Learning Task-Distribution Reward Shaping with Meta-Learning

Haosheng Zou¹, Tongzheng Ren², Dong Yan¹, Hang Su¹, Jun Zhu¹*

¹Dept. of Comp. Sci. & Tech., Institute for AI, BNRist Lab, Bosch-Tsinghua Joint ML Center, Tsinghua University
²Department of Computer Science, UT Austin

zouhaosheng@163.com, (rtz19970824, sproblvem)@gmail.com, (suhangss, dcszj)@tsinghua.edu.cn

Abstract

Reward shaping is one of the most effective methods to tackle the crucial yet challenging problem of credit assignment and accelerate Reinforcement Learning. However, designing shaping functions usually requires rich expert knowledge and hand-engineering, and the difficulties are further exacerbated given multiple tasks to solve. In this paper, we consider reward shaping on a distribution of tasks that share state spaces but not necessarily action spaces. We provide insights into optimal reward shaping, and propose a novel meta-learning framework to automatically learn such reward shaping to apply on newly sampled tasks. Theoretical analysis and extensive experiments establish us as the state-of-the-art in learning task-distribution reward shaping, outperforming previous such works (Konidaris and Barto 2006; Snel and Whiteson 2014). We further show that our method outperforms learning intrinsic rewards (Yang et al. 2019; Zheng et al. 2020), outperforms Rainbow (Hessel et al. 2018) in complex pixel-based CoinRun games, and is also better than hand-designed reward shaping on grid mazes. While the goal of this paper is to learn reward shaping rather than to propose new general meta-learning algorithms as PEARL (Rakelly et al. 2019) or MQL (Fakoor et al. 2020), our framework based on MAML (Finn, Abbeel, and Levine 2017) also outperforms PEARL / MQL, and could combine with them for further improvement.

1 Introduction

Credit assignment (Minsky 1961) remains a key challenge on the learning efficiency (sample complexity) of Reinforcement Learning (RL) (Mnih et al. 2015; Silver et al. 2018), which usually requires many samples from the environment to learn well. Reward shaping is one of the most intuitive, popular and effective remedies to accelerate RL (Dorigo and Colombetti 1994; Mataric 1994; Randlov and Alstrom 1998). It also played an important role in recent successes like Doom (Wu and Tian 2017) and Dota (OpenAI et al. 2019).

However, most shaping functions are hand-designed by human experts and are potentially tedious and inconvenient to code in especially complex environments (Wu and Tian 2017; OpenAI et al. 2019). Direct shaping functions may also change the optimal policies (Ng, Harada, and Russell 1999). Furthermore, in practice we are usually interested in solving multiple similar tasks as a whole (e.g., training one general agent for all possible 2D mazes instead of one per map). Such shared but not identical task-structures naturally induce a distribution over tasks, such as distributions over maze configurations (Wilson et al. 2007), system parameters for different robot-hand sizes (Lazaric and Ghavamzadeh 2010) and game maps for RTS games (Jaderberg et al. 2019). The ability to quickly solve new similar tasks drawn from such distributions is mastered by human infants quite young (Smith and Slonc 2017). However, the human effort in reward shaping for RL would be further exacerbated as we have to either design a different shaping per task or come up with a general shaping function presumably harder to design.

Motivated by such inconvenience under task multiplicity, we study the generally hard problem of learning automatic reward shaping on a distribution of tasks (termed learning task-distribution reward shaping). Our goal is not to propose new general meta-learning algorithms, but instead use them to learn good shaping. To the best of our knowledge, the only previous such works are (Konidaris and Barto 2006; Snel and Whiteson 2014), both using naive objectives and simple models. They first learn optimal Q-/V-values for each task and then fit one potential function to all the learned values. They have to learn as many value functions as tasks in the first step. More importantly, some tasks’ value functions might be in disagreement with others’ and would be inappropriately averaged out in the second step. Besides, its training doesn’t prepare for adaptation to individual tasks at all. To mitigate those problems, our main novelties are an overall framework that formulates task-distribution reward shaping as meta-learning, and an original algorithm for reward shaping in meta-testing. Our contributions include: 1) we present a state-of-the-art method using meta-learning and deep nets

*J.Z is the corresponding author.

Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.
to learn a general and flexible task-distribution reward shaping, requiring only shared state space and applicable to new tasks either directly or adaptively; 2) we provide insightful theory, and extensive empirical improvements over strong competitors (highlighted in Fig. 1), including Rainbow (Hessel et al. 2018), MQL (Fakoor et al. 2020), hand-designed shaping and learning intrinsic rewards (Zheng et al. 2020).

2 Preliminaries

We consider the setting of multi-task model-free RL, where the tasks follow a distribution \( p(T) \). Each sampled task \( T_i \sim p(T) \) is a standard Markov Decision Process (MDP) \( M_i = (S, A_i, T_i, \gamma, R_i) \), where \( S \) is state space, assumed to be shared by all tasks, \( A_i \) is the action space, \( T_i : S \times A_i \times S \to [0, 1] \) is the unknown state transition probability (hence model-free), \( \gamma \in [0, 1] \) is the discount factor and \( R_i : S \times A_i \times S \to \mathbb{R} \) is the reward function. Note we use the subscript \( i \) to denote potentially different action spaces \( A_i \), transition probabilities \( T_i \) and reward functions \( R_i \). We assume basic knowledge of optimality (value functions and (deep) Q-learning), and briefly introduce the techniques on which our method is based as follows.

2.1 Potential-based Shaping Function

A shaping function \( F \) transforms the original MDP \( M = (S, A, T, \gamma, R) \) into a shaped MDP \( M' = (S, A, T', \gamma, R' = R + F) \). An arbitrary \( F \) could change the original optimal policies. One particular form of \( F \), potential-based shaping function (Ng, Harada, and Russell 1999), keeps the optimal policies invariant (Policy invariance theorem, Appx. A):

**Definition 1** (Potential-based shaping function (Ng, Harada, and Russell 1999)). \( F : S \times A \times S \to \mathbb{R} \) is a potential-based shaping function if there exists a real-valued function \( \Phi : S \to \mathbb{R} \), such that \( \forall (s, a, s') \in S \times A \times S, \)

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

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

Ng, Harada, and Russell (1999) noted \( \Phi(s) = V^*_M(s) \) (the optimal state value) gives \( V^*_{M'}(s) \equiv 0 \) and leaves only the non-zero Q-values to learn. We provide further insightful analysis and motivate it in our method in §3.1.

2.2 Meta-Learning

Meta-learning has proved effective for task distributions in RL. (Duan et al. 2016; Wang et al. 2016a; Yu et al. 2019). It operates on two task sets: meta-training set \( \{T_i\}_{i=1}^N \) and meta-testing set \( \{T_j\}_{j=N+1}^{N+M} \), both drawn from the same task distribution \( p(T) \). One popular algorithm is Model-Agnostic Meta-Learning (MAML) (Finn, Abbeel, and Levine 2017):

In meta-training, MAML learns a parameterization \( \theta \) of a neural net \( f \), iterating in “meta-iterations” within which it samples a minibatch of tasks \( \{T_i\} \) and updates:

\[
\phi_i(\theta) \leftarrow \theta - \alpha \nabla_{\theta} \mathcal{L}_{T_i}(f_{\theta}),
\]

\[
\theta \leftarrow \theta - \beta \nabla_{\theta} \mathbb{E}_{T_i} \mathcal{L}_{T_i}(f_{\phi_i(\theta)}),
\]

Here \( \theta \) are the parameters to be learned, \( \phi_i \) are the task-specific parameters updated from \( \theta \) as initialization (Eqn. 1).

3 Methods

We propose a novel framework (depicted in Fig. 2) to learn task-distribution reward shaping. It casts task-distribution reward shaping onto meta-learning and learns a potential function \( \Phi(s) \) for reward shaping on newly sampled tasks. In essence, we would parameterize \( \Phi(s) \) as a dueling-DQN (Wang et al. 2016b) and meta-learn its parameter initialization during meta-training. We then propose a novel algorithm to adapt from the learned initialization for reward shaping during meta-testing.

We generally follow the notations in [1] and summarize the main notations to appear for reference in Table 1 in Appendix. Since we seek to extract prior knowledge from meta-training tasks \( \{M_i\} \) into \( \Phi(s) \), we call \( \Phi(s) \) the prior (also following recent works (Yoon et al. 2018; Finn, Xu, and Levine 2018)). Correspondingly, in meta-testing we adapt it towards the task-posterior \( \Phi(s|T_i) \) for better shaping. We parameterize the prior with \( \theta \): \( \Phi(s; \theta) \), and the task-posterior with \( \phi_i : \Phi(s; T_i; \phi_i) \).

We use the same letter \( \Phi \) for both prior and posterior because they would have the same neural network architecture though different parameter values. We use the same parameters \( \theta, \phi_i \) in equations like \( Q_{\phi_i}(s, a) = V_{\phi_i}(s) + A_{\phi_i}(s, a) \) for simplicity due to the dueling architecture and shared parameters. These would become clear later.

We first motivate our choice of the task-posterior \( \Phi(s|T_i) \).

\footnotetext[1]{To be more specific, its V-value head would serve as the potential function.}

\footnotetext[2]{We implement them as ordinary neural networks rather than distributions for optimization simplicity. We leave a fuller Bayesian treatment than such point estimates for future work.}
Algorithm 1 Meta-learning potential function prior

**Input:** $p(T)$: a distribution over tasks; $\alpha$, $\beta$: step sizes

**Output:** Learned prior MD3QN $Q_\theta$

Randomly initialize parameters $\theta$ for prior $Q_\theta$

for meta_iteration $= 0, 1, 2...$ do

Sample a mini-batch of tasks $\{T_i\} \sim p(T)$

for all $T_i$ do

Initialize replay buffer $D_i$
Collect data with $Q_\theta(s, a)$ (e-greedy) into $D_i$

$\phi_i(\theta) \leftarrow \theta - \alpha \nabla_{\theta} \mathcal{L}_{T_i}(Q_\theta(s)|\theta)(\text{Eqn. } 3) \text{ with } D_i$

end for

$\theta \leftarrow \theta - \beta \nabla_{\theta} \mathbb{E}_{T_i} \mathcal{L}_{T_i}(Q_{\phi_i}(\theta))(\text{Eqn. } 4)$, all $\{D_i\}$

end for

Algorithm 2 Meta-testing (adaptation with advantage head)

**Input:** $T_j$: new task to solve; meta-learned MD3QN $Q_\theta$

**Output:** adapted task-posterior $\phi_j$

Initialize $\phi_j \leftarrow \theta$ and replay buffer $D$

for gradient_step $= 0, 1, 2...$ do

Collect data with $A_{\phi_j}(s, a)$ into $D$

Update $A_{\phi_j}(s, a)$ with Eqn. (10), using samples from $D$ and the current potential function $V_{\phi_j}(s)$ for shaping

Update $V_{\phi_j}(s)$ with Eqn. (11), using samples from $D$

end for


3.1 Efficient Credit Assignment with Optimal Potential Functions

We provide more insights on the potential function $\Phi(s) = V_M^*(s)$ than previous works (e.g. Ng, Harada, and Russell 1999; Wiewiora 2003):

**Theorem 1** (Dependency Removal, proof in Appx. 3). For all states, $\Phi(s) = V_M^*(s)$ shaping gives $R(s, a) = \mathbb{E}_{s'} R'(s', a, s') \leq 0$ (non-positive immediate rewards) with optimal actions’ immediate rewards being exclusively 0.

It essentially removes long-term dependencies along trajectories, since immediate rewards (zero or negative) are sufficient to determine optimal policies. There’s no need to consider future rewards at the current state, and correspondingly there would be no need of credit assignment (attributing future results to past actions), which is arguably the optimal credit assignment for learning efficiency.

Therefore, we choose $V_M^*(s)$ as the adaptation target of the task-posterior $\Phi(s|T_i)$ to be used in Eqn. (4).

**Remarks:** 1) This only holds in expectation of environmental randomness, and has to be approximated by mini-batches of samples. Still, $V_M^*(s)$ is a much appealing choice. 2) Knowing $V^*$ seems equivalent to having solved the task partly ($V^*$ still cannot give policies in the model-free setting), but we are not using it for the same task in a circle: $V_M^*(s)$ serves as the learning target on meta-training tasks $\{M_i\}$ in order to better generalize and approximate it for shaping on meta-testing tasks $\{M_j\}$.

3.2 Meta-Learning Potential Function Prior

Having set the adaptation update (Eqn. 1), we now consider how to learn the prior (Eqn. 2) taking into account the adaptation and reward shaping. We would arrive at Alg. 1.

We design $\Phi(s; \theta)$ and $\Phi(s|T_i; \phi_i)$ to be of the same network architecture (hence both $\Phi$). For each $T_i$, $\phi_i$ is initialized to $\theta$ and adapts to $V_M^*(s)$ under some loss function $\mathcal{L}_{T_i}$:

$$\phi_i(\theta) \leftarrow \theta - \alpha \nabla_{\theta} \mathcal{L}_{T_i}(\Phi(s|T_i; \phi_i))$$

(3)

Then from Eqn. 2, our objective for the prior is:

$$\min_{\theta} \mathbb{E}_{T_i} \mathcal{L}_{T_i}(\Phi(s|T_i; \phi_i(\theta)))$$

(4)

In contrast, previous works of learning task-distribution reward shaping (Konidaris and Barto 2006; Snell and Whiteson 2014) use $\min_{\theta} \mathbb{E}_{T_i} \mathcal{L}_2^Q(Q_\theta)$, which is essentially different and we refer to as NAIVE AVERAGING. Regarding meta-learning, this is actually the “pretraining one network on all training tasks” baseline in Finn, Abbeel, and Levine (2017), where it’s already shown to be inferior to MAML.

**Choice of $L_{T_i}$ and $\Phi$:** The closest off-the-shelf RL way to learn $V_M^*(s)$ is arguably (deep) Q-learning (Watkins and Dayan 1992; Mnih et al. 2015), which learns $Q_M^*(s, a)$ and gives $V_M^*(s)$ simply with $V_M^*(s) = \max_a Q_M^*(s, a)$.

Therefore, we use a deep Q-network (DQN) (Mnih et al. 2015) parameterized as $Q_\theta$ and $Q_{\phi_i}$, for the potential:

$$\Phi(s|T_i; \phi_i) = \max_a Q_{\phi_i}(s, a)$$

(5)

Correspondingly, $L_{T_i}$ is the (deep) Q-learning objective on $Q$ instead of $\Phi$:

$$L_{T_i}^Q(Q_{\phi_i}) = \mathbb{E}_{D_i}(R(s, a, s') + \gamma \max_a Q_{\phi_i}(s', a) - Q_{\phi_i}(s, a))^2$$

computed on some sampled (off-policy) trajectory data $D_i$.

**Remark:** We could also use modern policy-gradient algorithms (Mnih et al. 2016; Schulman et al. 2017) that learn $V^*(s)$ in the meantime of optimizing the policy $\pi(s)$, although they don’t directly learn $V^*(s)$ and $V^*(s) \rightarrow V^*(s)$ only when $\pi(s) \rightarrow \pi^*(s)$, which may require more data to learn. Therefore, we only consider DQN variants in this paper, and leave policy-gradients (e.g., off-policy actor-critic (Lillicrap et al. 2015) for future work.

**DQN Design:** Taking $\max$ (Eqn. 3) in vanilla DQNs isn’t very robust to approximation error and noise, and the learned Q-values are sometimes not necessary good estimates of true values (Wang et al. 2016b; Tallec, Blier, and Olivier 2019). So we further introduce a key design of decomposed multi-head architecture (also justified in §5):

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

(6)

where we’d like the advantage head $A_{\phi_i}(s, a)$ to learn the advantage-value function. This explicit decomposition gives the arguably directest way to learn $V^*$:

Firstly, note an issue of identifiability in Eqn. 3 that an arbitrary bias could be added to $V_{\phi_i}$ and subtracted from $A_{\phi_i}$ while keeping $Q_{\phi_i}$ unchanged. Therefore, the final design is to subtract the maximum of the advantage value from $Q_{\phi_i}$:

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

(7)

Then as $Q_{\phi_i}$ attains $Q_{\phi_i}^*$, by taking $\max_a$ on both sides of Eqn. 7, we get:

$$V_{\phi_i}(s) = \max_a Q_{\phi_i}^*(s, a) = V^*(s)$$

So we can directly learn $V^*$ in the meantime of learning $Q^*$.
This decomposition directly mimics and is justified by the value functions’ relation \( Q^{\pi_i}_{\lambda_i}(s, a) = V_{\lambda_i}(s) + A^{\pi_i}_{\lambda_i}(s, a) \), by definition for any policy \( \pi \) and MDP \( M \). It was first introduced as dueling DQN (Wang et al. 2016b) but for a different purpose of speeding up training (“generalize learning across actions” (Wang et al. 2016b)). Here we exploit the architecture in learning the optimal \( V \)-values \( V^* \) as we meta-train at the combined \( Q(s, a) \) (Fig. 2).

Now the potential function becomes \( \Phi(s|T_i; \phi_i) = V_{\phi_i}(s) \), i.e., the \( V(s) \) head as shown in Fig. 2.

The final algorithm is as Alg. 1. Eqn. 3 and 4 become:

\[
\phi_i(\theta) \leftarrow \theta - \alpha \nabla_{\theta} L_{T_j}^Q(Q_{\theta}),
\]

(8)

\[
\min_{\theta} \mathbb{E}_{T_i} L_{T_j}^Q(Q_{\phi_i}(\theta)) = \min_{\theta} \mathbb{E}_{T_i} L_{T_j}^Q(Q_{\phi_i}(\theta)).
\]

(9)

Roughly, Alg 1 extends MAML and dueling-DQN for our purposes. However, since DQN is off-policy RL, we could re-use experience and require much less experience every meta-iteration than the original (on-policy) policy-based MAML (Finn, Abbeel, and Levine 2017), which re-samples data in each environment with the task-posterior \( \phi_i \) for the update in Eqn. 8.

In practice, we also incorporate value-estimation corrections Double DQN (Van Hasselt, Guez, and Silver 2016) and multi-step returns (Hessel et al. 2018), resulting in multi-step Dueling Double DQN (mDDQN). A baseline without dueling is thus mDDQN. It’s non-trivial to combine other parts of Rainbow (Hessel et al. 2018) with meta-learning, and meta-learning the simpler mDDQN already outperforms RAINBOW + NAIVE AVERAGING (S3.3).

### 3.3 Meta-Testing with Potential Function Prior

Naturally we would like to adapt the \( \theta \)-initialized shaping \( V_{\phi_0}(s) | \phi_0 = \theta \) towards \( V_{\lambda_i}(s) \) when learning on a meta-testing task \( T_j \). To facilitate this, we made an additional assumption first that the action space is shared across the task distribution (we’ll drop it and handle the unshared case later), so the mDDQN is still applicable on \( T_j \). We call this meta-testing setting adaptation with advantage head, as later in the unshared case we would discard the advantage head.

We’d like an algorithm that 1) utilizes the meta-trained mDDQN and 2) still does RL on the shaped \( M_j \) and 3) adapts the shaping function \( V_{\phi_0}(s) \) in a simultaneous but also clear, disentangled manner. We propose Alg. 2 which updates \( A_{\phi_0} \) and \( V_{\phi_0} \) alternately with stepsizes \( \alpha \):

- Update \( A_{\phi_0}(s, a) \) with sampled data from replay buffer:

\[
\phi_0 \leftarrow \phi_0 - \alpha \nabla_{\phi_0} \mathbb{E}_{D} \left[-R_j(s, a, s') + \gamma \max_{a'} A_{\phi_0}(s', a') - A_{\phi_0}(s, a)\right]^2.
\]

(10)

- Update \( V_{\phi_0}(s) \) towards target \( y \) with sampled data from replay buffer (where \( y = \max_{a} A_{\phi_0}(s, a) + V_{\phi_0}(s) \)):

\[
\phi_0 \leftarrow \phi_0 - \alpha \nabla_{\phi_0} \mathbb{E}_{D} \left[ V_{\phi_0}(s) - y \right]^2.
\]

(11)

They update intermediate \( A(s, a) \) and \( V(s) \) heads (Fig. 2).

**Theorem 2** (Meta-Testing Objective, proof in Appx. C). Eqn. 10 optimizes for the optimal policy. Eqn. 11 optimizes for the task-posterior \( V_{\lambda_i}(s) \).

We now return to unshared action spaces, which could arise in practice simply because the task requires a different action space (e.g., from discrete to fine-grained continuous spaces in S5), or because of other constraints forbidding the advantage head. In this case (which we call shaping only), we simply discard the advantage head and fix the meta-learned \( V_0 \) for reward shaping to train a new RL agent from scratch on \( T_j \). Due to the close (though nonequivalent) relation between Alg 1 and Q-learning, \( V_0 \) could still be expected to be much better than random potential functions or no shaping at all, which we empirically verify in Appendix.

### Meta-testing without reward shaping

In case of shared action space, one may wonder: why not simply do MAML’s meta-testing (Finn, Abbeel, and Levine 2017), just Q-learning starting from \( \theta \)? We refer to this baseline as VANILLA MAML. Though viable, VANILLA MAML merely exploits the meta-learned parameter initialization, while our Alg. 1 also explicitly exploits reward shaping from the potential function prior. The shaped rewards are easier for policy learning, and the adapting shaping (Eqn. 11) further boosts policy learning (Eqn. 10) immediately in the next loop. It also jointly does RL and adapts \( V_{\phi_0} \) in a much more clear, disentangled manner. These are empirically supported by the faster and more stable curves of our method in §5.3. Also, note that we are more general on action spaces, while VANILLA MAML requires a shared one.

### 4 Related Work

#### Learning task-distribution reward shaping

The only direct previous works are (Konidaris and Barto 2006; Snel and Whiteson 2014). They do consider multi-task transfer, but are based on naive (weighted) averaging in principle (whose downsides are discussed in §1) and only learn simple linear models on simple tasks. We include them in §5 as all NAIVE AVERAGING baselines with our deep models.

**Automatic reward shaping**: Potential-based reward shaping with policy invariance guarantee (Ng, Harada, and Russell 1999) have been extended to various settings (Asmuth, Littman, and Zinkev 2008; Lu, Schwartz, and Givigi 2011; Gao and Tom 2015; Eck et al. 2016; Mannion, Duggan, and Howley 2017; Grees and applications (Devlin, Grzes, and Kudenko 2011; Efthymiadis and Kudenko 2013). The exact form of the potential function could also be generalized to depend on action and/or timestep (Wiewiora, Cotrell, and Elkan 2003; Devlin and Kudenko 2012; Harutyunyan et al. 2015) in addition to state. Here we simply stick with its original form in (Ng, Harada, and Russell 1999). Potential-based shaping functions are usually pre-defined manually. Earlier attempts of automatic reward shaping rely on pre-defined representations (Konidaris and Barto 2006; Marthi 2007), pre-defined higher-level shaping function (Laud and DeJong 2002) or evolutionary search of shaping functions (Niekkum, Barto, and Spector 2010). Above all, those works are almost entirely based on tabular RL, lacking the ability to generalize/adapt. One recent related work on a task distribution is (Jaderberg et al. 2019), but they still learn in a tabular space.
way on a limited set of predefined novel states of their task.
In contrast to all those works, our method is quite general, assuming no task knowledge or heuristics, with a more general, principled meta-learning objective, flexible application settings, insightful theoretical analysis, and both neural agents and shaping functions with gradient optimization.

Another line of work is learning intrinsic rewards which learns the reward itself for policy optimization. It replaces the original reward and is fundamentally different from reward shaping, and policy invariance are usually only guaranteed asymptotically. Some learn on the program/code level (Alet et al. 2020). Some are for single tasks (Zheng, Oh, and Singh 2018; Stadie, Zhang, and Ba 2020; Du et al. 2019). Works comparable with ours on task distributions (Yang et al. 2019; Zheng et al. 2020) are built on on-policy RL (see §3.3).

5 Experiments

This sections seeks to answer two key questions: 1) what exactly is being learned and 2) how our framework compares to relevant baselines. We experimented on (of increasing complexity) CartPoles, grid mazes and CoinRun games through comparison with the only previous baselines (Konidaris and Barto 2006; Snel and Whiteson 2014) and other strong competitors (incl. RAINBOW (Hessel et al. 2018), hand-designed shaping, MQL (Fakoor et al. 2020) and intrinsic rewards (Zheng et al. 2020)). As a reminder, our method (“OURS”) meta-trains a mD3QN with Alg. 1 and meta-tests with Alg. 2 in the adaptation with advantage head (in short, adaptation) case or with the meta-learned $V_b$ in the shaping only case. We’ve included further experimental details and results in Appendix.

5.1 General Settings

We obtain each result over 5 different runs (random seeds 0–4). We sample, separate and fix the meta-training and meta-testing tasks at the beginning of each run, keeping all hyper-parameters the same. For fair comparison, meta-training of all methods uses same number of environment interactions and updates. For each meta-testing curve (# gradient steps as the x-axis in e.g., Fig. 4), we keep all data collection hyperparameters the same across all methods (e.g., # environment interactions per update) so that the number of gradient steps accurately mirrors sample complexity. For CartPoles and grid mazes, we aggregate raw performance statistics on all meta-testing tasks combined (so e.g., 5 × 40 = 200 tasks on CartPoles as follows), and then take median and interquartile curves. For CoinRun we take mean and standard deviation of all runs’ solve ratios following (Cobbe et al. 2019).

The task distribution on CartPoles is defined varying the pole length in the range of $[0.25, 5.00]$. Such change in dynamics is already harder than random noise. The mD3QN uses an MLP with two hidden layers of size 32 before the value heads. We meta-train on 500 sampled tasks for 200 meta iterations with 10 tasks per iteration and meta-test on 40 unseen tasks.

Figure 3: Visualization of meta-learned prior $V_b$ (right) on the left map at all possible agent positions.

The task distribution on grid mazes is defined on all possible maps of size $8 \times 8$, which is of exponentially many configurations varying start, goal and obstacles. The mD3QN uses a CNN similar as (Mnih et al. 2015) before the value heads. We meta-train on 1000 sampled maps for 1000 meta iterations and meta-test on 40 unseen maps. Pixel grid mazes is already non-trivial for non-tabular agents (Tamar et al. 2016).

The task distribution on CoinRun games (Cobbe et al. 2019) is defined on all possible level configurations. As Fig. 4 it’s a very difficult pixel game. The mD3QN uses pixel input and is also a CNN. We meta-train on 2000 generated levels for 1500 meta iterations. Meta-testing is done on a held-out set of 1000 levels.

Computationally, meta-learning is ~20% slower than non-meta-learning baselines due to the two-level MAML update, but on par with the more complex Rainbow (Hessel et al. 2018). Meta-testing is similar for all methods, only a bit slower than no shaping.

5.2 Qualitative Investigation of Learned Shaping

A natural question is what exactly is being learned in meta-training. We therefore plot the meta-learned prior, $V_b$, on one randomly sampled unseen grid maze (from start (S) to goal (G)) in Fig. 5 by iterating the agent location over all viable grid positions to get the value prediction for each state. We can see that the prior learns to generalize over state representations: plotted values overall match true values under the setting of +1 reward at goal, 0 reward elsewhere and discount factor 0.99. Although the map configuration varies across the task distribution, the representations of the start, goal, obstacles and agent positions are the same and meta-training could generalize over the detailed representations.

The prior (Fig. 5) is good for shaping only, but are not perfect (so could benefit from adaptation): they discount more rapidly than 0.99, and not all positions with same distance to goal have same values, especially farther away from goal. These match our analysis in §3.3. We also observe that some baselines we consider could get similar visualizations but with poorer quality.

On CoinRun we observe that the learned shaping is more complex since the game requires more maneuver, but it generally tends to encourage agent movement towards goal.

Another natural question is: is our learned shaping better than pure random shaping, since optimal policies are preserved under arbitrary potential-based reward shaping? we ablate with two random baselines to prove the effectiveness of our learned shaping:

- **Random Noise Shaping**: generate random noise and substitute the reward shaping whenever needed.
- **Random Network Shaping**: reward shaping with a
We experiment on grid mazes. Expectedly, the random baselines learn very poorly in both the adaptation and shaping only cases. The median returns remain zero until the same update number as of Fig. 4 (grid maze column). This proves that our learned shaping which generalizes over state representations does provide meaningful rewards.

5.3 Meta-Testing Performance

The key evaluation is learning curve comparison in both cases of meta-testing. We consider mostly the baselines emerging from §3 with their difference from OURS as follows:

- **NAIVE AVERAGING** (Konidaris and Barto 2006; Snell and Whiteson 2014): its prior-learning objective is $\min_\theta E_T \mathcal{L}_{\theta}^T (Q_{\theta})$ instead of Eqn. (9) as discussed in §3.2. Its original version needs to first learn the same number of value functions as the number of tasks, which does not scale well to large number of tasks. We instead merge its two steps and directly train an RL agent in all meta-training tasks, which is theoretically equivalent. This is realized by simply using only one replay buffer for all tasks and do conventional deep Q-learning with it. (Cobble et al. 2019) studies the same approach on CoinRun but with different setting details.

- **RAINBOW + NAIVE AVERAGING**: same as NAIVE AVERAGING but with better agent design (Hessel et al. 2018).

- **MDDQN**: taking away only the dueling part; to be compatible with Alg. 2 we further introduce $V_\phi(s) = \max_a Q_\phi(s, a)$ and $A_\phi(s, a) = Q_\phi(s, a) - V_\phi(s)$.

- **VANILLA MAML**: (only in adaptation case) adapts without shaping as (Finn, Abbeel, and Levine 2017).

- **POLICY-BASED MAML**: (only in adaptation case) the original MAML-RL (Finn, Abbeel, and Levine 2017), which is policy-based instead of value-based.

- **NoRML**: (only in adaptation case) as (Yang et al. 2019), does the inner update of POLICY-BASED MAML with a learnable advantage function without external rewards.

- **LIRPG-META**: (only in shaping only case) as (Zheng et al. 2020), meta-learns an intrinsic reward function that replaces external rewards (more details in Appendix).

Remark: The last three baselines (from POLICY-BASED MAML) are all essentially on-policy. They uses REINFORCE (Williams 1992) / TRPO (Schulman et al. 2015) / PPO (Schulman et al. 2017) for Eqn. (1) and Eqn. (2). In our experiments, they require more times the amount of training to achieve comparable performance. For clarity we exclude them from the main plots and leave them in the Appendix.

We meta-train all methods following §5.1 and plot meta-testing performances in Fig. 4. In shaping only case, training identically randomly-initialized RL agents, we also plot training without any reward shaping (“NO SHAPING”). Note that oscillation could not be completely avoided since it’s to some extent inherent to off-policy RL algorithms (Q-learning here).

Across all task distributions and both cases, OURS achieves state-of-the-art in learning task-distribution reward shaping, outperforming previous such works (Konidaris and Barto 2006; Snell and Whiteson 2014). Comparison with MDDQN and VANILLA MAML justifies our dueling design and Alg. 2 reward shaping respectively. We reiterate that pixel-based grid mazes is already non-trivial for non-tabular agents (Tamar et al. 2016), let alone CoinRun.

Note that in our setting the amount of meta-training experience and update steps per task on average are quite limited compared to conventional deep RL. This may be another reason why NAIVE AVERAGING performs poorly, and RAINBOW + NAIVE AVERAGING improves it only to some extent.
We let it take as input actions of continuous values. This also test under unshared action space. We model this further experimented on the well-studied domain for meta-learning part, MAML, is at least not much better than PEARL and MQL, our better result could only be attributed mainly to the reward shaping part. On the other hand, both PEARL and MQL are boosted when combined with ours, with ours + MQL finally better than ours. This again justifies our reward shaping. Again we note that our goal is not to propose a new general meta-learning algorithm as PEARL / MQL, but to learn task-distribution reward shaping with meta-learning.

5.6 PEARL and MQL on CoinRun

In the adaptation case, we further consider PEARL (Rakelly et al. 2019) and MQL (Pakoo et al. 2020) that are shown to outperform MAML on some but not all tasks (Yu et al. 2019). Generally, they both use the NAIVE AVERAGING (multi-task) objective and condition the RL agent on episode history so the problems of the naive objective are mitigated. The following comparison and combination with ours are considered:

- PEARL, MQL: their meta-training/testing algorithms but substituting their RL part with mDQN.
- ours + PEARL, ours + MQL: our Alg. 1 & 2 but substituting our meta-learning part with PEARL / MQL. Meta-training is the same as the above PEARL / MQL respectively (but not meta-testing, more details in Appendix).

We again meta-train all methods fairly following §5.1 and show meta-testing results in Fig. 5c. We outperform almost all the methods. On one hand, we surpass PEARL and MQL, though MQL is indeed competitive. Since generally our meta-learning part, MAML, is at least not much better than PEARL and MQL, our better result could only be attributed mainly to the reward shaping part. On the other hand, both PEARL and MQL are boosted when combined with ours, with ours + MQL finally better than ours. This again justifies our reward shaping. Again we note that our goal is not to propose a new general meta-learning algorithm as PEARL / MQL, but to learn task-distribution reward shaping with meta-learning.

6 Discussions

Our method builds on meta-learning, and a common question about meta-learning is on what task distributions meta-learning could work (Yu et al. 2019). Although we build on MAML, our experiments already show we could generalize on task distributions more difficult than most simple distributions varying only reward functions in e.g. (Finn, Abbeel, and Levine 2017) (Rakelly et al. 2019): in our case the underlying transition/dynamics vary with different CartPole lengths and the grid and CoinRun configurations, which is essentially more diverse than simply varying target velocities/goal locations in one same environment (Finn, Abbeel, and Levine 2017) (Rakelly et al. 2019). However, as empirically tested (Yu et al. 2019), being built on MAML still suggests we couldn’t generalize well to ultimately essentially different tasks (“non-parametric variation” (Yu et al. 2019)), where e.g. the task distribution could cover all robot-hand pushing, sliding and pressing combined.

One implicit assumption of ours is no drastic change in reward scale (e.g., 100 times larger or smaller). However, reward scale also affects policy gradient methods (Henderson et al. 2018), so it’s not our specific limitation.

There are many other ways to transfer knowledge across a task distribution, e.g., learn environment dynamics (Clavera et al. 2019) and visual features (Finn et al. 2017). Here we focus on reward shaping and leave a comprehensive study as future work. We would also consider continuous action spaces in meta-training, and single-task reward shaping.

Acknowledgements

This work was supported by NSFC Projects (Nos. 62061136001, 61620106010, U19B2034, U1811461), Beijing NSF Project (No. L172037), Beijing Academy of Artificial Intelligence (BAAI), and the Xplorer Prize.
References


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


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


Marom, O.; and Rosman, B. 2018. Belief reward shaping in reinforcement learning. In AAAI.

Marthi, B. 2007. Automatic shaping and decomposition of reward functions. In ICML.


