190] to the continuous case – which can also be seen as continuous multi-fidelity BO [93, 177] as in our setting, they both consider cost-sensitivity and iteration-efficiency. We, therefore, label the two baselines

as continuous multi-task/fidelity BO (CM-T/F-BO). We have ignored the minor difference in these settings, such as multi-task approaches jointly optimises the fidelity and input while BOCA [93] first selects the input and then the fidelity.

Our focus is to demonstrate the effectiveness of optimising the learning curve using compression and augmentation techniques. We therefore omit the comparison of various acquisition functions and kernel choices which can easily be used in our model. We also do not compare with Fabolas [98] which is designed to vary dataset sizes, not iteration numbers. We would expect the performance of Fabolas to be close to CM-T/F-BO. We are unable to compare with FreezeThaw as the code is not available. However, the curves in our setting are not exponential decays and thus ill-suited to their model (see last figure in the appendix). We have considered an ablation study in the appendix using a time kernel following the exponential decay proposed in Freeze-thaw method [189].

Task descriptions. We consider three DRL settings including a Dueling DQN (DDQN) [203] agent in the CartPole-v0 environment and Advantage Actor Critic (A2C) [125] agents in the InvertedPendulum-v2 and Reacher-v2 environments. In addition to the DRL applications, we tune 6 hyperparameters for training a convolutional neural network [110] on the SVHN dataset and CIFAR10. Due to space considerations, we refer to the appendix for further details.

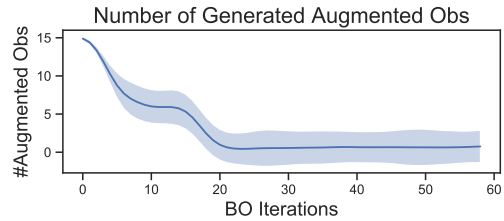


Figure 5.4: DDQN on CartPole. The number of augmented observations reduces over time.

5.5.1 Model illustration

We first illustrate the estimated compression functions $l(m_0^*, g_0^*)$ in Fig. 5.1. The parameters g_0^* and m_0^* are estimated by maximising the GP marginal likelihood and used for compressing training progressions. We see that the estimated curve

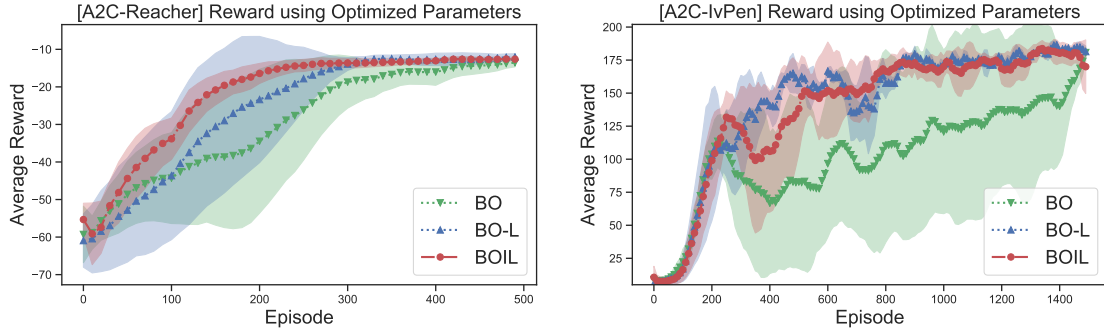


Figure 5.5: The learning curves of the best found parameters by different approaches. The curves show that BO-L and BOIL reliably identify parameters leading to stable training. BOIL takes only half total time to find this optimal curve.

from CartPole tends to reach the highest performance much earlier than Reacher, presumably because CartPole is an easier task than Reacher.

We next examine the count of additional observations generated per iteration in Fig. 5.4. Although this number is fluctuating, it tends to reduce over time. BOIL does not add more augmented observations at the later stage when we have gained sufficient information and GP covariance conditioning falls below our threshold $\delta = 20$.

5.5.2 Ablation study: curve compression

To demonstrate the impact of our training curve compression, we compare BOIL to vanilla Bayesian optimisation (BO) and with compression (BO-L) given the same number of iterations at $T_{\max}$. We show that using the curve compression leads to stable performance, as opposed to the existing technique of averaging the last iterations. We plot the learning curves of the best hyperparameters identified by BO, BO-L and BOIL. Fig. 5.5 shows the learning progress over $T_{\max}$ episodes for each of these. The curves are smoothed by averaging over 100 consecutive episodes for increased clarity. We first note that all three algorithms eventually obtain similar performance at the end of learning. However, since BO-L and BOIL take into account the preceding learning steps, they achieve higher performance more quickly. Furthermore, they achieve this more reliably as evidenced by the smaller error bars (shaded regions).

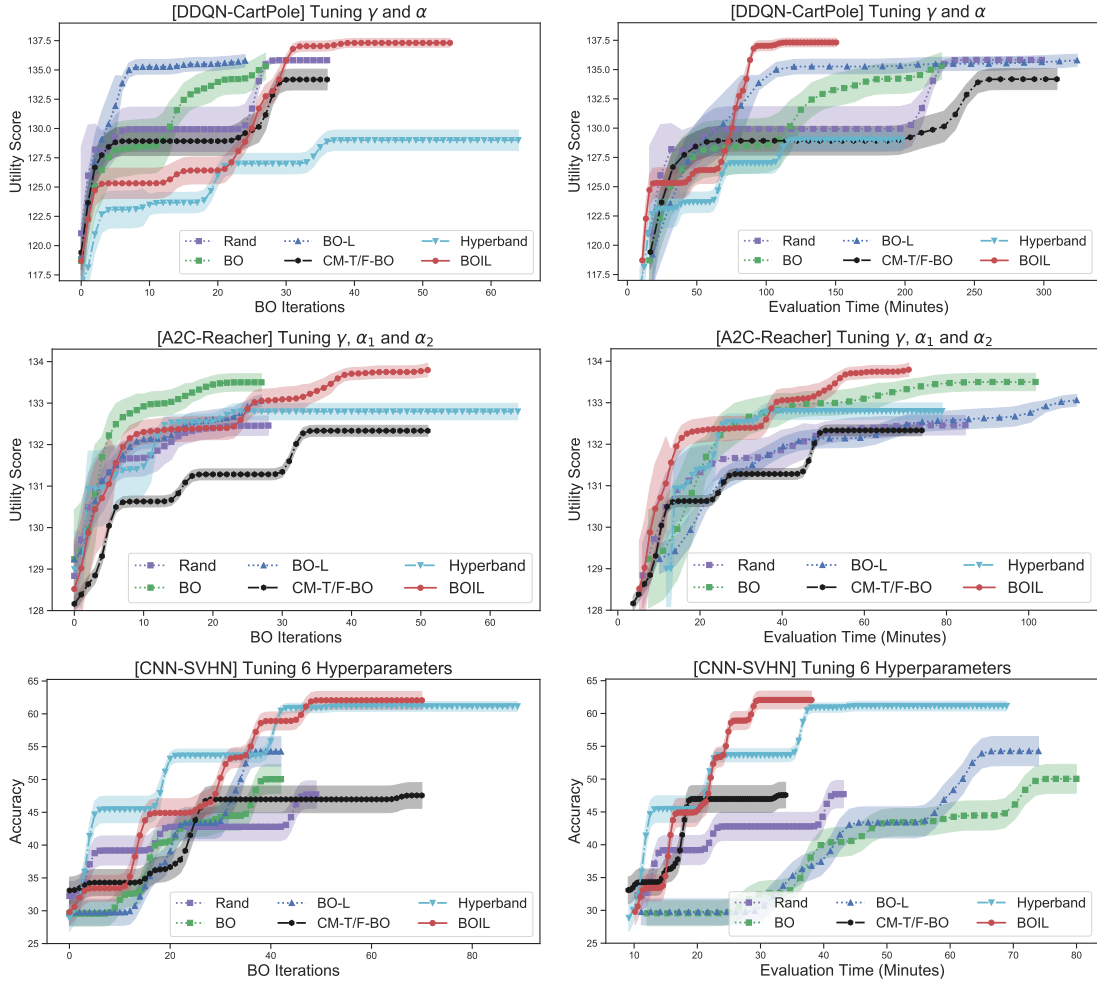


Figure 5.6: Comparison over BO evaluations (Left) and real-time (Right). Given the same time budget, CM-T/F-BO, Hyperband and BOIL can take more evaluations than vanilla BO, BO-L and Rand. BOIL outperforms other competitors in finding the optimal parameters in an iteration-efficient manner. CM-T/F-BO does not augment the observations from the curve and requires more evaluations. The results of InvertedPendulum and CNN-CIFAR10 are in the appendix.

5.5.3 Tuning deep reinforcement learning and CNN

We now optimise hyperparameters for deep reinforcement learning algorithms; in fact, this application motivated the development of BOIL. The combinations of hyperparameters to be tuned, target DRL algorithm and environment can be found in the appendix.

Comparison by iterations and real-time. Fig. 5.6 illustrates the performance of different algorithms against the number of iterations as well as real-time (the

plots for CIFAR10 are in the appendix). The performance is the utility score of the best hyperparameters identified by the baselines. Across all three tasks, BOIL identifies optimal hyperparameters using significantly less computation time than other approaches.

The plots show that other approaches such as BO and BO-L can identify well-performing hyperparameters in fewer iterations than BOIL. However, they do so only considering costly, high-fidelity evaluations resulting in significantly higher evaluation times. In contrast to this behaviour, BOIL accounts for the evaluation costs and chooses to initially evaluate low-fidelity settings consuming less time. This allows fast assessments of a multitude of hyperparameters. The information gathered here is then used to inform later point acquisitions. Hereby, the inclusion of augmented observations is crucial in offering useful information readily available from the data. In addition, this augmentation is essential to prevent from the GP kernel issue instead of adding the full curve of points into our GP model.

Hyperband [113] exhibits similar behaviour in that it uses low fidelity (small t) evaluations to reduce a pool of randomly sampled configurations before evaluating at high fidelity (large t). To deal with noisy evaluations and other effects, this process is repeated several times. This puts Hyperband at a disadvantage particularly in the noisy DRL tasks. Since early performance fluctuates hugely, Hyperband can be misled in where to allocate evaluation effort. It is then incapable of revising these choices until an entirely new pool of hyperparameters is sampled and evaluated from scratch. In contrast to this, BOIL is more flexible than Hyperband in that it can freely explore-exploit the whole joint space. The GP surrogate hereby allows BOIL to generalise across hyperparameters and propagate information through the joint space.

5.6 Conclusion and Future work

Our framework complements the existing BO toolbox for hyperparameter tuning with iterative learning. We present a way of leveraging our understanding that later stages of the training process are informed by progress made in earlier ones.

This results in a more iteration-efficient hyperparameter tuning algorithm that is applicable to a broad range of machine learning systems. We evaluate its performance on a set of diverse benchmarks. The results demonstrate that our model surpasses the performance of well-established alternatives while consuming significantly fewer resources. Finally, we note that our approach is not necessarily specific to machine learning algorithms, but more generally applies to any process exhibiting an iterative structure to be exploited.

Statement of authorship

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor (**only required where there isn't already a statement of contribution within the paper itself**).

Title of Paper	Bayesian Optimization for Iterative Learning
Publication Status	☐ Published ☒ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and unsubmitted work written in a manuscript style
Publication Details	Nguyen, Vu, Sebastian, Schulze, and Michael, Osborne. "Bayesian Optimization for Iterative Learning." In <i>Advances in Neural Information Processing Systems</i> (2020)(pp. 9361–9371).

Student Confirmation

Student Name:	Sebastian Schulze
Contribution to the Paper	- joint discussion of overall theoretical approach and specific extensions of the Bayesian Optimisation framework - implementation of RL setups - joint experiment design and result analysis - write-up of manuscript
Signature: 	Date 23/04/2021

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Prof Michael A. Osborne	
Supervisor comments All above confirmed.	
 Signature	Date 14 April 2021

This completed form should be included in the thesis, at the end of the relevant chapter.

6

Marginalising Stationary Kernels across Spectral Densities

Contents

6.1	Abstract	89
6.2	Introduction	89
6.3	Background	91
6.3.1	Gaussian Processes	91
6.3.2	Bayesian Quadrature	92
6.4	Our Method: MASKERADE	93
6.4.1	The generative model	93
6.4.2	Posterior Inference	95
6.4.3	Bayesian Quadrature	96
6.4.4	Computational Complexity	99
6.5	Related Work	100
6.5.1	Kernel Learning	100
6.5.2	Bayesian Quadrature	101
6.5.3	Gaussian Processes on Spaces of Measures	102
6.6	Results	102
6.6.1	Experiment setup	103
6.6.2	Medium scale data sets	103
6.6.3	Large scale data sets	105
6.6.4	Ablation Study	105
6.7	Discussion	106

Preface

Optimisation of hyperparameters as discussed in the previous chapter allows practitioners to adjust general model behaviour. Taking an additional step backwards raises the question which model class is appropriate for which application in the first instance. Sadly, there is no universal answer and various factors affect this decision. These include but are not limited to the amount of training data and time available, the types of relationships to be modelled, the kind of predictions required and “soft” requirements such as interpretability and explainability of eventual results.

This chapter expands the capacity and applicability of Gaussian Processes [153], which were employed in the previous section. Their probabilistic foundation allows for principled, data-efficient inference with clear mechanisms for the description of prior knowledge and interpretation of derived posteriors. Traditionally, this made them popular in use-cases in which rich prior information could be exploited to enable learning progress based on very little data and early research often focused on how to encode problem specific priors. This choice of setting was also motivated by the computational cost inference (specifically the computation of likelihoods) incurs which scales poorly with data set size.

With rising interest in data-driven approaches to machine learning, poor scalability in data has become a significant limitation to GPs. As information of interest has to be extracted from large data sets with little access to prior knowledge, the standard GP formulation cannot match performance of e.g. connectionist models which lend themselves more easily to batch-training. However, as those models struggle with issues of interpretability – particularly the provision of well-calibrated uncertainty information – GPs have retained a significant research following. Accordingly, the focus of GP research has shifted towards scalability and the design of universal priors whilst retraining probabilistic soundness.

The former is typically achieved by making a variational, parametric approximation to the full GP model [193]. This approximation is amenable to training via local gradient-based optimisation in a batch fashion similar to connectionist models. Since likelihoods are computed per batch, scaling issues are mitigated.

This work is orthogonal to the variational GP approach – rather than reducing the cost of likelihood evaluations, we aim to reduce the number of evaluations necessary in the first instance. Following recent work, we consider the general class of stationary kernels and inference based on marginalisation rather than computing a point estimate.

The required integral across the entire kernel family is approximated efficiently by adopting a model-based Bayesian Quadrature approach. To adequately compare kernels, we introduce a novel “hyper-kernel” based on the maximum mean discrepancy between spectral representations. This encodes known representation invariances and maximises the reuse of acquired likelihood information. To further increase efficiency we describe how general properties of a dataset translate into a prior over kernels and extend targeted, information-theoretic point acquisition of likelihood evaluations to the batch setting.

We believe that our contributions bring Gaussian Processes one step closer to being a viable, highly flexible model choice in large data regimes. However, several limitations remain and we discuss those as well as how further research can address them in the future.

6.1 Abstract

Marginalising over families of Gaussian Process kernels produces flexible model classes with well-calibrated uncertainty estimates. Existing approaches require likelihood evaluations of many kernels, rendering them prohibitively expensive for larger datasets. We propose a Bayesian Quadrature scheme to make this marginalisation more efficient and thereby more practical. Through use of the maximum mean discrepancies between distributions, we define a kernel over kernels that captures invariances between Spectral Mixture (SM) Kernels. Kernel samples are selected by generalising an information-theoretic acquisition function for warped Bayesian Quadrature. We show that our framework achieves accurate predictions with better calibrated uncertainty than state-of-the-art baselines, especially when given limited (wall-clock) time budgets.

6.2 Introduction

Gaussian Processes (GPs) [153] are a rich class of models, which place probability distributions directly on classes of functions. Crucially, the success of these models is tied to the choice of their kernels.

Analogously to architecture design in deep learning, kernels control the expressiveness and complexity of a GP model. If the correct kernel is chosen, GPs have shown the ability to make accurate predictions based on comparatively small training data sets. Beyond that, they natively provide predictive uncertainties with little additional computation. These confidence measurements can be vital for various down-stream tasks such as decision-making in the real world.

If, however, the chosen kernel is misspecified, little probability mass will be assigned to the neighbourhood of the true function, resulting in a poor fit. Many commonly used kernels, such as the Radial Basis Function and Matérn kernels, have been shown to be universal kernels [119], meaning they can approximate arbitrary functions given sufficient data. Despite this they, in practice, encode

strong and frequently task-inappropriate inductive biases, significantly delaying learning progress.

To increase model flexibility and avoid such pathologies practitioners typically define parameterised *kernel families*. The selection of a particular kernel is delayed until training and made data-dependent, e.g. via a maximum likelihood estimate (MLE).

A fully Bayesian approach gains further expressivity and robustness by marginalising across the chosen family instead of settling on a single kernel instance for test-time predictions. This is particularly valuable when working with large, complex data sets that exhibit complicated interactions between data points.

Our approach aims to create a flexible regression model capable of representing arbitrary stationary kernels and efficiently learning complex functions from large data sets. We draw on several previously separate strands of research to achieve this.

The analysis of kernels in the spectral domain has been shown to be an effective tool in learning expressive kernels [209]. Due to Bochner [21] a link between stationary kernel functions and frequency measures can be established. The former is a general class of kernels encompassing many popular families (e.g. RBF, Matérn, periodic). The link to the latter allows the construction of posterior distributions in a principled and computationally efficient way with little user intervention and opens up clear avenues for their interpretation.

Rather than settling on a specific kernel via MLE, we conduct inference in the spectral-kernel framework by marginalising the entire kernel family against parameter likelihoods. The resulting integrals are computationally intractable and solutions can only be approximated. Previous approaches [136, 19, 173] based on Monte Carlo variants effectively average across likelihood evaluations of kernels randomly sampled from the posterior. Each likelihood evaluation requires an inversion of the kernel Gram matrix over all data points, making such approaches increasingly expensive as we scale to larger data sets.

To stay efficient even in large data, expensive likelihood settings, we adopt a model-based approach to integral approximation. Bayesian Quadrature (BQ) [135, 64] methods model the function to be integrated directly, to incorporate

and exploit prior knowledge of regularities (e.g. smoothness and non-negativity of likelihood surfaces [140]). This enables more careful sample acquisition, yielding performance superior to MCMC methods in wall-clock time for moderate-dimensional problems [71].

Throughout this paper, we make the following contributions. Firstly, the construction of a custom *hyper*-kernel based on the maximum mean discrepancy between distributions. This kernel is cheap to evaluate, whilst producing meaningful covariances between spectral densities, regardless of their parameterisation. Secondly, a BQ framework based on this kernel to conduct computationally efficient GP inference on large data sets. Thirdly, the derivation of an information-theoretic acquisition function for the selection of new parameter evaluations. Finally, we empirically demonstrate the viability of the resulting approach on several synthetic examples and real world data sets. Our algorithm chooses informative likelihood evaluations and achieves high performance (log-likelihoods) on limited computational budgets measured in wall-clock time.

6.3 Background

6.3.1 Gaussian Processes

A Gaussian Process (GP) defines a probability distribution over the space of functions $f : \mathcal{X} \rightarrow \mathbb{R}$, such that for any finite subset $\mathcal{X}' \subset \mathcal{X}$, the vector $\{f(x)\}_{x \in \mathcal{X}'}$ is normally distributed. We denote such a distribution $f \sim \mathcal{GP}(m, k)$, where the mean $m(x) = E[f(x)]$ and kernel functions $k(x_1, x_2) = E[(f(x_1) - m(x_1))(f(x_2) - m(x_2))]$ encode our prior beliefs about the function. Analytic expressions for posterior mean and kernel of a process conditioned on a set of observations D under a Normal likelihood are readily available.¹

¹A detailed discussion of the GP framework can be found in [153].

Spectral Covariance Functions

In this work the prior kernel functions $k(x_1, x_2)$ are assumed to be stationary. With slight abuse of notation, they take the form:

$$k(x_1, x_2) = k(|x_1 - x_2|) = k(\rho), \quad (6.1)$$

such that two function values covary only depending on their separation ρ and not their location.

Bochner's Theorem [21] states that $k(\rho)$ is a positive-definite function on $\mathbb{R}$ (and valid kernel) if and only if its Fourier transform, $S(\omega)$, is a positive spectral density, i.e.:

$$k(\rho) = \int e^{2\pi i \rho \omega} dS(\omega). \quad (6.2)$$

As a consequence, any parameterisation of a set of spectral measures induces a corresponding parameterisation of stationary kernels².

Kernels based on Distance Metrics

Distance-based kernels which often take the form $k(x_1, x_2) = \lambda^2 \exp\left(-\frac{d(x_1, x_2)^q}{l^2}\right)$ (λ , and l being hyper-parameters), are valid, i.e. positive definite, for $q = 1$ if and only if the underlying metric $d(x_1, x_2)$ is conditionally negative definite [88, 55], meaning that it must give rise to matrices, D , such that $c^T D c < 0, \forall c : \sum_i c_i = 0$. For $q = 2$ the underlying metric must be Hilbertian, meaning that there must be an isometry between the metric space defining d and a Hilbert space [55].

6.3.2 Bayesian Quadrature

Bayesian inference in machine learning frequently involves computation of intractable integrals of the form:

$$Z = \int f(x)p(x)dx \quad , \quad (6.3)$$

where $p(x)$ is a known prior density and $f(x)$ a (likelihood-)function. Bayesian quadrature is a model-based approach to approximately evaluating such integrals

²For $k(\rho)$ to be real, we only consider symmetric $S(\omega)$

by modelling f as a GP. Since GPs are closed under affine transforms the posterior over Z is then Gaussian. For suitable kernel-prior combinations the quadrature weights can be evaluated analytically.

While the maintenance of the GP model requires some computational effort, it enables the generalisation of sample information across the integration domain. Consequently, samples can be chosen in a targeted fashion and BQ has been shown to be a competitive, more evaluation-efficient alternative to Monte Carlo methods, particularly in moderate-dimensional domains.

Recent work [140, 71, 30] further increases sample efficiency by encoding model-constraints such as the positivity of likelihood functions using warped GPs. This work builds upon WSABI [71] in particular, which places a GP prior over the square-root of f and is described in more detail in Appendix C.1.

6.4 Our Method: MASKERADE

In the following we present a flexible, data efficient Gaussian Process framework and show how to conduct Bayesian inference within it.

6.4.1 The generative model

Without loss of generality, we assume input dimensions of the training data to be rescaled to the interval $[0, 1]$ and outputs to be normalised to have zero mean and unit variance.

By this construction, a zero mean prior for our GP model is the most logical choice. As mentioned above, kernels are only assumed to be stationary. Following Wilson and Adams [209] and Oliva et al. [136], we parameterise spectral densities through Gaussian Mixture Models (GMMs) to induce a corresponding parameterisation of stationary kernels. In the limit of infinitely many components GMMs can approximate any spectral density to arbitrary precision.

The kernel, matching a spectral density parameterised by θ , can be recovered via (6.2). To create only real-valued kernels, we reflect all mixtures at the origin. We also note that an additional output-scale $k(0)$ is required to cover the space

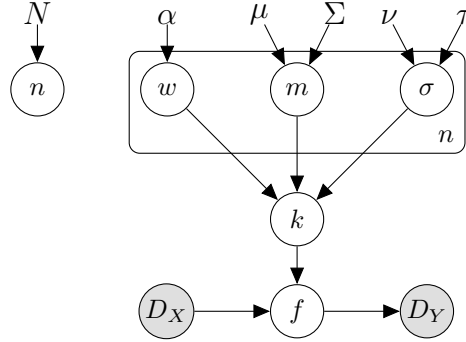


Figure 6.1: Bayesian Network of our model. The number of mixture components n is drawn from a uniform prior; α parametrises a Dirichlet distribution over mixture weights w ; μ and Σ parameterise Gaussians over mixture means; ν and τ parameterise Log-Normal distributions over mixture scales. Not shown are hyperparameters of the hyper-kernel, λ and l .

of (unnormalised) measures and recover arbitrary stationary kernels. Since data has unit variance, however, we omit this in the following.

To perform Bayesian inference and marginalise across stationary kernels, we define a hyperprior $p(\theta)$ (see next section). A graphical representation of the resulting generative model is shown in Figure 6.1.

The Hyperprior $p(\theta)$

The definition of our Bayesian model necessitates the choice of a suitable hyperprior. We see this is an advantage to the practitioner, as they are able to include existing knowledge to reduce the space of kernels needing to be explored. In the absence of such information, we motivate a set of heuristics based on general properties of the training data.

The prior over the number of mixture components n is uniform up to some maximum N , limiting the size of the overall parameter space. The n components are weighted against one another through a Dirichlet prior.

Mixture means are drawn from a normal distribution with zero mean and standard deviation $F_s/5$ so that negligible mass lies above the Nyquist frequency F_s of the data. Recall that the Nyquist frequency is the highest frequency identifiable

by a data set.³

Mixture scales can be seen as approximate inverse lengthscales of the kernel. Accordingly their log-normal prior is set such that most of its mass lies below the inverse of the minimum distance between data points. Additionally we allow for learning long range trends by placing prior mass below the inverse of the maximum distance between data points, as well as above it. This leads to a mean of $\log(1/|D_x|)$ and a scale of $\log(\sqrt{2})$ (where $|D_x|$ is the size of the data window).

We stress that the prior outlined above is, in our view, the widest reasonable prior. It provides a good starting point for initial estimates, but performance is likely to be improved considerably by specifying narrower priors, if possible, especially for high dimensional datasets.

6.4.2 Posterior Inference

Since we do not know the full parameterisation of our GP model, we marginalise out the unknown kernel parameters. Accordingly, the predictive distribution at some location x_* under the above model, after observing data $D = \{x_D, y_D\}$, is given by:

$$\begin{aligned} p(y_* | x_*, D) &= \int p(y_* | x_*, \theta, D) p(\theta | D) d\theta \\ &= \int p(y_* | x_*, \theta, D) \frac{p(D | \theta) p(\theta)}{p(D)} d\theta \\ &= \frac{\int p(y_* | x_*, \theta, D) p(D | \theta) p(\theta) d\theta}{\int p(D | \theta) p(\theta) d\theta}, \end{aligned} \tag{6.4}$$

where $\theta = (n, \{w_i, m_i, \sigma_i\}_{i=1}^n)$ is the parameterisation of a spectral GMM with n components, each described by a weight w_i , mean m_i and scale σ_i . The predictive distributions $p(y_* | x_*, \theta, D)$ and likelihoods $p(D | \theta)$ for the spectral kernel corresponding to a particular θ are given by the GP framework.

Unfortunately, both integrals in (6.4) are intractable and require approximation. We note that a major cost here is the evaluation of Gaussian Process likelihoods. These involve an inversion of the kernel Gram matrix of the entire data set whose computational cost increases quickly with data set size (typically cubically). To

³For unevenly-spaced data we choose F_s as the Nyquist frequency of a fictitious dataset with sampling frequency equal to the inverted average distance between datapoints.

reduce the number of likelihood evaluations required, we propose to adopt a Bayesian Quadrature approach.

6.4.3 Bayesian Quadrature

We begin by constructing GP models of the functions $p(y_* | x_*, \theta, D) p(D | \theta)$ and $p(D | \theta)$ that are to be integrated against the prior. Since the locations x^* of test points are unknown at training time, inference separates into two steps.

During training, a “pseudo-dataset” $\mathfrak{D}$ of decompositions and matching likelihood evaluations (and model evidences respectively) is compiled through active sampling (see Section 6.4.3) to shape our surrogate models and improve approximation quality. Hereby, we follow Gunter et al. [71] and enforce positivity of the likelihood surrogate by placing a GP prior over the square-root of the likelihood function $L(\theta) = P(D | \theta)$:

$$z = \sqrt{2(L(\theta) - \epsilon)} \sim \mathcal{GP}(0, \kappa), \quad (6.5)$$

where $\kappa(\theta_1, \theta_2)$ is a kernel between spectral decompositions (see section 6.4.3) and ϵ is a small non-negative constant.

Under this model $L(\theta)$ is not a GP, but can be approximated with one using a linear transformation of the surrogate. The posterior mean and covariance functions $\mathfrak{m}_{\mathfrak{D}}(\theta)$ and $\mathfrak{K}_{\mathfrak{D}}(\theta_1, \theta_2)$ then remain functions of the kernel κ and the likelihood observations. To ensure a good fit to the actual likelihood function, hyperparameters are re-optimised after each evaluation. Since GPs are closed under continuous bounded linear transformations [22], we obtain a Gaussian posterior over the model evidence – the integral in the denominator of (6.4). The mean of this Gaussian takes the form $z^T Q z$, where Q are the BQ weights – a full derivation is given in Appendix C.1.

To make a prediction, we compute the integral in the numerator of (6.4) in the same fashion. $p(y_* | x_*, \theta, D)$ is a GP posterior. Since we employ the same kernel κ to measure similarity between spectral densities, and we reuse the same likelihood evaluations from the training stage, the quadrature weights are identical and only z differs, allowing for efficient computation of the desired quantities.

The Hyper-Kernel

Noting that the densities under consideration are GMMs with a limited number of components, it becomes apparent that a suitable *hyper*-kernel, κ , has to overcome two obstacles. Firstly, it needs to cope with varying numbers of parameters, or otherwise assign a predefined covariance between GMMs with differing numbers of components. Secondly, it should capture a number of invariances in the parameterisation of GMMs (e.g. reordering of components, subdivision or recombination of components etc.). A naïve approach for the construction of such a kernel would base distances between densities on the Euclidean distance between their parameters and fails to achieve either desiderata.

Instead we construct a kernel based on the distance between the spectral densities represented by those parameters. Accordingly, we seek a conditionally-negative definite or Hilbertian metric between distributions. The maximum mean discrepancy meets our criteria, and is cheap to compute.

We obtain the following kernel:

$$\kappa(\theta_1, \theta_2) = \lambda^2 \exp \left(-\frac{d(\theta_1, \theta_2)^q}{l^2} \right), \quad (6.6)$$

where $\lambda > 0$ is an output-scale, $l > 0$ is a length-scale parameter, $q = \{1, 2\}$ and d is a maximum mean discrepancy between the spectral densities (GMMs) parameterised by θ_1 and θ_2 . The exact form of the MMD depends on the underlying kernel. Our experiments use the Energy Distance MMD [56], given by

$$d(\theta_1, \theta_2) = w_1^T M_{12} w_2 - \frac{1}{2} w_1^T M_{11} w_1 - \frac{1}{2} w_2^T M_{22} w_2 \quad (6.7)$$

where w_1 and w_2 are vectors of component weights and M_{12} is a matrix of euclidean distances between all pairs of component parameters $\{(m_i, \sigma_i)\}_{i=1}^{n_1}$ and $\{(m_i, \sigma_i)\}_{i=1}^{n_2}$, but our framework allows for alternatives (e.g. the Gaussian MMD, which can also be computed analytically for GMMs). Note that, since the MMD is analytically computed (rather than approximated through sampling as is usual in the MMD

literature), it is a valid distance between unnormalised densities. The hyperkernel therefore remains valid when kernels with arbitrary, non-unit output-scales are considered.

The only drawback is that integration of this kernel against the prior distribution is no longer possible analytically, so Monte Carlo integration is necessary. Arguably, however, the integration of the kernel against the prior lends itself better to such an approximation than that of the data likelihood. Intuitively, the likelihood function will have many sharp peaks for large datasets requiring samples to be drawn at very specific (unknown) locations for an accurate estimate. The kernel function by comparison is smoother, cheaper to evaluate and the resulting integral easier to approximate via random sampling from the prior. Our empirical results justify this approach, showing that we outperform competing methods despite the requirement for sampling.

Information-theoretic point acquisition

The GP surrogate is trained based on a set of sampled spectral distributions, $\mathfrak{D} = \{(\theta_i, L_i)\}$, and the likelihood values of their corresponding kernels. Since adding a new point θ^{new} to this set requires a computationally expensive likelihood evaluation, it should be chosen to maximise its informativeness w.r.t. the quantity we care about – the predictive distribution at test locations x_* .

As we do not expect the model to have access to the test locations at training time, we choose a different acquisition criterion for the new point. Instead, we aim to improve the approximation of the denominator of (6.4) – and thereby the fit of the posterior distribution over spectral mixtures – at training time. In Gunter et al. [71] an acquisition function based on the uncertainty of the integrand is proposed. We extend the work of Gessner, Gonzalez, and Mahsereci [63] and Ru et al. [159] to propose an information-theoretic criterion instead.

We observe that the expected reduction in the entropy of our integral estimate after making an observation at sample location θ^s is given by

$$\alpha(\theta_*) = H(L_* \mid \mathfrak{D}, \theta_*) - E_{p(Z \mid \mathfrak{D}, \theta_*)}[H(L_* \mid \mathfrak{D}, \theta_*, Z)]. \quad (6.8)$$

Here L_* is the predicted observation at location θ_* and Z is the integral in the denominator of (6.4). Within the GP framework, the entropy of the predictive distribution $H(L_* \mid \mathfrak{D}, \theta_*)$ is the entropy of a Normal distribution and can be calculated in closed form. The second term turns out to be the expectation of a constant and the predictive distribution conditioned on Z can once more be computed in closed form. Gessner, Gonzalez, and Mahsereci [63] discuss this acquisition function in the context of ordinary and multi-source Bayesian Quadrature. We note that it is available for warped Bayesian Quadrature. In particular, the acquisition function can be computed analytically when the linearisation approximation is used, provided that the kernel integrals are analytic. This is because the composition of the approximation and integration remains an affine transformation. Therefore, the integral of the approximate warped surrogate is jointly Gaussian distributed with the unwarped surrogate.

The acquisition function (6.8) is also valid for selecting batches of a fixed size. Importantly, our combination of hyper-kernel and acquisition function allows for the selection of (batches of) GMMs that are maximally informative about the overall evidence, $\int p(D \mid \theta)p(\theta)d\theta$, rather than selecting GMMs for each parameter domain (corresponding to a fixed number of components) separately. This is because the kernel is able to define covariances between GMMs with different numbers of mixture components. Incorporating such information represents an improvement over Chai et al. [29] whose framework, in this instance, would be naïve as it assumes that the GPs over each domain are independent.

6.4.4 Computational Complexity

During learning, each iteration requires optimising the hyperparameters of the surrogate GP and the acquisition function. The time complexity of hyperparameter optimisation is dominated by the cost of making likelihood evaluations, $\mathcal{O}(h^3)$ where h is the number of Spectral Mixture kernels evaluated thus far. Acquisition function optimisation incurs an initial cost of $\mathcal{O}(h^3 + mh^2 + m^2)$, where m is the number of Monte Carlo samples used to approximate the kernel integrals.

Subsequent evaluations require $\mathcal{O}(bh^2)$ operations, where b is the batch size. The memory complexity during hyperparameter optimisation is dominated by the cost of storing the hyper-kernel evaluated between observed SM kernels $\mathcal{O}(h^2)$. During the initialisation of the acquisition function, $\mathcal{O}(m^2)$ memory is additionally required to store the hyper-kernel evaluated between the MC samples.

Once the evaluation budget is exhausted the quadrature weights can be computed in $\mathcal{O}(h^3 + mh^2 + m^2)$ time and $\mathcal{O}(h^2 + m^2)$ memory. Inference takes $\mathcal{O}(h|D|^3)$ operations, though this can be reduced significantly by disregarding terms that have low quadrature weight. The memory complexity of inference is $\mathcal{O}(h^2|D|^2)$ for a naïve implementation, and can be reduced to $\mathcal{O}(h|D|^2)$ by looping appropriately (note that this leaves the asymptotic time complexity unchanged).

6.5 Related Work

6.5.1 Kernel Learning

Early work in kernel learning focussed on searching over compositions (sums and products) of a set of basic covariance functions [50]. More recent work has proposed learning kernels using their spectral representations [209, 101, 108, 60, 102, 136, 155, 87, 2, 194, 191, 19]. Beyond the ability to capture more complex structures, this approach also lends itself to approximate inference, reducing learning time complexity [60, 151, 80]. A model similar to ours uses a Dirichlet Process prior over the number of components in a Gaussian Mixture Model (GMM) residing in the spectral domain [136]. Simpson, Lalchand, and Rasmussen [173] use Nested Sampling to marginalise over the parameters of a Spectral Mixture kernel with a fixed number of mixture components.

Inference in spectral models reduces to the marginalisation of the likelihood function against the posterior over kernels (or, equivalently, their spectral decompositions). The resulting integrals are intractable and have to be approximated – usually through the use of an appropriate MCMC scheme, such as Gibbs sampling, RJ-MCMC [67] or elliptical slice sampling [131]. Scaling MCMC marginalisation over the spectral kernel parameters up to large datasets is problematic. The

computational cost of likelihood evaluations scales poorly in the size of the training data set and the collection of sufficiently many samples for an accurate Monte-Carlo estimate becomes prohibitively expensive. As above, BQ may be a more appropriate choice in these settings.

The recently proposed baselines we will compare against in our experiments include:

Variational Sparse Spectrum Gaussian Process Gal and Turner [60] which performs Variational Inference over a sparse spectrum approximation.

Bayesian Nonparametric Kernel Learning Oliva et al. [136] which learns the posterior distribution over Spectral Mixture kernels via a parameterisation of spectral densities based on GMMs and a Gibbs sampling scheme.

Functional Kernel Learning Benton et al. [19] which places a GP prior over the kernel’s spectral density and infers a posterior over kernels using an Elliptical Slice Sampling scheme.

Various approximations to reduce the cost of likelihood evaluations have been proposed in the literature, such as PCG [34] and BBMM [61] – modified conjugate gradient methods – and Random Fourier Features [151]. This is orthogonal to our investigation and our framework is optionally capable of leveraging both of these for scalability.

6.5.2 Bayesian Quadrature

Recent work in BQ [140, 71, 30] has proposed the use of warped GPs [175] to incorporate a priori known model-constraints into the surrogate. This work builds upon WSABI [71] in particular, which places a GP prior over the square-root of the integrand. The acquisition function developed in Section 6.4.3 also applies to this setting and is an extension of the information-theoretic scheme discussed in Gessner, Gonzalez, and Mahsereci [63]. Osborne et al. [141] also proposed a BQ framework to infer ratios of integrals, which is relevant to the computation of marginalised posteriors. However, the combination with WSABI and our framework introduces

further intractable integral terms, which must be approximated with Monte Carlo sampling and would prohibitively raise the cost of inference. Additionally, the work of Xi, Briol, and Girolami [215] and Chai et al. [29] is related as it concerns the use of BQ for computing correlated integrals, and for automatic model selection.

6.5.3 Gaussian Processes on Spaces of Measures

The BQ integrand model requires the specification of a positive-definite kernel across the integration domain – the space of spectral densities. While numerous metrics and divergences have been proposed to measure differences between probability distributions, many do not give rise to valid kernels.

Recent work has used Wasserstein distances [13] to define Gaussian Processes on a space of measures. However, Wasserstein distances are only Hilbertian in 1D [147] so extensions to multidimensional distributions rely on Hilbert space embeddings of optimal transport maps to a reference distribution [14]. Furthermore, evaluating Wasserstein distances can be expensive as it involves finding the solution to an optimal transport problem. The Independence kernel based on the Sinkhorn distance [35] has been shown to be a valid positive definite kernel, but the distance between identical distributions may not be zero. An elegant alternative that satisfies our desiderata are maximum mean discrepancies (MMDs), which are defined in terms of kernel mean embeddings. MMDs are Hilbertian, so are a valid metric for kernels of the form (6.6) [130].

6.6 Results

In the following we empirically assess the performance of our model on a variety of tasks.

6.6.1 Experiment setup

The experiments were conducted on an Nvidia Titan V⁴ and a Nvidia GeForce GTX 1080⁵ GPUs. Since the cost of likelihood evaluations depends on the dataset size, so does the training budget. The numbers given indicate either mean performance or mean performance and standard deviation.

We compare our approach of MArginalising Spectral KERnels As DEnsities (MASKERADE)⁶ to models previously proposed in the literature including VSSGP [60], BaNK [136], and FKL [19]. No reference implementations for VSSGP and BaNK are publically available and results are taken directly from the respective publications. For FKL experiments the implementation and default hyperparameters in the `spectralgp`⁷ Python package were used.

For MASKERADE data is preprocessed so that the training labels have unit variance, and the training inputs lie in the range $[0, 1]$ in all dimensions. The former heuristically imposes a fixed outputscale equal to the standard deviation of the training labels. The latter rescales features to facilitate the formulation of suitable hyperpriors (see Section 6.4). We limit the parameter space MASKERADE considers to GMMs with up to 5 components ($N=5$) and set the concentration of the Dirichlet prior over weights to $\alpha = 1$. BQ surrogate GPs are initialised with likelihood evaluations of parameters randomly chosen from the prior and thereafter acquire additional evaluations in batches of size 10 (respective 20 for the large datasets). LBFGS is used to optimise both the acquisition function as well as the parameters of the hyper-kernel (which happens every 2 batches).

6.6.2 Medium scale data sets

We begin by fitting MASKERADE to the Solar dataset [109] to compare against VSSGP. We replicate the setup from Gal and Turner [60], withholding 5 sets of length 20 as the test set.

⁴Comparisons to BaNK and VSSGP, ablation studies

⁵Comparison to FKL

⁶<https://github.com/saadhamidml/maskerade>

⁷<https://github.com/wjmaddox/spectralgp>

Table 6.1: Test set RMSE for the Solar Irradiance Data Set. Results for VSSGP are taken from Gal and Turner [60].

	VSSGP	MASKERADE
RMSE	0.41	0.17

We further examine the algorithms’ performance on four datasets from the UCI Machine Learning Repository [46]: Yacht Hydrodynamics (308 instances, 6 input dimensions) [118], Auto MPG (398 instances, 8 input dimensions) [150], Concrete Compressive Strength (1030 instances, 8 input dimensions) [218], and Airfoil Self-Noise (1503 instances, 5 input dimensions) [117].

To compare against FKL we follow Benton et al. [19] and average over 10 runs. Instead of randomly partitioning the data into 90/10 train-test-sets, we use 10-fold cross validation. Thus, test sets remain independent from one another, enabling us to assess the statistical significance of our results using paired Student-t tests. For each of the medium dataset we set a limited training time budget of 5 minutes and present the test results in Table 6.2.

Table 6.2: Posterior log likelihood of and RMSE on the test set for UCI MLR datasets.

DATASET	LL		P-VALUE
	FKL	MASKERADE	
YACHT	-32.071 $\pm$ 8.45	47.163 5.757	2.97×10^{-9}
AUTOMPG	-103.57 $\pm$ 5.638	-57.265 $\pm$ 3.95	3.56×10^{-9}
CONCRETE	-300.953 $\pm$ 34.136	-122.696 $\pm$ 6.485	9.72×10^{-8}
AIRFOIL	-863.545 $\pm$ 35.959	-151.54 $\pm$ 13.263	1.65×10^{-13}

DATASET	RMSE		P-VALUE
	FKL	MASKERADE	
YACHT	0.631 $\pm$ 0.358	2.013 $\pm$ 0.576	1.08×10^{-6}
AUTOMPG	3.089 $\pm$ 0.358	8.137 $\pm$ 0.655	6.77×10^{-9}
CONCRETE	5.808 $\pm$ 2.651	13.148 $\pm$ 1.314	2.78×10^{-5}
AIRFOIL	83.854 $\pm$ 12.335	4.592 $\pm$ 0.623	1.06×10^{-8}

Similarly, to compare against BaNK we follow Oliva et al. [136] and perform 3

repeats of 5-fold cross-validation on the Concrete Compressive Strength and Airfoil Self-Noise datasets⁸, for which we present results in Table 6.3.

Table 6.3: Test set MSE on two UCI data sets. Results for BaNK taken from Oliva et al. [136].

DATASET	BANK	MASKERADE
CONCRETE	0.1195 $\pm$ 0.0108	0.6889 $\pm$ 0.0168
AIRFOIL	0.3359 $\pm$ 0.0354	0.3244 $\pm$ 0.0214

6.6.3 Large scale data sets

Finally we compare MASKERADE against FKL on two large datasets, with a limited time budget of 20 minutes for learning. The setup remains the same as for the UCI datasets and results are reported in Table 6.4.

The datasets are the Sterling Broad Based Exchange Rate [51] (data from 1990-01-02–2020-10-08, totalling 7781 data points) and the UCI 3D Road Network [94] (10,000 data point subset of 250,000 data points. We choose the first 10,000 instances with a unique OSM-ID) for which the task is to predict altitude from latitude and longitude.

As on medium datasets, results show that MASKERADE has a higher prediction error (RMSE) than FKL. However, as indicated by the higher LL, uncertainty estimates are better calibrated and MASKERADE is “less confidently wrong” (respectively “more confidently correct”). In our opinion a main motivation for the use of probabilistic models is the computation of uncertainty estimates for downstream tasks and we see this trade-off slightly favouring MASKERADE.

6.6.4 Ablation Study

We conduct an ablation study to verify the effectiveness of two components of our model in Table 6.5. Firstly, we verify that marginalising across the kernel family does indeed lead to better predictive performance. As a baseline, we compare against

⁸the lack of a reference implementation precludes further statistical comparison

Table 6.4: Posterior log likelihood of and RMSE on the test set for FKL and MASKERADE fit to two large datasets. Both methods were given a time budget of 20 minutes for training.

DATASET	LL		P-VALUE
	FKL	MASKERADE	
STERLING	-1053.566 ± 30.697	335.37 ± 53.582	7.81×10^{-8}
ROAD ALT	-4355.296 ± 219.847	-1070.571 ± 21.548	7.19×10^{-12}

DATASET	RMSE		P-VALUE
	FKL	MASKERADE	
STERLING	1.042 ± 0.028	0.523 ± 0.042	1.77×10^{-10}
ROAD ALT	10.17 ± 0.62	12.798 ± 0.427	3.68×10^{-6}

a GP using Spectral Mixture kernel (SM) whose hyperparameters were optimised via gradient descent to maximise the likelihood of the training data. Secondly, we compare our proposed acquisition function against uncertainty sampling [71] and random point acquisition. We set an evaluation budget of 1000 likelihood evaluations and assess performance on the Airline Passenger [120] and Mauna Loa Atmospheric CO₂ Concentration [134] datasets. As before, we partition according to a 10-fold cross validation and assess statistical significance. In both cases, our chosen approach compares favourably to alternative designs. However, on the Airline Passenger data differences between the MASKERADE variants are not statistically significant.

6.7 Discussion

We have introduced a novel framework for kernel learning for Gaussian Processes that seeks to marginalise over Spectral Mixture Kernels using Bayesian Quadrature. Specifically, we use a maximum mean discrepancy as a metric underlying an exponential kernel to define a Gaussian Process on a space of GMMs, and show how this can be efficiently computed. This elegantly incorporates invariances between Spectral Mixture Kernels into our model. Additionally we show that an information-theoretic acquisition function is applicable for warped Bayesian Quadrature, and

how to use it within our framework. We empirically evaluate our method on several datasets and find that it is competitive with state-of-the-art baselines.

The key limitation of our method is due to the curse of dimensionality. The most scalable instantiation of our framework handles multi-dimensionality by using GMMs in which each Gaussian has a diagonal covariance structure. This means that the number of parameters for an SM kernel with M mixtures is $M(1 + 2D)$, which grows faster than the number of data dimensions, D . This limits the effectiveness of any Bayesian Quadrature regime. Note that Monte Carlo methods also suffer in high dimensional spaces, especially if likelihood evaluations are expensive. (Our method relies on MC integration of kernel integrals, but these are considerably easier to compute than the marginalisation integrals because our method only requires samples from the priors over hyperparameters rather than the posteriors, and the kernel is much cheaper to evaluate than the likelihood.) Our approach is most beneficial for inference based on large, low dimensional datasets.

Future work could investigate the other families of spectral densities. In theory, our model can use arbitrary distributions, provided their maximum mean discrepancy can be computed. This may give rise to alternative parameterisations allowing for better approximation via BQ or the inclusion of task specific knowledge. A second, promising direction of research concerns an extension to non-stationary

Table 6.5: Normalised test set RMSE for the Airline Passenger and Mauna Loa datasets. MASKERADE-I indicates the use of our proposed information theoretic acquisition function, MASKERADE-U indicates uncertainty sampling [71], and MASKERADE-R indicates random sampling under the prior. P-values are computed using paired t-tests between the respective method and MASKERADE-I.

ALGORITHM	AIRPASS		MAUNA-LOA	
	RMSE	P-VALUE	RMSE	P-VALUE
SM	0.788 ± 0.264	4.29×10^{-5}	1.057 ± 0.082	3.06×10^{-11}
MASKERADE-R	0.188 ± 0.073	0.783	0.03 ± 0.002	0.01
MASKERADE-U	0.187 ± 0.064	0.78	0.029 ± 0.003	0.02
MASKERADE-I	0.183 ± 0.066	-	0.023 ± 0.007	-

kernels. The corresponding spectral decompositions are bi-variate, requiring a more elaborate representation and a tractable solution to integral approximation in the resulting, high-dimensional parameter spaces.

Statement of authorship

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor (**only required where there isn't already a statement of contribution within the paper itself**).

Title of Paper	Marginalising over Stationary Kernels with Bayesian Quadrature
Publication Status	☐ Published ☐ Accepted for Publication ☒ Submitted for Publication ☐ Unpublished and unsubmitted work written in a manuscript style
Publication Details	Under review at "Thirty-eighth International Conference on Machine Learning" (ICML 2021)

Student Confirmation

Student Name:	Sebastian Schulze
Contribution to the Paper	- joint discussion of theoretical approaches / research impulses - various code reviews / debug iterations - joint experiment design and result analysis - write-up of manuscript
Signature: 	Date: 13/04/2021

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Prof Michael A. Osborne		
Supervisor comments All above confirmed.		
 Signature	Date	14 April 2021

This completed form should be included in the thesis, at the end of the relevant chapter.

7

Concluding remarks

This thesis highlights different aspects in the design of agents learning from interaction with the real-world. We argue that data-efficient adaptation is only possible by exploiting the ubiquitous regularities practical settings exhibit, but standard RL-systems remain unaware of.

In Chapter 3 we constructed an agent that learns to reason about the environment it lives in and uses that knowledge for the development of efficient exploration strategies. Chapter 4 elaborates on the second step by discussing the need to not only move through, but explicitly identify the key interactions of interest. The resulting learning processes are complex and often require specific, favourable starting conditions for their eventual success. Consequently, Chapter 5 moves towards automating their repeated initialisation and accelerating the identification of such starting points. Our as well as other machine learning systems are highly dependent on the ability to represent and manipulate information. Therefore, the development and extension of ever-more expressive and efficient model classes remains an on-going research effort. Here, Chapter 6 makes a contribution towards the advancement of Gaussian processes which are often a central part of Bayesian modelling.

Ultimately, we believe that our individual contributions will merge with other on-going machine learning research into a single pre-deployment learning process. This process will equip autonomous agents not only to meet the requirement for

data-efficiency discussed, but also encode other desiderata such as a need for safe and explainable behaviour.

Appendices



Supplementary Material to Chapter 3

A.1 Full ELBO derivation

Equation (3.9) can be derived as follows.

$$\begin{aligned}
\mathbb{E}_{\rho(M, \tau:H)} [\log p_{\theta}(\tau:H)] &= \mathbb{E}_{\rho} \left[\log \int p_{\theta}(\tau:H, m) \frac{q_{\phi}(m|\tau:t)}{q_{\phi}(m|\tau:t)} dm \right] \\
&= \mathbb{E}_{\rho} \left[\log \mathbb{E}_{q_{\phi}(m|\tau:t)} \left[\frac{p_{\theta}(\tau:H, m)}{q_{\phi}(m|\tau:t)} \right] \right] \\
&\geq \mathbb{E}_{\rho, q_{\phi}(m|\tau:t)} \left[\log \frac{p_{\theta}(\tau:H, m)}{q_{\phi}(m|\tau:t)} \right] \\
&= \mathbb{E}_{\rho, q_{\phi}(m|\tau:t)} [\log p_{\theta}(\tau:H|m) + \log p_{\theta}(m) - \log q_{\phi}(m|\tau:t)] \\
&= \mathbb{E}_{\rho} \left[\mathbb{E}_{q_{\phi}(m|\tau:t)} [\log p_{\theta}(\tau:H|m)] - KL(q_{\phi}(m|\tau:t) || p_{\theta}(m)) \right] \\
&= ELBO_t.
\end{aligned} \tag{A.1}$$

A.2 Experiments: Gridworld

Here we provide additional remarks and figures for the GridWorld results from Section 3.6.1.

A.2.1 Additional Remarks

Figure 3.3c visualises how the latent space changes as the agent interacts with the environment. As we can see, the value of the latent dimensions starts around

mean 1 and variance 0, which is the prior we chose for the beginning of an episode. Given that the variance increases for a little bit before the agent finds the goal, this prior might not be optimal. A natural extension of variBAD is therefore to also learn the prior to match the task at hand.

A.2.2 Comparison to RL^2

Figure A.1a shows the learning curves for variBAD and RL^2 , in comparison to an oracle policy (which has access to the goal position). We trained these policies on a horizon of $H^+ = 4 \times H = 60$, i.e., on a BAMDP in which the agent has to maximise online return within four episodes. We indicate the values of a hard-coded Bayes-optimal policy, and a hard-coded posterior sampling policy using dashed lines.

Figure A.1b shows the end-performance of variBAD and RL^2 , compared to the hard-coded optimal policy (which has access to the goal position), Bayes-optimal policy, and posterior sampling policy. variBAD and RL^2 both closely approximate the Bayes-optimal solution. By inspecting the individual runs, we found that variBAD learned the Bayes-optimal solution for 4 out of 20 seeds, RL^2 zero times. Both otherwise find solutions that are very close to Bayes-optimal, with the difference that during the second rollout, the cells left to search are not all on the shortest path from the starting point.

Note that both variBAD and RL^2 were trained on only four episodes, but we evaluate them on six episodes here. After the fourth rollout, we do *not* fix the latent / hidden state, but continue rolling out the policy as before. As we can see, the performance of RL^2 drops again after the fourth episode: this is likely due to instabilities in the 128-dimensional hidden state. VariBAD’s latent representation, the approximate task posterior, is concentrated and does not change with more data.

A.3 Experiments: MuJoCo

In this section we provide the learning curves (A.3.1) for the MuJoCo environments from Section 3.6.3. We also provide additional analyses of the learned agent behaviour (A.3.2 and how the latent space behaves at test time A.3.3). A.3.4

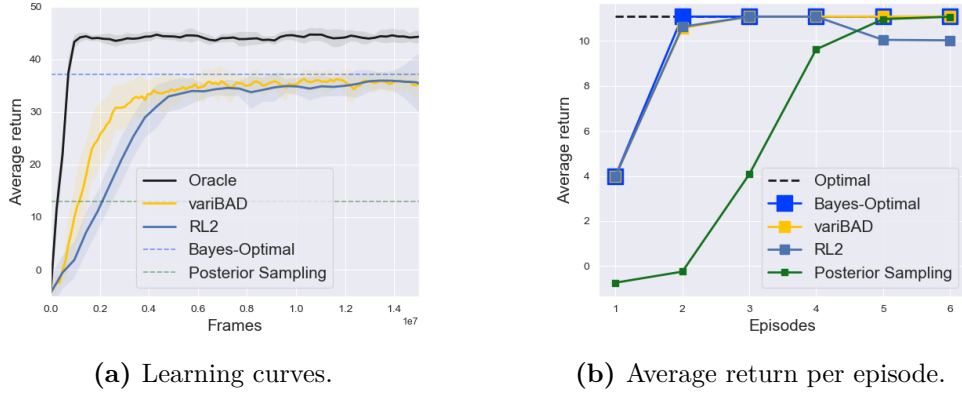


Figure A.1: Results for the gridworld toy environment. Results are averages over 20 seeds (with 95% confidence intervals for the learning curve).

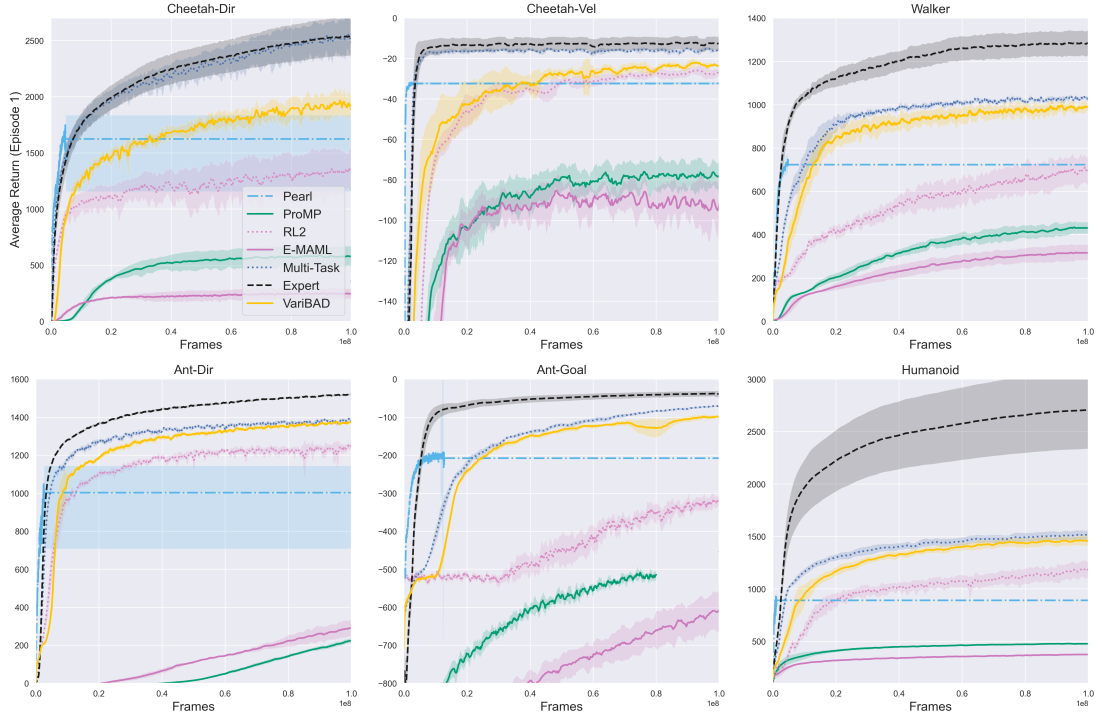


Figure A.2: Learning curves for the MuJoCo results presented in Section 3.6.3. The plots show the performance at the N -th rollout. For variBAD and RL², $N = 2$. For PEARL, $N = 10$. For ProMP and E-MAML, $N = 30 - 60$ (3 gradient steps on rollouts of length 10 – 20 depending on the environment).

provides a runtime comparison, and A.3.5 an ablation on backpropagating the RL loss through the VAE encoder.

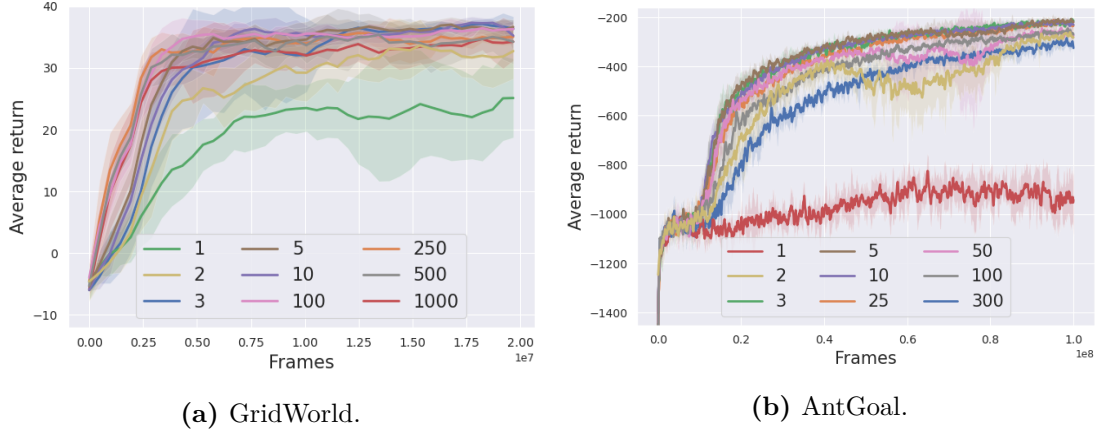


Figure A.3: Learning curves for different VAE latent dimensions, for the GridWorld environment (a) and the MuJoCo AntGoal environment (b).

A.3.1 Learning Curves

Figure A.2 shows the learning curves for the MuJoCo environments for all approaches. The oracle policy was trained using PPO. PEARL [152] was trained using the reference implementation provided by the authors. The environments we used are also taken from this implementation.

As we can see, PEARL is much more sample efficient in terms of number of frames than the other methods (Fig A.2), which is because it is an off-policy method. On-policy vs off-policy training is an orthogonal issue to our contribution, but an extension of variBAD to off-policy methods has been done in Dorfman, Shenfeld, and Tamar [43]. Doing posterior sampling using off-policy methods also requires PEARL to use a different encoder (to maintain order invariance of the sampled trajectories) which is non-recurrent (and hence faster to train, see next section).

For all MuJoCo environments, we trained variBAD with a reward decoder only (even for Walker, where the dynamics change, we found that this has superior performance).

A.3.2 CheetahDir Test Time Behaviour

To get a sense for where these differences between the different approaches might stem from, consider Figure A.4 which shows example behaviour of the policies during the first three rollouts in the HalfCheetahDir environment, when the task

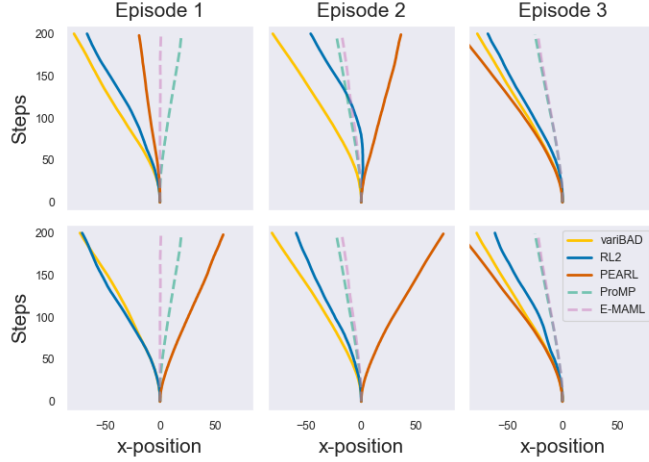


Figure A.4: Behaviour at test time for the task “walk left” in HalfCheetahDir. The x-axis reflects the position of the agent; the y-axis the steps in the environment (to be read from bottom to top). Rows are separate examples, columns the number of rollouts.

is “go left”. Both variBAD and RL² adapt to the task online, whereas PEARL acts according to the current sample, which in the first two rollouts can mean walking in the wrong direction. For a visualisation of the variBAD latent space at test time for this environment see Figure A.5.

A.3.3 Latent Space Visualisation

A nice feature of variBAD is that it can give us insight into the uncertainty of the agent about what task it is in. Figure A.5 shows the latent space for the HalfCheetahDir tasks “go right” (top row) and “go left” (bottom row). We observe that the latent mean and log-variance adapt rapidly, within just a few environment steps (left and middle figures). This is also how fast the agent adapts to the current task (right figures). As expected, the variance decreases over time as the agent gets more certain. It is interesting to note that the values of the latent dimensions swap signs between the two tasks.

Visualising the belief in the reward/state space directly, as we have done in the gridworld example, is more difficult for MuJoCo tasks, since we now have continuous states and actions. What we could do instead, is to additionally train a model that predicts a ground-truth task description (separate from the main objective and just

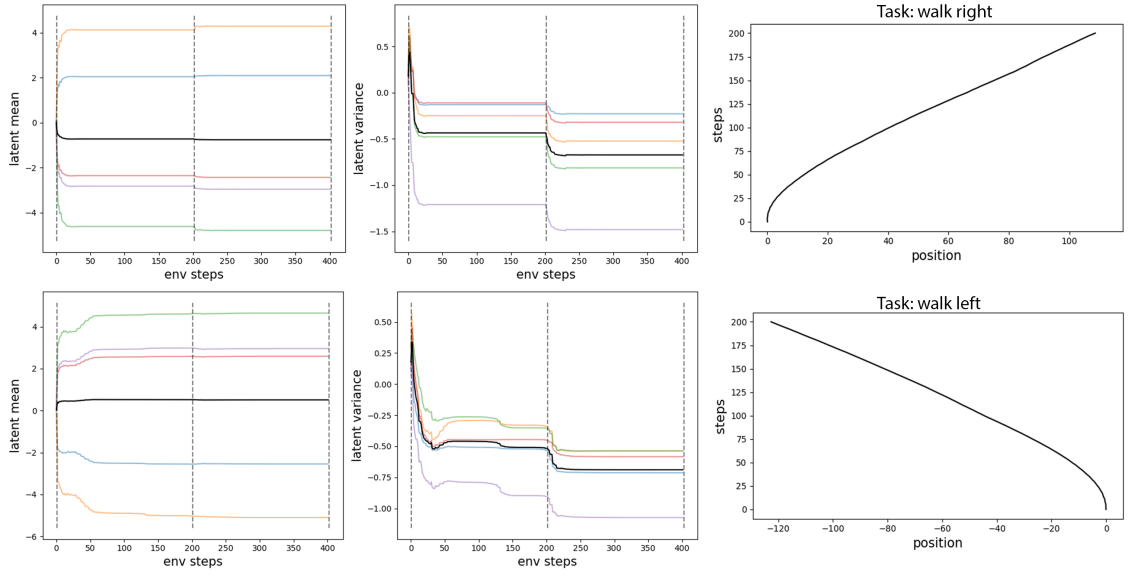


Figure A.5: Visualisation of the latent space at meta-test time, for the HalfCheetahDir environment and the tasks "go right" (top) and the task "go left" (bottom). **Left:** value of the posterior mean during a single rollout (200 environment steps). The black line is the average value. **Middle:** value of the posterior log-variance during a single rollout. **Right:** Behaviour of the policy during a single rollout. The x-axis shows the position of the Cheetah, and the y-axis the step (should be read from bottom to top).

for further analysis, since we do not want to use this privileged information for meta-training). This would give us a more direct sense of what task it thinks it is in.

A.3.4 Runtime Comparison

The following are rough estimates of average run-times for the HalfCheetah-Dir environment (from what we have experienced; we often ran multiple experiments per machine, so some of these might be overestimated and should be mostly understood as giving a relative sense of ordering).

- ProMP, E-MAML: 5 – 8 hours
- variBAD: 48 hours
- RL²: 60 hours
- PEARL: 24 hours

E-MAML and ProMP have the advantage that they do not have a recurrent part such as variBAD or RL². Forward and backward passes through recurrent networks can be slow, especially with large horizons.

Even though both variBAD and RL² use recurrent modules, we observed that variBAD is faster when training the policy with PPO. This is because we do not backpropagate the RL-loss through the recurrent part, which allows us to make the PPO mini-batch updates without having to re-compute the embeddings (so it saves us a lot of forward/backward passes through the recurrent model). This difference is less pronounced with other RL methods that do not rely on this many forward/backward passes per policy update.

A.3.5 Ablation Study: Backpropagating the RL loss Through the Encoder

In the main experiments presented in the paper, we do not backpropagate the RL loss through the encoder. Instead, we alternate between updating the VAE with the ELBO loss, and updating the policy with the PPO loss (and detaching the gradient when feeding the belief into the policy). We do so because this performs sufficiently well, and has two advantages: first, we do not have to calibrate the relative weighting between the RL and the VAE loss; second, it is much faster in practice because otherwise we would have to re-compute the belief embedding for each PPO mini-batch.

Figure A.6 shows the learning curves of the variBAD agent in the AntDir and AntGoal environments, when backpropagating the RL loss compared to our standard setting and compared to RL². These results show that sometimes, combining the RL and VAE loss can marginally improve performance (Figure A.6a), but it can also significantly hurt performance (Figure A.6b) if the relative weighting is not calibrated correctly. In terms of experiment runtime, when backpropagating the RL loss through the encoder is as slow as RL² (around 66% slower in these environments).

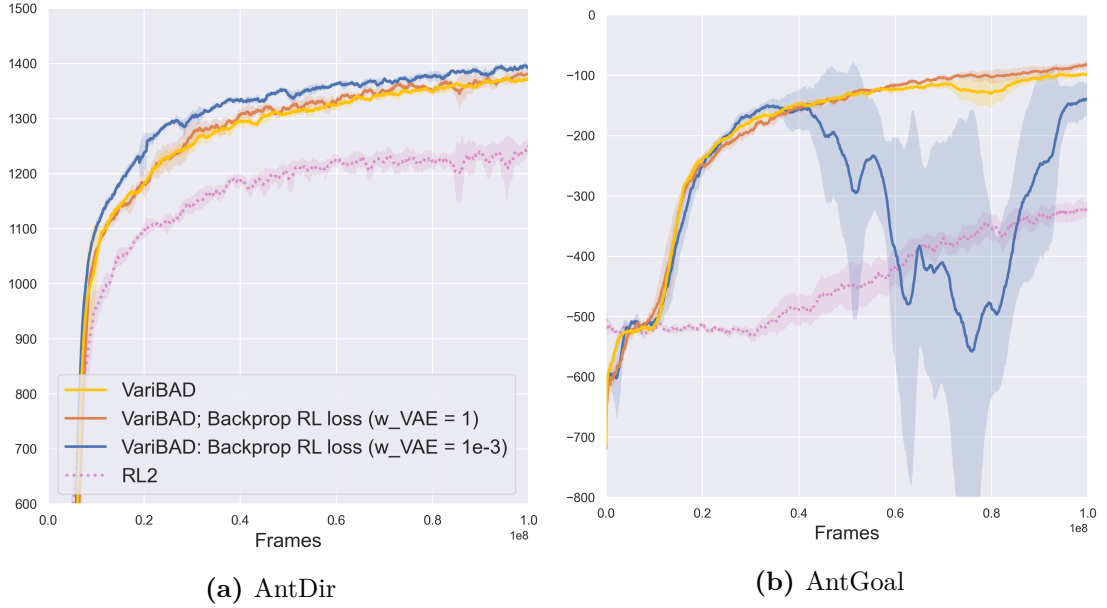


Figure A.6: Learning curves for the MuJoCo AntDir (a) and AntGoal (b) environments, when backpropagating the RL loss through the encoder. Depending on the task, this can help (a) or hurt (b) performance, and critically depends on the relative weight between VAE and RL loss.

A.4 Experiments: Meta-World

Figure A.7 shows the learning curves for the Meta-World ML1 tasks for variBAD (20 seed averages with 95% confidence intervals). We followed the evaluation protocol of Yu et al. (2019).

A.5 Hyperparameters

We used the PyTorch framework [143] for our experiments. The hyperparameters for GridWorld, MuJoCo CheetahDir, PointRobot and MetaWorld ML1-Push can be found in the tables below. For more details, see our reference implementation at <https://github.com/lmzintgraf/variBAD>.

We used different number of seeds per experiment to balance significance of results and computational required, due to the inherent randomness/difficulty of different tasks. For the main experiments, we used 20 seeds for GridWorld/Navigation/MetaWorld, and 10 seeds per MuJoCo environment. For the ablation studies, we used

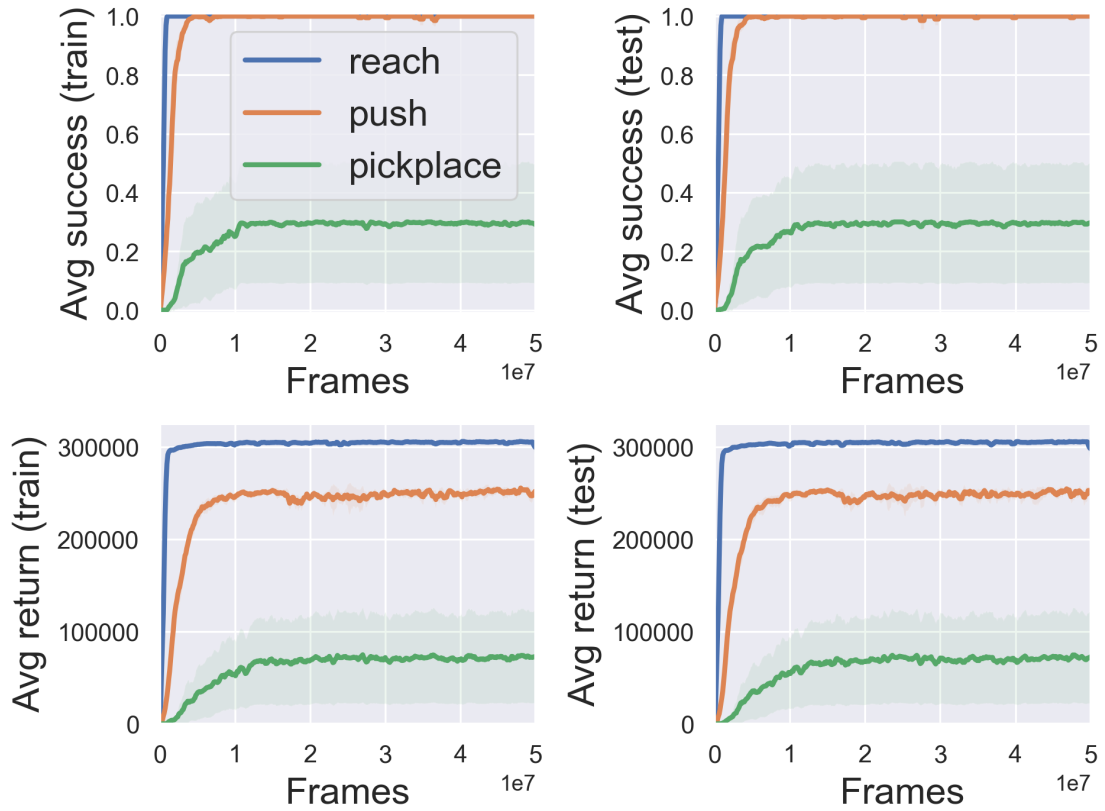


Figure A.7: Learning curves for the different ML1 Meta-World tasks. Shown are the success rates (top row) and average returns (bottom row) for the training set (first column) and test set (second column)

fewer for MuJoCo (5 instead of 10), and GridWorld (15 instead of 20) due to computational constraints.

	Grid World	Cheetah Dir	Point Robot	ML1 Push
max _rollouts _per _task	4	2	3	3
policy _state _embedding _dim	16	64	64	64
policy _latent _embedding _dim	16	64	64	64
norm _state _for _policy	True	True	True	True
norm _latent _for _policy	True	True	True	True
norm _rew _for _policy	True	True	True	True
norm _actions _pre _sampling	False	True	False	False
norm _actions _post _sampling	False	False	False	False
policy _layers	[32]	[128, 128]	[128, 128, 128]	[128, 128, 128]
policy _activation _function	tanh	tanh	tanh	tanh
policy _initialisation	normc	normc	normc	normc
policy _anneal _lr	False	False	False	False
policy	ppo	ppo	ppo	ppo
policy _optimiser	adam	adam	adam	adam
ppo _num _epochs	2	16	2	2
ppo _num _minibatch	4	4	8	8
ppo _clip _param	0.05	0.1	0.1	0.1
lr _policy	0.0007	0.0007	0.0007	0.0007
num _processes	16	16	16	16
policy _num _steps	60	800	200	200
policy _eps	1e-08	1e-08	1e-08	1e-08
policy _value _loss _coef	0.5	0.5	0.5	0.5
policy _entropy _coef	0.01	0.01	0.001	0.001
policy _gamma	0.95	0.97	0.99	0.99
policy _use _gae	True	True	True	True
policy _tau	0.95	0.9	0.9	0.9
use _proper _time _limits	False	True	True	True
encoder _max _grad _norm	None	1.0	None	None
decoder _max _grad _norm	None	1.0	None	None
lr _vae	0.001	0.001	0.001	0.001
size _vaebuffer	100000	10000	10000	10000
precollect _len	5000	5000	5000	5000
vae _buffer _add _thresh	1	1	1	1
vae _batch _num _trajs	25	10	15	15
tbptt _stepsize	None	50	None	None
vae _subsample _elbos	None	50	None	None
vae _subsample _decodes	None	50	None	None
vae _avg _reconstruction _terms	False	False	False	False
num _vae _updates	3	1	3	3
pretrain _len	0	0	0	0
kl _weight	0.01	0.1	1.0	1.0

Table A.1: VariBAD hyperparameters for MuJoCo experiments

	Grid World	Cheetah Dir	Point Robot	ML1 Push
action _embedding _size	0	16	16	16
state _embedding _size	8	32	32	32
reward _embedding _size	8	16	16	16
encoder _layers _before _gru	[]	[]	[]	[]
encoder _gru _hidden _size	64	128	128	128
encoder _layers _after _gru	[]	[]	[]	[]
latent _dim	5	5	5	5
decode _reward	True	True	True	True
rew _loss _coeff	1.0	1.0	1.0	1.0
input _prev _state	False	True	False	False
input _action	False	True	False	False
reward _decoder _layers	[32, 32]	[64, 32]	[64, 32]	[64, 32]
decode _state	False	False	False	False
state _loss _coeff	1.0	1.0	1.0	1.0
state _decoder _layers	[32, 32]	[64, 32]	[64, 32]	[64, 32]
disable _kl _term	False	False	False	False
rlloss _through _encoder	False	False	False	False

Table A.2: VariBAD hyperparameters for MuJoCo experiments

B

Supplementary Material to Chapter 5

The following sections are intended to give the reader further insights into our design choices and a deeper understanding of the algorithms properties.

B.1 GP kernels and treatment of GP hyperparameters

We detail the GP kernels and treatment of GP hyperparameters for the black-box function f .

Although the raw learning curve in DRL is noisy, the transformed version using our proposed curve compression makes the resulting curve smooth. Therefore, we use two squared exponential kernels for hyperparameters and training iterations, respectively. That is $k_x(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma_x^2}\right)$ and $k_t(t, t') = \exp\left(-\frac{\|t-t'\|^2}{2\sigma_t^2}\right)$ where the observation $\mathbf{x}$ and t are normalised to $[0, 1]^d$. Outcome observations y are similarly standardised such that $y \sim \mathcal{N}(0, 1)$. As a result, our product kernel becomes

$$k([\mathbf{x}, t], [\mathbf{x}', t']) = k(\mathbf{x}, \mathbf{x}') \times k(t, t') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma_x^2} - \frac{\|t - t'\|^2}{2\sigma_t^2}\right).$$

The length-scales σ_x and σ_t are learnable parameters indicating the variability of the function with regards to the hyperparameter input $\mathbf{x}$ and number of training

Table B.1: Duelling DQN on CartPole problem.

Variables	Min	Max	Best Found $\mathbf{x}^*$
γ discount factor	0.8	1	0.95586
learning rate model	$1e^{-6}$	0.01	0.00589
#Episodes	300	800	-

iterations t . Estimating appropriate values for them is critical as this represents the GPs prior regarding the sensitivity of performance w.r.t. changes in the number of training iterations and hyperparameters.

We fit the GP hyperparameters by maximising their posterior probability (MAP), $p(\sigma_x, \sigma_t \mid \mathbf{x}, \mathbf{t}, \mathbf{y}) \propto p(\sigma_x, \sigma_t, \mathbf{x}, \mathbf{t}, \mathbf{y})$, which, thanks to the Gaussian likelihood[153], is available in closed form as

$$\ln p(\mathbf{y}, \mathbf{x}, \mathbf{t}, \sigma_x, \sigma_t) = \frac{1}{2} \mathbf{y}^T (\mathbf{K} + \sigma_y^2 \mathbf{I}_N)^{-1} \mathbf{y} - \frac{1}{2} \ln |\mathbf{K} + \sigma_y^2 \mathbf{I}_N| + \ln p_{hyp}(\sigma_x, \sigma_t) + \text{const} \quad (\text{B.1})$$

where $\mathbf{I}_N$ is the identity matrix in dimension N (the number of points in the training set), and $p_{hyp}(\sigma_x, \sigma_t)$ is the prior over hyperparameters, described in the following.

We optimise Eq. (B.1) with a gradient-based optimiser, providing the analytical gradient to the algorithm. We start the optimisation from the previous hyperparameter values θ_{prev} . If the optimisation fails due to numerical issues, we keep the previous value of the hyperparameters. We refit the hyperparameters every $3 \times d$ function evaluations where d is the dimension.

B.2 Experiment settings

We summarise the hyperparameters to be optimised, their search ranges and values identified as optimal for the different experiment settings in Tables B.1, B.2 and B.3. Additional details of the RL agents are given in Table B.4.

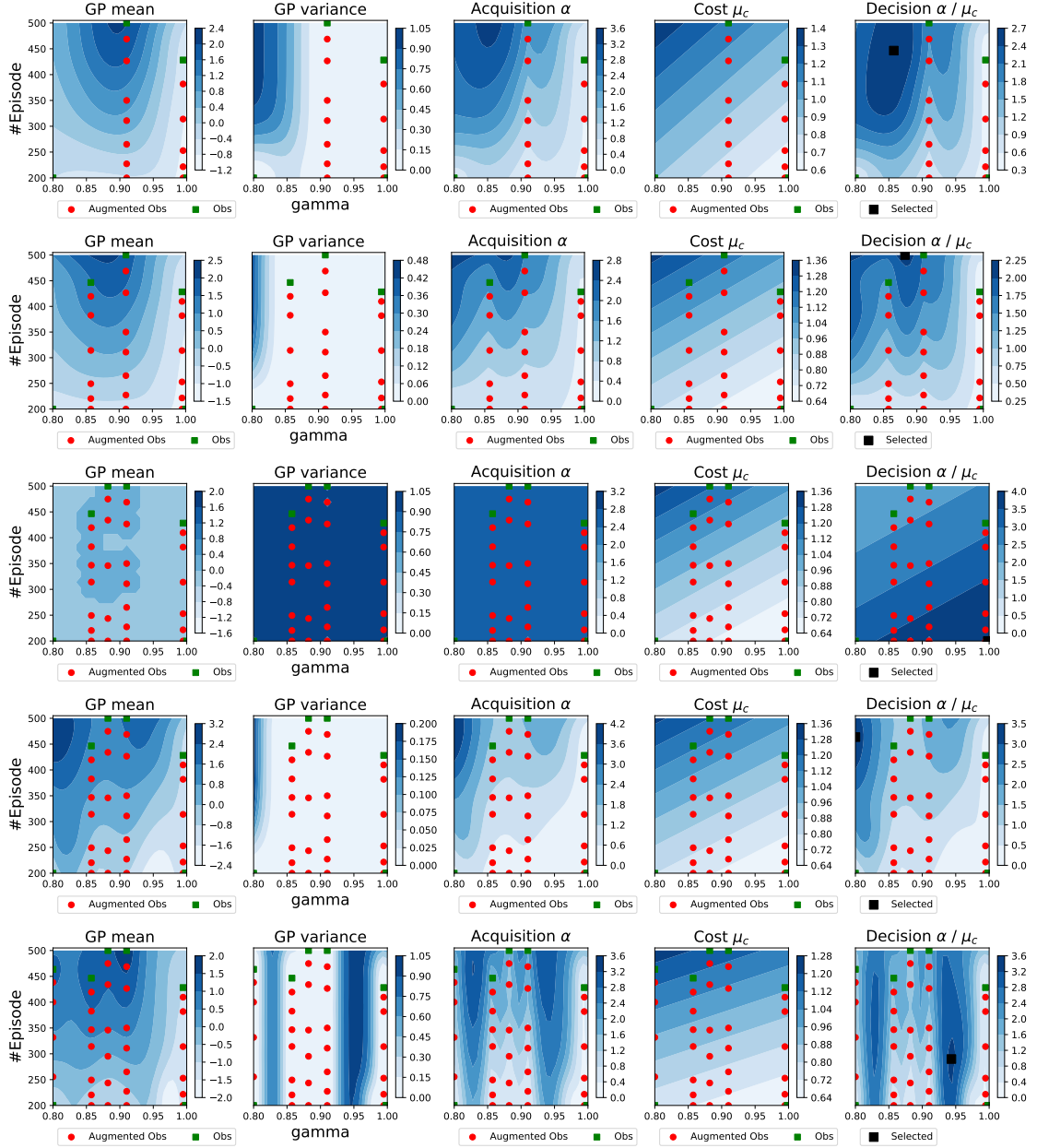


Figure B.1: Illustration of BOIL on a 2-dimensional optimisation task of DDQN on CartPole. The augmented observations fill the joint hyperparameter-iteration space quickly to inform our surrogate. Our decision balances utility α against cost τ for iteration-efficiency. Especially in situations of multiple locations sharing the same utility value, our algorithm prefers to select the cheapest option.

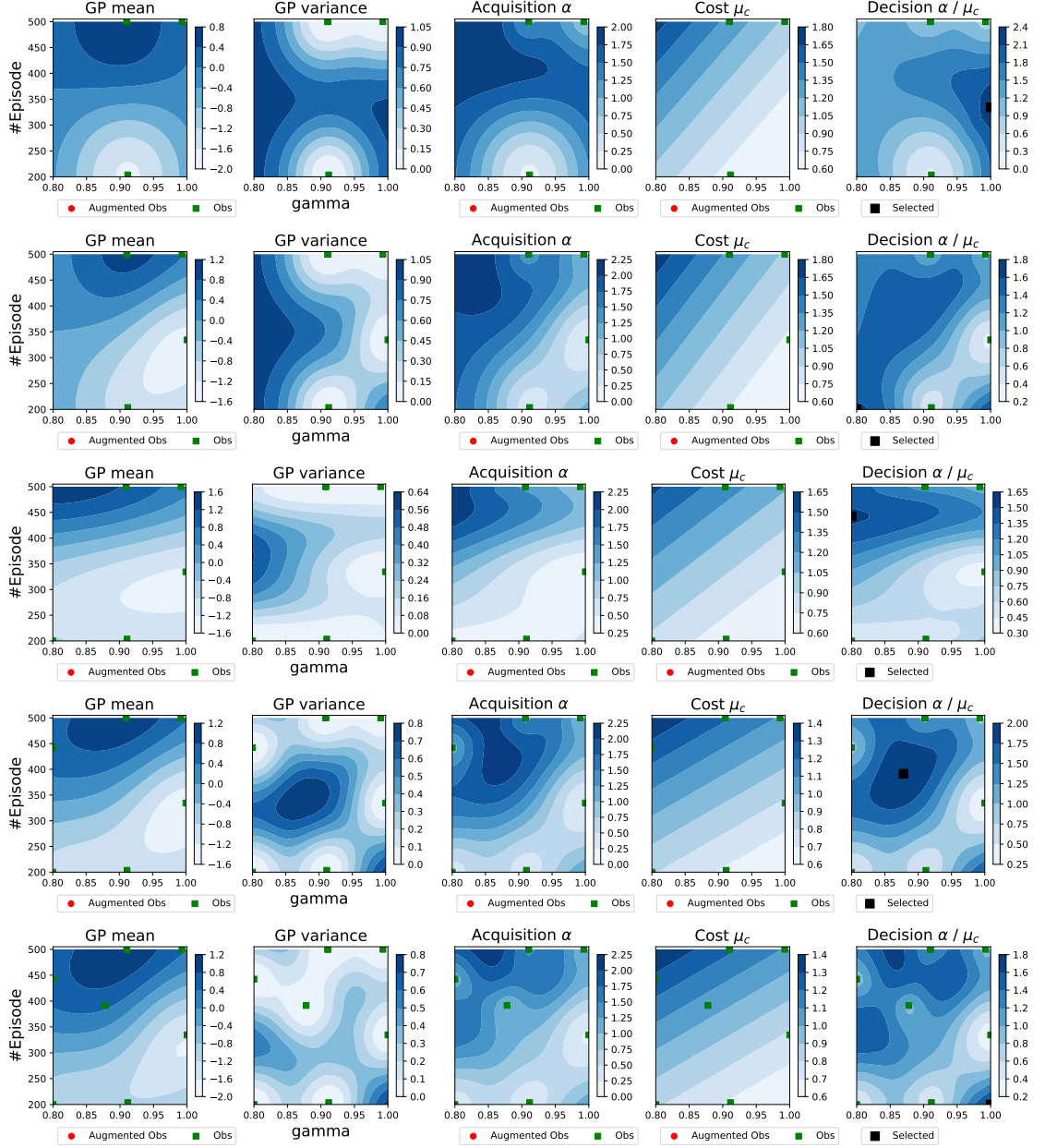


Figure B.2: Illustration of the Continuous Multi task/fidelity BO (CM-T/F-BO) – this is the case of BOIL **without** using augmented observations (same setting as Fig. B.1). This version leads to less efficient optimisation as the additional iteration dimension requires more evaluation than optimising the hyperparameters on their own.

Table B.2: A2C on Reacher (left) and InvertedPendulum (right).

Variables	Min	Max	Best Found $\mathbf{x}^*$	Min	Max	Best Found $\mathbf{x}^*$
γ discount factor	0.8	1	0.8	0.8	1	0.95586
learning rate actor	$1e^{-6}$	0.01	0.00071	$1e^{-6}$	0.01	0.00589
learning rate critic	$1e^{-6}$	0.01	0.00042	$1e^{-6}$	0.01	0.00037
#Episodes	200	500	-	700	1500	-

Table B.3: Convolutional Neural Network.

Variables	Min	Max	Best Found $\mathbf{x}^*$
filter size	1	8	5
pool size	1	5	5
batch size	16	1000	8
learning rate	$1e^{-6}$	0.01	0.000484
momentum	0.8	0.999	0.82852
decay	0.9	0.999	0.9746
number of epoch	30	150	-

B.3 Algorithm Illustration and Further Experiments

Fig. B.1 and Fig. B.2 illustrate the behaviour of our proposed algorithm BOIL on the example of optimising the discount factor γ of Duelling DQN [203] on the CartPole problem. The two settings differ in the inclusion augmented observations into BOIL in Fig. B.1 and CM-T/F-BO (or BOIL without augmented observations) in Fig. B.2.

In both cases, we plot the GP predictive mean in Eq. (1), GP predictive variance in Eq. (2), the acquisition function in Eq. (3), the predicted function and the final decision function in Eq. (8). These equations are defined in the main manuscript.

Table B.4: Further DRL agent specifications

Duelling DQN		A2C	
Q-network	[50, 50]	Critic-network	[32, 32]
ϵ -greedy (annealed)	(1.0, 0.05, 10000)	Actor-network	[32, 32]
Buffer size	10000	Entropy coefficient	0.01
Batch size	64		
PER- α	1.0		
PER- β (annealed)	(1.0, 0.6, 1000)		

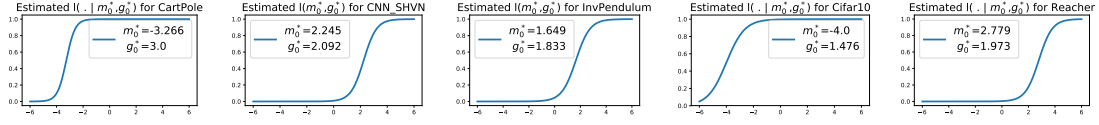


Figure B.3: Learned logistic scoring functions l estimated for different environments and datasets.

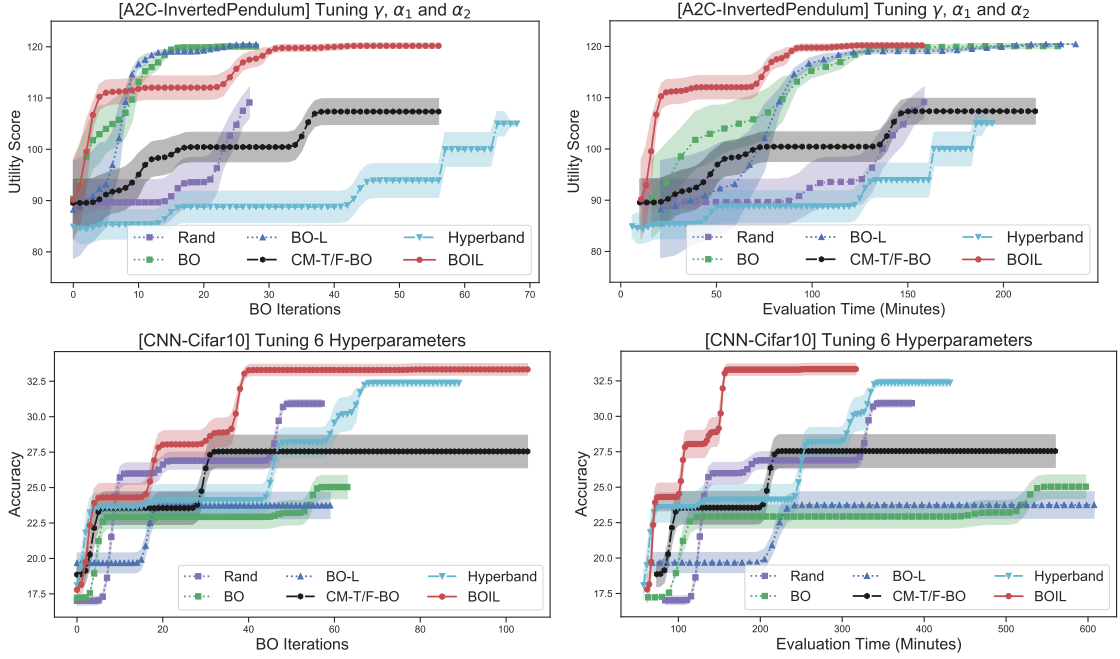


Figure B.4: Tuning hyperparameters of a DRL on InvertedPendulum and a CNN model on CIFAR10.

As shown in the respective figures the final decision function balances between utility and cost of any pair (γ, t) to achieve iteration efficiency. Especially in situations where multiple locations share the same utility value, our decision will prefer to select the cheapest option. Using the augmented observations in Fig. B.1, our joint space is filled quicker with points and the uncertainty (GP variance) across it reduces faster than in Fig. B.2 – the case of vanilla CM-T/F-BO without augmenting observations. A second advantage of having augmented observations is that the algorithm is discouraged to select the same hyperparameter setting at lower fidelity than a previous evaluation.

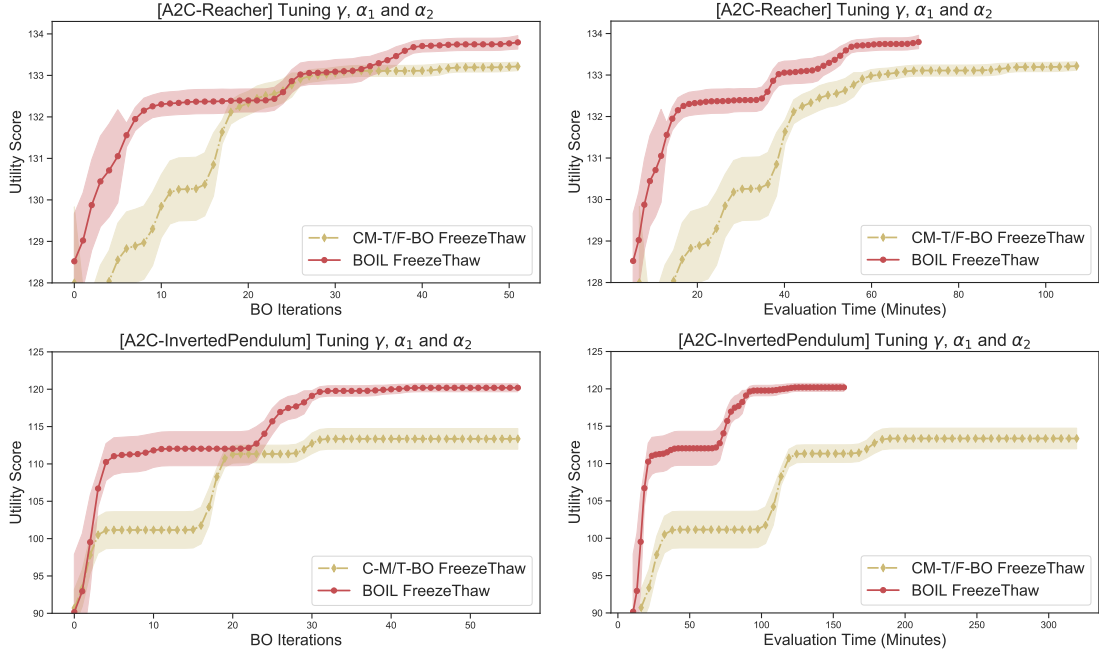


Figure B.5: Comparison to a FreezeThaw time kernel.

B.3.1 Ablation Study using a Freeze-Thaw Time Kernel

In the joint modelling framework of hyperparameter and time (iteration), we can replace the kernel either $k(\mathbf{x}, \mathbf{x})$ or $k(t, t)$ with different choices. We, therefore, set up a new baseline of using the time-kernel $k(t, t')$ in FreezeThaw approach [189] which encodes the monotonously exponential decay from the curve. Particularly, we use the kernel defined as

$$k(t, t') = \frac{\beta^\alpha}{(t + t' + \beta)^\alpha}$$

for parameters $\alpha, \beta > 0$ which are optimised in the GP models.

We present the result in Fig. B.5 that CM-T/F-BO is still less competitive to BOIL using this specific time kernel. The results again validate the robustness our approach cross different choices of kernel.

B.3.2 Additional Experiments for Tuning DRL and CNN

We present the additional experiments for tuning a DRL model using Inverted-Pendulum environment and a CNN model using a subset of CIFAR10 in Fig. B.4.

Again, we show that the proposed model clearly outperforms the baselines in tuning hyperparameters for models with iterative learning available.

C

Supplementary Material to Chapter 6

C.1 Summary of WSABI

For completeness we include a summary of the Warped Sequential Active Bayesian Integration (WSABI) algorithm proposed by Gunter et al. [71], upon which our work builds heavily.

Recall that we are interested in computing integrals of the form $\int f(\theta) d\pi(\theta)$, where $f(\theta)$ is non-negative. We enforce this constraint by modelling $g(\theta) = \sqrt{2(f(\theta) - \epsilon)} \sim \mathcal{GP}(0, \kappa)$. This induces a non-central Chi-squared distribution over $f(\theta)$.

Now, denote by Θ the set of observation locations, the matching transformed observations by $z = \sqrt{2(f(\Theta) - \epsilon)}$, the covariance between all pairs of observations as $\kappa_{\Theta\Theta}$, the covariance between an arbitrary point θ and the set of observations as $\kappa_{\theta\Theta}$ and the kernel between two arbitrary points as $\kappa_{\theta_1\theta_2}$. Additionally, let $\mathfrak{D} = (\Theta, z)$.

Further, recall that the posterior over $g(\theta)$ conditioned on z is given by $\mathcal{GP}(\mu_{\mathfrak{D}}, \Sigma_{\mathfrak{D}})$ with:

$$\mu_{\mathfrak{D}} = \kappa_{\theta\Theta} \kappa_{\Theta\Theta}^{-1} z$$

$$\Sigma_{\mathfrak{D}} = \kappa_{\theta_1\theta_2} - \kappa_{\theta_1\Theta} \kappa_{\Theta\Theta}^{-1} \kappa_{\theta_2\Theta}^T$$

By taking a Taylor expansion around the posterior mean of $g(\theta)$, $\mu_{\mathfrak{D}}$, we can approximate

$$\begin{aligned} f(g) &\approx f(\mu_{\mathfrak{D}}) + (g - \mu_{\mathfrak{D}}) \frac{df}{dg} \Big|_{g=\mu_{\mathfrak{D}}} \\ &= \epsilon - \frac{1}{2} \mu_{\mathfrak{D}}^2 + \mu_{\mathfrak{D}} g. \end{aligned}$$

As this is a linear transform of g , we can approximate the distribution over f as $\mathcal{GP}(\mathbf{m}_{\mathfrak{D}}, \mathfrak{K}_{\mathfrak{D}})$ with moments

$$\begin{aligned} \mathbf{m}_{\mathfrak{D}}(\theta) &= \epsilon + \frac{1}{2} \mu_{\mathfrak{D}}(\theta)^2, \\ \mathfrak{K}_{\mathfrak{D}}(\theta_1, \theta_2) &= \mu_{\mathfrak{D}}(\theta_1) \Sigma_{\mathfrak{D}}(\theta_1, \theta_2) \mu_{\mathfrak{D}}(\theta_2). \end{aligned}$$

Then the first two moments of the integral of interest are given by

$$\begin{aligned} \mathbb{E} \left[\int f(\theta) d\pi(\theta) \right] &= \int \mathbb{E}[f(\theta)] d\pi(\theta) = \int \left(\epsilon + \frac{\mu_{\mathfrak{D}}(\theta)^2}{2} \right) d\pi(\theta) \\ &= \epsilon + \frac{1}{2} \left(z^T \kappa_{\Theta\Theta}^{-1} \int \kappa_{\theta\Theta}^T \kappa_{\theta\Theta} d\pi(\theta) \kappa_{\Theta\Theta}^{-1} z \right) \\ &= \epsilon + \frac{1}{2} z^T Q z. \end{aligned}$$

and

$$\begin{aligned} \text{Cov} \left[\int f(\theta) d\pi(\theta) \right] &= \int \int \mu_{\mathfrak{D}}(\theta) \Sigma_{\mathfrak{D}}(\theta, \theta') \mu_{\mathfrak{D}}(\theta') d\pi(\theta) d\pi(\theta') \\ &= z^T \kappa_{\Theta\Theta}^{-1} \left(\int \int \kappa_{\theta\theta'} \kappa_{\theta\Theta}^T \kappa_{\theta'\Theta} d\pi(\theta) d\pi(\theta') \right. \\ &\quad \left. - \int \kappa_{\theta\Theta}^T \kappa_{\theta\Theta} d\pi(\theta) \kappa_{\Theta\Theta}^{-1} \int \kappa_{\theta'\Theta}^T \kappa_{\theta'\Theta} d\pi(\theta') \right) \kappa_{\Theta\Theta}^{-1} z \end{aligned}$$

Where the quadrature weights, $Q = \kappa_{\Theta\Theta}^{-1} \int \kappa_{\theta\Theta}^T \kappa_{\theta\Theta} d\pi(\theta) \kappa_{\Theta\Theta}^{-1}$, only depend on the particular kernel choice and locations of already observed function values.

For the numerator of Equation (4) in the main text we simply substitute $z_* = \sqrt{2(p(y_* | x_*, D, \theta)p(D | \theta) - \epsilon)}$ for $z = \sqrt{2(p(D | \theta) - \epsilon)}$. The predictive posterior is then a weighted sum of products of Gaussians.

C.2 A Remark on Posterior Inference (6.4)

The predictive posterior given in (6.4) is typically viewed through the hierarchical Bayesian framework:

$$\begin{aligned}
 p(y_* | x_*, D) &= \dots = \frac{\int p(y_* | x_*, \Theta, D) p(D | \Theta) p(\Theta) d\Theta}{\int p(D | \Theta) p(\Theta) d\Theta} \\
 &= \frac{\int \int p(y_* | x_*, \theta^{(n)}, D) p(D | \theta^{(n)}) p(\theta^{(n)}, n) d\theta^{(n)} dn}{\int \int p(D | \theta^{(n)}) p(\theta^{(n)}, n) d\theta^{(n)} dn} \\
 &= \frac{\sum_{n=1}^N p(n) \int p(y_* | x_*, \theta^{(n)}, D) p(D | \theta^{(n)}) p(\theta^{(n)}) d\theta^{(n)}}{\sum_{n=1}^N p(n) \int p(D | \theta^{(n)}) p(\theta^{(n)}) d\theta^{(n)}},
 \end{aligned} \tag{C.1}$$

where $\theta^{(n)}$ is the description of a GMM with exactly n components. For our experiments we choose uniform priors over n , which cancel (note that this is not an inherent limitation of our method).

A naïve approach performs the integrals over $\theta^{(n)}$ separately before computing the sums over n . The ability of MASKERADE to share information across models enables the selection of acquisitions which are maximally informative about the *sum of integrals rather than each individual integral separately*, improving the convergence rate. Additionally, accounting for the covariance between integrals over $\theta^{(n)}$ improves the quality of the posterior over the result of the summations.

C.2.1 Derivation of Information Theoretic Sample Acquisition

The acquisition function is the expected information gained about the integral by making a set of likelihood observations

$$\alpha(\theta_*^{(n)}) = \int \left(H[Z | z, \Theta] - H[Z | z_*, \theta_*^{(n)}, z, \Theta] \right) p(z_* | \theta_*^{(n)}, z, \Theta) dz_*$$

Due to the symmetry of the mutual information, we can swap of Z and z_* [52]

$$\begin{aligned}
 \alpha(\theta_*^{(n)}) &= H[z_* | z, \Theta] - \int H[z_* | \theta_*^{(n)}, Z, z, \Theta] p(Z | z, \Theta) dZ \\
 &= H[z_* | z, \Theta] - H[z_* | \theta_*^{(n)}, Z, z, \Theta]
 \end{aligned}$$

where H is differential entropy (always of Gaussians). The second line follows because the entropy of the Gaussian $p(z_* | \theta_*^{(n)}, Z, z, \Theta)$ does not depend on the value of Z .

Table C.1: The effect of the number of Monte Carlo samples used to estimate the kernel integrals (i.e. the those described in Section C.1). For a MASKERADE model that marginalises over up to 5 mixture components we randomly sample sets of 100, 500 and 1000 hyperparameters from the prior, and compute their corresponding likelihoods on the UCI Airfoil Self-Noise dataset. We then infer the model evidence with WSABI using the MASKERADE hyper-kernel. For a given number of MC samples, we repeat this five times and report the mean and SEM for the posterior mean of the model evidence. We observe that the estimate for the model evidence is not highly sensitive to the number of monte carlo samples used to compute the kernel integrals.

No. LIKELIHOOD OBSERVATIONS	NO. OF MC SAMPLES		
	100	1000	10000
100	0.02546 ± 0.00149	0.02542 ± 0.00020	0.02526 ± 0.00012
500	0.00369 ± 0.00035	0.00379 ± 0.00013	0.00358 ± 0.00003
1000	0.00333 ± 0.00095	0.00292 ± 0.00031	0.00292 ± 0.00006

C.2.2 Monte Carlo Approximation to the Kernel Integrals

The kernel integrals are approximated using Quasi Monte Carlo sampling under the prior. Table C.1 shows the effect of varying the number of QMC samples.

C.3 Full Algorithm Description

For completeness we once more give the full generative model:

$$\begin{aligned}
n &\sim \text{Uniform}(\{1, 2, \dots, N\}), \\
w &\sim \text{Dirichlet}(\alpha), \\
m_{1..n} &\sim \mathcal{N}(\mu, \Sigma), \\
\sigma_{1..n} &\sim \text{Log-Normal}(\nu, \tau), \\
S(\omega) &= \sum_{j=1}^n \frac{w_j}{2} \left(\mathcal{N}(\omega; m_j, \sigma_j) + \mathcal{N}(\omega; -m_j, \sigma_j) \right), \\
k(x, x') &= \int e^{2\pi i |x-x'| \omega} S(\omega) d\omega, \\
f &\sim \mathcal{GP}(0, k).
\end{aligned} \tag{C.2}$$

Below, we provide a summary of the MASKERADE framework in Algorithm 1. A schematic outlining our method is made available in Figure C.1.

Algorithm 3 Pseudocode for both the learning and prediction phases of our algorithm.

```

obtain initial samples  $\Theta, z$   $\{z$  are likelihood evaluations at  $\Theta\}$ 
 $\lambda, l \leftarrow \operatorname{argmax}_{\lambda, l} p(z \mid \Theta, \lambda, l)$   $\{\text{Optimise BQ Surrogate}\}$ 
while  $i \geq 0$  do
   $\{\theta_s^{(n)}\}_{1:b} \leftarrow \operatorname{argmax}_{\{\theta_s^{(n)}\}_{1:b}} \alpha(\{\theta_s^{(n)}\}_{1:b})$   $\{\text{Optimise acquisition function}\}$ 
  append  $\{\theta_s^{(n)}\}_{1:b}$  to  $\Theta$ .
  append  $\text{likelihood}(\{\theta_s^{(n)}\}_{1:b})$  to  $z$ .
   $\lambda, l \leftarrow \operatorname{argmax}_{\lambda, l} p(z \mid \Theta, \lambda, l)$   $\{\text{Optimise BQ surrogate}\}$ 
   $i \leftarrow i - 1$ 
end while
 $Q \leftarrow \text{compute\_bq\_weights}(\Theta, z, \lambda, l, \phi)$   $\{Q \text{ defined as in Appendix on WSABI above. } \phi \text{ are all prior parameters.}\}$ 
for  $\theta_s^{(n)}$  in  $\Theta$  do
   $\mu_*, \Sigma_* \leftarrow \text{predict}(x_*, D, \theta_s^{(n)})$ 
end for
return  $\text{predictive\_posterior}(Q, \{\mu_*\}, \{\Sigma_*\}, z)$   $\{\text{See Appendix on WSABI for details.}\}$ 

```

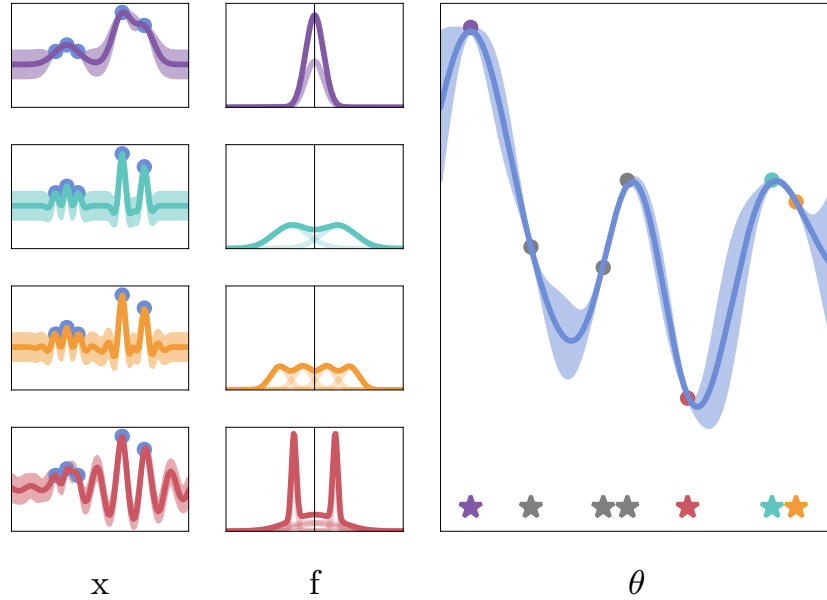


Figure C.1: A schematic representation of the inference procedure under our model. The left column shows the GP posterior for different Spectral Mixture Kernels on the same dataset. The centre column shows the SM kernels in their spectral domain. The right column illustrates a GP posterior on a space indexed by the spectral densities of the SM kernels. BQ can then be used to marginalise over SM kernels which may have different numbers of mixture components.

C.4 Qualitative Analysis

We qualitatively inspect the behaviour of MASKERADE in Figure C.2 by examining the spectra of kernels assigned the highest weights in the posterior, and the effect of varying the number of mixture components that are marginalised over.

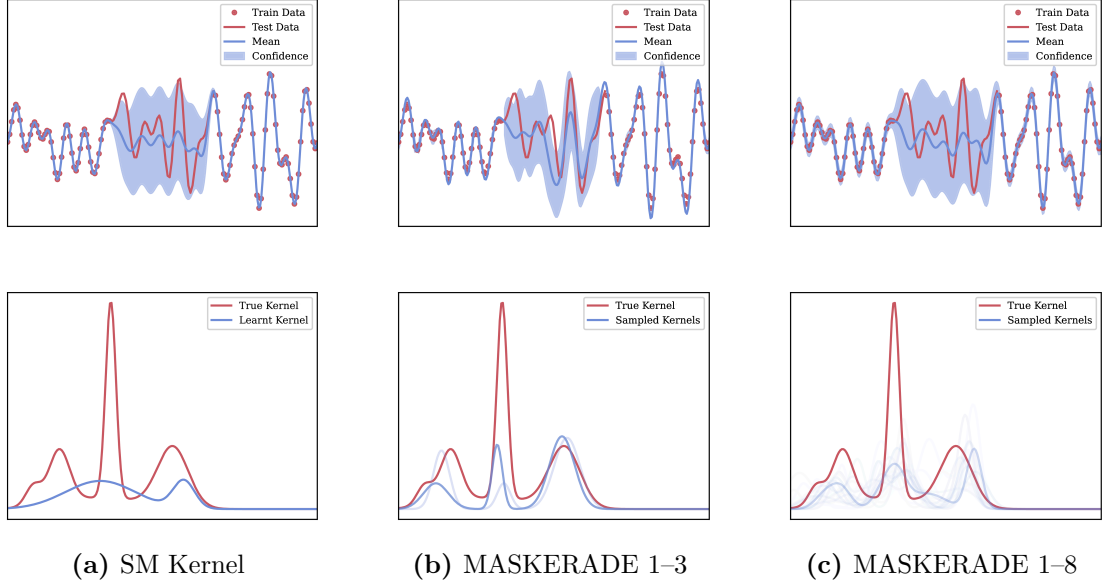


Figure C.2: A comparison of a 5 component SM kernel with optimised hyperparameters, and two variants of MASKERADE – one that places a prior over up to 3 mixture components, and the other up to 8 mixture components – on a toy dataset drawn from a 5 component SM kernel. Each column plots attributes of the labelled model. The first row shows the posterior conditioned on the training data (for the MASKERADE models we show the moment-matched posterior). The second row shows the spectra (for positive frequencies) of the data generating kernel, and (for the 5 component SM kernel model) the optimised or (for the MASKERADE models) the sampled kernels. For the MASKERADE model, the opacity of a sampled kernel is proportional to the quadrature weight for all GP products of which that kernel is a part. (Recall that the posterior is a weighted sum of products of GP posteriors.) MASKERADE 1-3 is able to select samples near the data generating kernel, and therefore produce a posterior that generalises better than the other two models. Despite having the same number of mixture components as the data generating kernel, the optimised SM kernel sets the weights of 3 components to be very small. MASKERADE 1-8 struggles to explore its larger hyperparameter space with the same budget (500 likelihood evaluations) as MASKERADE 1-3. This can be seen by the fact that it spreads its posterior mass more evenly over a larger number of samples.

Next, we visually verify that our method improves upon baselines by plotting fits on the Mauna Loa dataset. These are shown in Figure C.3. MASKERADE is parameterised to marginalise over 5 mixture components, with priors set as

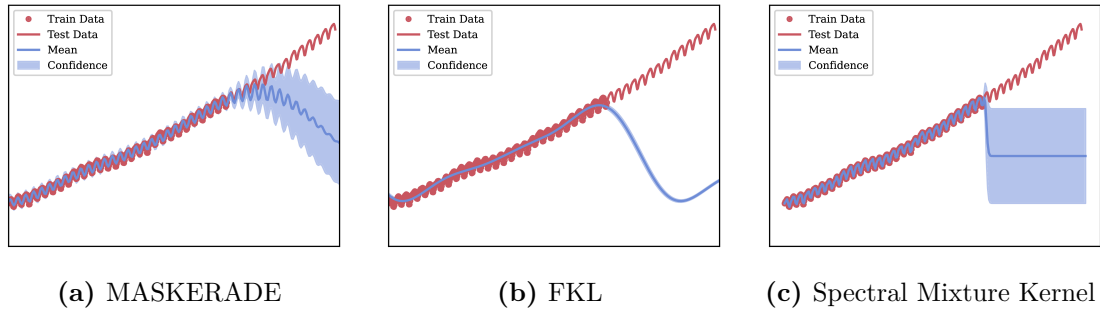


Figure C.3: Posteriors for several methods on the Mauna Loa dataset (For MASKERADE and FKL we show moment matched posteriors). All methods struggle to model the linear trend, but MASKERADE is best able to extrapolate the periodic structure.

in Section 3.1.1. FKL uses the defaults in the `spectralgp`¹ Python package, with the maximum frequency set to the Nyquist frequency of the Mauna Loa dataset. The Spectral Mixture kernel uses 5 mixtures, initialised using the empirical spectrum of the dataset.

¹<https://github.com/wjmaddox/spectralgp>

References

- [1] Pieter Abbeel and Andrew Y. Ng. “Apprenticeship Learning via Inverse Reinforcement Learning”. In: *Proceedings of the 21st International Machine Learning Conference*. ACM Press, 2004.
- [2] Luca Ambrogioni and Eric Maris. “Integral Transforms from Finite Data: An Application of Gaussian Process Regression to Fourier Analysis”. In: *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics*. Vol. 84. Proceedings of Machine Learning Research. 2018, pp. 217–225.
- [3] Dario Amodei and Danny Hernandez. *AI and Compute*. <https://openai.com/blog/ai-and-compute/>. Blog Post. 2018.
- [4] Dario Amodei et al. *Concrete Problems in AI Safety*. 2016. arXiv preprint arXiv: [1606.06565](https://arxiv.org/abs/1606.06565) (cs.AI).
- [5] Thomas Anthony, Zheng Tian, and David Barber. “Thinking Fast and Slow with Deep Learning and Tree Search”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017.
- [6] Mauricio Araya-López, Olivier Buffet, and Vincent Thomas. “Near-Optimal BRL using Optimistic Local Transitions”. In: *Proceedings of the 29th International Conference on Machine Learning*. Omnipress, 2012.
- [7] Isac Arnekvist, Danica Kragic, and Johannes A. Stork. “VPE: Variational Policy Embedding for Transfer Reinforcement Learning”. In: *International Conference on Robotics and Automation*. 2019, pp. 36–42.
- [8] Kai Arulkumaran et al. “Deep Reinforcement Learning: A Brief Survey”. In: *IEEE Signal Processing Magazine* 34.6 (2017), pp. 26–38.
- [9] John Asmuth and Michael L. Littman. “Learning is Planning: Near Bayes-optimal Reinforcement Learning via Monte-Carlo Tree Search”. In: *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*. 2011.
- [10] John Asmuth et al. “A Bayesian Sampling Approach to Exploration in Reinforcement Learning”. In: *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*. AUAI Press. 2009, pp. 19–26.
- [11] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. “Finite-time Analysis of the Multiarmed Bandit Problem”. In: *Machine learning* 47.2-3 (2002), pp. 235–256.
- [12] Mohammad G. Azar, Ian Osband, and Rémi Munos. “Minimax Regret Bounds for Reinforcement Learning”. In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. Proceedings of Machine Learning Research. 2017, pp. 263–272.

- [13] François Bachoc et al. “A Gaussian Process Regression Model for Distribution Inputs”. In: *IEEE Transactions on Information Theory* 64.10 (2018), pp. 6620–6637.
- [14] François Bachoc et al. “Gaussian Processes with Multidimensional Distribution Inputs via Optimal Transport and Hilbertian Embedding”. In: *Electronic Journal of Statistics*. Vol. 14. 2. Institute of Mathematical Statistics and Bernoulli Society, 2020, pp. 2742–2772.
- [15] Leemon Baird. “Residual Algorithms: Reinforcement Learning with Function Approximation”. In: *Proceedings of the 12th International Conference on Machine Learning*. Morgan Kaufmann Publishers, Inc., 1995, pp. 30–37.
- [16] Marc G. Bellemare et al. “The Arcade Learning Environment: An Evaluation Platform for General Agents”. In: *Journal of Artificial Intelligence Research* 47 (2013), pp. 253–279.
- [17] Richard Bellman. “A Problem in the Sequential Design of Experiments”. In: *Sankhyā: The Indian Journal of Statistics (1933-1960)* 16.3/4 (1956), pp. 221–229.
- [18] Richard Bellman. “The Theory of Dynamic Programming”. In: *The Bulletin of the American Mathematical Society* 60 (1954), pp. 503–515.
- [19] Gregory Benton et al. “Function-Space Distributions over Kernels”. In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019.
- [20] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- [21] Salomon Bochner. *Lectures on Fourier Integrals*. Vol. 42. Princeton University Press, 1959.
- [22] Vladimir Igorevich Bogachev. *Gaussian Measures*. 1961.
- [23] Ronen I. Brafman and Moshe Tennenholtz. “R-max a General Polynomial Time Algorithm for near-optimal Reinforcement Learning”. In: *Journal of Machine Learning Research* 3 (2002), pp. 213–231.
- [24] Greg Brockman et al. *OpenAI Gym*. 2016. arXiv preprint arXiv: [1606.01540](https://arxiv.org/abs/1606.01540) (cs.LG).
- [25] Emma Brunskill. “Bayes-optimal Reinforcement Learning for Discrete Uncertainty Domains”. In: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*. 2012, pp. 1385–1386.
- [26] Sebastien Bubeck and Nicolo Cesa-Bianchi. “Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems”. In: *Foundations and Trends® in Machine Learning* 5.1 (2012), pp. 1–122.
- [27] Anthony R. Cassandra, Leslie P. Kaelbling, and Michael L. Littman. “Acting Optimally in Partially Observable Stochastic Domains”. In: *Proceedings of the 12th National Conference on Artificial Intelligence*. Vol. 2. AAAI Press, 1994, pp. 1023–1028.
- [28] Michael Castronovo et al. “Benchmarking for Bayesian Reinforcement Learning”. In: *PloS One* 11.6 (2016).

- [29] Henry Chai et al. “Automated Model Selection with Bayesian Quadrature”. In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. Proceedings of Machine Learning Research. 2019, pp. 931–940.
- [30] Henry R. Chai and Roman Garnett. “Improving Quadrature for Constrained Integrands”. In: *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*. Vol. 89. Proceedings of Machine Learning Research. 2019, pp. 2751–2759.
- [31] Yutian Chen et al. *Bayesian Optimization in AlphaGo*. 2018. arXiv preprint arXiv: [1812.06855](#) (cs.LG).
- [32] Paul F. Christiano et al. “Deep Reinforcement Learning from Human Preferences”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017.
- [33] Rémi Coulom. “Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search”. In: *5th International Conference on Computers and Games*. Springer, 2006, pp. 72–83.
- [34] Kurt Cutajar et al. “Preconditioning Kernel Matrices”. In: *Proceedings of the 33rd International Conference on Machine Learning*. Vol. 48. Proceedings of Machine Learning Research. 2016, pp. 2529–2538.
- [35] Marco Cuturi. “Sinkhorn Distances: Lightspeed Computation of Optimal Transport”. In: *Advances in Neural Information Processing Systems*. Vol. 26. Curran Associates, Inc., 2013.
- [36] Zhongxiang Dai et al. “Bayesian Optimization Meets Bayesian Optimal Stopping”. In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. Proceedings of Machine Learning Research. 2019, pp. 1496–1506.
- [37] Christian Daniel et al. “Active Reward Learning”. In: *Robotics: Science and System X*. 2014.
- [38] Christoph Dann et al. “Policy Certificates: Towards Accountable Reinforcement Learning”. In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. Proceedings of Machine Learning Research. 2019, pp. 1507–1516.
- [39] Pieter-Tjerk De Boer et al. “A Tutorial on the Cross-entropy Method”. In: *Annals of Operations Research* 134.1 (2005), pp. 19–67.
- [40] Richard Dearden, Nir Friedman, and Stuart Russell. “Bayesian Q-learning”. In: *Proceedings of the 15th National Conference on Artificial Intelligence*. AAAI Press, 1998, pp. 761–768.
- [41] Prafulla Dhariwal et al. “Openai Baselines”. In: *GitHub, GitHub repository* (2017).
- [42] Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. “Speeding up Automatic Hyperparameter Optimization of Deep Neural Networks by Extrapolation of Learning Curves”. In: *Proceedings of the 24th International Joint Conference on Artificial Intelligence*. AAAI Press, 2015, pp. 3460–3468.
- [43] Ron Dorfman, Idan Shenfeld, and Aviv Tamar. *Offline Meta Learning of Exploration*. 2020. arXiv preprint arXiv: [2008.02598](#) (cs.LG).

- [44] Finale Doshi-Velez and George Konidaris. “Hidden Parameter Markov Decision Processes: A Semiparametric Regression Approach for Discovering Latent Task Parametrizations”. In: *Proceedings of the 25th International Joint Conference on Artificial Intelligence*. AAAI Press, 2016, pp. 1432–1440.
- [45] Anca D. Dragan. *Robot Planning with Mathematical Models of Human State and Action*. 2017. arXiv preprint arXiv: [1705.04226](#) (cs.RO).
- [46] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017. URL: <http://archive.ics.uci.edu/ml>.
- [47] Yan Duan et al. “One-Shot Imitation Learning”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017.
- [48] Yan Duan et al. *RL²: Fast Reinforcement Learning via Slow Reinforcement Learning*. 2016. arXiv preprint arXiv: [1611.02779](#) (cs.AI).
- [49] Michael Duff. “Optimal Learning: Computational Procedures For Bayes-Adaptive Markov Decision Processes”. PhD thesis. Univ of Massachusetts at Amherst, 2002.
- [50] David Duvenaud et al. “Structure Discovery in Nonparametric Regression through Compositional Kernel Search”. In: *Proceedings of the 30th International Conference on Machine Learning*. Vol. 28. Proceedings of Machine Learning Research 3. 2013, pp. 1166–1174.
- [51] Bank of England. *Broad Effective Exchange Rate Index, Sterling*. 2020. URL: <https://www.quandl.com/data/BOE/XUDLBK82-Broad-Effective-Exchange-%20Rate-Index-Sterling-jan-2005-100>.
- [52] Arkady Epshteyn, Adam Vogel, and Gerald DeJong. “Active Reinforcement Learning”. In: *Proceedings of the 25th International Conference on Machine learning*. OmniPress, 2008, pp. 296–303.
- [53] Owain Evans, Andreas Stuhlmüller, and Noah D. Goodman. “Learning the Preferences of Ignorant, Inconsistent Agents”. In: *Proceedings of the 30th AAAI Conference on Artificial Intelligence*. AAAI Press. 2016, pp. 323–329.
- [54] Stefan Falkner, Aaron Klein, and Frank Hutter. In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. Proceedings of Machine Learning Research. 2018, pp. 1437–1446.
- [55] Aasa Feragen, Francois Lauze, and Søren Hauberg. “Geodesic Exponential Kernels: When Curvature and Linearity Conflict”. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3032–3042.
- [56] Jean Feydy. “Geometric data analysis, beyond convolutions”. PhD thesis. Paris-Saclay University, 2020, p. 94.
- [57] 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*. Vol. 70. Proceedings of Machine Learning Research. 2017, pp. 1126–1135.
- [58] Vincent Francois-Lavet et al. “An Introduction to Deep Reinforcement Learning”. In: *Foundations and Trends in Machine Learning* 11.3-4 (2018), pp. 219–354.

- [59] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. “Deep Bayesian Active Learning with Image Data”. In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. Proceedings of Machine Learning Research. 2017, pp. 1183–1192.
- [60] Yarin Gal and Richard Turner. “Improving the Gaussian Process Sparse Spectrum Approximation by Representing Uncertainty in Frequency Inputs”. In: *Proceedings of the 32nd International Conference on Machine Learning*. Vol. 37. Proceedings of Machine Learning Research. 2015, pp. 655–664.
- [61] Jacob Gardner et al. “GPpyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration”. In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., 2018.
- [62] Marta Garnelo et al. “Neural Processes”. In: *Workshop on Theoretical Foundations and Applications of Deep Generative Models at the 35th International Conference on Machine Learning*. 2018.
- [63] Alexandra Gessner, Javier Gonzalez, and Maren Mahsererci. “Active Multi-Information Source Bayesian Quadrature”. In: *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*. Vol. 115. Proceedings of Machine Learning Research. 2020, pp. 712–721.
- [64] Zoubin Ghahramani and Carl E. Rasmussen. “Bayesian Monte Carlo”. In: *Advances in Neural Information Processing Systems*. Vol. 15. MIT Press, 2003.
- [65] Mohammad Ghavamzadeh et al. “Bayesian Reinforcement Learning: A Survey”. In: *Foundations and Trends® in Machine Learning* 8.5-6 (2015).
- [66] Jonathan Gordon et al. “Meta-learning Probabilistic Inference for Prediction”. In: *Proceedings of the 7th International Conference on Learning Representations*. OpenReview.net, 2019.
- [67] Peter J. Green. “Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination”. In: *Biometrika* 82.4 (1995), pp. 711–732.
- [68] Arthur Guez. “Sample-Based Search Methods For Bayes-Adaptive Planning”. PhD thesis. University College London, 2015.
- [69] Arthur Guez, David Silver, and Peter Dayan. “Efficient Bayes-Adaptive Reinforcement Learning using Sample-Based Search”. In: *Advances in Neural Information Processing Systems*. Vol. 25. Curran Associates, Inc., 2012.
- [70] Arthur Guez, David Silver, and Peter Dayan. “Scalable and Efficient Bayes-Adaptive Reinforcement Learning Based on Monte-Carlo Tree Search”. In: *Journal of Artificial Intelligence Research* 48 (2013), pp. 841–883.
- [71] Tom Gunter et al. “Sampling for Inference in Probabilistic Models with Fast Bayesian Quadrature”. In: *Advances in Neural Information Processing Systems*. Vol. 27. Curran Associates, Inc., 2014.
- [72] Abhishek Gupta et al. “Meta-Reinforcement Learning of Structured Exploration Strategies”. In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., 2018.
- [73] Dylan Hadfield-Menell et al. “Inverse Reward Design”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017.

- [74] Assaf Hallak, Dotan Di Castro, and Shie Mannor. *Contextual Markov Decision Processes*. 2015. arXiv preprint arXiv: [1502.02259](#) (stat.ML).
- [75] Nikolaus Hansen, Sibylle D. Müller, and Petros Koumoutsakos. “Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES)”. In: *Evolutionary computation* 11.1 (2003), pp. 1–18.
- [76] Karol Hausman et al. “Learning an Embedding Space for Transferable Robot Skills”. In: *Proceedings of the 6th International Conference on Learning Representations*. OpenReview.net, 2018.
- [77] Nicolas Heess et al. “Emergence of Locomotion Behaviours in Rich Environments”. In: (2017). arXiv preprint arXiv: [1707.02286](#) (cs.AI).
- [78] Peter Henderson et al. “Deep Reinforcement Learning that Matters”. In: *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*. 1. 2018.
- [79] Philipp Hennig and Christian J. Schuler. “Entropy Search for Information-efficient Global Optimization”. In: *Journal of Machine Learning Research* 13 (2012), pp. 1809–1837.
- [80] James Hensman, Nicolas Durrande, and Arno Solin. “Variational Fourier Features for Gaussian Processes”. In: *Journal of Machine Learning Research* 18 (2018), pp. 1–52.
- [81] José Miguel Hernández-Lobato, Matthew W. Hoffman, and Zoubin Ghahramani. “Predictive Entropy Search for Efficient Global Optimization of Black-box Functions”. In: *Advances in Neural Information Processing Systems*. Vol. 27. Curran Associates, Inc., 2014.
- [82] Jonathan Ho and Stefano Ermon. “Generative Adversarial Imitation Learning”. In: *Advances in Neural Information Processing Systems*. Vol. 29. Curran Associates, Inc., 2016.
- [83] Rein Houthooft et al. “Evolved Policy Gradients”. In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., 2018.
- [84] Jan Humplik et al. *Meta Reinforcement Learning as Task Inference*. 2019. arXiv preprint arXiv: [1905.06424](#) (cs.LG).
- [85] Maximilian Igl et al. “Deep Variational Reinforcement Learning for POMDPs”. In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. Proceedings of Machine Learning Research. 2018, pp. 2117–2126.
- [86] Max Jaderberg et al. “Reinforcement Learning with Unsupervised Auxiliary Tasks”. In: *Proceedings of the 5th International Conference on Learning Representations*. OpenReview.net, 2017.
- [87] Phillip A. Jang et al. “Scalable Levy Process Priors for Spectral Kernel Learning”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017.
- [88] Sadeep Jayasumana et al. “Kernel Methods on Riemannian Manifolds with Gaussian RBF Kernels”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.12 (2015), pp. 2464–2477.

- [89] Nan Jiang et al. “Contextual Decision Processes with Low Bellman Rank are PAC-Learnable”. In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. Proceedings of Machine Learning Research. 2017, pp. 1704–1713.
- [90] Michael I. Jordan and Tom M. Mitchell. “Machine Learning: Trends, Perspectives, and Prospects”. In: *Science* 349.6245 (2015), pp. 255–260.
- [91] Kshitij Judah, Alan Fern, and Thomas G. Dietterich. “Active Imitation Learning via Reduction to I.I.D. Active Learning”. In: *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2012, pp. 428–437.
- [92] Leslie P. Kaelbling, Michael L. Littman, and Anthony R. Cassandra. “Planning and Acting in Partially Observable Stochastic Domains”. In: *Artificial Intelligence* 101.1-2 (1998), pp. 99–134.
- [93] Kirthevasan Kandasamy et al. “Multi-fidelity Bayesian Optimisation with Continuous Approximations”. In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. Proceedings of Machine Learning Research. 2017, pp. 1799–1808.
- [94] Manohar Kaul. *3D Road Network (North Jutland, Denmark) Data Set*. 2013. URL: [https://archive.ics.uci.edu/ml/datasets/3D+Road+Network+\(North+Jutland,+Denmark\)#](https://archive.ics.uci.edu/ml/datasets/3D+Road+Network+(North+Jutland,+Denmark)#).
- [95] Michael Kearns and Satinder Singh. “Near-optimal Reinforcement Learning in Polynomial Time”. In: *Machine learning* 49.2-3 (2002), pp. 209–232.
- [96] Taylor W. Killian et al. “Robust and Efficient Transfer Learning with Hidden Parameter Markov Decision Processes”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017.
- [97] Diederik P. Kingma and Max Welling. “Auto-Encoding Variational Bayes”. In: *Proceedings of the 2nd International Conference on Learning Representations*. 2014.
- [98] Aaron Klein et al. “Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets”. In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. Vol. 54. Proceedings of Machine Learning Research. 2017, pp. 528–536.
- [99] Aaron Klein et al. “Learning Curve Prediction with Bayesian Neural Networks”. In: *Proceedings of the 5th International Conference on Learning Representations*. OpenReview.net, 2017.
- [100] J. Zico Kolter and Andrew Y. Ng. “Near-Bayesian Exploration in Polynomial Time”. In: *Proceedings of the 26th International Conference on Machine Learning*. Omnipress, 2009, pp. 513–520.
- [101] Yves-Laurent Kom Samo. “Advances in Kernel Methods”. PhD thesis. University of Oxford, 2017.
- [102] Yves-Laurent Kom Samo and Stephen Roberts. *Generalized Spectral Kernels*. 2015. arXiv preprint arXiv: [1506.02236](https://arxiv.org/abs/1506.02236) (stat.ML).
- [103] Andreas Krause and Cheng Ong. “Contextual Gaussian Process Bandit Optimization”. In: *Advances in Neural Information Processing Systems*. Vol. 24. Curran Associates, Inc., 2011.

- [104] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “Imagenet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012.
- [105] David Krueger et al. “Active Reinforcement Learning: Observing Rewards at a Cost”. In: *Future of Interactive Learning Machines Workshop at the 29th Conference on Neural Information Processing Systems*. 2016.
- [106] Volodymyr Kuleshov and Doina Precup. *Algorithms for Multi-armed Bandit Problems*. 2014. arXiv preprint arXiv: [1402.6028](https://arxiv.org/abs/1402.6028) (cs.AI).
- [107] Lin Lan et al. “Meta Reinforcement Learning with Task Embedding and Shared Policy”. In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, 2019, pp. 2794–2800.
- [108] Miguel Lázaro-Gredilla et al. “Sparse Spectrum Gaussian Process Regression”. In: *Journal of Machine Learning Research* 11 (2010), pp. 1–17.
- [109] J. Lean. *Solar Irradiance Reconstruction Data Set, NOAA/NGDC Paleoclimatology Program, Boulder CO, USA*. 2004. URL: ftp://ftp.ncdc.noaa.gov/pub/data/paleo/climate_forcing/solar_variability/lean2000_irradiance.txt.
- [110] Yann LeCun et al. “Gradient-based Learning Applied to Document Recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.
- [111] Gilwoo Lee et al. “Bayesian Policy Optimization for Model Uncertainty”. In: *Proceedings of the 7th International Conference on Learning Representations*. OpenReview.net, 2019.
- [112] Benjamin Letham et al. “Constrained Bayesian Optimization with Noisy Experiments”. In: *Bayesian Analysis* 14.2 (2019), pp. 495–519.
- [113] Lisha Li et al. “Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization”. In: *Journal of Machine Learning Research* 18 (2018), pp. 1–52.
- [114] Timothy P. Lillicrap et al. *Continuous Control with Deep Reinforcement Learning*. 2015. arXiv preprint arXiv: [1509.02971](https://arxiv.org/abs/1509.02971) (cs.LG).
- [115] Long-Ji Lin. “Self-improving Reactive Agents based on Reinforcement Learning, Planning and Teaching”. In: *Machine learning* 8.3-4 (1992), pp. 293–321.
- [116] Yun-En Liu et al. “Trading Off Scientific Knowledge and User Learning with Multi-Armed Bandits”. In: *Proceedings of the 7th International Conference on Educational Data Mining*. 2014, pp. 161–168.
- [117] Roberto Lopez. *Airfoil Self-Noise Data Set*. 2014. URL: <https://archive.ics.uci.edu/ml/datasets/airfoil+self-noise>.
- [118] Roberto Lopez. *Yacht Hydrodynamics Data Set*. 2013. URL: <https://archive.ics.uci.edu/ml/datasets/yacht+hydrodynamics>.
- [119] Gabor Lugosi. “Universal Kernels”. In: *Journal of Machine Learning Research* 7 (2006), pp. 2651–2667.
- [120] S. Makridakis, S. Wheelwright, and R. Hyndman. *Airline Passenger Data Set*. 1998. URL: <https://github.com/robjhyndman/fma/blob/master/data/airpass.rda>.

- [121] James J. Martin. *Bayesian Decision Problems and Markov Chains*. John Wiley & Sons Ltd., 1967.
- [122] Michael McCloskey and Neal J. Cohen. “Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem”. In: *Psychology of Learning and Motivation*. Vol. 24. Elsevier, 1989, pp. 109–165.
- [123] Mark McLeod, Stephen Roberts, and Michael A. Osborne. “Optimization, Fast and Slow: Optimally Switching between Local and Bayesian Optimization”. In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. Proceedings of Machine Learning Research. 2018, pp. 3443–3452.
- [124] Vladimir Mikulik et al. *Meta-trained Agents Implement Bayes-optimal Agents*. 2020. arXiv preprint arXiv: [2010.11223](#) (cs.AI).
- [125] Volodymyr Mnih et al. “Asynchronous Methods for Deep Reinforcement Learning”. In: *Proceedings of The 33rd International Conference on Machine Learning*. Vol. 48. Proceedings of Machine Learning Research. 2016, pp. 1928–1937.
- [126] Volodymyr Mnih et al. “Human-level Control Through Deep Reinforcement Learning”. In: *Nature* 518.7540 (2015), pp. 529–533.
- [127] Volodymyr Mnih et al. *Playing Atari with Deep Reinforcement Learning*. 2013. arXiv preprint arXiv: [1312.5602](#) (cs.LG).
- [128] Aditya Modi and Ambuj Tewari. “Contextual Markov Decision Processes using Generalized Linear Models”. In: *Reinforcement Learning for Real Life Workshop at the 36th International Conference on Machine Learning*. 2019.
- [129] Thomas M. Moerland, Joost Broekens, and Catholijn M. Jonker. *Model-based Reinforcement Learning: A Survey*. 2021. arXiv preprint arXiv: [2006.16712](#) (cs.LG).
- [130] Krikamol Muandet et al. “Kernel Mean Embedding of Distributions: A Review and Beyond”. In: *Foundations and Trends in Machine Learning* 10.1 (2017), pp. 1–141.
- [131] Iain Murray, Ryan Adams, and David MacKay. “Elliptical Slice Sampling”. In: *Proceedings of Machine Learning Research* 9 (2010), pp. 541–548.
- [132] Vu Nguyen and Michael A. Osborne. “Knowing The What But Not The Where in Bayesian Optimization”. In: *Proceedings of the 37th International Conference on Machine Learning*. Vol. 119. Proceedings of Machine Learning Research. 2020, pp. 7317–7326.
- [133] Alex Nichol and John Schulman. *On First-Order Meta-Learning Algorithms*. 2018. arXiv preprint arXiv: [1803.02999v3](#) (cs.LG).
- [134] NOAA, USA. *Mauna Loa Atmospheric CO₂ Observations*. 2020. URL: ftp://aftp.cmdl.noaa.gov/ccg/co2/trends/co2_mm_gl.csv.
- [135] A. O’Hagan. “Bayes–Hermite quadrature”. In: *Journal of Statistical Planning and Inference* 29.3 (1991), pp. 245–260.
- [136] Junier B. Oliva et al. “Bayesian Nonparametric Kernel-Learning”. In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*. Vol. 51. Proceedings of Machine Learning Research. 2016, pp. 1078–1086.

- [137] Pedro A. Ortega et al. *Meta-learning of Sequential Strategies*. 2019. arXiv preprint arXiv: [1905.03030](https://arxiv.org/abs/1905.03030) (cs.LG).
- [138] Ian Osband, John Aslanides, and Albin Cassirer. “Randomized Prior Functions for Deep Reinforcement Learning”. In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., 2018.
- [139] Ian Osband, Daniel Russo, and Benjamin Van Roy. “(More) Efficient Reinforcement Learning via Posterior Sampling”. In: *Advances in Neural Information Processing Systems*. Vol. 26. Curran Associates, Inc., 2013.
- [140] Michael A. Osborne et al. “Active Learning of Model Evidence Using Bayesian Quadrature”. In: *Advances in Neural Information Processing Systems*. Vol. 25. Curran Associates, Inc., 2012.
- [141] Michael A. Osborne et al. “Bayesian Quadrature for Ratios”. In: *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics*. Vol. 22. Proceedings of Machine Learning Research. 2012, pp. 832–840.
- [142] Jack Parker-Holder, Vu Nguyen, and Stephen J. Roberts. “Provably Efficient Online Hyperparameter Optimization with Population-Based Bandits”. In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 17200–17211.
- [143] Adam Paszke et al. “Automatic Differentiation in Pytorch”. In: *Autodiff Workshop at the 31st Conference on Neural Information Processing Systems*. 2017.
- [144] Deepak Pathak et al. “Curiosity-driven Exploration by Self-supervised Prediction”. In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. Proceedings of Machine Learning Research. 2017, pp. 2778–2787.
- [145] Christian F. Perez, Felipe P. Such, and Theofanis Karaletsos. “Efficient Transfer Learning and Online Adaptation with Latent Variable Models for Continuous Control”. In: *Continual Learning Workshop at the 31st Conference on Neural Information Processing Systems*. 2018.
- [146] Jan Peters and Stefan Schaal. “Natural Actor-critic”. In: *Neurocomputing* 71.7-9 (2008), pp. 1180–1190.
- [147] Gabriel Peyré and Marco Cuturi. “Computational Optimal Transport”. In: *Foundations and Trends in Machine Learning* 11.5 (2019), pp. 1–257.
- [148] Pascal Poupart et al. “An Analytic Solution to Discrete Bayesian Reinforcement Learning”. In: *Proceedings of the 23rd International Conference on Machine Learning*. Omni Press, 2006.
- [149] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons Ltd., 2014.
- [150] Ross Quinlan. *Auto MPG Data Set*. 1993. URL: <https://archive.ics.uci.edu/ml/datasets/auto+mpg>.
- [151] Ali Rahimi and Benjamin Recht. “Random Features for Large-Scale Kernel Machines”. In: *Advances in Neural Information Processing Systems*. Vol. 20. Curran Associates, Inc., 2008.

- [152] Kate Rakelly et al. “Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables”. In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. Proceedings of Machine Learning Research. 2019, pp. 5331–5340.
- [153] Carl E. Rasmussen and Christopher Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.
- [154] Roger Ratcliff. “Connectionist Models of Recognition Memory: Constraints Imposed by Learning and Forgetting Functions”. In: *Psychological review* 97.2 (1990), pp. 285–308.
- [155] Sami Remes, Markus Heinonen, and Samuel Kaski. “Non-Stationary Spectral Kernels”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017.
- [156] John D. Co-Reyes et al. “Self-Consistent Trajectory Autoencoder: Hierarchical Reinforcement Learning with Trajectory Embeddings”. In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. Proceedings of Machine Learning Research. 2018, pp. 1008–1017.
- [157] Christopher D. Rosin. “Multi-armed Bandits with Episode Context”. In: *Annals of Mathematics and Artificial Intelligence* 61.3 (2011), pp. 203–230.
- [158] Jonas Rothfuss et al. “ProMP: Proximal Meta-Policy Search”. In: *Proceedings of the 7th International Conference on Learning Representations*. OpenReview.net, 2019.
- [159] Binxin Ru et al. “Fast Information-theoretic Bayesian Optimisation”. In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. Proceedings of Machine Learning Research. 2018, pp. 4384–4392.
- [160] Gavin A. Rummery and Mahesan Niranjana. *Online Q-learning using Connectionist Systems*. Vol. 37. University of Cambridge, Department of Engineering, 1994.
- [161] Dorsa Sadigh et al. “Active Preference-based Learning of Reward Functions”. In: *Robotics: Science and Systems XIII*. 2017.
- [162] Steindór Sæmundsson, Katja Hofmann, and Marc Peter Deisenroth. “Meta Reinforcement Learning with Latent Variable Gaussian Processes”. In: *Proceedings of the 34th Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2018, pp. 642–652.
- [163] William Saunders et al. “Trial without Error: Towards Safe Reinforcement Learning via Human Intervention”. In: (2018), pp. 2067–2069.
- [164] John Schulman et al. *Proximal Policy Optimization Algorithms*. 2017. arXiv preprint arXiv: [1707.06347](https://arxiv.org/abs/1707.06347) (cs.LG).
- [165] John Schulman et al. “Trust Region Policy Optimization”. In: *Proceedings of the 32nd International Conference on Machine Learning*. Vol. 37. Proceedings of Machine Learning Research. 2015, pp. 1889–1897.
- [166] Steven L. Scott. “Multi-armed Bandit Experiments in the Online Service Economy”. In: *Applied Stochastic Models in Business and Industry* 31.1 (2015), pp. 37–45.

- [167] Rajat Sen, Kirthevasan Kandasamy, and Sanjay Shakkottai. “Multi-Fidelity Black-Box Optimization with Hierarchical Partitions”. In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. Proceedings of Machine Learning Research. 2018, pp. 4538–4547.
- [168] Burr Settles. “Active Learning”. In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6 (2012), pp. 1–114.
- [169] Bobak Shahriari et al. “Taking the Human out of the Loop: A Review of Bayesian Optimization”. In: *Proceedings of the IEEE* 104.1 (2016), pp. 148–175.
- [170] David Silver et al. “Deterministic Policy Gradient Algorithms”. In: *Proceedings of the 31st International Conference on Machine Learning*. Vol. 32. Proceedings of Machine Learning Research. 2014, pp. 387–395.
- [171] David Silver et al. “Mastering the Game of Go with Deep Neural Networks and Tree Search”. In: *Nature* 529.7587 (2016), pp. 484–489.
- [172] David Silver et al. “Mastering the Game of Go without Human Knowledge”. In: *Nature* 550.7676 (2017), pp. 354–359.
- [173] Fergus Simpson, Vidhi Lalchand, and Carl E. Rasmussen. “Marginalised Spectral Mixture Kernels with Nested Sampling”. In: *Advanced in Neural Information Processing Systems*. Vol. 34. 2021.
- [174] Leslie N. Smith. *A Disciplined Approach to Neural Network Hyper-parameters: Part 1 – Learning Rate, Batch Size, Momentum, and Weight Decay*. 2018. arXiv preprint arXiv: [1803.09820](#) (cs.LG).
- [175] Edward Snelson, Zoubin Ghahramani, and Carl E. Rasmussen. “Warped Gaussian Processes”. In: *Advances in Neural Information Processing Systems 17*. 2004, pp. 1–8.
- [176] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. “Practical Bayesian Optimization of Machine Learning Algorithms”. In: *Advances in Neural Information Processing Systems*. Vol. 25. Curran Associates, Inc., 2012.
- [177] Jialin Song, Yuxin Chen, and Yisong Yue. “A General Framework for Multi-fidelity Bayesian Optimization with Gaussian Processes”. In: *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*. Vol. 89. Proceedings of Machine Learning Research. 2019, pp. 3158–3167.
- [178] Jonathan Sorg, Satinder Singh, and Richard L. Lewis. “Variance-based Rewards for Approximate Bayesian Reinforcement Learning”. In: *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2010.
- [179] Niranjana Srinivas et al. “Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design”. In: *Proceedings of the 27th International conference on Machine learning*. OmniPress, 2010, pp. 1015–1022.
- [180] Bradley C. Stadie et al. *Some Considerations on Learning to Explore via Meta-Reinforcement Learning*. 2018. arXiv preprint arXiv: [1803.01118](#) (cs.AI).
- [181] Malcolm Strens. “A Bayesian Framework for Reinforcement Learning”. In: *Proceedings of the 17th International Conference on Machine learning*. 2000, pp. 943–950.

- [182] Emma Strubell, Ananya Ganesh, and Andrew McCallum. “Energy and Policy Considerations for Deep Learning in NLP”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2019.
- [183] Kaushik Subramanian, Charles L. Isbell, and Andrea L. Thomaz. “Exploration from Demonstration for Interactive Reinforcement Learning”. In: *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems. 2016, pp. 447–456.
- [184] Flood Sung et al. *Learning to Learn: Meta-Critic Networks for Sample Efficient Learning*. 2017. arXiv preprint arXiv: [1706.09529](#) (cs.LG).
- [185] Richard S. Sutton. “First Results with Dyna, an Integrated Architecture for Learning, Planning and Reacting”. In: *Neural Networks for Control* (1990), pp. 179–189.
- [186] Richard S. Sutton. “Learning to Predict by the Methods of Temporal Differences”. In: *Machine learning* 3.1 (1988), pp. 9–44.
- [187] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Vol. 1. 1. MIT Press, Cambridge MA, 1998.
- [188] Richard S. Sutton et al. “Policy Gradient Methods for Reinforcement Learning with Function Approximation”. In: *Advances in Neural Information Processing Systems*. Vol. 12. MIT Press, 2000, pp. 1057–1063.
- [189] Kevin Swersky, Jasper Snoek, and Ryan P. Adams. *Freeze-thaw Bayesian Optimization*. 2014. arXiv preprint arXiv: [1406.3896](#) (stat.ML).
- [190] Kevin Swersky, Jasper Snoek, and Ryan P. Adams. “Multi-Task Bayesian Optimization”. In: *Advances in Neural Information Processing Systems*. Vol. 26. Curran Associates, Inc., 2013.
- [191] Tong Teng et al. “Scalable Variational Bayesian Kernel Selection for Sparse Gaussian Process Regression”. In: *Proceedings of the 34th AAAI Conference on Artificial Intelligence*. Vol. 34. AAAI Press, 2020, pp. 5997–6004.
- [192] William R Thompson. “On the Likelihood that one Unknown Probability exceeds another in View of the Evidence of two Samples”. In: *Biometrika* 25.3/4 (1933), pp. 285–294.
- [193] Michalis Titsias. “Variational Learning of Inducing Variables in Sparse Gaussian Processes”. In: *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*. Vol. 5. 2009, pp. 567–574.
- [194] Felipe Tobar. “Bayesian Nonparametric Spectral Estimation”. In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., 2018.
- [195] Emanuel Todorov, Tom Erez, and Yuval Tassa. “Mujoco: A Physics Engine for Model-based Control”. In: *International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 5026–5033.
- [196] Sebastian Tschiatschek et al. *Variational Inference for Data-Efficient Model Learning in POMDPs*. 2018. arXiv preprint arXiv: [1805.09281](#) (stat.ML).

- [197] Hado Van Hasselt, Arthur Guez, and David Silver. “Deep Reinforcement Learning with Double Q-learning”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 30. 2016.
- [198] Hado Van Hasselt et al. *Deep Reinforcement Learning and the Deadly Triad*. 2018. arXiv preprint arXiv: [1812.02648](#) (cs.AI).
- [199] Jane X. Wang et al. *Learning to Reinforcement Learn*. 2016. arXiv preprint arXiv: [1611.05763](#) (cs.LG).
- [200] Tingwu Wang et al. *Benchmarking Model-Based Reinforcement Learning*. 2019. arXiv preprint arXiv: [1907.02057](#) (cs.LG).
- [201] Zi Wang and Stefanie Jegelka. “Max-value Entropy Search for Efficient Bayesian Optimization”. In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. Proceedings of Machine Learning Research. 2017, pp. 3627–3635.
- [202] Ziyu Wang and Nando de Freitas. *Theoretical Analysis of Bayesian Optimisation with Unknown Gaussian Process Hyper-Parameters*. 2014. arXiv preprint arXiv: [1406.7758](#) (stat.ML).
- [203] Ziyu Wang et al. “Dueling Network Architectures for Deep Reinforcement Learning”. In: *Proceedings of The 33rd International Conference on Machine Learning*. Vol. 48. Proceedings of Machine Learning Research. 2016, pp. 1995–2003.
- [204] Ziyu Wang et al. “Robust Imitation of Diverse Behaviors”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017.
- [205] Garrett Warnell et al. “Deep TAMER: Interactive Agent Shaping in High-Dimensional State Spaces”. In: (2018), pp. 1545–1554.
- [206] Christopher J. C. H. Watkins and Peter Dayan. “Q-learning”. In: *Machine learning* 8.3-4 (1992), pp. 279–292.
- [207] Daniel Willemsen, Hendrik Baier, and Michael Kaisers. “Value Targets in Off-policy AlphaZero: a new greedy Backup”. In: *Neural Computing and Applications* (2021).
- [208] Ronald J. Williams. “Simple Statistical Gradient-following Algorithms for Connectionist Reinforcement Learning”. In: *Machine learning* 8.3-4 (1992), pp. 229–256.
- [209] Andrew Wilson and Ryan Adams. “Gaussian Process Kernels for Pattern Discovery and Extrapolation”. In: *Proceedings of the 30th International Conference on Machine Learning*. Vol. 28. Proceedings of Machine Learning Research 3. 2013, pp. 1067–1075.
- [210] David Wingate et al. “Bayesian Policy Search with Policy Priors”. In: *Proceedings of the 22nd International Joint Conference. IJCAI/AAAI*, 2011, pp. 1565–1570.
- [211] Christian Wirth et al. “A Survey of Preference-Based Reinforcement Learning Methods”. In: *Journal of Machine Learning Research* 18.136 (2017), pp. 1–46.
- [212] Jian Wu and Peter Frazier. “The Parallel Knowledge Gradient Method for Batch Bayesian Optimization”. In: *Advances in Neural Information Processing Systems*. Vol. 29. Curran Associates, Inc., 2016.

- [213] Jian Wu et al. “Practical Multi-fidelity Bayesian Optimization for Hyperparameter Tuning”. In: *Proceedings of the 35th Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2019.
- [214] Mike Wu et al. “Meta-Amortized Variational Inference and Learning”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 4. AAAI Press, 2020, pp. 6404–6412.
- [215] Xiaoyue Xi, François-Xavier Briol, and Mark Girolami. “Bayesian Quadrature for Multiple Related Integrals”. In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. Proceedings of Machine Learning Research. 2018, pp. 8533–8564.
- [216] Jiayu Yao et al. “Direct Policy Transfer via Hidden Parameter Markov Decision Processes”. In: *LLARLA Workshop at the 28th International Conference on Flexible Automation and Intelligent Manufacturing*. 2018.
- [217] Gregory Yauney and Pratik Shah. “Reinforcement Learning with Action-Derived Rewards for Chemotherapy and Clinical Trial Dosing Regimen Selection”. In: *Proceedings of the 3rd Machine Learning for Healthcare Conference*. Vol. 85. Proceedings of Machine Learning Research. 2018, pp. 161–226.
- [218] I-Cheng Yeh. *Concrete Compressive Strength Data Set*. 2007. URL: <https://archive.ics.uci.edu/ml/datasets/concrete+compressive+strength>.
- [219] Tianhe Yu et al. “Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning”. In: *Proceedings of the Conference on Robot Learning*. Vol. 100. Proceedings of Machine Learning Research. 2020, pp. 1094–1100.
- [220] Manzil Zaheer et al. “Deep Sets”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017.
- [221] Amy Zhang, Harsh Satija, and Joelle Pineau. “Decoupling Dynamics and Reward for Transfer Learning”. In: *Workshop paper at the 6th International Conference on Learning Representations*. 2018.
- [222] Luisa Zintgraf et al. *Exploration in Approximate Hyper-State Space for Meta Reinforcement Learning*. 2020. arXiv preprint arXiv: [2010.01062](https://arxiv.org/abs/2010.01062) (cs.LG).
- [223] Luisa Zintgraf et al. “Fast Context Adaptation via Meta-Learning”. In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. Proceedings of Machine Learning Research. 2019, pp. 7693–7702.