Adaptive Interest for Emphatic Reinforcement Learning

Martin Klissarov [*]
Mila, McGill University

Rasool Fakoor
Amazon Web Services

Jonas Mueller
Cleanlab

Kavosh Asadi
Amazon Web Services

Taesup Kim
Seoul National University

Alex Smola
Amazon Web Services

Abstract

Emphatic algorithms have shown great promise in stabilizing and improving reinforcement learning by selectively emphasizing the update rule. Although the emphasis fundamentally depends on an interest function which defines the intrinsic importance of each state, most approaches simply adopt a uniform interest over all states (except where a hand-designed interest is possible based on domain knowledge). In this paper, we investigate adaptive methods that allow the interest function to dynamically vary over states and iterations. In particular, we leverage meta-gradients to automatically discover online an interest function that would accelerate the agent's learning process. Empirical evaluations on a wide range of environments show that adapting the interest is key to provide significant gains. Qualitative analysis indicates that the learned interest function emphasizes states of particular importance, such as bottlenecks, which can be especially useful in a transfer learning setting.

1 Introduction

A fundamental challenge in reinforcement learning (RL) is to approximate key quantities such as value functions and optimal policies. Under the assumption that the world in which an RL agent interacts is large and the computational capacity is limited, a natural trade-off emerges in which certain quantities are more accurately predicted than others over the course of learning. Standard RL algorithms, such as temporal differences (TD) [41], perform updates at every state, thereby spending more resources on such frequent states at the expense of other potentially more useful ones. A possible solution could be to selectively emphasize certain updates, for example through a state-dependent *interest function*. However, when combined with standard bootstrapping as in TD(λ), such update rules are known to be unstable [26].

Emphatic algorithms propose a solution in which state-dependent selective updating can be applied while maintaining stability under linear function approximation [45]. At their core, emphatic algorithms determine the emphasis to be applied at each update by accounting for how much the current state is being bootstrapped from as well as an intrinsic measure of its importance relative to other states. This intrinsic measure is encoded through an arbitrary state-dependent *interest* function which the practitioner can set to any desired positive value. However, with this added flexibility comes a problematic question: how should one select the interest function?

Previous works propose hand-crafted solutions for the interest function that showcase the usefulness of selective updating [28, 3, 30]. However, when applying emphatic algorithms to complex environments

[*]Correspondance to martin.klissarov@mail.mcgill.ca

where external domain knowledge may be too hard to encode, other than rare and specific exceptions [60], most practitioners use a uniform interest over states [45, 54, 21, 61, 22, 18]. That is, they simply set the interest to 1 for all states. Building on the intuition that it can be beneficial to learn more from certain states than others, we argue that *different* emphases may be useful at various stages of the learning process. Indeed, as the RL learning process is inherently non-stationary, the relative importance of a particular state in the agent's updates should likely vary over training iterates as well.

In this work, we study how to adaptively learn the interest function in complex environments where hand-crafting an effective interest function is impractical. A good approach should allow for fast and flexible adaptation based on the agent's interactions with its environment. Considering the previous success of meta-gradient framework in discovering hyperparameters [56, 59], objective functions [55], intrinsic rewards [64], and temporal abstractions [52], we here propose to learn and adapt the interest function based on meta-gradients in an online fashion. The interest function in our method is parameterized by *meta-parameters*, which are updated by gradient descent along with the parameters of the policy and value function.

We empirically investigate the merits of adapting the interest function on a wide variety of environments and settings, ranging from prediction with linear function approximation to control on vision-based tasks. Our contributions are the following. (1) In the off-policy setting, we see substantial gains in performance and sample efficiency when adapting the interest function. (2) We extend the traditional application of emphatic algorithms from the off-policy setting to on-policy control, where we find it is crucial to adapt the interest function in order to observe consistent gains. (3) Qualitatively, our learned interest function appears to naturally discover states of importance, such as bottlenecks [40]. Such discovery is demonstrated to be very useful in transfer learning experiments. Our results highlight the general applicability of emphatic algorithms beyond the off-policy single-task setting considered in most previous studies of emphatic RL.

2 Background

We assume a Markov Decision Process $\mathcal{M}$, defined as a tuple $\langle \mathcal{S}, \mathcal{A}, r, P\rangle$ with a finite state space $\mathcal{S}$, a finite action space $\mathcal{A}$, a transition probability distribution $P : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \to [0,1]$, and a scalar reward function $r(s,a)$ depending on action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$. The policy $\pi : \mathcal{A}\times\mathcal{S} \to [0,1]$ specifies the agent's behaviour and its expected discounted return starting from any state is represented as the value function: $V^\pi(s) = \mathbb{E}_\pi\left[\sum_{i=t}^{\infty}\gamma^{i-t}R_{i+1}\middle|S_t = s\right]$, where $\gamma \in [0,1)$ is the discount factor and R_{t+1} is the sampled reward after performing action A_t in state S_t. Under linear function approximation, the value function is defined with parameters $\theta \in \mathbb{R}^n$ and features $\phi(s) \in \mathbb{R}^n$, that is $\hat{V}^\pi(s;\theta) = \theta^\top\phi(s)$[2]. An efficient family of algorithms for learning such functions builds on the Temporal Difference (TD) algorithm [41] where the value parameters, θ, are updated as follows:

$$\theta_{t+1} = \theta_t + \alpha(R_{t+1} + \gamma\theta_t^\top\phi_{t+1} - \theta_t^\top\phi_t)\phi_t \tag{1}$$

with α denoting the step size. In the control setting, the policy gradient theorem [46] for the episodic case provides the gradient of the expected discounted return from an initial state distribution $d(s_0)$ with respect to a stochastic policy $\pi(\cdot \mid s;\nu)$ now parameterized by ν:

$$\frac{\partial J_\pi(\nu)}{\partial\nu} = \sum_s d_\pi^\gamma(s)\sum_a \frac{\partial\pi\left(a|s;\nu\right)}{\partial\nu}Q^\pi(s,a) \tag{2}$$

where $d_\pi^\gamma(s) = \sum\limits_{s_0} d(s_0)\sum_{t=0}^{\infty}\gamma^t P^\pi(S_t = s|S_0 = s_0)$ is the discounted state occupancy measure of the target policy π and $Q^\pi(s,a) = \mathbb{E}_\pi\left[\sum_{i=t}^{\infty}\gamma^{i-t}R_{i+1}\middle|S_t = s, A_t = a\right]$ is the state-action value function. For a more detailed presentation of the notation please refer to App. C.

2.1 Emphatic Algorithms

Emphatic algorithms [45, 29] provide a way to emphasize and de-emphasize the updates made at each iteration while preserving convergence. Their development was motivated by the challenges that arise under off-policy learning when using function approximation and bootstrapping [51, 43]. In the off-policy setting, a behavior policy $b(a|s)$ generates the data to learn value functions or

[2]To simplify notation, we drop s in ϕ (i.e. ϕ_t instead of $\phi(S_t)$ where S_t is the sampled state at time t).

policies evaluated under the target policy $\pi(a|s)$. Emphatic algorithms generalize TD in various ways, however of particular interest to our work is the added flexibility of arbitrarily defining the intrinsic importance of each state through the interest function. In the following we present emphatic algorithms in the general off-policy setting, as learning on-policy is a special case.

Policy Evaluation: In its simplest one-step bootstrapping form, the Emphatic Temporal Difference (ETD) update rule for the value parameters θ takes the following form

$$\theta_{t+1} = \theta_t + \alpha \rho_t F_t (R_{t+1} + \gamma \theta_t^\top \phi_{t+1} - \theta_t^\top \phi_t)\phi_t \tag{3}$$

where $\rho_t = \frac{\pi(A_t|S_t)}{b(A_t|S_t)}$ is the importance sampling ratio at time t and F_t, the followon trace, is defined as,

$$F_t = i(S_t) + \gamma \rho_{t-1} i(S_{t-1}) + \gamma^2 \rho_{t-1}\rho_{t-2} i(S_{t-2}) + ... = i(S_t) + \gamma \rho_{t-1} F_{t-1}$$

$$F_t = i(S_t) + \gamma i(S_{t-1}) + \gamma^2 i(S_{t-2}) + ... = i(S_t) + \gamma F_{t-1}$$

where $i(\cdot) : \mathcal{S} \to \mathbb{R}^+$ is the arbitrary user-defined interest function. The specific form of this trace depends in part on the interest function, but also on how much a state is bootstrapped from by previous states, discounted over time. This specific form is what confers stability and convergence to ETD [58], without introducing the full product of importance ratios used for prior correction [32].

Control: In the actor-critic setting, [11] proposed to maximize the excursions objective $J_b(\nu) = \sum_s d_b(s) V^\pi(s)$ where $d_b(s)$ is the stationary distribution of the policy b. We explain in App. C the reason why the stationary distribution appears instead of the discounted state occupancy measure of (2). They proposed a way to approximate the policy gradient, where such approximation is only valid in the tabular case. [21] later derived the correct gradient for the more general objective that now includes the state dependent interest function,

$$J_b(\nu) = \sum_s d_b(s) i(s) V^\pi(s) \tag{4}$$

where the correct stochastic gradient update for the policy parameters ν takes the following form,

$$\nu_{t+1} = \nu_t + \alpha F_t \rho_t \nabla \log \pi(A_t|S_t;\nu_t) Q^\pi(S_t, A_t) \tag{5}$$

Interestingly, the same trace F_t from the off-policy policy evaluation setting appears in the off-policy control setting. Note that ρ_t would be equal to 1 in the on-policy setting where target policy and behavior policy are the same.

3 Adaptive Interest

The interest function was designed as a way to emphasize some states more than others and as such can be an efficient way to encode useful inductive bias. Although it may be possible to find an interest function that is effective for a specific and simple case (e.g. when additional knowledge about the task is readily available), it is not convenient to hand-design interest functions that effectively work for complex domains. For this reason, most previous works on emphatic algorithms consider a simple uniform interest over states [45, 22].

Furthermore, we hypothesize that the usefulness of a particular interest function can vary through the learning process itself. This is obvious in the case of a changing environment, for example in continual learning, but is also relevant in the single task setting where the agent's policy or bootstrapping targets vary in a non-stationary manner. In the next section, we further motivate the advantage of an adaptive interest function through the example of a simple chain MDP.

3.1 Motivating Example

In Fig. 1, we consider the case of off-policy control in a simple chain MDP made of four non-terminal states. Here our agent uses one-step SARSA [34] to learn the action-value function (Q-value). The agent starts in state S_0 and can reach either the terminal state on the right with reward of 1 or the more distant terminal state on the left with reward of 100. In our example, suppose the behavior policy is biased towards going right in the three rightmost states. The resulting target policy learned via SARSA without any emphatic weighting is misguided toward a suboptimal solution (Fig. 1). App. B describes additional details of this experiment.

When designing a fixed interest function for SARSA with emphatic weighting, it would be advantageous to emphasize the states on the left and de-emphasize the states on the right, as a way to try and avoid the sub-optimal solution. However, Fig. 1 shows that although a well-designed fixed interest function can improve upon the this baseline, the resulting emphatic SARSA is still unable to converge to a good policy within 500 updates.

Finally we consider using an adaptive interest function. Here we leverage the same pattern as in the previous fixed interest function, but we only activate the interest in some states at certain times. Particularly, at the start of training, only the left-most state is emphasized, and all other states have interest set to near-zero. As credit is propagated from the terminal left state towards the rest of the chain, the interest of the second left-most state is increased. This continues until credit assignment reaches the starting state and the optimal action is selected. Emphatic SARSA with such a dynamic interest function is able to quickly converge to a good policy (Fig. 1).

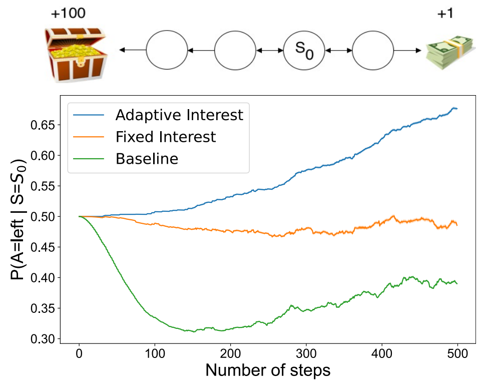


Figure 1: **Four State Chain MDP with off-policy control** where the behavior policy is biased towards going right. We plot the probability that the learned target policy (implemented as a Boltzmann policy) goes left in the initial state S_0, which is the optimal action to take. A **baseline** SARSA agent not using emphatic weighting will struggle to overcome this bias, whereas an emphatic agent with **fixed** interest ($[10, 1, .1, .001]$ for each state from left to right) will require many samples to obtain the optimal policy. Using an **adaptive** interest inside emphatic updates guides credit assignment towards the starting state efficiently, converging towards the optimal policy early on. Results are averaged over 500 runs.

Fig. 1 empirically shows that, in our example MDP, a standard SARSA agent (without emphatic weighting) is outperformed by emphatic SARSA with a fixed interest (supporting the general utility of emphatic algorithms), which is in turn outperformed by emphatic SARSA with an adaptive interest (supporting the additional utility of adaptive interest). This example highlights that even in tabular settings without function approximation, the additional flexibility of an adaptive interest function can be quite beneficial. It is important to note that the fixed interest agent eventually finds the right solution in our example, but it has much worse sample complexity compared to our adaptive interest agent.

3.2 Meta-Gradient Interest (MINT)

As discussed in the previous section, we seek an adaptive interest function that would improve the learning process during training. However, automatically discovering such a function is not straightforward, in part as it can take any arbitrary value, and because we have to evaluate the effect of a particular change in the interest function with regards to the agent's parameters. We explore adapting the interest function with a wide variety of heuristics such as the prediction error and find that such approaches do not generally provide improvements. Alternatively, meta-gradients [56, 65] are a natural candidate as they can automatically discover such interest functions at each stage of learning through the interaction of an inner loop and an outer loop of optimization. We therefore propose to learn and adapt the interest function parameterized by meta-parameters η in an online manner, within a single lifetime and within a single environment.

During the inner loop, the meta-parameters η, together with the agent's policy and value parameters $\{\nu, \theta\}$, appear in the base objective, $J^B(\theta, \nu, \eta)$. Only the parameters are updated through this objective while the meta-parameters η remain fixed and influence the gradients. To illustrate the influence of the meta-parameters on the resulting parameters, we can write them as functions of η, i.e. $\{\nu'(\eta), \theta'(\eta)\}$.

During the outer loop, the updated parameters are evaluated with respect to a meta-objective, $J^M(\nu'(\eta), \theta'(\eta))$, from which we derive the gradients with respect to η. This is referred to as the *meta-gradient*, which evaluates how the values of the meta-parameters affected the performance of the updated parameters. By repeating this process, meta-gradients will adapt the meta-parameters in order to more efficiently improve the parameters themselves.

We now describe the specific choices behind applying meta-gradients to emphatic algorithms when updating the policy. For simplicity, the derivation for the value function is relegated to App. H.3.

In the inner loop, the agent maximizes the following inner objective,

$$J^B(\nu, \eta) = \sum_s d_b(s) i(s; \eta) V^\pi(s) \tag{6}$$

where the interest function i is parameterized by the meta-parameters η and the policy π is parameterized by ν. This inner objective is based on the excursions objective [21], that is, the future reward achieved by following the target policy π starting from the distribution of states generated by the behavior b. Another possibility would have been to consider the counterfactual objective [60] or the alternative life objective [31], however these choices imply additional complexities which we leave for future work. We provide a more detailed discussion on the choice of the objective function in the App. H.1. The inner loop can be written by using (5), obtaining:

$$\nu' \leftarrow \nu + \alpha_b \rho_t F_{t,\eta} \nabla_\nu \pi(A_t|S_t; \nu) Q^\pi(S_t, A_t) \tag{7}$$

where $F_{t,\eta} = i(S_t; \eta) + \gamma \rho_{t-1} F_{t-1,\eta}$ emphasizes the current state according to the current meta interest and the followon trace at the previous timestep.

When considering the meta-objective, practitioners usually employ the same form as inner objective. In our case this would be written as

$$J^M(\nu'(\eta)) = \sum_s d_b(s) V^\pi(s) \tag{8}$$

where π is defined by the updated parameters $\nu'(\eta)$. Recently, [15] argue that such an approach may lead to a poor meta-optimisation landscape, as both objectives share the same curvature. In App. D, we verify different meta-objectives, such as the variance of the reward-to-go [48], and report no increase in performance when compared to (8) in our setting. It is likely that obtaining their improvements also relies tackling myopia in meta-gradients. From (8) we obtain the following meta-gradient

$$\eta' \leftarrow \eta + \alpha_m \nabla_\eta J^M = \eta + \alpha_m \nabla_{\nu'} J^M \nabla_\eta \nu' \tag{9}$$

where $\nabla_\eta \nu'$ encodes how the meta-parameters affected the new parameters. A stochastic sample of this quantity at time t can be expanded as (see also App. H.2),

$$\Big(\sum_{i=0}^{t} \gamma^{t-i} \nabla_\eta i_\eta(S_i) \rho_{i:t} \Big) \nabla_\nu \log \pi(A_t|S_t; \nu) Q^\pi(S_t, A_t)$$

where $\rho_{i:t}$ is a product of importance sampling ratios. Pseudocode for our approach, which we call **MINT** (**M**eta-gradient **Int**erest), is presented in Algorithm 1 of App. A.

Performing the update rule in (9) would require a new set of samples. In practice, the same samples are re-used for both loops [65, 63, 52] through a sliding window of experience. In our work we opt to use the importance sampling ratio method of [65]. Finally in the present derivation we only consider 1-step meta-gradient [53] as it greatly simplifies the exposition (see App. H.4 for derivations). We also show in App. H.4 why the sampling correction term [2] does not appear in the off-policy setting.

4 Experiments

We now validate our method on a wide range of scenarios to assess the following questions: 1) Can we automatically learn an interest function on complex environment in order to improve performance? 2) How robust is the meta interest with respect to the agent's hyperparameters? 3) Is it possible to leverage the information encoded by the learned interest function for downstream tasks?

We first conduct experiments in the off-policy policy evaluation setting under linear function approximation. Next, we extend the usual field of study of emphatic algorithms and verify their general utility. In particular, we study how they can improve the performance of on-policy algorithms under the control setting, where considerable gains are witnessed only under an adaptive interest. Finally, we further extend our investigation to the transfer learning setting by leveraging the interest function learned in a previous task in order to greatly speed up the learning process in a second task. All hyperparameters are available in the App. E.

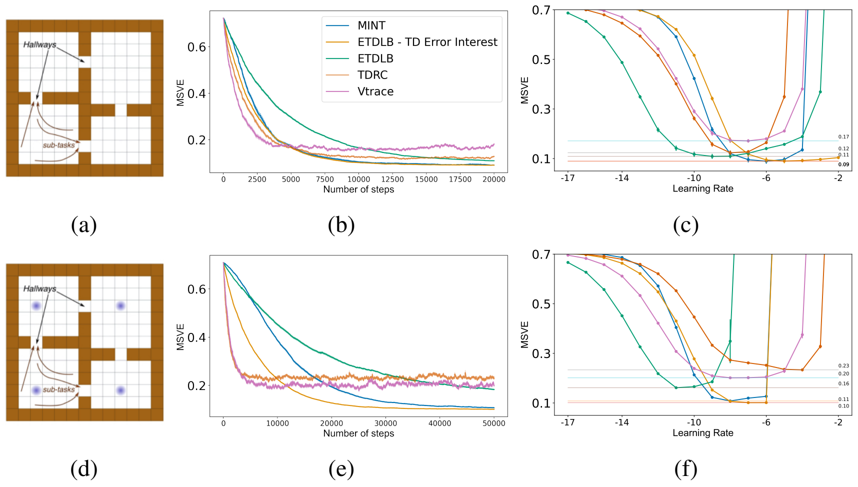


Figure 2: **Off-Policy Evaluation under linear function approximation** where we build on the empirical setup of [17]. The top row presents results for the 4Rooms-8Tasks (4R8T) domain, while the bottom row presents for the HighVariance-4Rooms-8Tasks (H4R8T) domain, where states in blue present high variance. We compare emphatic algorithms to high performing off-policy baselines such as TDRC and Vtrace. For both environments we notice that by adapting the interest function, either through meta gradients as in MINT or through the absolute value of TD error, the final prediction error is significantly improved, especially in the second environment where MINT *reduces by half* the error when compared to non-emphatic methods.

4.1 Linear Function Approximation

Setup. Our first experiments are done in the off-policy policy evaluation setting with linear function approximation using tile coding [43]. We adopt the setup from [17], who considered two variations of the classical Four Rooms domain, shown in Fig. 2. We name these variations 4Rooms-8Tasks (4R8T) and HighVariance-4Rooms-8Tasks (H4R8T) to highlight their characteristics (See App. D for their details).

Quantitative Results. Fig. 2 shows the results for the 4R8T (top) and H4R8T (bottom row). The y-axis indicates the mean squared value error (MSVE) averaged across all policies. In Fig. 2c and 2f, we vary the learning rate on the x-axis and for each value report each algorithm's best final performance chosen across all values of the bootstrapping coefficient λ (and other possible hyperparameters). Fig. 2b and 2e show the best learning curves for each algorithm.

In these two domains, we compare learning the interest through meta gradients, MINT, to a baseline that adapts the interest with respect to the absolute value of the TD error (ETDLB - TD Error Interest, where ETDLB refers to the generalized version of emphatic TD [20]). We compare these adaptive methods to the standard emphatic baseline ETDLB. Finally, we also compare to the recent TD with Regularized Corrections (TDRC) algorithm [16], which follows the line of work on Gradient TD [44], as well as the V-trace [12], which is representative of the performance of methods that use truncated importance sampling ratios [27]. We notice that across both tasks as shown in Fig. 2e, leveraging an adaptive interest leads to a better final value error and especially on H4R8T, our method *almost halves the error* when compared to non-emphatic methods.

In App. D we present additional figures that take into account a different metric: the area under the curve (instead of the final performance) and notice a similar pattern. When looking at the learning curves on the left, we notice that ETDLB pays a price in terms of slower convergence in order to achieve a better final performance. However, when using an adaptive interest, the difference with non-emphatic methods is greatly reduced, especially in the 4R8T domain.

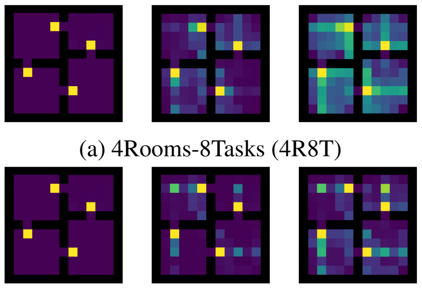
Examining closely Fig. 2c and 2f, we notice that leveraging an adaptive interest moves the bottom of the U-shaped curve to the right. By selectively emphasizing some states, emphatic algorithm using an adaptive interest are able to learn on a higher learning rate. However, the shape of the U curve tends to cut drastically after a certain threshold, at which point the updates become unstable.

When we compare adapting the interest through meta gradients to the one defined as the absolute value of the TD error, we notice that their performance is almost equal. It can perhaps seem surprising that this would be case, as the meta gradients method can in theory learn any function. For our particular choice of objectives, we show in App. H.3 the form of the meta gradients under linear function approximation. For simplicity, if we further assume that features are tabular and we are under the on-policy setting, we get that the stochastic sample at time t of the gradient is,

$$\nabla_\eta J_t^M = \mathbf{e}_t(\delta_t)^2 \tag{10}$$

where $\mathbf{e}_t$ indicates the one-hot vector represent the current state at time t. (10) shows that the meta gradient updates in the direction of the squared TD error. Although the convergence of the meta-parameters will not be to the sampled squared TD error, at each iteration the meta-parameters are affected in a similar way when compared to the absolute value of the TD error heuristic. In practice, the features are not exactly tabular and therefore the updates made on one state may affect another, which would also explain why MINT is slower to converge.

Qualitative Results. In Fig. 3, we inspect the learned interest functions obtained by MINT on four of the eight tasks (one per room). The top row shows the learning process for 4R8T, while the bottom row shows the process for H4R8T. In both domains, the first state to be highlighted is the one next to the goal in the hallway. Indeed this state is highly influential since all states bootstrap from it, directly or indirectly.


(a) 4Rooms-8Tasks (4R8T)

(b) HighVariance-4Rooms-8Tasks (H4R8T)

Figure 3: **Visualization of the interest function across iterations.** These show results at the start (left column), mid-training (middle column) and at the the end (right column). Depending on the environment, different patterns are being encoded in the interest function. In the H4R8T environment, the high variant state is being emphasized as it requires more resources in order to be estimated accurately.

As training progresses, the interest in the 4R8T diffuses to neighbouring states (this bears a close resemblance to the example presented in Fig. 1). In the H4R8T, a different pattern emerges where the state with high variance is being highlighted early on in training. As this state is visited by many trajectories that the target policy would take, it influences the values of many other states that need to bootstrap from it. However, since it exhibits higher variance than neighbouring states, it requires more computation to be correctly estimated.

At the end of training, we observe that some states are particularly less important, like state near the opposite corners of the hallways. As the target policy does not visit them often and not many states bootstrap from them, it is less important for them to be accurate. Moreover, it is interesting to note that the diffusion of interest observed in the 4R8T domain is not perfectly uniform. Since tile coding [43] is used as the function approximator, a specific pattern of the interest function may be needed at different states to lower the overall prediction error.

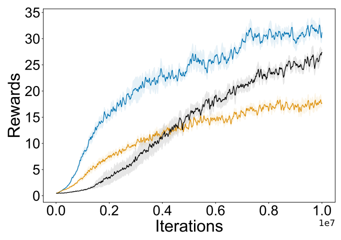
4.2 Experiments at Scale

Emphatic algorithms were initially derived for the off-policy setting. However, the flexibility given by the interest function is generally applicable, even in the on-policy case. To showcase this flexibility, in this section we investigate the performance of emphatic algorithms in the on-policy setting using non-linear function approximation.

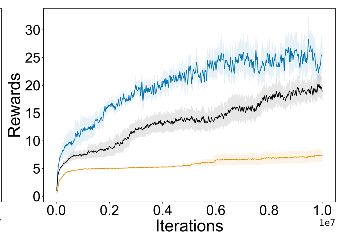
4.2.1 MinAtar

Setup. We verify the generality of the proposed method by considering the MinAtar domain [57], which is a miniaturized version of some of the games from the classic Atari 2600 testbed [7]. The environment provides $10 \times 10 \times n$ state representations, where n varies for each game. The environments are implemented using sticky actions and randomization [25]. For all games we use 10 random seeds and report the mean and standard deviation after 10M timesteps.

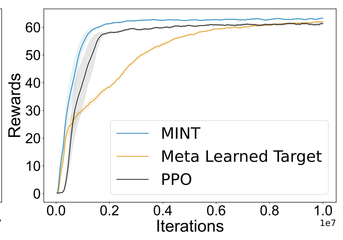
Results. Fig. 4 shows that MINT provides good gains when compared to the two baselines. A standard PPO agent [39] and a meta-gradient approach [55] that meta-learns the target function which appears in RL update rules. We explain in detail this baseline in the App. F and theoretically show that learning the interest function is not simply special case of their approach. Results in Fig 4 clearly demonstrate that our method outperforms both baselines. Our findings also agree with the experiment



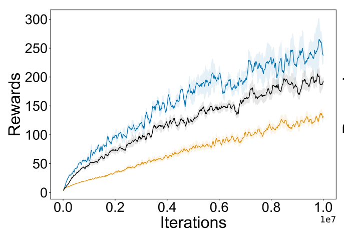
(a) Asterix-v0



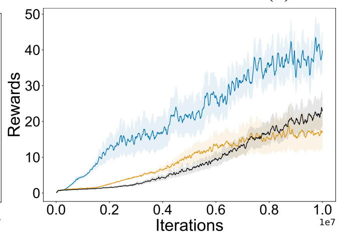
(b) Breakout-v0



(c) Freeway-v0



(d) SpaceInvaders-v0



(e) Seaquest-v0

Figure 4: **Performance on MinAtar.** Adapting the interest provides consistent gains in sample efficiency as well as final performance. Meta-learning the target function is a more general approach than ours, but also requires more samples to provide improvements over the standard baseline.

in [55] (Fig 3a in their paper) that meta-learning the target function requires many more samples before it can match the baseline's performance. [55] also compares to an approach that meta-learns the complete loss function (which could recover our update rule) and find that the agent is not able to learn in the online, single-lifetime setting (which is our setting).

These results highlight the difficulties of more general meta-learning formulations and their impact on sample efficiency. This suggests that meta-learning the interest-function may be a good trade-off between generality and the amount of inductive bias. We additionally present results in Fig. 10 where we vary the learning rate and present at the U shaped curves of performance, which seem to behave similarly to the linear function approximation case.

4.2.2 Continuous Control

Setup. We perform experiments on the MuJoCo domain [50, 8], where states and actions are continuous. We report the mean and standard error averaged across 10 random seeds. We include several emphatic baselines where we explore using a fixed interest and various heuristics for adapting the interest function. Additionally, we investigate the usefulness of adapting the learning rate itself using hypergradient descent [5]. We provide a description of all the baselines in App. E.

Results. As Fig. 5 shows, utilizing adaptive interest function is the key to get consistent improvement over PPO (across almost all environments). Interestingly, the adaptive heuristic based on the TD error that worked well in the prediction setting does not generalize to this one. One way to understand this is to consider that a low TD error may not be indicative of a high performing policy in control.

We also compare our method to hypergradient descent (HD) [6] which dynamically updates the learning rate during training. We notice that HD does not seem to provide gains, except in Humanoid-v3 where it reaches the performance of MINT at the cost of a slower learning process. This is in contrast to our method which generally does not suffer from increased sample complexity to achieve better performance. We highlight that an important difference between HD and MINT is that the interest function is a state-dependent quantity, which can provide additional flexibility. In the App. E we compare to additional baselines, such as meta-learning the reward function [65] and various adaptive heuristics for the interest function.

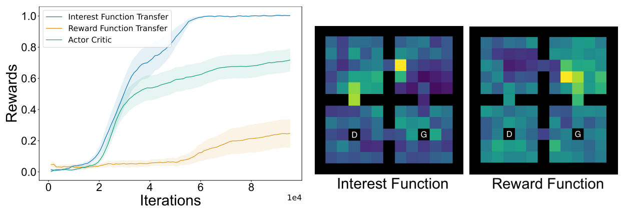
4.3 Transfer Learning across RL Tasks

Setup. As the interest function is automatically learned, we observe that it likely encodes knowledge and information that can be useful later. In particular, this knowledge may serve to speed up learning in new environment over learning from scratch. We investigate this hypothesis in the

FourRoomsTransfer environment [10], shown in Fig. 6a where we show mean and standard deviation across 30 seeds. The agent starts in the top left corner and has to get to the goal location in green. The state highlighted in orange is a distractor state that provides a random reward, $R \sim \mathcal{N}(0, 1)$.

For the transfer setting, we change the location of each of these entities (see App. G). In this setting, the agent learns in the first environment and only transfers the interest function to the second one, which remains fixed thereafter. We also compare to an agent that uses meta gradients to learn an intrinsic reward function before transferring it to the second task, similarly to [63].

Results. Fig. 6 shows that the learned interest function provides a significant speed-up when compared to the actor-critic baseline that learns from scratch without an interest function. Here, transferring the intrinsic reward does not seem to help. Note we *do not* claim that using meta-gradients to learn an intrinsic function is always a better choice than to learn an intrinsic reward function, as it likely depends on the exact transfer learning setup[3].

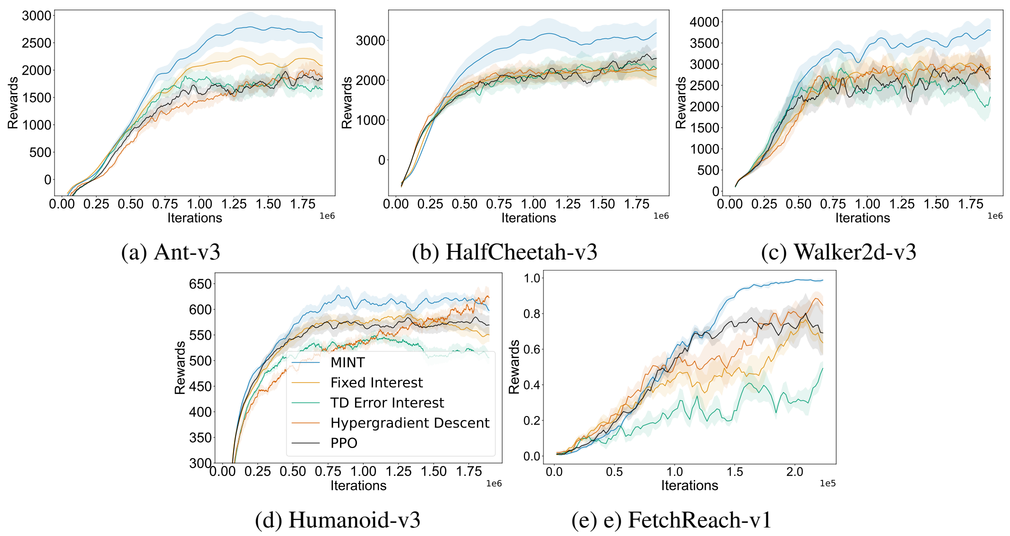


(a) (b) (c)

Figure 6: **Transfer experiment** where the interest or the reward function is transferred to help an agent learn a policy from scratch in a variant of the task. In b) and c) we visualize the values of each function, where brighter color means higher value. Interestingly, the interest function highlights states near the hallways of the starting state room (top-left), also referred to as *bottleneck states*.

To understand how the interest function helps in transfer, we present in Fig. 6b and Fig. 6c the learned interest function and the learned reward function. We notice that the interest function highlights states that are near the goal, but also the hallways of the starting room (top left), which are usually referred to as bottleneck states [40]. Such states are of particular importance as they influence the trajectory an agent takes as well as many of the predictions it makes during such trajectory. As we notice, the interest function highlights the hallway leading to the goal, but also the one leading to the distractor. On the opposite, the reward function naturally highlights the path to the goal and de-emphasizes the one leading to the distractor state. This illustrates one useful property of the interest function: it highlights the location of rewards, whether they are positive or negative. This kind of invariance is key for a

[3]Note, we do not consider using recurrent neural networks or a lifetime value function here, which were found critical for meta-learned intrinsic rewards to work well in transfer [63]. Also, the interest/reward functions are trained on a single environment at first, before being transferred to a new one, without further training.



(a) Ant-v3 (b) HalfCheetah-v3 (c) Walker2d-v3

(d) Humanoid-v3 (e) e) FetchReach-v1

Figure 5: **Results on continuous control.** We compare MINT to various baselines including an emphatic variant of PPO using a fixed interest, as well as an interest based on the absolute value of the TD error. We also verify whether updating the learning rate via hypergradient descent can match the performance of MINT. Across environments, we notice that *adapting the interest via meta-gradients* is key to obtain consistent gains.

better transfer performance in our setting, and could be used more generally in continual learning [33]. This experiment also points to an interesting future direction where a universal interest function could be defined similarly to universal value functions [35].

5 Related Work

Emphatic Methods. Initially derived by [45, 28] as a stable and simple one-time-step solution to the problem of off-policy prediction. Its convergence is shown in [58] when employing the full trace and later [62] show convergence for the Truncated ETD algorithm. The ideas in prediction were extended to control by [21, 19, 61]. Emphatic algorithms have been shown to be a strong baseline in many benchmarks under linear function approximation [17, 18], even in the on-policy case [54, 3]. Recently, emphatic algorithms have been extended to the deep RL by building on a variant of the IMPALA agent [12] with auxiliary heads and have shown superior performance on Atari [22, 23].

Selective Updating. Emphatic methods can be seen more generally as performing selective updating, whereby through a scalar we emphasize or de-emphasize the updates to each state. This has previously been investigated in the context of model free learning [36] and model based planning [1]. Recently, [9] provide a unifying view of various algorithms as a form of selective credit assignment. In hierarchical reinforcement learning, selective updating in an integral part of the options framework either through initiation sets [47] or interest functions [24].

Meta Gradients. Meta learning [37, 49, 14] is a class of methods that have better capability in adapting to new tasks by learning a better prior from previously seen related tasks in the past [38, 4]. While these methods mainly focus on multi-task learning [13], meta-gradients [64, 65] based methods instead focus on learning the meta parameters online within a single task, based on online cross validation [42].

6 Conclusion

We propose to learn and adapt the interest function based on meta-gradients in an online fashion in complex environments where hand-coded solutions are not feasible. Comprehensive experiments on various settings suggest that automatically adapting the interest function from a stream of data leads to improved performance. Although certain heuristics for adapting the interest function are occasionally beneficial, our experiments point that consistency and general usefulness are achieved through meta-gradients.

References

[1] Z. Abbas, S. Sokota, E. J. Talvitie, and M. White. Selective dyna-style planning under limited model capacity. *CoRR*, abs/2007.02418, 2020.

[2] M. Al-Shedivat, T. Bansal, Y. Burda, I. Sutskever, I. Mordatch, and P. Abbeel. Continuous adaptation via meta-learning in nonstationary and competitive environments. *CoRR*, abs/1710.03641, 2017.

[3] N. Anand and D. Precup. Preferential temporal difference learning. *CoRR*, abs/2106.06508, 2021.

[4] J. Baxter. A model of inductive bias learning. *Journal of artificial intelligence research*, 12:149–198, 2000.

[5] A. G. Baydin, R. Cornish, D. Martínez-Rubio, M. Schmidt, and F. D. Wood. Online learning rate adaptation with hypergradient descent. *CoRR*, abs/1703.04782, 2017.

[6] A. G. Baydin, R. Cornish, D. M. Rubio, M. Schmidt, and F. Wood. Online learning rate adaptation with hypergradient descent. In *International Conference on Learning Representations*, 2018.

[7] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *CoRR*, abs/1207.4708, 2012.

[8] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016.

[9] V. Chelu, D. Borsa, D. Precup, and H. van Hasselt. Selective credit assignment, 2022.

[10] M. Chevalier-Boisvert, L. Willems, and S. Pal. Minimalistic gridworld environment for openai gym. `https://github.com/maximecb/gym-minigrid`, 2018.

[11] T. Degris, M. White, and R. S. Sutton. Off-policy actor-critic. *CoRR*, abs/1205.4839, 2012.

[12] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, S. Legg, and K. Kavukcuoglu. IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures. *CoRR*, abs/1802.01561, 2018.

[13] R. Fakoor, P. Chaudhari, S. Soatto, and A. J. Smola. Meta-q-learning. In *International Conference on Learning Representations*, 2020.

[14] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *CoRR*, abs/1703.03400, 2017.

[15] S. Flennerhag, Y. Schroecker, T. Zahavy, H. van Hasselt, D. Silver, and S. Singh. Bootstrapped meta-learning. *CoRR*, abs/2109.04504, 2021.

[16] S. Ghiassian, A. Patterson, S. Garg, D. Gupta, A. White, and M. White. Gradient temporal-difference learning with regularized corrections. *CoRR*, abs/2007.00611, 2020.

[17] S. Ghiassian and R. S. Sutton. An empirical comparison of off-policy prediction learning algorithms in the four rooms environment. *CoRR*, abs/2109.05110, 2021.

[18] S. Ghiassian and R. S. Sutton. An empirical comparison of off-policy prediction learning algorithms on the collision task. *CoRR*, abs/2106.00922, 2021.

[19] E. Graves, E. Imani, R. Kumaraswamy, and M. White. Off-policy actor-critic with emphatic weightings. *CoRR*, abs/2111.08172, 2021.

[20] A. Hallak, A. Tamar, R. Munos, and S. Mannor. Generalized emphatic temporal difference learning: Bias-variance analysis, 2015.

[21] E. Imani, E. Graves, and M. White. An off-policy policy gradient theorem using emphatic weightings. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[22] R. Jiang, T. Zahavy, Z. Xu, A. White, M. Hessel, C. Blundell, and H. Van Hasselt. Emphatic algorithms for deep reinforcement learning. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5023–5033. PMLR, 18–24 Jul 2021.

[23] R. Jiang, S. Zhang, V. Chelu, A. White, and H. van Hasselt. Learning expected emphatic traces for deep rl, 2021.

[24] K. Khetarpal, M. Klissarov, M. Chevalier-Boisvert, P. Bacon, and D. Precup. Options of interest: Temporal abstraction with interest functions. *CoRR*, abs/2001.00271, 2020.

[25] M. C. Machado, M. G. Bellemare, E. Talvitie, J. Veness, M. J. Hausknecht, and M. Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *CoRR*, abs/1709.06009, 2017.

[26] A. Mahmood. Incremental off-policy reinforcement learning algorithms. 2017.

[27] A. R. Mahmood, H. Yu, and R. S. Sutton. Multi-step off-policy learning without importance sampling ratios. *CoRR*, abs/1702.03006, 2017.

[28] A. R. Mahmood, H. Yu, M. White, and R. S. Sutton. Emphatic temporal-difference learning. *CoRR*, abs/1507.01569, 2015.

[29] A. R. Mahmood, H. Yu, M. White, and R. S. Sutton. Emphatic temporal-difference learning, 2015.

[30] M. K. McLeod, C. Lo, M. K. Schlegel, A. Jacobsen, R. Kumaraswamy, M. White, and A. M. White. Continual auxiliary task learning. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[31] A. Patterson, A. White, S. Ghiassian, and M. White. A generalized projected bellman error for off-policy value estimation in reinforcement learning. *CoRR*, abs/2104.13844, 2021.

[32] D. Precup, R. Sutton, and S. Dasgupta. Off-policy temporal-difference learning with function approximation. *Proceedings of the 18th International Conference on Machine Learning*, 06 2001.

[33] D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, and G. Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[34] G. A. Rummery and M. Niranjan. On-line Q-learning using connectionist systems. Technical Report TR 166, Cambridge, England, 1994.

[35] T. Schaul, D. Horgan, K. Gregor, and D. Silver. Universal value function approximators. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1312–1320, Lille, France, 07–09 Jul 2015. PMLR.

[36] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay, 2015. cite arxiv:1511.05952Comment: Published at ICLR 2016.

[37] J. Schmidhuber. A neural network that embeds its own meta-levels. *IEEE International Conference on Neural Networks*, pages 407–412 vol.1, 1993.

[38] J. Schmidhuber, J. Zhao, and M. Wiering. Shifting inductive bias with success-story algorithm, adaptive levin search, and incremental self-improvement. *Machine Learning*, 28(1):105–130, Jul 1997.

[39] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[40] A. Solway, C. Diuk, N. Córdova, D. M. Yee, A. G. Barto, Y. Niv, and M. M. Botvinick. Optimal behavioral hierarchy. *PLoS Computational Biology*, 10, 2014.

[41] R. S. Sutton. Learning to predict by the methods of temporal differences. *Mach. Learn.*, 3(1):9–44, aug 1988.

[42] R. S. Sutton. Adapting bias by gradient descent: An incremental version of delta-bar-delta. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, AAAI'92, page 171–176. AAAI Press, 1992.

[43] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[44] R. S. Sutton, H. Maei, and C. Szepesvári. A convergent o(n) temporal-difference algorithm for off-policy learning with linear function approximation. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2009.

[45] R. S. Sutton, A. R. Mahmood, and M. White. An emphatic approach to the problem of off-policy temporal-difference learning. *J. Mach. Learn. Res.*, 17(1):2603–2631, jan 2016.

[46] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 2000.

[47] R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1):181–211, 1999.

[48] A. Tamar, D. D. Castro, and S. Mannor. Learning the variance of the reward-to-go. *Journal of Machine Learning Research*, 17(13):1–36, 2016.

[49] S. Thrun and L. Pratt, editors. *Learning to Learn*. Kluwer Academic Publishers, USA, 1998.

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

[51] H. van Hasselt, Y. Doron, F. Strub, M. Hessel, N. Sonnerat, and J. Modayil. Deep reinforcement learning and the deadly triad, 2018.

[52] V. Veeriah, T. Zahavy, M. Hessel, Z. Xu, J. Oh, I. Kemaev, H. van Hasselt, D. Silver, and S. Singh. Discovery of options via meta-learned subgoals. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[53] R. Vuorio, J. A. Beck, G. Farquhar, J. N. Foerster, and S. Whiteson. No DICE: An investigation of the bias-variance tradeoff in meta-gradients. In *Deep RL Workshop NeurIPS 2021*, 2021.

[54] A. M. White and M. White. Investigating practical, linear temporal difference learning. *CoRR*, abs/1602.08771, 2016.

[55] Z. Xu, H. P. van Hasselt, M. Hessel, J. Oh, S. Singh, and D. Silver. Meta-gradient reinforcement learning with an objective discovered online. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15254–15264. Curran Associates, Inc., 2020.

[56] Z. Xu, H. P. van Hasselt, and D. Silver. Meta-gradient reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[57] K. Young and T. Tian. Minatar: An atari-inspired testbed for thorough and reproducible reinforcement learning experiments. *arXiv preprint arXiv:1903.03176*, 2019.

[58] H. Yu. On convergence of emphatic temporal-difference learning. *CoRR*, abs/1506.02582, 2015.

[59] T. Zahavy, Z. Xu, V. Veeriah, M. Hessel, J. Oh, H. van Hasselt, D. Silver, and S. Singh. A self-tuning actor-critic algorithm, 2021.

[60] S. Zhang, W. Boehmer, and S. Whiteson. Generalized off-policy actor-critic. *CoRR*, abs/1903.11329, 2019.

[61] S. Zhang, B. Liu, H. Yao, and S. Whiteson. Provably convergent off-policy actor-critic with function approximation. *CoRR*, abs/1911.04384, 2019.

[62] S. Zhang and S. Whiteson. Truncated emphatic temporal difference methods for prediction and control. *CoRR*, abs/2108.05338, 2021.

[63] Z. Zheng, J. Oh, M. Hessel, Z. Xu, M. Kroiss, H. van Hasselt, D. Silver, and S. Singh. What can learned intrinsic rewards capture? *CoRR*, abs/1912.05500, 2019.

[64] Z. Zheng, J. Oh, M. Hessel, Z. Xu, M. Kroiss, H. Van Hasselt, D. Silver, and S. Singh. What can learned intrinsic rewards capture? In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 11436–11446. PMLR, 13–18 Jul 2020.

[65] Z. Zheng, J. Oh, and S. Singh. On learning intrinsic rewards for policy gradient methods. *arXiv preprint arXiv:1804.06459*, 2018.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [No] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] We provide discussion in Section 3.2 and App. H
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material [No] We are currently working to clean up the code and release it. We will make it public before the rebuttal.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] The code also will be released upon publication.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix: Adaptive Interest for Emphatic Reinforcement Learning

A Algorithm

Algorithm 1 MINT

```
1: Initialize policy and value parameters $\{\nu, \theta\}$
2: Initialize meta-parameters for the interest function $\{\eta\}$
3: while not done do
4:     Generate trajectory either using the target policy (on-policy case) or the behavior policy (off-policy case).
5:     Update policy and value function using Equations 7-11
6:     Update interest function using Equation 9.
7: end while
```

B Motivating Example Implementation Details

In the example of Figure 1, the behavior policy chooses to go right with probability 0.9, which misguides the agent towards a sub-optimal policy as it will visit the right terminal state considerably more often. Notice that the left terminal state gives a higher reward. We consider using SARSA [34] to learn the agent's action value function. To obtain a probability distribution over actions, we use a Boltzmann policy in conjunction of the action value function with unitary temperature. As we see in Figure 1 where we plot the probability of going left in the starting state S_0, the baseline SARSA agent not using emphatic is stuck with a sub-optimal policy after 500 steps. We additionally consider using emphatic weightings with a fixed interest function. The specific values of this function are $[10, 1, 0.1, 0.001]$ for each of the four states in the chain MDP, going left to right. Notice that the values are increasing as we go left, as an attempt to counter the behavior policy's bias towards going right. This agent performs much better than the baseline, however it may require many more samples to finally converge to the optimal solution. The agent using an adaptive interest function is on the other hand much more sample efficient. This agent also uses the values $[10, 1, 0.1, 0.001]$ for the interest function, however it only activates some values at certain stages of learning. In particular, at first, only the left most value is active, and the remainder is set to near-zero. Once credit assignment has reached the left most state, the first and second values are active, while the third and fourth remain masked. As more credit assignment is propagated, more values of the interest become active. With this simple strategy we notice in Figure 1 that the agent using an adaptive interest is prefers to go left in the starting state at almost anytime during learning.

C Background and Notation

In the main text we present the TD and ETD algorithms for policy evaluation under linear function approximation, as a way to recognize the existing literature on emphatic algorithms [28]. We here present the derivation for policy evaluation under general function approximation. Following standard notation [43], capital letters for states, actions or rewards represent the random variable at time t (i.e. S_t is the random variable at time t) and lowercase letters represent their instantiation (i.e. $S_t = s$ is the random variable S_t taking value s at time t). The TD algorithm under general function approximation updates the value function parameters θ in the following way,

$$\theta_{t+1} = \theta_t + \alpha\big(R_{t+1} + \gamma \hat{V}^\pi(S_{t+1}; \theta_t) - \hat{V}^\pi(S_t; \theta_t)\big)\nabla_{\theta_t} \hat{V}^\pi(S_t; \theta_t)$$

where $\hat{V}^\pi$ is the function approximation to the true value function V^π when following policy π. The true value function is defined as the expected discounted return from a given state when following policy π,

$$V^\pi(s) = \mathbb{E}_\pi\big[\sum_{i=t}^{\infty} \gamma^{i-t} R_{i+1} | S_t = s\big]$$

We can also define the action value function as the expected discounted return from a given state and given action when following policy π,

$$Q^\pi(s,a) = \mathbb{E}_\pi\big[\sum_{i=t}^{\infty}\gamma^{i-t}R_{i+1}|S_t=s, A_t=a\big] = \mathbb{E}[R_{t+1}+\gamma V^\pi(S_{t+1})|S_t=s, A_t=a]$$

In the control setting, we use function approximation to parameterize the policy $\pi(a|s;\nu)$ using parameters ν. In the episodic on-policy setting, we update the policy using gradient ascent on the following objective,

$$J_\pi(\nu) = \sum_s d_0(s)V^\pi(s)$$

where d_0 is an arbitrary starting state distribution. Using the policy gradient theorem [46], we get the following,

$$\frac{\partial J_\pi(\nu)}{\partial \nu} = \sum_s d_\pi^\gamma(s)\sum_a \frac{\partial \pi\,(a|s;\nu)}{\partial \nu}Q^\pi(s,a)$$

where $d_\pi^\gamma(s) = \sum\limits_{s_0} d(s_0)\sum_{t=0}^{\infty}\gamma^t P^\pi(S_t=s|S_0=s_0)$ is the discounted state occupancy measure of the policy π. The quantity $P^\pi(S_t=s|S_0=s_0)$ is defined as

$$P^\pi(S_t=s|S_0=s_0) = \prod_{i=1}^{t-1}\sum_{S_i}\sum_{A_i}P(S_{i+1}|S_i,A_i)\pi(A_i|S_i;\nu)\sum_{A_0}P(S_1|S_0,A_0)\pi(A_0|S_0;\nu)$$

using the environment transition distribution P and the policy π.

Policy evaluation done under general function approximation using ETD algorithm to update the value function parameters takes the following form,

$$\theta_{t+1} = \theta_t + \alpha F_t\rho_t\big(R_{t+1}+\gamma\hat{V}^\pi(S_{t+1};\theta_t) - \hat{V}^\pi(S_t;\theta_t)\big)\nabla_{\theta_t}\hat{V}^\pi(S_t;\theta_t)$$

where F_t, the followon trace, is defined as,

$$F_t = i(S_t) + \gamma\rho_{t-1}i(S_{t-1}) + \gamma^2\rho_{t-1}\rho_{t-2}i(S_{t-2}) + ... = i(S_t) + \gamma\rho_{t-1}F_{t-1}$$

The emphatic algorithm for off-policy control is defined through the excursions objective generalized through the state-dependent interest function,

$$J_b(\nu) = \sum_s d_b(s)i(s)V^\pi(s)$$

where d_b is the stationary distribution of the behavior policy b. The excursions objective is based on the continuing setting and as a result the starting state distribution d_0 is replaced with the behavior stationary distribution d_b. Intuitively, this objective encodes the expected return from executing the target policy starting from the distribution of states. It is possible to reconcile the continuing setting, which is more theoretical in nature, to the episodic setting, which is more practical, by considering generalizations of the discount factor and bootstrapping parameter [80].

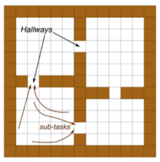
D Linear Function Approximation

We borrow the experimental setup from [17] which considers the off-policy prediction setting. The agent uses linear function approximation to learn a value function for a target near-optimal policy while the data is generated by an near-uniform policy. We also adopt the environmental setup from them which they considered two variations of the classical Four Rooms domain (depicted in Figure 2). We name these variations 4Rooms-8Tasks and HighVariance-4Rooms-8Tasks to highlight their characteristics.

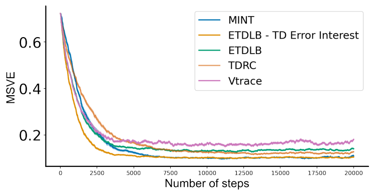
In both domains, the agent starts randomly in any state and follows an infinitely long trajectory where the value functions are updated online. Each of the four rooms consists of two independent tasks: reaching each of the two hallways. As such, a total of eight value functions for eight target policies are being evaluated simultaneously, where each of the policies follows the shortest path to the hallway location.

In 4Rooms-8Tasks, the behavior policy is simply a uniform distribution over actions. In the HighVariance-4Rooms-8Tasks domain, the behavior policy is uniform, except for the states highlighted in blue, as shown in Figure 2d, where it takes one particular action with probability 0.97: in two left rooms this action is to go left, whereas in the two right rooms it is to go right.

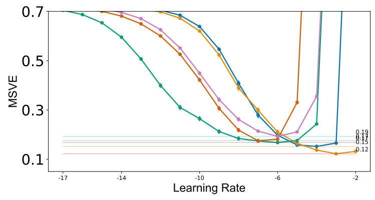
The setup from [17] benchmarks a wide variety of algorithms, from which we choose some of the most performing ones. We also pick these algorithms such that they are representative of different families of algorithms. In Figure 2 we presented the results in terms of final performance. We present in 7 the results in terms of area under curve (AUC). We notice once again that adapting the interest function can provide improvements, especially on HighVariance-4Rooms-8Tasks environment. The performance of MINT compared to the ETDLB baseline using the absolute value of the TD error is slightly lower in terms of AUC. Meta-optimization can bring a set of challenges that need to be addressed in order to improve on the performance we report. In particular we did not use any special tricks such as gradient clipping.



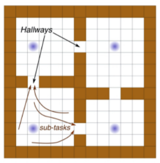
(a) a)



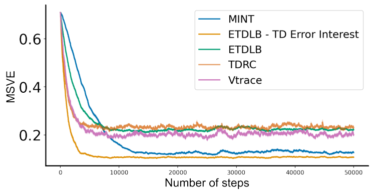
(b) b)



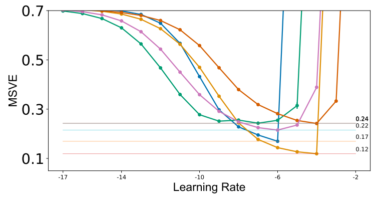
(c) c)



(d) d)



(e) e)



(f) f)

Figure 7: **Off-Policy Evaluation under linear function approximation** where we build on the empirical setup of [17]. The top row presents results for the 4Rooms-8Tasks domain, while the bottom row presents for the HighVariance-4Rooms-8Tasks domain, where states in blue present high variance. We compare emphatic algorithms to high performing off-policy baselines such as TDRC and Vtrace. For both environments we notice that by adapting the interest function, either through meta gradients as in MINT or through the TD error, the area under the curve is significantly improved.

For each algorithm, we plot the best performing curves by looking across learning rate values 2^{-x} where $x \in \{0, 1, 2, .., 18\}$ and bootstrapping coefficient $\{0.0, 0.1, 0.2, 0.3, 0.5, 0.9, 1.0\}$. For the emphatic algorithms we searched the β parameter in $\{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$. We used a meta learning rate of 0.25 for both environments and across all other values of hyperparameters. The interest function is implemented using tabular representation to allow for better stability of the meta-optimization. We do not use any activation function for the interest function as it naturally remains greater than zero.

We also try a couple of different meta objectives, as a way to decouple the curvature of the meta optimisation from the optimisation of the base objective. We present in Fig. 8 results for the reward to go (RTG) [78] and variance temporal difference learning (VTD) [81]. We do not observe an improvement in performance. This highlights that the meta optimisation landscape is complex in nature and researchers would benefits by probing further how to improve the curvature, as suggested by [15].

E Continuous Control

We consider the standard environment from [8] and compare our approach to various baselines building on top of PPO [39]. In Figure 5 we consider an emphatic version of PPO where the interest is fixed (**Fixed Interest**) and where the interest is defined through the absolute value of the TD error

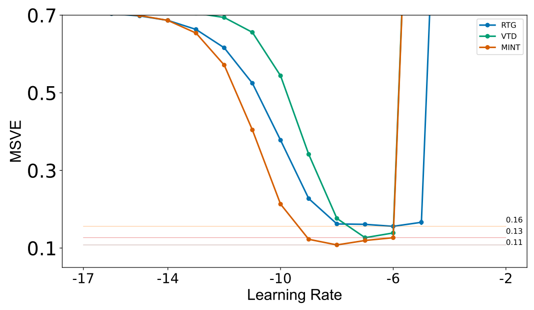
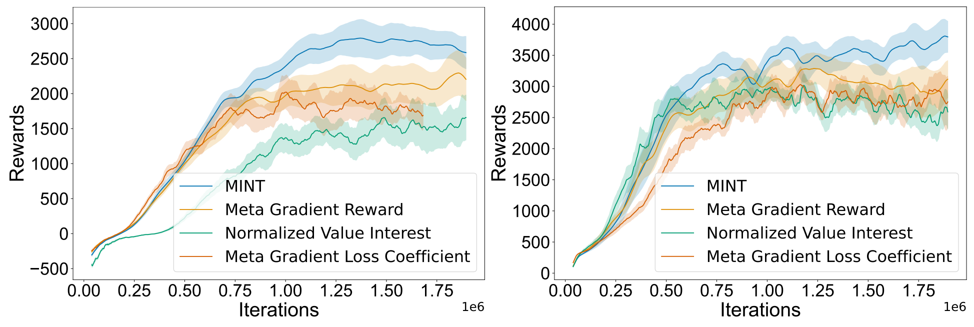


Figure 8: Off-Policy Evaluation under linear function approximation where we build on the empirical setup of [17]. We compare MINT using the excursions objective as the meta-objective to MINT using the reward to go (RTG) and variance temporal difference learning (VTD) as part of the meta-optimisation.

(**TD Error Interest**). Additionally, we compare with a method using hypergradient descent [5] (**Hypergradient Descent**) to learn the learning rate.



(a) a) Ant-v3 (b) b) Walker2d-v3

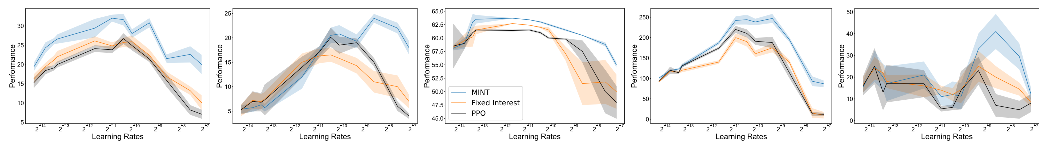
Figure 9: **Additional Baselines on MuJoCo.** We compare MINT to various baselines including an emphatic variant of PPO using the sigmoid of the value function. Additionally we use meta-gradients to learn the loss coefficient as in [82]. Finally, we also try learning an intrinsic reward funtcion through meta gradients, as a way to verify whether the same compute resources used for learning the interest function can be better spent.

In Figure 9 we provide another set of baselines. We verify adapting the interest function by setting it to the sigmoid of the value function (**Normalized Value Interest**), with the intuition that rewarding states may be more valuable. However, this does not help. We also used meta gradients to learn the loss coefficient (**Meta Gradient Loss Coefficient**) as in [82]. This method was shown to provide significant improvements, at the condition of using auxiliary losses, which we do not employ. Finally, we verify whether using the resources for learning an interest function can be better served by learning other quantities. In particular, we try learning an intrinsic reward function (**Meta Gradient Reward**), which helps more than other baselines but still does not match the performance of our method.

Across all environments we use the default hyperparameters from [68] for the base learner. For the meta learning rate we verify values in $\{1 \times 10^{-4}, 3 \times 10^{-4}\}$, and we use the former for all games, except Walker2d-v3 where the latter provided slightly better performance. The neural network for the interest function is the same network used for the policy and value function: a two layer MLP with hidden sizes $\{64, 64\}$. The interest function uses an exponential activation.

F MinAtar

For the experiments on MinAtar we use convolutional neural networks for feature extraction within the interest function, the policy and value function. Across all environments and across all learning rates we use the same default meta learning rate of 1×10^{-4}. The interest function uses an exponential activation.



(a) a) Asterix-v0 (b) b) Breakout-v0 (c) c) Freeway-v0 (d) d) SpaceInvaders-v0 (e) e) Seaquest-v0

Figure 10: **Sensitivity analysis on MinAtar.** Across almost all learning rates, adapting the interest is key to improve upon the baseline. For most of the games, MINT's inverse-U shaped curve tends reaches its peak for *higher learning rates* than the baseline.

G MiniGrid Transfer

For the experiments on MiniGrid-FourRoomsTransfer-v0 [10], we also use convolutional neural networks for feature extraction within the interest function, the policy and value function. In the MiniGrid-FourRoomsTransfer-v0 environment, the agent must reach the goal location within 60 steps. There is also a distractor state that outputs a random reward distributed as a Gaussian with mean zero. We use a learning rate of 3×10^{-4}. The interest function uses an exponential activation. For the transfer learning setting, we consider the following setup,

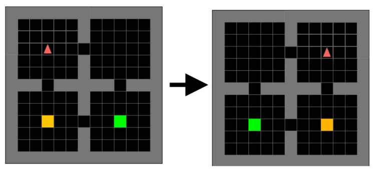


Figure 11: **Transfer learning setting.** We first train the agent on the environment on the left, before vary the location of each entity in the environment. In this new environment we transfer the previously learned interest function or reward function.

H Meta-Gradients

H.1 Choice of objective

The meta-objective is based on the excursions objective, that is,

$$J_b(\nu) = \sum_s d_b(s) i(s) V^\pi(s)$$

Another possibility would have been to consider the alternative life objective,

$$J_\pi(\nu) = \sum_s d_\pi(s) i(s) V^\pi(s)$$

The latter better encodes the performance of the target policy after deployment, as the states are weighted according to the target policy π instead of the behavior policy b. However, optimising this objective will require estimating the density ratio between the stationary distribution of the behavior and target policies, that is $\frac{d_\pi(s)}{d_b(s)}$. Although methods exist [70], they usually require more complex optimisation methods and have not yet shown great performance on complex domains. Another possibility for the policy evaluation case is to consider at each timestep t the full ratio of importance sampling ratios between the behavior and target policies [31, 32]. However, this approach is prone to excessively large variance.

Another possibility is to consider mixing the excursions and alternative life objective through a scalar $\hat{\gamma} \in [0, 1]$ to obtain the counterfactual objective [60], which is written as,

$$J_{\hat{\gamma}}(\nu) = \sum_s d_{\hat{\gamma}}(s) i(s) V^\pi(s)$$

where $d_{\hat{\gamma}}(s)$ is defined as,

$$\mathbf{d}_{\hat{\gamma}} = (1 - \hat{\gamma})(\mathbf{I} - \hat{\gamma}\mathbf{P}_\pi^\top)\mathbf{d}_b$$

When $\hat{\gamma} = 1$, we recover the alternative life objective, whereas when $\hat{\gamma} = 0$ we recover the excursions objective. Optimising this objective requires keeping track of two traces. In practice, it was shown to provide improvements on control tasks. As the best performing agent had a value of $\hat{\gamma}$ was closer to the excursions objective, we decide to proceed with the standard excursions objective to avoid the additional traces.

H.2 Policy parameters

Consider the base learner objective

$$J^B = \sum_s d_b(s) i(s; \eta) V^\pi(s)$$

with the associated base learner updates,

$$\nu' \leftarrow \nu + \alpha_b F_t \rho_t \nabla_\nu \log \pi(A_t|S_t; \nu) Q^\pi(S_t, A_t)$$

For the meta-objective we consider the following excursions objective,

$$J^M = \sum_s d_b(s) V^\pi(s)$$

The meta updates can then be written as,

$$\eta' \leftarrow \eta + \alpha_m \nabla_\eta J^M$$

where using the chain rule gives us,

$$\nabla_\eta J^M = \nabla_{\nu'} J^M \nabla_\eta \nu'$$

Expanding the second term, we get that the stochastic sample of the gradient is,

$$\begin{aligned}
\nabla_\eta \nu' &= \nabla_\eta \big(\nu + \alpha_b F_t \rho_t \nabla_\nu \log \pi(A_t|S_t; \nu) Q^\pi(S_t, A_t)\big) \\
&= \nabla_\eta \big(\alpha_b F_t \rho_t \nabla_\nu \log \pi(A_t|S_t; \nu) Q^\pi(S_t, A_t)\big) \\
&= \nabla_\eta \big(\alpha_b \sum_{i=0}^{t} \gamma^{t-i} i(S_i; \eta) \rho_{i:t} \nabla_\nu \log \pi(A_t|S_t; \nu) Q^\pi(S_t, A_t)\big) \\
&= \alpha_b \sum_{i=0}^{t} \gamma^{t-i} \nabla_\eta i(S_i; \eta) \rho_{i:t} \nabla_\nu \log \pi(A_t|S_t; \nu) Q^\pi(S_t, A_t)
\end{aligned}$$

where we used the log derivative trick [46] and the definition of the followon trace.

H.3 Value parameters

Consider the base learner objective,

$$J^B = \sum_s d_b(s) i(s; \eta) \frac{1}{2} (v^\pi(s) - V^\pi(s; \theta))^2$$

where $v^\pi(s)$ is the true value function (which is inaccessible), with the following base learner updates,

$$\theta' \leftarrow \theta + \alpha_b F_t \rho_t (R_{t+1} + \gamma V^\pi(S_{t+1}; \theta) - V^\pi(S_t; \theta)) \nabla_\theta V^\pi(S_t; \theta) \tag{11}$$

where we consider the semi-gradient update rule [43]. Notice that different values of the bootstrapping coefficient will lead to different update rules [45]. Consider the following meta-objective,

$$J^M = \sum_s d_b(s) \frac{1}{2} (v^\pi(s) - V^\pi(s; \theta'(\eta)))^2$$

with the following meta-update,

$$\eta' \leftarrow \eta + \alpha_m \nabla_\eta J^M$$

where

$$\nabla_\eta J^M = \nabla_{\theta'} J^M \nabla_\eta \theta'$$

In practice the term $\nabla_{\theta'} J^M$ will as well be a semi-gradient due to bootstrapping. In this equation, $\nabla_\eta J^M$ is of size $1 \times m$, where m is the size of the meta-parameters η. Consequently, $\nabla_{\theta'} J^M$ is of size $1 \times n$, where n is the size of the parameters θ and $\nabla_\eta \theta'$ is of size $n \times m$.

Consider the case where the parameters and meta parameters use linear function approximation, that is $i(s; \eta) = \eta^\top \psi(s)$ and $V^\pi(s; \theta) = \theta^\top \phi(s)$, where ϕ are the features for the value function and ψ are the features for the interest function. Furthermore, consider the on-policy case. We can then write that the stochastic sample at time t is,

$$\nabla_{\theta'} J^M_t = \phi_t \delta_t$$

where δ_t is the TD(1) error and ϕ_t indicates the features at time t. We also have that

$$\nabla_\eta \theta'_t = D_{\phi_t} \Psi_t \delta_t$$

where D_{ϕ_t} is the diagonal matrix with ϕ_t on its diagonal and where Ψ_t is the matrix where each row is made of the interest function's features ψ_t.

If we further consider that all features are tabular representations, we have that,

$$\nabla_{\theta'} J^M_t = \mathbf{e}_t \delta_t$$

where $\mathbf{e}_t$ is the one-hot vector that indicates the state at time t. We also have that $\nabla_\eta \theta'$ is a matrix where all elements are zero, except at position s, s (the state at time t) where it is equal to δ_t. Multiplying the $\nabla_{\theta'} J^M$ and $\nabla_\eta \theta'$ together gives us,

$$\nabla_\eta J^M_t = \mathbf{e}_t (\delta_t)^2$$

That is, the meta gradient updates in the direction of the L_2 norm of the TD error.

H.4 Multi-step derivation

In the main text we present the derivation for the meta-gradient obtained from one step updates in the inner loop. This was done to help the clarity of the presentation, but in practice it is possible to do multiple updates to the parameters before updating the meta-parameters. This sequence of K updates can be written as,

$$\nu_K = \nu_0 + \sum_{k=0}^{K-1} U(\eta, \nu_k, \mathcal{D}_k)$$

where U is the update rule of (7) using the data $\mathcal{D}_k$ collected at iteration k (i.e. a set of trajectories). If the data is collected on-policy, taking the gradient of meta-objective with respect to the meta parameters would result in,

$$\nabla_\eta J^M(\eta) = \sum_{k=0}^{K} \mathbb{E}_{\{\mathcal{D}_i\}_{i=0}^{k-1}} \mathbb{E}_\pi \big[\sum_t \big(\sum_{j=0}^{k-1} \nabla_\eta \nu_j \nabla_{\nu_j} \log p(\mathcal{D}_j | \nu_j) + \\ \nabla_\eta \nu_k \nabla_{\nu_k} \log \pi(A_t | S_t; \nu_k) \big) Q^\pi(S_t, A_t) \big]$$

Let's explain this equation. The sum over k comes from considering the evaluation of the meta-objective after each of the inner updates. This is known to optimize the area under the curve of the meta-objective after each update, rather than final performance [79]. The outer expectation is over the datasets collected at each of the previous iterations, as this data affects the current parameters. The sum over t is with respect to the current trajectory obtained from the current parameters. The second term multiplying the action value function, $\nabla_\eta \nu_k \nabla_{\nu_k} \log \pi(A_t|S_t;\nu_k)$ is the part of the meta gradient that considers the effect of the meta parameters on the current trajectory's performance.

The first term multiplying the action value function, $\sum_{j=0}^{k-1} \nabla_\eta \nu_j \nabla_{\nu_j} \log p(\mathcal{D}_j|\nu_j)$, is the *sampling correction* term and was first derived by [2]. As the current parameters were obtained by updates using sampled datasets, the parameters are random variables as well. The sampling correction term then allows for the meta gradients to propagate through inner update iterations. It is usually not included in most of the meta gradient literature, which renders the meta updates biased. However, including them may lead to increased variance [53], and as such there is a trade-off to consider.

In the off-policy control case using the excursions objective, the sampling correction term does not appear as the data is collected through a behavior policy, which is not influenced by the meta-parameters. However, under the alternative life objective we would have extra terms similar to the sampling correction as optimising this objective would require using the full product of importance sampling ratios between target and behavior policies, or the ratio between stationary distributions of the target and behavior policies. Please see section App. H.1 for more details on the alternative life objective.

In the policy evaluation setting, whether we are on or off-policy, the sampling correction term would also not appear, as the parameters of the policy generating the data are not updated. In the policy evaluation setting only the value function estimating the expected return is parametrised.

Additional References for the Appendix

[5] A. G. Baydin, R. Cornish, D. Martínez-Rubio, M. Schmidt, and F. D. Wood. Online learning rate adaptation with hypergradient descent. *CoRR*, abs/1703.04782, 2017.

[8] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016.

[68] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov. Openai baselines. `https://github.com/openai/baselines`, 2017.

[17] S. Ghiassian and R. S. Sutton. An empirical comparison of off-policy prediction learning algorithms in the four rooms environment. *CoRR*, abs/2109.05110, 2021.

[70] Q. Liu, L. Li, Z. Tang, and D. Zhou. Breaking the curse of horizon: Infinite-horizon off-policy estimation. *CoRR*, abs/1810.12429, 2018. URL `http://arxiv.org/abs/1810.12429`.

[28] A. R. Mahmood, H. Yu, M. White, and R. S. Sutton. Emphatic temporal-difference learning. *CoRR*, abs/1507.01569, 2015.

[31] A. Patterson, A. White, S. Ghiassian, and M. White. A generalized projected bellman error for off-policy value estimation in reinforcement learning. *CoRR*, abs/2104.13844, 2021.

[32] D. Precup, R. Sutton, and S. Dasgupta. Off-policy temporal-difference learning with function approximation. *Proceedings of the 18th International Conference on Machine Learning*, 06 2001.

[39] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[43] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[46] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 2000.

[45] R. S. Sutton, A. R. Mahmood, and M. White. An emphatic approach to the problem of off-policy temporal-difference learning. *J. Mach. Learn. Res.*, 17(1):2603–2631, jan 2016. ISSN 1532-4435.

[78] A. Tamar, D. D. Castro, and S. Mannor. Learning the variance of the reward-to-go. *Journal of Machine Learning Research*, 17(13):1–36, 2016. URL `http://jmlr.org/papers/v17/14-335.html`.

[79] V. Veeriah, M. Hessel, Z. Xu, R. L. Lewis, J. Rajendran, J. Oh, H. van Hasselt, D. Silver, and S. Singh. Discovery of useful questions as auxiliary tasks. *CoRR*, abs/1909.04607, 2019. URL `http://arxiv.org/abs/1909.04607`.

[80] M. White. Unifying task specification in reinforcement learning. *CoRR*, abs/1609.01995, 2016.

[81] M. White and A. White. A greedy approach to adapting the trace parameter for temporal difference learning. *CoRR*, abs/1607.00446, 2016. URL `http://arxiv.org/abs/1607.00446`.

[82] T. Zahavy, Z. Xu, V. Veeriah, M. Hessel, J. Oh, H. van Hasselt, D. Silver, and S. Singh. A self-tuning actor-critic algorithm, 2021.

[60] S. Zhang, W. Boehmer, and S. Whiteson. Generalized off-policy actor-critic. *CoRR*, abs/1903.11329, 2019.