Reward Shaping via Meta-Learning

Haosheng Zou [*1] **Tongzheng Ren** [*1] **Dong Yan** [1] **Hang Su** [1] **Jun Zhu** [1]

Abstract

Reward shaping is one of the most effective methods to tackle the crucial yet challenging problem of *credit assignment* in Reinforcement Learning (RL). However, designing shaping functions usually requires much expert knowledge and hand-engineering, and the difficulties are further exacerbated given multiple similar tasks to solve. In this paper, we consider reward shaping on a distribution of tasks, and propose a general meta-learning framework to automatically learn the efficient reward shaping on newly sampled tasks, assuming *only* shared state space but not necessarily action space. We first derive the theoretically *optimal* reward shaping in terms of credit assignment in model-free RL. We then propose a value-based meta-learning algorithm to extract an effective prior over the optimal reward shaping. The prior can be applied directly to new tasks, or *provably* adapted to the task-posterior while solving the task within few gradient updates. We demonstrate the effectiveness of our shaping through significantly improved learning efficiency and interpretable visualizations across various settings, including notably a successful transfer from DQN to DDPG.

1. Introduction

Reinforcement Learning (RL) has recently attracted much attention with its success in various domains such as Atari (Mnih et al., 2015) and Go (Silver et al., 2018). However, the problem of *credit assignment* (Minsky, 1961) still troubles its learning efficiency. It is rather difficult for RL agents to answer the following question: how to distribute credit for success (or penalty for failure) among the sequence of decisions involved in producing the result from naturally delayed (even sparse) rewards. If the agent could know exactly which actions are right or wrong, RL would be no more difficult than supervised learning. Such inefficiency in credit assignment is one major reason for the unsatisfactory learning efficiency of current model-free RL methods.

Reward shaping is one of the most intuitive, popular and effective solutions to credit assignment, whose very goal is to shape the original delayed rewards to properly reward or penalize intermediate actions as in-time credit assignment. The technique first emerges in animal training (Skinner, 1990), and is then introduced to RL (Dorigo & Colombetti, 1994; Mataric, 1994) to tackle increasingly complex problems like Doom (Wu & Tian, 2017) and Dota 2 (OpenAI, 2018). While most shaping functions could be directly applied, it is proved that optimal policies remain invariant under certain ones, namely potential-based shaping functions (Ng et al., 1999).

However, almost all reward shapings are *hand-crafted* and need to be carefully designed by experienced human experts (Wu & Tian, 2017; OpenAI, 2018). On one hand, coding those shaping functions in programming languages is potentially tedious and inconvenient especially in complex large-scale environments such as Doom (Wu & Tian, 2017) and Dota 2 (OpenAI, 2018). On the other hand, humans have to theoretically justify the shaping rewards to ensure that they lead to expected behavior but not other local optima. Together this makes effective reward shapings hard to design/code, and easily coded shapings usually ineffective.

Furthermore, in practice we are usually interested in solving multiple similar tasks as a whole. For example, when training an RL agent to solve 2D grid mazes, we wouldn't like to train individual agents for each maze map, but would naturally hope for *one* general agent for all possible mazes. The shared but not identical task-structures naturally induce a *distribution over tasks*, which in this case is a distribution over maze configurations (Wilson et al., 2007) and could elsewhere be a distribution over system parameters (Lazaric & Ghavamzadeh, 2010) for different robot-hand sizes or over game maps for RTS games (Jaderberg et al., 2018). The ability to quickly solve new similar tasks drawn from such distributions is much expected for general intelligence, since it is mastered by human infants quite young (Smith & Slone, 2017). However, the human effort in reward shaping would be further exacerbated, where we have to either design a different shaping per task or come up with a general

[*]Equal contribution [1]Dept. of Comp. Sci. & Tech., State Key Lab of Intell. Tech. & Sys., TNList Lab, CBICR Center Tsinghua University, Beijing, China. Correspondence to: Jun Zhu <dcszj@mail.tsinghua.edu.cn>, Haosheng Zou <zouhs16@mails.tsinghua.edu.cn>, Tongzheng Ren <rtz19970824@gmail.com>.

task-dependent function presumably harder to design.

To this end, we consider the generally hard problem of reward shaping on a distribution of tasks. Motivated by the inconvenience in reward shaping under task multiplicity, we seek to design a *general, automatic* reward shaping mechanism that works well on the task distribution without hand-engineering of human experts. We first derive the theoretically *optimal* reward shaping in terms of credit assignment in model-free RL to be the optimal V-values. By spotting that there exists shared knowledge across tasks on the same distribution, we then propose a novel value-based algorithm based on Model-Agnostic Meta-Learning (MAML) (Finn et al., 2017b), leveraging meta-learning to extract such prior knowledge. This **prior** approximates the optimal potential-based shaping function (Ng et al., 1999) for each task. The meta-learned prior conducts reward shaping on newly sampled tasks either directly (zero-shot) or adapting to the **task-posterior** optimum (few-shot) to shape rewards in the meantime of solving the task. We provide theoretical guarantee for the latter. Extensive experiments demonstrate the effectiveness of our reward shaping in both two cases.

To summarize, our contributions are: (1) We present a first attempt to conduct *general, automatic* reward shaping with meta-learning on a distribution of tasks for better credit assignment and learning efficiency; (2) Our framework requires *only* a shared state space across tasks, and could be applied either directly or adaptively on newly sampled tasks, which is quite general and flexible compared with most existing meta-learning methods and multi-task reward shaping works; (3) We theoretically derive and analyze the optimal reward shaping (w.r.t. credit assignment based on potential functions (Ng et al., 1999)) and our shaping algorithm.

2. Preliminaries

We consider the setting of multi-task reinforcement learning (RL), where the tasks follow a distribution $p(\mathcal{T})$. Each sampled task $\mathcal{T}_i \sim p(\mathcal{T})$ is a standard Markov Decision Process (MDP) $M_i = (\mathcal{S}, \mathcal{A}_i, T_i, \gamma, R_i)$, where $\mathcal{S}$ is the state space, assumed to be shared by all tasks, $\mathcal{A}_i$ is the action space, $T_i : \mathcal{S} \times \mathcal{A}_i \times \mathcal{S} \to [0, 1]$ is the state transition probability, $\gamma \in [0, 1]$ is the discount factor and $R_i : \mathcal{S} \times \mathcal{A}_i \times \mathcal{S} \to \mathbb{R}$ is the reward function. Here, we use the subscript i to denote that the tasks may have *different* action spaces $\mathcal{A}_i$, different transition probabilities T_i and different reward functions R_i.

In this section, we briefly introduce the techniques on which our method is based, namely general Q-learning variants to solve individual MDPs, reward shaping functions to accelerate learning with theoretical guarantees, and meta-learning to tackle reward shaping on task distributions.

2.1. Q-Learning

Given any MDP M, a policy is a distribution $\pi(a|s)$. The **V-value** $V^\pi_M(s)$ and **Q-value** $Q^\pi_M(s, a)$ are correspondingly defined for (M, π) as cumulative rewards. The goal of standard RL on a single task is to find the optimal π that gives maximal V-(and Q-)values: $V^*_M(s) = \sup_\pi V^\pi_M(s), Q^*_M(s, a) = \sup_\pi Q^\pi_M(s, a)$.

Q-Learning (Watkins & Dayan, 1992) provides one solution to directly learn Q^*_M and induce π from it. Different from previously tabular representations, Deep Q-Network (DQN) (Mnih et al., 2015) parameterizes the Q-value with a neural network Q_θ and minimizes the temporal difference (TD) error (Sutton & Barto, 1998) with gradient descent:

$$\min_\theta \|R(s, a, s') + \gamma \max_{a'} Q_\theta(s', a') - Q_\theta(s, a)\|^2,$$

where θ represents the parameters of the neural network. A periodic target network is usually adopted.

Dueling-DQN (Wang et al., 2016b) specifically parameterizes Q_θ as $Q_\theta(s, a) = V_\theta(s) + A_\theta(s, a)$ so as to " generalize learning across actions" for better learning efficiency and performance. The neural network's penultimate layer outputs a V-value head V_θ and an advantage head A_θ that sum to the ultimate Q-value. Still, the delayed (or even sparse) nature of rewards poses great challenge on learning.

2.2. Potential-based shaping function

A reward-shaping function $F : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ modifies the original reward function and attempts to make RL methods (e.g., Q-learning) converge faster with more "instructive" rewards. It generally resides in the same functional space as the reward function R, and transforms the original MDP $M = (\mathcal{S}, \mathcal{A}, T, \gamma, R)$ into another **shaped** MDP $M' = (\mathcal{S}, \mathcal{A}, T, \gamma, R' = R + F)$. Of all possible shapings, potential-based shaping functions (Ng et al., 1999) retain the optimal policy, as summarized below.

Definition 2.1 (Potential-based shaping function (Ng et al., 1999))**.** *$F : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is a potential-based shaping function if there exists a real-valued function $\Phi : \mathcal{S} \to \mathbb{R}$, such that $\forall (s, a, s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$,*

$$F(s, a, s') = \gamma\Phi(s') - \Phi(s).$$

$\Phi(s)$ is thus called the potential function.

Theorem 2.1 (Policy Invariance under Reward Shaping (Ng et al., 1999))**.** *The condition that F is a potential-based shaping function is necessary and sufficient for it to guarantee consistency with the optimal policy. Formally, for $M = (\mathcal{S}, \mathcal{A}, T, \gamma, R)$ and $M' = (\mathcal{S}, \mathcal{A}, T, \gamma, R + F)$, if $F(s, a, s') = \gamma\Phi(s') - \Phi(s)$, then $\forall (s, a) \in \mathcal{S} \times \mathcal{A}$*

$$\begin{aligned} Q^*_{M'}(s, a) =& Q^*_M(s, a) - \Phi(s), \\ V^*_{M'}(s) =& V^*_M(s) - \Phi(s), \end{aligned} \tag{1}$$

so the policy derived from $Q^*_{M'}$ remains the same.

Consequently, if we choose $\Phi(s) = V^*_M(s)$, then $V^*_{M'}(s) \equiv 0$, and "all that would remain to be done would be to learn the non-zero Q-values" (Ng et al., 1999).

However, why are the "non-zero Q-values" easier to learn for RL? Agents could never know *a priori* which actions' Q-values are zero, and we cannot directly induce policies from V-values without access to the underlying MDP model. We found that the true advantage this particular reward shaping brings about is under-appreciated in previous works, and in Sec. 3.2 we provide formal analysis and identify its theoretical optimal efficiency in *credit assignment*, motivating our framework based on such shaping functions.

2.3. Meta-Learning

Meta-learning is an effective strategy to deal with a distribution of tasks. Specifically, it operates on two sets of tasks: **meta-training** set $\{\mathcal{T}_i\}_{i=1}^N$ and **meta-testing** set $\{\mathcal{T}_j\}_{j=N+1}^{N+M}$, both drawn from the same task distribution $p(\mathcal{T})$. The meta-learner attempts to learn the structure of tasks during meta-training, and in meta-testing, it leverages the structure to learn efficiently on new tasks with a limited number of newly observed examples from new tasks.

Meta-learning methods have been developed in both supervised learning (Santoro et al., 2016; Vinyals et al., 2016) and RL settings (Duan et al., 2016; Wang et al., 2016a). One of the most popular algorithms is Model-Agnostic Meta-Learning (MAML) (Finn et al., 2017a), which meta-learns an versatile initialization θ of model parameters by:

$$\phi_i \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta), \tag{2}$$

$$\theta \leftarrow \theta - \beta \nabla_\theta \mathbb{E}_{\mathcal{T}_i} \mathcal{L}_{\mathcal{T}_i}(f_{\phi_i}). \tag{3}$$

where θ are the parameters to be learned, ϕ_i are the task-specific parameters updated from θ as initialization (Eqn. (2)), α and β are learning rates and $\mathcal{L}_{\mathcal{T}_i}$ is the loss function on each $\mathcal{T}_i$. Note that ϕ_i depend on θ and the gradients back-propagated through ϕ_i to θ (Eqn. (3)). In meta-testing, given data from the new task $\mathcal{T}_j$, MAML adapts model parameters starting from θ. MAML has also been recently extended to a more Bayesian treatment (Grant et al., 2018; Yoon et al., 2018; Finn et al., 2018).

3. Methods

Based on the notions and notations in Sec. 2, we first formulate the problem of learning shaping functions on a distribution of tasks. Then we derive the optimal shaping function we'd like to learn and introduce our algorithm to learn the shaping function on sampled tasks from the distribution. Lastly we introduce how to use the learned shaping function on newly sampled tasks.

3.1. Problem Formulation

Our goal is to learn a potential function $\Phi(s) : \mathcal{S} \to \mathbb{R}$ capable of effective reward shaping on tasks sampled from the distribution to accelerate their learning. We seek to learn $\Phi(s)$ via mete-learning on a certain number of sampled tasks. In terms of meta-learning, this is the *meta-training* phase to extract *prior* knowledge from the task distribution. In light of this and recent works (Grant et al., 2018; Yoon et al., 2018; Finn et al., 2018), we call $\Phi(s)$ the potential function **prior**. During *meta-testing* phase, we seek to directly plug in the prior to shape rewards as a general test, or to adapt it to the **task-posterior** $\Phi_i(s|\mathcal{T}_i)$ under more restricted conditions for more effective shaping.

Note that in implementation we instantiate the prior as $\Phi(s;\theta)$ and task-posterior as $\Phi_i(s|\mathcal{T}_i;\phi_i)$, i.e., ordinary neural networks rather than distributions. However, our method could still be understood from a Bayesian perspective by treating the prior as a delta function, the task-posterior as maximum-a-posteriori inference and the overall algorithm as empirical Bayes, the details of which are beyond the scope of this paper and readers may refer to (Grant et al., 2018; Yoon et al., 2018; Finn et al., 2018).

Next, we first derive the ideal task-posterior $\Phi(s|\mathcal{T}_i)$.

3.2. Efficient Credit Assignment with Optimal Potential Functions

Delving deeper into the particular potential function $\Phi(s) = V^*_M(s)$ in Sec. 2.2, we first show that the substantial advantage it brings to credit assignment, which the "non-zero Q-values" fail to identify, is the following:

Theorem 3.1. *Shaping with $\Phi(s) = V^*_M(s)$ is optimal for credit assignment and learning efficiency.*

Proof. We first show that the reward shaping gives non-positive immediate rewards with the optimal actions' rewards exclusively zero. To see this, consider a general MDP M and the corresponding **shaped** MDP M', we have

$$\begin{aligned} R'(s,a) =& \mathbb{E}_{s'} R'(s,a,s') \\ =& \mathbb{E}_{s'}[R(s,a,s') + \gamma V^*_M(s') - V^*_M(s)] \\ =& Q^*_M(s,a) - \max_a Q^*_M(s,a) \\ \leq& 0, \end{aligned}$$

where the last equality holds iff $a = \arg\max_a Q^*_M(s,a)$.

Therefore, after shaping the rewards with $\Phi(s) = V^*_M(s)$, at *any* state, only the optimal action(s) give zero immediate reward, and all the other actions give strictly negative rewards right away. As a result, *credit assignment* could be achieved the most efficiently since the agent could spot a deviation from the optimal policy as soon as it receives a

Algorithm 1 Meta-learning potential function prior

Input: $p(\mathcal{T})$: a distribution over tasks
Input: α, β: step sizes
Output: Learned prior θ
Randomly initialize parameter θ for prior
for `meta_iteration` $= 0, 1, 2...$ **do**
  Sample a batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
  for all $\mathcal{T}_i$ **do**
    Initialize replay buffer $\mathcal{D}_i$
    Collect experience $\{s_0, a_0, r_0, \cdots\}$ with ϵ-greedy using $Q_\theta(s, a)$ and add to the replay buffer $\mathcal{D}_i$
    Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(Q_\theta)$ using samples from $\mathcal{D}_i$ ($\mathcal{L}_{\mathcal{T}_i}$ defined in Eqn. (5))
    Compute adapted parameters with gradient descent: $\phi_i = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(Q_\theta)$
  end for
  Update $\theta \leftarrow \theta - \beta \nabla_\theta \mathbb{E}_{\mathcal{T}_i} \|Q_\theta(s, a) - Q_{\phi_i}(s, a)\|^2$ with previous samples from all $\mathcal{D}_i$
end for

negative reward. The optimality of any action could be determined instantaneously after it's taken without any need to consider future rewards, and any RL algorithm could penalize negative-reward actions without any fear that they might lead to better rewards in the future, hence the theoretically optimal efficiency in credit assignment. □

We thus choose $V^*_{M_i}(s)$ as the adaptation target of task-posterior $\Phi_i(s|\mathcal{T}_i)$. In practical RL, the non-positivity may not always hold with sampled experience and rewards from the environment, but the property still holds under expectation, and mini-batches of data approximate the very expectation. Learning efficiency could therefore be still improved, which will be demonstrated through our experiments.

3.3. Meta-Learning Potential Function Prior

The optimal shaping function $V^*_{M_i}(s)$ is task-specific without a universal optimum for all tasks $\mathcal{T}$. Inspired by MAML's idea to learn a proper prior capable of fast adaptation to the task-posterior, we propose Alg. 1, as detailed below.

Formally, we specify the prior as $\Phi(s; \theta)$ defined on $\mathcal{S}$ with parameters θ. For each task $\mathcal{T}_i$, the task-posterior $\Phi_i(s|\mathcal{T}_i; \phi_i)$ adapts in the direction of $V^*_{M_i}(s)$ initialized from θ. Then, a natural objective of learning prior is:

$$\min_\theta \mathbb{E}_{\mathcal{T}_i} \|\Phi(s; \theta) - V^*_{M_i}(s)\|^2. \tag{4}$$

However, $V^*_{M_i}(s)$ is not directly accessible for any MDP, and neither is there any RL algorithm to directly learn optimal V-values. We therefore first specify the adaptation from prior to $V^*_{M_i}(s)$, and then return to the learning of prior.

Algorithm 2 Meta-testing (adaptation with advantage head)

Input: $\mathcal{T}_j$: new task to solve
Input: ϕ_j: task-posterior parameters, initialized as learned prior θ
Output: adapted task-posterior ϕ_j
Initialize replay buffer $\mathcal{D}$
for `gradient_step` $= 0, 1, 2...$ **do**
  Collect experience $\{s_0, a_0, r_0, \cdots\}$ with ϵ-greedy using $A_{\phi_j}(s, a)$ and add to replay buffer $\mathcal{D}$
  Update $A_{\phi_j}(s, a)$ with Eqn. (8) with samples from $\mathcal{D}$ and current potential function $V_{\phi_j}(s)$ for shaping
  Update $V_{\phi_j}(s)$ with Eqn. (9) with samples from $\mathcal{D}$
end for

Task-Posterior Adaptation: Existing policy-based RL methods either don't estimate values or simply use the value output as baseline or bootstrap, leaving value-based RL methods more suitable for our framework. In this paper, we simply choose Q-learning (Watkins & Dayan, 1992), though in principle any value-based algorithm explicitly estimating optimal values could be adopted.

Q-learning still cannot directly estimate optimal V-values. To address this, we decompose the optimal values as:

$$Q^*_{M_i}(s, a) = V^*_{M_i}(s) + A^*_{M_i}(s, a),$$

where $A^*_{M_i}(s, a)$ is the advantage-value function. We implement this by separating the V-value head and advantage head before the network outputs Q-value:

$$Q_{\phi_i}(s, a) = V_{\phi_i}(s) + A_{\phi_i}(s, a),$$

where ϕ_i is initialized as θ.

Note that $V_{\phi_i}(s)$ (and $V_\theta(s)$) are the potential functions we need, so $\Phi_i(s|\mathcal{T}_i; \phi_i)$ (and $\Phi(s; \theta)$) are just part of the whole network, but for completeness we denote the overall parameters ϕ_i (and θ) and treat $\Phi_i(s|\mathcal{T}_i; \phi_i)$ (and $\Phi(s; \theta)$) as "augmented" potential functions.

Task-posterior adapts by following Q-learning and minimizing the TD error:

$$\mathcal{L}_{\mathcal{T}_i}(Q_{\phi_i}) = \|R_i(s, a, s') + \gamma \max_{a'} Q_{\phi_i}(s', a') - Q_{\phi_i}(s, a)\|^2. \tag{5}$$

This method was first introduced in dueling-DQN (Wang et al., 2016b) but for a different purpose of speeding up training. Here we exploit the architecture in estimating the optimal V-values. To see this, first note that for identifiability of V and A, the maximum of the output advantage function is further subtracted from Q in implementation:

$$Q_{\phi_i}(s, a) = V_{\phi_i}(s) + A_{\phi_i}(s, a) - \max_{a'} A_{\phi_i}(s, a'). \tag{6}$$

As Q_{ϕ_i} attains $Q^*_{M_i}$ during Q-learning, by taking $\max_a$ on both sides of Eqn. 6, we get $V_{\phi_i}(s) = \max_a Q^*_{M_i}(s, a) =$

$V^*(s)$. We can therefore learn the optimal V-values with dueling-DQN, adapting to task-posterior from prior.

Prior Learning: Following the design of the task-posterior, the prior is naturally instantiated also as a dueling-DQN $Q_\theta(s,a) = V_\theta(s) + A_\theta(s,a)$. Similar as MAML (Finn et al., 2017a), we explicitly model the desired property of the prior to be able to efficiently adapt to the task-posterior. Based on that each task-posterior adapts from θ to ϕ_i on $\mathcal{T}_i$ with N steps of gradient update, we could finally rewrite the impractical prior-learning problem (4) as a practical one:

$$\min_\theta \mathbb{E}_{\mathcal{T}_i} \|Q_\theta(s,a) - Q_{\phi_i}(s,a)\|^2. \tag{7}$$

It is worth noting that this problem is in essence different from that of DQN, as it does *not* compute bootstrapped Q-values for TD error but directly uses Q_{ϕ_i} under the expectation of $\mathcal{T}_i$ as the learning target for Q_θ.

Also note that in implementation we keep the full computational graph of task-posterior adaptation so ϕ_i is dependent on θ and gradients could back-propagate through ϕ_i to θ. For all our experiments we set $N = 1$ for simplicity, but $N > 1$ is a natural implementational extension. Possibly thanks to task multiplicity, we didn't find target networks necessary for the Q-networks. Besides, since it's still an overall off-policy algorithm, we don't need to re-sample data for θ update, contrary to MAML.

3.4. Meta-Testing with Potential Function Prior

During meta-testing, we aim to find the optimal policy on newly sampled tasks $\mathcal{T}_j$ with reward shaping by the learned potential function prior. We use the meta-learned $V_\theta(s)$ to directly shape the MDP, which transforms the original MDP $M_j = (\mathcal{S}, \mathcal{A}_j, T_j, \gamma, R_j)$ into the shaped MDP $M'_j = (\mathcal{S}, \mathcal{A}_j, T_j, \gamma, R'_j := R_j + F)$, where $F(s,a,s') = \gamma V_\theta(s') - V_\theta(s)$. Intuitively, $V_\theta(s)$ provides a good estimate of $V^*_{M_j}(s)$ from meta-training on the task distribution, thus learning on M'_j can be much simpler than learning on M_j as the reward shaping is close to optimal.

We identify two cases of meta-testing with our dueling-DQN-based meta-learning algorithm. **Shaping only** is one case where $V_\theta(s)$ is directly applied on new tasks without adaptation. This applies to new tasks with different action spaces, or when the advantage head simply could not be used for some reason (e.g., constraints on the new policy). According to Thm. 2.1, any RL algorithm could be used on the shaped MDP with the optimal policy unchanged. **Adaptation with advantage head** is the other case where the action space doesn't change and the DQN-policy is still applicable. We can then jointly adapt $V^*_{M_j}(s)$ to the task-posterior and find the optimal policy efficiently within a few updates, initializing the whole ϕ_j as θ.

In the latter case, we still shape the MDP with the task-posterior being adapted. We iteratively collect experience using $A_{\phi_j}(s,a)$ with ϵ-greedy and update $A_{\phi_j}(s,a)$ and $V_{\phi_j}(s)$ alternating the following two steps (step size α):

◦ Update $A_{\phi_j}(s,a)$ with sampled data from replay buffer:

$$\phi_j \leftarrow \phi_j - \alpha \nabla_{\phi_j} \|R'_j(s,a,s') + \gamma \max_{a'} A_{\phi_j}(s',a') - A_{\phi_j}(s,a)\|^2. \tag{8}$$

◦ Update $V_{\phi_j}(s)$ with sampled data from replay buffer:

$$\phi_j \leftarrow \phi_j - \alpha \nabla_{\phi_j} \|V_{\phi_j}(s) - \texttt{stop\_gradient}\big(\max_a A_{\phi_j}(s,a) + V_{\phi_j}(s)\big)\|^2. \tag{9}$$

Theorem 3.2. *Eqn. (8) optimizes for the optimal policy. Eqn. (9) optimizes for the task-posterior* $V^*_{M_j}(s)$.

We defer the proof to Appx. A.

For faster adaptation on new tasks, we simply optimize Eqn. (8) and Eqn. (9) alternately, which we find sufficient in experiments. We summarize such **adaptation with advantage head** in Alg. 2.

Advantage over MAML: Note that in the latter case of meta-testing, one can directly adapt as the original MAML. However, direct adaptation merely exploits the parameter initialization, while our Alg. 2 also explicitly exploits the efficient reward shaping of the potential function prior in addition. The shaped rewards are easier for policy learning, and the adapting shaping (Eqn. (9)) further boosts policy learning (Eqn. (8)) immediately in the next loop. Thus our Alg. 2 is faster and more stable than direct MAML, and in Sec. 5 we compare with and outperform MAML. We also emphasize that we only assume shared state space, facilitating adaptation across discrete and continuous action spaces, which MAML cannot achieve.

4. Related Work

To the best of our knowledge, the only recent work on *automatic* reward shaping on a task distribution is Jaderberg et al. (2018). In addition to being independent of our work, the difference of Jaderberg et al. (2018) is that they access the limited novel states (termed "game events") of the game engine of their specific task and only need to evolve the rewards for those states. Such rewards are simply stored in a short, fixed-length table and optimized with evolution strategies, with the meta-optimization objective of evolution being also designed task-specifically. Earlier similar works (Konidaris & Barto, 2006; Snel & Whiteson, 2010) are also restricted in various ways such as relying on specific feature choice and evolution heuristics, being unable to adapt to new tasks as ours, lacking theoretical analysis of reward shaping on credit assignment or being unable to scale to complex environments with simple models. In contrast to

those works, our method is quite general, assuming no task knowledge or model access, with a more general, principled meta-learning objective, flexible application settings, novel theoretical analysis and gradient-based optimization.

Apart from Jaderberg et al. (2018), almost all other recent RL successes in complex environments either *manually* design reward shaping based on game elements, with examples in Doom (Wu & Tian, 2017) and Dota 2 (OpenAI, 2018), or simply depart from the scalar-reward RL approach and exploit rich supervision signals of other source with supervised learning (Dosovitskiy & Koltun, 2017; Silver et al., 2018; Huang et al., 2019; Wu et al., 2019).

5. Experiments

We demonstrate the effectiveness and generality of our framework under various settings. First we conduct experiments on the classic control task, CartPole (Barto et al., 1983), where the task distribution is defined varying the pole length and the action space could be either discrete or continuous. We then consider grid games whose state space is of much higher dimensionality and the maps of which hold exponential many possibilities (the task distribution is also defined on all the possible maps). Depending on whether the action space shares across the task distribution, the advantage head in our dueling-DQN model (and thus the Q-values) may not be applicable to newly sampled tasks. We therefore experiment under both settings to test the learning efficiency on new tasks. Since we are more interested in general complex MDPs where shaping rewards are hard to code and our meta-training relies on function approximators to generalize on the task distribution, we use neural-network agents in all experiments under the model-free setting.

5.1. Discrete and Continuous CartPoles

In CartPole (Barto et al., 1983), the agent tries to keep a pole upright by applying horizontal forces to the cart supporting the pole. Although a single particular CartPole is not very difficult, it's non-trivial to consider infinitely many CartPole tasks with different pole lengths, since the pole length affects the pole mass, mass center and, therefore, the whole dynamics of the environment. Besides, the applied forces could also be represented in either a discrete or continuous way in different tasks, posing further difficulties in solving them altogether.

A positive reward of 1 is provided every timestep as long as the pole stays within a pre-defined "upright" range of 15 degrees from vertical (Barto et al., 1983; Brockman et al., 2016). This reward is not sparse, but is still far from optimal in terms of *credit assignment* since it does not distinguish between "really" upright positions and dangerous ones where the pole is yet about to fall. To design a properly distinguishing reward shaping obviously requires much expert knowledge of the underlying physics. Therefore, automatic reward shaping on the distribution of CartPoles is of much significance.

Basic Training Settings: We modify the CartPole environment in OpenAI Gym (Brockman et al., 2016) so that it accepts pole length as a construction parameter and changes the physical dynamics accordingly. The pole length is uniformly sampled within the range of $[0.25, 5.00]$ and defines a distribution over CartPoles. All the state spaces $\mathcal{S} \subset \mathbb{R}^4$. We use the discrete two-action setting (a fixed amount of force to the left or right) and the aforementioned original reward during meta-training. Episodes terminate after 200 timesteps, so the maximum achievable return is 200.

For the dueling-DQN we use an MLP with two hidden layers of size 32, followed by one linear layer for the advantage head and one for the value head to aggregate the output Q-values as Eqn. (6). The prior θ is meta-trained with Alg. 1 for 500 meta iterations with 10 sampled tasks per iteration. Note that the tasks are merely used for the meta-update in Alg. 1 with no performance guarantee on single tasks. All results are taken across five random seeds from 0 to 4.

Intuitively, Alg. 1 is learning to generalize over different dynamics to assess how good/bad a state is.

Meta-Testing with Advantage Head: We first test the case of *adaptation with advantage head* as per Sec. 3.4, where test tasks share the action space with meta-training tasks. The meta-trained prior θ is evaluated on 40 newly sampled unseen discrete CartPoles with Alg. 2 to see how fast and how well the potential function (value head), as well as the advantage head, adapts to each new task after re-initializing their weights to θ. As mentioned in Sec. 3.4, we compare with the meta-testing procedure of MAML as a baseline, keeping θ and all common hyperparameters the same.

We track the episodic returns of the agent after each gradient update step, aggregate all such returns across different meta-test tasks and different runs, and plot their medians and quartiles in Fig. 1 (left). As can be seen, our method performs better than MAML, achieving the max 200 two times faster (in 4 steps c.f. 8 steps) and oscillates milder, with improvement even clearer in Sec. 5.2. The relatively high initial return also indicates the quality of the meta-learned prior on new tasks. While MAML could also exploit the prior over the entire model, it's with the additional reward shaping that our method could adapt and learn on new tasks faster. Note that oscillation could not be completely avoided since it's to some extent inherent to off-policy RL algorithms, as is shown in later experiments.

Meta-Testing from Discrete to Continuous: We then test the *shaping only* case as per Sec. 3.4. With V_θ directly used for reward shaping zero-shot, we train: (1) a vanilla DQN

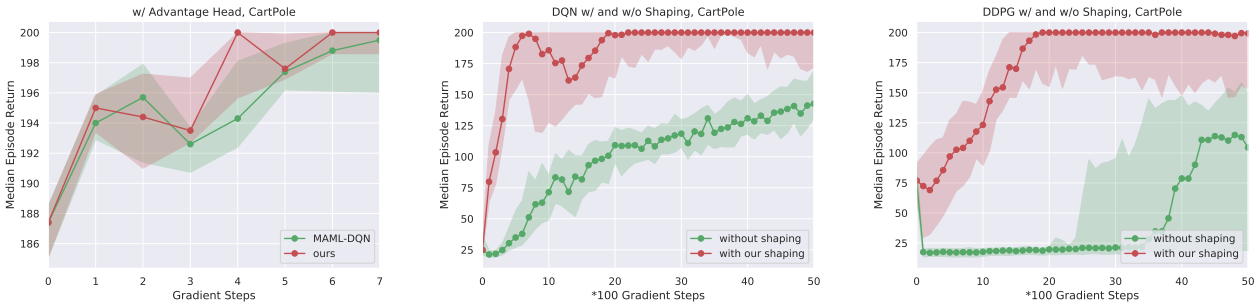

Figure 1. Median and quartile return curves of meta-testing on newly sampled tasks on CartPoles. We uniformly outperform baselines in learning efficiency and stability. Left: meta-testing with advantage head. Improvement over directly applying MAML to the learned prior (green) is even clearer on later grid games. Middle: learning randomly initialized vanilla DQNs with (red) and without (green) our meta-learned zero-shot reward shaping. Right: learning randomly initialized continuous policies using DDPG with (red) and without (green) our meta-learned zero-shot reward shaping in continuous action space.

with randomly initialized weights on discrete CartPoles, corresponding to situations where the advantage head could not be used, and (2) a deterministic policy network using DDPG on continuous CartPoles, corresponding to situations where meta-test tasks have *different* actions spaces, which disqualifies almost all existing meta-learning methods.

The vanilla DQN has only two hidden layers of size 32 without dueling. It's randomly re-initialized for each test task, and we track and plot the test progress similarly as before, except that we evaluate episodic returns every 100 updates. Naturally, we compare with training the same vanilla DQN with the same common hyperparameters but without any reward shaping to test the effectiveness of the meta-learned reward shaping. As shown in Fig. 1 (middle), the zero-shot reward shaping still significantly boost the learning process on new tasks, achieving the max 200 remarkably faster while "without shaping" hasn't achieved yet.

To test with continuous action, we further modify the CartPole environment to accept a scalar real value as action, whose sign determines the direction and absolute value determines the force magnitude. We use a deterministic policy also with two hidden layers of size 32, and an additional two-hidden layer critic network for DDPG. Similar as with the vanilla DQN, we run DDPG with or without our reward shaping. As shown in Fig. 1 (right), learning on new tasks is again significantly accelerated with our reward shaping. Note that because we apply `tanh` nonlinearity to the action output to bound the actions, the initial policy appears more stable with higher initial returns than in the discrete case. However, due to the non-optimal original reward in terms of credit assignment, DDPG without shaping confuses and struggles at first with returns dropping to below 25.

5.2. Grid Games

Grid games are clean but still challenging environments for model-free RL agents in terms of navigation and planning, especially when using neural nets as the agent model (Tamar et al., 2016) since tabular representations could not generalize across grids. Many real-world environments could be modeled as grids in 2D or 3D. While represented simple, grids could have many variations with different start and goal positions on a 8×8 grid incurring $64 \times 63 = 4032$ different tasks. Introducing additional obstacles on the maps leads to combinatorial explosion of further possibilities.

Furthermore, grids almost always come with sparse rewards with rewards obtained only in novel states like goals or traps. Such rewards are probably the most difficult for credit assignment, and to manually design reward shapings requires full access to the environment model and much human knowledge and heuristics which usually pre-compute the shortest paths or some distance metrics. Therefore, it's very important to study automatic reward shaping on the distribution of grid games.

We randomly generate grid maps specifying start and goal positions and possibly obstacles and traps. Agents start from the start position, move in the four canonical top, down, left and right directions and only receive a positive reward of 1 upon reaching the goal. The discount factor assures that the optimal V-/Q-values display certain notion of shortest path. Episodes terminate if the agent hasn't reached the goal in certain timesteps (50 in our experiments).

We use the same representations for start, goal and obstacles respectively across different maps, so intuitively, Alg. 1 learns to recognize and generalize concepts of map positions and, more importantly, the notion of shortest path to goal.

Grid Games with Clean Maps: We first experimented with a simpler version of grid mazes with only start and goal positions but no obstacles. We generate 800 such maps of size 10×10 (e.g., Fig. 2 (left)) for meta-training, where all state spaces $\mathcal{S} \subset \mathbb{R}^{10\times10\times4}$ with the last dimension corresponding to the 4 channels of 0-1 maps of start, goal and current position as well as obstacles (all 0 in this case).

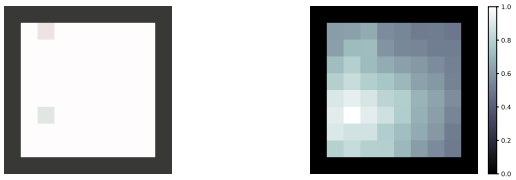

Figure 2. Left: One instance of the clean maps with only the start position (red) and goal (green). Right: Visualization of the meta-learned prior (V-values) on the left map. It matches the intuition that the closer to goal, the higher the value, and is expectedly not optimal yet with values not strictly symmetric w.r.t. goal.

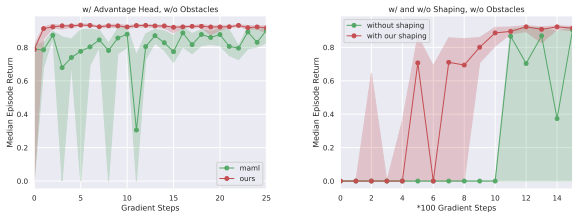

Figure 3. Median and quartile return curves of meta-testing on newly sampled tasks on clean maps. We again uniformly outperform baselines in learning efficiency and stability. Left: meta-testing with advantage head. Right: learning randomly initialized vanilla DQNs with (red) and without (green) our meta-learned zero-shot reward shaping.

For the dueling-DQN we use a CNN with four convolutional layers with 32 kernels of 3×3 and stride 1, followed by two fully connected layers and then the dueling module. Meta-training is conducted similarly as in Sec. 5.1.

We also meta-test the two cases as per Sec. 3.4: *adaptation with advantage head* from the whole prior θ, and *shaping only* to train a vanilla DQN with zero-shot V_θ shaping. We mainly follow the procedure as in Sec. 5.1, except that we don't construct continuous-action grids. All common hyperparameters are the same between any pair of our method and baseline.

As can be seen from Fig. 3, our method performs much better in both cases of meta-testing in terms of learning efficiency and stability, displaying the high potential of our method in scaling to complex environments and agent models. Visualization of the learned V-values on an unseen map (Fig. 2 (right)) also justifies the meta-learned prior θ.

Grid Games with Obstacles: We then experimented with a fuller version of grid mazes where obstacles may be present at each grid position with probability 0.2 during map generation. We generate 4000 such maps of size 8×8 (e.g., Fig. 4 (left)) for meta-training, so all state spaces $\mathcal{S} \subset \mathbb{R}^{8\times8\times4}$. We used the same convolutional dueling-DQN architecture as on clean maps, and the same meta-training/-testing protocols.

Figure 4. Left: One instance of the maps with obstacles. Right: Visualization of the meta-learned prior (V-values) on the left map. It also matches the shortest path intuition with non-optimal, not strictly symmetric values.

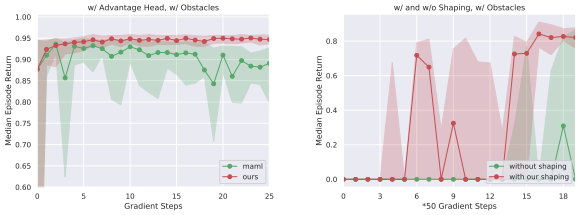

Figure 5. Median and quartile return curves of meta-testing on newly sampled tasks on maps with obstacles. We constantly outperform baselines in learning efficiency and stability. Left: meta-testing with advantage head. Right: learning randomly initialized vanilla DQNs with (red) and without (green) our meta-learned zero-shot reward shaping.

As can be seen from Fig. 5, our method constantly learns more efficiently than the baselines in both cases of *adaptation with advantage head* and *shaping only* on new tasks. The meta-learned V_θ of an unseen map also passes intuitive sanity check (Fig. 4 (right)). Oscillation is a bit more severe than before due to the harder tasks and off-policy algorithmic nature, but ours is still superior in relative performance and stability.

6. Conclusions

In this paper, we consider the problem of reward shaping on a distribution of tasks. We first prove the optimality of optimal V-values for potential-based reward shaping in terms of credit assignment. We then propose a meta-learning algorithm to learn a flexible prior over the optimal V-values. The prior could be well applied directly to shape rewards and could also quickly adapt to the task-posterior optimum while solving the task. We provide additional theoretical guarantee for the latter case. Meanwhile, our framework only assumes that the state spaces of the task distribution are shared, leaving wide possibilities for potential applications. Extensive experiments demonstrate the effectiveness of our method in terms of learning efficiency and stability on new tasks. We plan to consider adapting the shaping prior without the advantage head, and also single-task setting in the future.

Acknowledgement

Haosheng Zou would personally like to thank his beautiful and sweet wife, Jiamin Deng, for her incredible suppport during the whole process of this paper by not being around most of the time.

References

Barto, A. G., Sutton, R. S., and Anderson, C. W. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, 1983.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym, 2016.

Dorigo, M. and Colombetti, M. Robot shaping: Developing autonomous agents through learning. *Artificial intelligence*, 71(2):321–370, 1994.

Dosovitskiy, A. and Koltun, V. Learning to act by predicting the future. In *ICLR*, 2017.

Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., and Abbeel, P. Rl^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017a.

Finn, C., Yu, T., Zhang, T., Abbeel, P., and Levine, S. One-shot visual imitation learning via meta-learning. In *CoRL*, 2017b.

Finn, C., Xu, K., and Levine, S. Probabilistic model-agnostic meta-learning. In *NeurIPS*, 2018.

Grant, E., Finn, C., Levine, S., Darrell, T., and Griffiths, T. Recasting gradient-based meta-learning as hierarchical bayes. In *ICLR*, 2018.

Huang, S., Su, H., Zhu, J., and Chen, T. Combo-action: Training agent for fps game with auxiliary tasks. In *AAAI*, 2019.

Jaderberg, M., Czarnecki, W. M., Dunning, I., Marris, L., Lever, G., Castaneda, A. G., Beattie, C., Rabinowitz, N. C., Morcos, A. S., Ruderman, A., et al. Human-level performance in first-person multiplayer games with population-based deep reinforcement learning. *arXiv preprint arXiv:1807.01281*, 2018.

Konidaris, G. and Barto, A. Autonomous shaping: Knowledge transfer in reinforcement learning. In *ICML*, 2006.

Lazaric, A. and Ghavamzadeh, M. Bayesian multi-task reinforcement learning. In *ICML*, 2010.

Mataric, M. J. Reward functions for accelerated learning. In *Machine Learning Proceedings 1994*, pp. 181–189. Elsevier, 1994.

Minsky, M. Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30, 1961.

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

Ng, A. Y., Harada, D., and Russell, S. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, 1999.

OpenAI. Openai five blog. https://blog.openai.com/openai-five/, 2018. Posted: 2018-06-25.

Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T. Meta-learning with memory-augmented neural networks. In *ICML*, 2016.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Skinner, B. F. *The behavior of organisms: An experimental analysis*. BF Skinner Foundation, 1990.

Smith, L. B. and Slone, L. K. A developmental approach to machine learning? *Frontiers in psychology*, 8(2124), 2017.

Snel, M. and Whiteson, S. Multi-task evolutionary shaping without pre-specified representations. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, 2010.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 1998.

Tamar, A., Wu, Y., Thomas, G., Levine, S., and Abbeel, P. Value iteration networks. In *NeurIPS*, 2016.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. Matching networks for one shot learning. In *NeurIPS*, 2016.

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

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. Dueling network architectures for deep reinforcement learning. In *ICML*, 2016b.

Watkins, C. J. and Dayan, P. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Wilson, A., Fern, A., Ray, S., and Tadepalli, P. Multi-task reinforcement learning: a hierarchical bayesian approach. In *ICML*, 2007.

Wu, B., Fu, Q., Liang, J., Qu, P., Li, X., Wang, L., Liu, W., Yang, W., and Liu, Y. Hierarchical macro strategy model for moba game ai. In *AAAI*, 2019.

Wu, Y. and Tian, Y. Training agent for first-person shooter game with actor-critic curriculum learning. In *ICLR*, 2017.

Yoon, J., Kim, T., Dia, O., Kim, S., Bengio, Y., and Ahn, S. Bayesian model-agnostic meta-learning. In *NeurIPS*, 2018.

A. Proof of Thm. 3.2

Proof. (Part I) *Eqn.* (8) *optimizes for the optimal policy.*

First, note that Eqn. (8),

$$\phi_j \leftarrow \phi_j - \alpha \nabla_{\phi_j} \|R'_j(s,a,s') + \gamma \max_{a'} A_{\phi_j}(s',a') - A_{\phi_j}(s,a)\|^2,$$

is naturally minimizing an objective in the form of TD error:

$$\|R'_j(s,a,s') + \gamma \max_{a'} A_{\phi_j}(s',a') - A_{\phi_j}(s,a)\|^2. \tag{10}$$

Bearing in mind that in Alg. 2 we use $\Phi(s) = V_{\phi_j}(s)$ as the potential-based shaping function to obtain $R'_j(s,a,s')$, we can rewrite objective (10) as:

$$\begin{aligned}
&\|R'_j(s,a,s') + \gamma \max_{a'} A_{\phi_j}(s',a') - A_{\phi_j}(s,a)\|^2 \\
=&\|\big(R_j(s,a,s') + \gamma V_{\phi_j}(s') - V_{\phi_j}(s)\big) + \gamma \max_{a'} A_{\phi_j}(s',a') - A_{\phi_j}(s,a)\|^2 \\
=&\|R_j(s,a,s') + \gamma(\max_{a'} A_{\phi_j}(s',a') + V_{\phi_j}(s')) - (A_{\phi_j}(s,a) + V_{\phi_j}(s))\|^2 \\
=&\|R_j(s,a,s') + \gamma \max_{a'} Q_{\phi_j}(s',a') - Q_{\phi_j}(s,a)\|^2,
\end{aligned}$$

where $A_{\phi_j}(\cdot,\cdot) + V_{\phi_j}(\cdot) = Q_{\phi_j}(\cdot,\cdot)$ simply because it's neural-network computation of the dueling architecture.

Now we've already arrived at exactly the Q-learning TD error on the original MDP M_j:

$$\mathcal{L}_{\mathcal{T}_j}(Q_{\phi_j}) = \|R_j(s,a,s') + \gamma \max_{a'} Q_{\phi_j}(s',a') - Q_{\phi_j}(s,a)\|^2. \tag{11}$$

Therefore, Eqn. (8) is in essence minimizing the Q-learning TD error on M_j, thus optimizing for the optimal policy (invariant with/without the potential-based reward shaping).

Remark: As an alternative understanding, first note that $A_{\phi_j}(s,a)$ is just a notation for the neural-network head. If we view it as an estimator of $Q^*_{M'_j}(s,a)$, then Eqn. (8) is actually performing Q-learning on the shaped MDP M'_j, with objective (10) directly being the corresponding TD error. It is therefore still optimizing for the invariant optimal policy.

(Part II) *Eqn.* (9) *optimizes for the task-posterior* $V^*_{M_j}(s)$.

Let $\phi'_j = \arg\min_{\phi_j} \|R'_j(s,a,s') + \gamma \max_{a'} A_{\phi_j}(s',a') - A_{\phi_j}(s,a)\|^2$, i.e., assume Eqn. (8) optimizes to minimum the parameters that it has gradients on, and get ϕ'_j. Following the remark in Part I, we have

$$A_{\phi'_j}(s',a') = Q^*_{M'_j}(s,a) \tag{12}$$

from Q-learning on the shaped MDP M'_j.

We also rearrange Eqn. (1) with the adopted $\Phi(s) = V_{\phi_j}(s)$ to get:

$$Q^*_{M'_j}(s,a) + V_{\phi_j}(s) = Q^*_{M_j}(s,a). \tag{13}$$

Substituting Eqn. (12) into (13), we get:

$$Q^*_{M_j}(s,a) = Q^*_{M'_j}(s,a) + V_{\phi_j}(s) = A_{\phi'_j}(s,a) + V_{\phi_j}(s). \tag{14}$$

Note that by definition,

$$Q^*_{M_j}(s,a) = V^*_{M_j}(s) + A^*_{M_j}(s,a). \tag{15}$$

So we have

$$V^*_{M_j}(s) + A^*_{M_j}(s,a) = A_{\phi'_j}(s,a) + V_{\phi_j}(s). \tag{16}$$

Taking $\max_a$ on both sides of Eqn. (16), we get

$$\begin{aligned} V^*_{M_j}(s) + \max_a A^*_{M_j}(s,a) &= \max_a A_{\phi'_j}(s,a) + V_{\phi_j}(s) \\ V^*_{M_j}(s) &= \max_a A_{\phi'_j}(s,a) + V_{\phi_j}(s), \end{aligned} \tag{17}$$

where $\max_a A^*_{M_j}(s,a) = \max_a Q^*_{M_j}(s,a) - V^*_{M_j}(s) = 0$ holds by definition.

In this way, we transform the inaccessible $V^*_{M_j}(s)$ into the computable $\max_a A_{\phi'_j}(s,a) + V_{\phi_j}(s)$, and to adapt $V_{\phi_j}(s)$ to the task-posterior $V^*_{M_j}(s)$ one should minimize

$$\| V_{\phi_j}(s)) - \texttt{stop\_gradient}\big(\max_a A_{\phi'_j}(s,a) + V_{\phi_j}(s) \big) \|^2, \tag{18}$$

where we stop the gradients because the latter part should be treated as a scalar learning target.

Therefore, Eqn. (9) is indeed optimizing for the task-posterior $V^*_{M_j}(s)$.

Remark: Here we assume ϕ_j is optimized to the final ϕ'_j. In practice this is not necessary nor desired, preventing fast adaptation. Therefore, we take only one step of Eqn. (8), and alternate between one step of Eqn. (8) and one step of Eqn. (9), where one pair constitutes one update step in Fig. 3 and 5 (left).

Also note that A_{ϕ_j} and V_{ϕ_j} may or may not share parameters, and ϕ'_j only corresponds to the parameters that Eqn. (8) has gradients on, so we keep the separate notations of ϕ_j and ϕ'_j. From the above derivation, we can see that Eqn. (17) holds for arbitrary ϕ_j, so nothing is violated if some parameters of ϕ_j is updated by Eqn. (8). □