How Should an Agent Practice?

Janarthanan Rajendran, Richard Lewis, Vivek Veeriah, Honglak Lee, Satinder Singh

University of Michigan

rjana@umich.edu, rickl@umich.edu, vveeriah@umich.edu, honglak@eecs.umich.edu, baveja@umich.edu

Abstract

We present a method for learning intrinsic reward functions to drive the learning of an agent during periods of *practice* in which extrinsic task rewards are not available. During practice, the environment may differ from the one available for training and evaluation with extrinsic rewards. We refer to this setup of alternating periods of practice and objective evaluation as *practice-match*, drawing an analogy to regimes of skill acquisition common for humans in sports and games. The agent must effectively use periods in the practice environment so that performance improves during matches. In the proposed method the intrinsic practice reward is learned through a meta-gradient approach that adapts the practice reward parameters to reduce the extrinsic match reward loss computed from matches. We illustrate the method on a simple grid world, and evaluate it in two games in which the practice environment differs from match: Pong with practice against a wall without an opponent, and PacMan with practice in a maze without ghosts. The results show gains from learning in practice in addition to match periods over learning in matches only.

Introduction

There are many applications of reinforcement learning (RL) in which the natural formulation of the reward function gives rise to difficult computational challenges, or in which the reward itself is unavailable for extended periods of time or is difficult to specify. These include settings with very sparse or delayed reward, multiple tasks or goals, reward uncertainty, and learning in the absence of reward or in advance of unknown future reward. A range of approaches address these challenges through *reward design*, providing *intrinsic* rewards to the agent that augment or replace the objective or *extrinsic* reward. The aim is to provide useful and proximal learning signals that drive behavior and learning in a way that improves performance on the main objective of interest (Ng, Harada, and Russell 1999; Barto, Singh, and Chentanez 2004; Singh et al. 2010). These intrinsic rewards are often hand-engineered, and based on either task-specific reward features developed from domain analysis, or task-general reward features, sometimes inspired by intrinsic mo-

tivations in animals and humans (Oudeyer and Kaplan 2009; Schmidhuber 2010) and sometimes based on heuristics such as learning diverse skills (Gupta et al. 2018). The optimal rewards framework (Singh et al. 2010) provides a general meta-optimization formulation of intrinsic reward design, and has served as the basis for algorithms that discover good intrinsic rewards; we discuss this further in Related Work.

In this work we address the challenges imposed by settings where a learning agent faces extended periods of no evaluation in which an extrinsic reward is unavailable and where the environment may differ from that of objective evaluation when extrinsic reward is available. We refer to such settings as *practice-match*, drawing an analogy to regimes of skill acquisition typical for humans in sports and games. For example, in team sports such as basketball it is common to practice skills such as dribbling and shooting in the absence of other players, and in sports such as tennis it is common to practice skills in environments other than a full court. In such settings, during practice, the agent must behave in the absence of the main match reward (e.g., winning games against opponents), but in such a way that performance on the future matches (defined by the extrinsic rewards during match) improves. Examples of practice-match settings beyond sports include an office robot using the evening after office-hours to practice for day-time tasks (match), household robotic assistants using free-time to practice, task-specific dialogue agents using down-time to practice with human-trainers or using opportunities for low-stakes on-line conversation practice, and multi-agent teams using down-time to practice coordination strategies.

We focus on the question of how an agent should practice given a practice environment in a setting of alternating periods of practice and match. We formulate this problem as one of *discovering good practice rewards*. Our primary contribution is a method that learns intrinsic reward functions for practice that improve the match policy during practice. The method uses meta-gradients to adapt the intrinsic practice reward parameters to reduce the extrinsic loss computed from matches. Our results show gains from learning in practice in addition to match periods over the performance achieved from learning in matches only.

Related work

We place our contributions in the context of three bodies of related work: (a) the design or discovery of intrinsic rewards that modify or replace an available extrinsic reward; (b) the design or discovery of intrinsic rewards to motivate learning and behavior in the absence of extrinsic reward; and (c) meta-gradient approaches to optimizing reinforcement learning agent parameters.

Optimal rewards and reward design. Reward functions serve as implicit specifications of desired policies, but the precise form of the reward also has consequences for the sample (and computational) complexity of learning. Approaches to *reward design* seek to modify or replace the extrinsic reward to improve the complexity of learning while still finding good policies. Approaches such as *potential rewards* (Ng, Harada, and Russell 1999) define a space of reward transformations guaranteed to preserve the implicit optimal policies. *Intrinsically-motivated RL* aims to improve learning by providing reward bonuses, e.g., to motivate effective exploration, often through hand-designed features that formalize notions such as curiosity or salience (Barto, Singh, and Chentanez 2004; Oudeyer and Kaplan 2009; Schmidhuber 2010). In contrast to this prior work, the practice reward discovery method proposed here does not commit to the form of the intrinsic reward and does not use hand-designed reward features. The *optimal rewards* framework of Singh et al. (2010) formulates a meta-optimization problem motivated by the insight that the optimal intrinsic reward for an RL agent depends on the bounds on the agent's learning algorithm and environment; algorithms exist for finding optimal intrinsic rewards for planning (Guo et al. 2016; Sorg, Lewis, and Singh 2010) and policy-gradient agents (Zheng, Oh, and Singh 2018). Our new work shares the meta-optimization framework of optimal rewards, but addresses the challenge of how to drive learning during periods of practice where extrinsic rewards are not available and the practice environment is different from the evaluation environment.

Learning in the absence of extrinsic reward. Recent work addresses the challenge faced by agents that must learn during a period of free exploration that precedes an objective evaluation in which the agent is tasked with a sequence of goals drawn from some distribution; the distribution parameters may be partially known to the agent in advance. This prior work includes methods for learning goal-conditioned policies via the automatic generation of a curriculum of goals (Florensa et al. 2018) or via information-theoretic loss functions (Eysenbach et al. 2018; Goyal et al. 2019). Gupta et al. (2018) generate tasks that lead to learning of diverse skills and use them to learn a policy initialization that adapts quickly to the objective evaluation. Our work shares with these approaches the challenge of motivating learning in the absence of extrinsic rewards, but differs in that our proposed practice reward method *discovers* intrinsic rewards through losses defined only in terms of an extrinsic reward, and the practice-reward setting concerns a single objective task and possibly different environments.

Meta-gradient approaches to optimizing RL agent parameters. Recently, researchers have developed several different meta-gradient approaches that optimize meta-parameters of a policy-gradient agent that affect the policy loss only indirectly through their effect on the policy parameters. For example, meta-gradient approaches have been used successfully to learn good policy network initializations that adapts quickly to new tasks (Finn, Abbeel, and Levine 2017; Rothfuss et al. 2018; Finn et al. 2017; Gupta et al. 2018), and RL hyper-parameters such as discount factor and bootstrapping parameters (Xu, van Hasselt, and Silver 2018). Zheng, Oh, and Singh (2018) developed a meta-gradient algorithm for discovering optimal intrinsic rewards for policy gradient agents. Our proposed method modifies and extends Zheng, Oh, and Singh (2018) to practice-match settings. Specifically, we derive the gradient of extrinsic reward loss during match with respect to practice reward parameters and use it to improve practice rewards over the course of alternating practices and matches. The success of the method thus contributes to the growing body of recent work demonstrating the utility of meta-gradient algorithms for RL.

Algorithm for learning practice rewards

In this section, we first describe briefly policy gradient-based RL and then our algorithm for learning practice rewards.

Policy gradient- based RL. At each time step t, the agent receives a state s_t and takes an action a_t from a discrete set A of possible actions. The actions are taken following a policy π (a mapping from states s_t to actions a_t), parameterized by θ and denoted as π_θ. The agent then receives the next state s_{t+1} and a scalar reward r_t. This process continues until the agent reaches a terminal state (which ends an episode) after which the process restarts and repeats.

Let $G(s_t, a_t)$ be the future discounted sum of rewards obtained by the agent until termination, i.e., $G(s_t, a_t) = \sum_{i=t}^{\infty} \gamma^{i-t} r(s_i, a_i)$, where γ is the discount factor. The value of the policy π_θ denoted by $J(\theta)$ is the expected discounted sum of rewards obtained by the agent when executing actions following the policy π_θ, i.e., $J(\theta) = E_\theta[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$. The policy gradient theorem of Sutton et al. (2000) shows that for all time steps t within an episode, the gradient of the value $J(\theta)$ with respect to the policy parameters θ can be obtained as follows:

$$\nabla_\theta J(\theta) = E_\theta[G(s_t, a_t)\nabla_\theta \log \pi_\theta(a_t|s_t)] \qquad (1)$$

Notation. We use the following notation throughout:

- θ : policy parameters
- $r^{ex} = r^{ex}(s,a)$: extrinsic reward (available during matches)
- $r_\eta^{in} = r_\eta^{in}(s,a)$: intrinsic reward parameterized by η
- $G^{ex}(s_t, a_t) = \sum_{i=t}^{\infty} \gamma^{i-t} r^{ex}(s_i, a_i)$: extrinsic reward return
- $G^{in}(s_t, a_t) = \sum_{i=t}^{\infty} \gamma^{i-t} r_\eta^{in}(s_i, a_i)$: intrinsic reward return
- $J^{ex} = E_\theta[\sum_{t=0}^{\infty} \gamma^t r^{ex}(s_t, a_t)]$: extrinsic value of policy
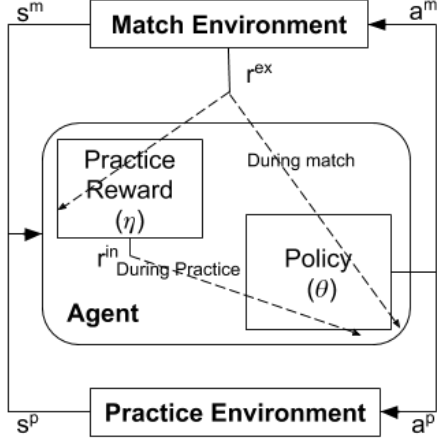- $J^{in} = E_\theta[\sum_{t=0}^{\infty} \gamma^t r_\eta^{in}(s_t, a_t)]$: intrinsic value of policy

Figure 1: The agent has two modules, the policy module parameterized by θ and the practice reward module parameterized by η. As the dashed lines show, θ is updated using extrinsic match reward r^{ex} during match and using the intrinsic practice reward r^{in} during practice; η is updated using the extrinsic match reward r^{ex} from the match environment.

Algorithm overview. The algorithm is specified in Algorithm 1 and the agent architecture is depicted in Figure 1. At each time step t the agent receives an observation from the environment and *concatenates the observation with a practice/match flag* indicating whether the agent is in practice or match. We denote this concatenated input as s_t^p for the practice environment and s_t^m for the match environment.

During *match*, the policy parameters are updated to improve performance in the match task as defined by the extrinsic reward; this happens by adjusting the policy parameters θ in the direction of the gradient of J^{ex}, which is the expected discounted sum of match time extrinsic rewards.

During *practice*, the policy parameters are updated to improve performance in the practice task as defined by the current intrinsic practice reward; this happens by adjusting θ in the direction of the gradient of J^{in}, which is the expected discounted sum of practice time intrinsic rewards.

After each practice update, the intrinsic practice reward parameters are updated in the key meta-gradient step. The aim is to adjust the intrinsic practice reward so that the policy parameter updates that result from practice improve the extrinsic reward performance on match. This is done by using match experience to evaluate the policy parameters that result from the practice update, and updating the intrinsic reward parameters η in the direction of the gradient of J^{ex} computed on the match experience. We explore two variants: updating based on the previous match experience, and updating based on the next match experience. We describe each step in detail below.

Our algorithm is a modification and extension of Zheng, Oh, and Singh (2018)'s algorithm (which discovers optimal intrinsic rewards for policy gradient agents in the regular RL setting) for practice-match settings and we follow their

derivations closely.

Updating policy parameters during match. Let $\mathcal{D}_m = \{s_0^m, a_0^m, s_1^m, a_1^m, \cdots\}$ be the trajectory taken by the agent in the match using the policy π_θ. The policy parameters θ are updated in the direction of the gradient of J^{ex}:

$$\theta' = \theta + \alpha^m \nabla_\theta J^{ex}(\theta; \mathcal{D}_m) \qquad (2)$$
$$\approx \theta + \alpha^m G^{ex}(s_t^m, a_t^m) \nabla_\theta \log \pi_\theta(a_t^m | s_t^m) \qquad (3)$$

using the empirical return G^{ex} in the approximation of the gradient.

Updating policy parameters during practice. Let $\mathcal{D}_p = \{s_0^p, a_0^p, s_1^p, a_1^p, \cdots\}$ be the trajectory taken by the agent in the practice environment using the policy $\pi_{\theta'}$. The policy parameters θ' are updated in the direction of the gradient of J^{in}:

$$\theta'' = \theta' + \alpha^p \nabla_{\theta'} J^{in}(\theta'; \mathcal{D}_p) \qquad (4)$$
$$\approx \theta' + \alpha^p G^{in}(s_t^p, a_t^p) \nabla_{\theta'} \log \pi_{\theta'}(a_t^p | s_t^p) \qquad (5)$$

using the empirical return G^{in} in the approximation of the gradient.

Updating intrinsic practice reward parameters. The intrinsic practice reward parameters η are updated in the direction of the gradient of J^{ex} of the match. The gradient of J^{ex} of the match with respect to η is computed using the chain rule as follows:

$$\eta' = \eta + \beta \nabla_\eta J^{ex}(\theta'') \qquad (6)$$
$$= \eta + \beta \nabla_\eta \theta'' \nabla_{\theta''} J^{ex}(\theta'') \qquad (7)$$

The second term $\nabla_{\theta''} J^{ex}(\theta'')$ evaluates the policy parameters θ'' (that resulted from the practice update using the intrinsic rewards) using match samples. We specify here two forms of the intrinsic practice reward update: when the match samples are from the *next* match, and when the match samples are from the *previous* match. If we use the next match to perform the update, the agent will act using the policy $\pi_{\theta''}$ in the next match and can use the new match samples from the trajectory $\mathcal{D}_{m'}$ to approximate $\nabla_{\theta''} J^{ex}(\theta'')$ as follows:

$$\nabla_{\theta''} J^{ex}(\theta'') = \nabla_{\theta''} J^{ex}(\theta''; \mathcal{D}_{m'})$$
$$\approx G^{ex}(s_t^{m'}, a_t^{m'}) \nabla_{\theta''} \log \pi_{\theta''}(a_t^{m'} | s_t^{m'}) \quad (8)$$

If we use the previous match samples, the agent can perform an off-policy update using an importance sampling correction:

$$\nabla_{\theta''} J^{ex}(\theta'') = \nabla_{\theta''} J^{ex}(\theta''; \mathcal{D}_m)$$
$$\approx G^{ex}(s_t^m, a_t^m) \frac{\nabla_{\theta''} \log \pi_{\theta''}(a_t^m | s_t^m)}{\log \pi_\theta(a_t^m | s_t^m)} \quad (9)$$

The first term $\nabla_\eta \theta''$ in Eq. 7 evaluates the effect of change in the intrinsic parameters η on the policy parameters that result after the practice time policy update, θ''. This term

Algorithm 1 Learning Practice Rewards
***
1: **Input:** step-size parameters α^m, α^p and β
2: **Init:** initialize θ and η with random values
3: **repeat**
4:     **Updating policy parameters during match**
5:     Sample a trajectory $\mathcal{D}_m = \{s_0^m, a_0^m, s_1^m, a_1^m, \cdots\}$ by interacting with the match environment using π_θ
6:     Approximate $\nabla_\theta J^{ex}(\theta; \mathcal{D}_m)$ by Equation 3
7:     Update $\theta' \leftarrow \theta + \alpha^m \nabla_\theta J^{ex}(\theta; \mathcal{D}_m)$
8:     **Updating policy parameters during practice**
9:     Sample a trajectory $\mathcal{D}_p = \{s_0^p, a_0^p, s_1^p, a_1^p, \cdots\}$ by interacting with the practice environment using $\pi_{\theta'}$
10:    Approximate $\nabla_{\theta'} J^{in}(\theta'; \mathcal{D}_p)$ by Equation 5
11:    Update $\theta'' \leftarrow \theta' + \alpha^p \nabla_{\theta'} J^{in}(\theta'; \mathcal{D}_p)$
12:    **Updating intrinsic reward parameters after practice update**
13:    Approximate $\nabla_{\theta''} J^{ex}(\theta'')$ by Equation 8 or 9
14:    Approximate $\nabla_\eta \theta''$ by Equation 13
15:    Compute $\nabla_\eta J^{ex} = \nabla_{\theta''} J^{ex}(\theta'') \nabla_\eta \theta''$
16:    Update $\eta' \leftarrow \eta + \beta \nabla_\eta J^{ex}$
17: **until** done
***

can be computed as follows:

$$\nabla_\eta \theta'' = \nabla_\eta \left(\theta' + \alpha^p G^{in}(s_t^p, a_t^p) \nabla_{\theta'} \log \pi_{\theta'}(a_t^p | s_t^p) \right) \tag{10}$$

$$= \nabla_\eta \left(\alpha^p G^{in}(s_t^p, a_t^p) \nabla_{\theta'} \log \pi_{\theta'}(a_t^p | s_t^p) \right) \tag{11}$$

$$= \nabla_\eta \left(\alpha^p (\sum_{i=t}^\infty \gamma^{i-t} r_\eta^{in}(s_i^p, a_i^p)) \nabla_{\theta'} \log \pi_{\theta'}(a_t^p | s_t^p) \right) \tag{12}$$

$$= \alpha^p \left(\sum_{i=t}^\infty \gamma^{i-t} \nabla_\eta r_\eta^{in}(s_i^p, a_i^p) \right) \nabla_{\theta'} \log \pi_{\theta'}(a_t^p | s_t^p) \tag{13}$$

For simplicity we have described our proposed algorithm using a basic policy gradient formulation. Our proposed algorithm is fully compatible with advanced policy gradient methods such as Advantage Actor-Critic that reduce the variance of the gradient and improve data efficiency.

Illustration on grid-world: Visualizing practice rewards

We now illustrate the algorithm in a simple grid world that allows us to visualize discovered practice rewards at different points in the agent's learning. The environment is a corridor world of length 8 shown in Figure 2a. The corridor world has *trash* (T) in the leftmost corner ($X = 0$) and a *bin* (B) in the rightmost corner ($X = 7$). The state input for the agent is its X position, a flag denoting if it has trash or not and flag denoting if it is in practice or match. The agent has two actions, move left and move right. The agent starts every episode at $X = 0$ with trash. If the agent moves to the bin, $X = 7$, with trash it gets a reward of $+1$ for delivering the trash and it automatically loses the trash at the following

time step. If it moves back to $X = 0$ without trash, it gets the trash automatically at the following time step. The agent undergoes 3 practice episodes before every match episode. Here, the match and the practice environment are the same. Each episode in both practice and match is of length between 45 and 50, sampled uniformly. The agent uses REINFORCE (Williams 1992) with our proposed algorithm for its learning. Next match samples are used for updating the intrinsic practice reward parameters using Equation 8.

Intuitively there are two important stages in the learning for this task. First, the agent must learn to take the trash from $X = 0$ to $X = 7$. Second, the agent must learn to come back to $X = 0$ to collect the trash again, so that the first step can be repeated. Figure 2b shows the return obtained by the agent across the matches. We observe that the agent quickly learns to get a episode reward of $+1$ and later, after about 100 matches, starts getting a episode reward of $+2$.

Visualization of learned intrinsic practice rewards. Our aim here is to visualize how good practice rewards vary as a function of the learning state of the agent. We do this by pausing the update of the policy at two different points during learning (Match 1 and Match 200), and allowing the intrinsic reward parameters to be updated (via additional samples of match and practice experience) until they converge. In other words, we are seeking to visualize an approximation of the optimal practice reward as a function of learning. (To be clear, the results in Figure 2b are from Algorithm 1 without pausing to allow intrinsic reward convergence.)

Figure 2c shows the (approximate) optimal practice reward over the state space at the start of agent's learning (Match 1). The top and bottom rows correspond to the agent carrying trash and not carrying trash respectively. The reward tends to be high (darker) towards the right and low (lighter) in the left of the corridor (irrespective of the presence or absence of trash), which indicates that it is asking the agent to practice going from left to right, which would allow it to get an extrinsic reward of $+1$ during match, as the agent always begins an episode at the leftmost corner with trash. Figure 2d shows the (approximate) optimal practice reward for an agent that has learned over 200 matches. At this point the agent consistently gets a reward of at least $+1$ (see Figure 2b), which means starting from $X = 0$ with trash at the beginning of the episode, the agent has learned to take the trash to $X = 7$ (bin) once. Now it needs to learn to go back to $X = 0$ from $X = 7$ (bin), so that it can collect the trash, and take it to the bin again to get an additional reward of $+1$. Figure 2d indicates that the (approximate) optimal practice reward encourages such behavior in practice. In order to reach the highest rewarding state of $X = 0$ and No Trash, the agent which starts at $X = 0$ with trash has to go to the bin, $X = 7$ (where it loses the trash) and come back to $X = 0$. In the following time step, it will automatically get trash. Now the agent has to repeat the above to reach the highest rewarding state ($X = 0$, No Trash) again, which leads to the desired behavior of repeatedly collecting and emptying the trash.

These visualizations show that our meta-gradient learning method finds practice rewards that have an intuitive and

(a) Corridor world
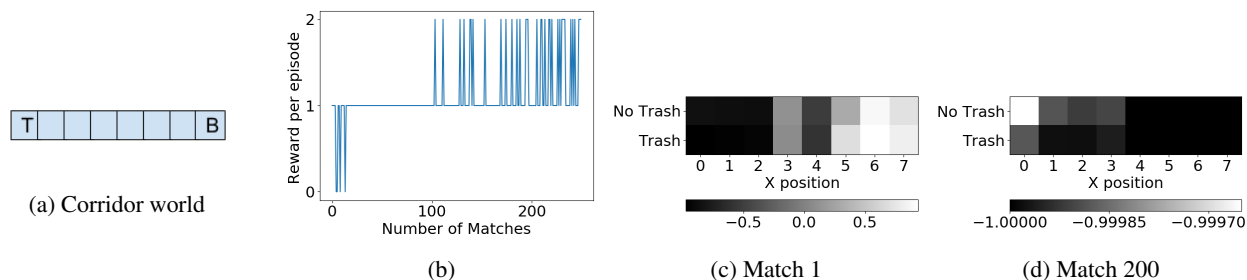
(b)

(c) Match 1

(d) Match 200

Figure 2: **(a)** shows the corridor world with trash (T) in the left most corner and the bin (B) in the rightmost corner. **(b)** shows the learning curve of the agent in the corridor world. The x-axis is number of matches during learning. The y-axis is the reward per episode during matches. **(c)** and **(d)** show the visualization of the learned intrinsic rewards for practice over the state space at two different points in the agent's learning. The top and bottom rows correspond to the agent carrying trash and not carrying trash respectively. The visualization maps the intrinsic reward values to the lightness of the color with dark (black) corresponding to the lowest value and fully illuminated (white) corresponding to the highest. The corresponding color bars show what exact value a color represents.

expected interpretation in this simple domain, and furthermore they highlight an important (and understudied) aspect of learning intrinsic rewards in general: that good intrinsic rewards are non-stationary because they depend on the state of the learner. We now move to evaluations in more challenging domains in which practice and match environments differ.

Evaluation on practice-match versions of two Atari games

In the following two experiments we create practice-match settings of two Atari games in which the practice environment differs from the match environment in an interesting way. We perform comparisons to baseline conditions to answer the following questions:

1. Does learning in practice environments in addition to matches improve performance compared to learning in matches only?

2. Is the meta-gradient update for improving the practice reward contributing to performance improvement above that obtained from training with a fixed random practice reward?

3. How does the proposed meta-gradient based method for learning practice rewards compare with a method that provides practice rewards that are similar to the match time extrinsic rewards?

4. How does the performance obtained from practice and match compare with the performance obtained if the time allotted to practice was instead replaced with additional matches?

To answer the first and fourth questions we measure and report on the comparisons therein below. To answer the second question we initialize the practice reward parameters with random weights using the same initialization method as in the meta-gradient agents, but we keep the practice reward parameters fixed during learning. In this way we directly test the effect of the meta-gradient update. To answer the third

question, we design a method where the intrinsic rewards used during practice come from a network that is trained to predict extrinsic rewards during matches. This is a sensible approach to learning potentially useful practice rewards and may be very effective in certain practice-match settings.

The two domains used for our evaluation are Pong and PacMan. In Pong, the practice environment has a wall on the side opposite to the agent instead of an opponent. In PacMan, the practice environment has the same maze as match but without any ghosts (ghosts are other agents that must be avoided). After every match, the agent is allowed a fixed time for practice in its practice environment.

Implementation details. The learning agent uses the open-source implementation of the A2C algorithm (Mnih et al. 2016) from OpenAI (Dhariwal et al. 2017) for the two games. A2C performs multiple updates to the policy parameters within a single episode (both in practice and match). Instead of waiting for the next match, we store the previous match samples in a buffer and use them to evaluate the practice policy updates as they happen within a practice episode and update the intrinsic reward parameters. The extrinsic reward provided to the agent during match is the change in game score as is standard in work on Atari games. The image pixel values and the practice/match flag are provided as state input to the A2C agent (policy and the practice reward modules). The practice reward module outputs a single scalar value (through a tanh non-linearity). There is a visual mismatch between the practice and match environments (described below) which the agent must learn to account for while transferring learning from practice to match. Note that the agent has the information of whether it is in practice or match as a part of its state input which enables the agent to learn different policies for practice and for match.

For both Pong and PacMan, we show learning curves for four A2C agents: an A2C agent that learns only in matches, an A2C agent that learns in both practice and match using our new algorithm (+ Meta-Gradients Practice), an A2C agent that learns in practice and match but using a fixed random practice reward network during practice (+ Random
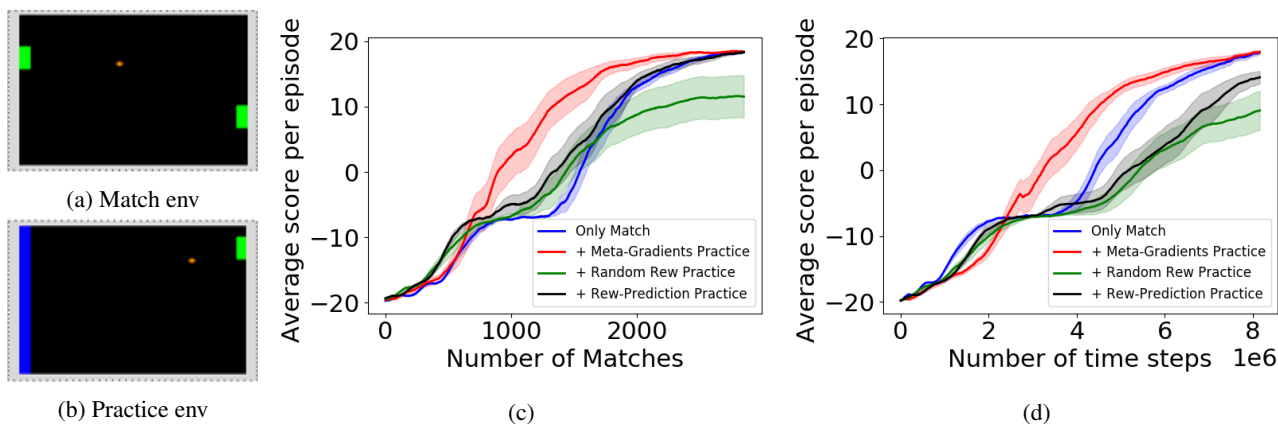
(a) Match env

(b) Practice env

(c)

(d)

Figure 3: *Results from Pong.* The blue, red, green and black curves show, respectively, performance for the baseline A2C agent learning only in matches, the practicing A2C agent using meta-gradient updates to improve the practice reward, the practicing A2C agent using fixed random rewards and, the practicing A2C agent using rewards from the extrinsic reward prediction network. The curves are the average of 10 runs with different random seeds, the shaded area shows the standard error. The y-axis is the mean reward over the last 100 training episodes. For **(c)** the x-axis is the number of matches during learning and for **(d)** the x-axis is the number of time steps during learning in both practice (when performed) and match combined.

Rew Practice) and an A2C agent that learns in practice and match but using the practice rewards during practice from a network that is trained to predict extrinsic rewards during matches (+ Rew-Prediction Practice).

Pong experiments. Pong is a two player game that simulates table tennis. Each player controls a paddle which can move vertically to hit a ball back and forth. The RL agent competes against a CPU player on the opposite side. The goal is to reach twenty points before the opponent does; a point is earned when the opponent fails to return the ball. The dynamics are interesting in that the return angle and speed of the ball depends on where the ball hits the paddle.

In the *practice environment* there is no opponent but instead a wall on the opponent's side can bounce the ball back. In contrast to an opponent's paddle, the angle of rebound is always the same as the angle of incidence irrespective of where the ball hits the wall, and the acceleration remains constant as well. Figures 3a and 3b show the match and practice environments.

To perform well in Pong, the agent needs to learn to track the ball and return it to the opponent so that the opponent misses it. This requires the agent to use the opponent's location to determine where on the paddle the ball should be hit to control the return direction and speed of the ball. The practice environment potentially allows the agent to practice tracking and returning the ball successfully without missing it, but it does not help prepare the agent for the varying speeds and direction of the ball when returned from an opponent's paddle. The practice environment also does not help practicing for directing the return of ball depending on the opponent's position. The agent practices in this modified practice environment for 3000 time steps after every match.

PacMan experiments. The player moves a PacMan through a maze containing stationary pellets and moving ghosts. The player earns points by eating pellets; the goal is

to eat as many pellets as possible while avoiding the ghosts. There are two power pellets that provide a temporary ability to eat ghosts and earn bonus points. The match ends if the PacMan eats all the pellets, the PacMan is eaten by the ghost, or the number of time steps reaches the limit of 200.

The *practice environment* has the same maze with pellets, but does not have any ghosts (Figs. 4a and 4b). Each practice episode lasts 100 time steps, and there are 3 practice episodes after every match. To perform well in a PacMan match, the agent must learn to identify where pellets are in the maze and navigate to them efficiently, while avoiding ghosts and taking alternate routes when needed. The practice environment allows the agent to learn to navigate the maze to eat pellets but does not allow it to learn to avoid ghosts and take alternate routes depending on the ghost's position during the process of trying to eat the pellets.

Pong and PacMan results. Figures 3c and 4c show the average score that the four A2C agents obtained per episode across matches in Pong and PacMan respectively. We see that learning in practice periods in addition to match periods using our proposed method (red curve) helps the agent reach good performance faster than just learning in the matches (blue curve), answering our first question above. This question of whether learning in practice in addition to match is helpful, is one that may be of significant applied interest. For example, this question is important in all of our motivating examples: basketball, tennis, or any sports, office robot, household robot, task-specific dialog agent and multi-agent teams. In all these scenarios practice can be done in addition to match without affecting the matches themselves. In other words, removing the practice (which is available in between the matches) will not speed up the availability of matches.

Figures 3c and 4c also show clearly that the benefit from practice is due to the meta-gradient update. The agent practicing with a fixed random intrinsic practice reward (green
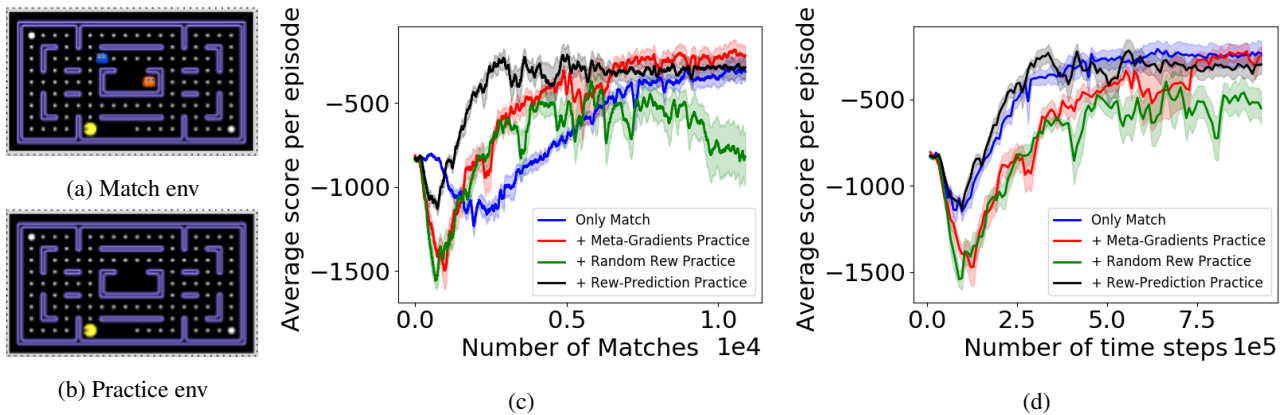
(a) Match env

(b) Practice env

(c)

(d)

Figure 4: *Results from PacMan*. The blue, red, green and black curves show, respectively, performance for the baseline A2C agent learning only in matches, the practicing A2C agent using meta-gradient updates to improve the practice reward, the practicing A2C agent using fixed random rewards and, the practicing A2C agent using rewards from the extrinsic reward prediction network. The curves average 10 runs with different random seeds, the shaded area shows the standard error. The y-axis is the mean reward over the last 100 training episodes. For **(c)** the x-axis is the number of matches during learning and for **(d)** the x-axis is the number of time steps during learning in both practice (when performed) and match combined.

curve) performs very poorly compared to the method that improves the intrinsic practice rewards using meta-gradient updates (red curve). This answers our second question.

The black curve (Rew-Prediction practice) shows the performance of the method where the intrinsic rewards used during practice come from a network that is trained to predict extrinsic match rewards during matches. This is a sensible approach to learning potentially useful practice rewards and may be very effective in certain practice-match settings such as in PacMan, where we expect it would provide practice rewards for eating pellets, a very good practice reward for practice without ghost.

In Pong this baseline performs worse than our proposed method for learning intrinsic rewards. In PacMan, in the initial stages of learning, this baseline provides much faster learning compared to our proposed method. However it ends up settling to a solution which is slightly worse than our proposed method. This is an interesting outcome because it suggests that, though it takes some time to learn the intrinsic practice rewards, our method can learn better practice rewards. We conjecture that this is because our method can adapt practice reward across the agent's lifetime and exploit the capacity to take into consideration how policy parameter changes during practice affect the match time policies which the baseline method cannot do. Further study is required to understand when our proposed method based on meta-gradients can provide faster learning compared to Rew-Prediction practice. This might be closely tied to the question of how the relationship between practice and match environment impact the performance of the two methods. This answers our third question.

Figures 3d and 4d show learning curves as a function of *time steps* in *both* practice and match combined. This compares the performance of an agent that learns in practice and match with that of an agent whose practice time

is replaced with additional matches (blue curve). In other words, it answers question 4. Surprisingly in Pong the agent could learn to perform better in matches faster if it uses some time on practice in the modified environment—while learning practice rewards using our proposed method (red curve)—instead of using that time playing additional matches. Whether it is possible to achieve faster and better learning in matches through practices instead of additional matches depends on how the practice environment is related to that of the match. In PacMan where the match policy is highly dependent on ghost position, practice without ghosts may not substitute for additional matches even if the agent performs the best practice possible. This is reflected in the results as well. In Pong we hypothesize that practice with a wall is an easier environment to learn returning the ball compared to a match with an opponent and hence leads to faster learning compared to having additional matches.

However in both Pong and PacMan, as we have seen, when we have practice in addition to matches, it leads to faster learning for a given number of matches compared to learning in matches only. As noted earlier, this evaluation of performance with respect to the number of matches is one of practical interest.

Conclusion

In this work we address the challenges encountered when a learning agent must learn in an environment in which the extrinsic reward of a primary task is not available, and where the environment itself may differ from the primary task environment; the *practice-match* setting. To address these challenges we formulated a *practice reward discovery* problem and proposed a principled meta-gradient method to solve the problem. We provided evidence from a simple grid world that shows that good practice rewards discovered by the method depend on the state of the learner.

In our primary evaluations on Pong and Pacman the practice environments differed from the standard match environments. The performance obtained from practicing in addition to match exceeded that in match alone, even though the agent had to learn *what* it should practice—that is, learn the practice reward—in addition to learning to improve the policy on the match task through the practice itself. The comparison to a poorly-performing fixed random practice reward provided evidence that performance gains are due to the meta-gradient update of the practice reward.

Conclusions concerning the generality of the method are limited by the properties of our present evaluations. We do not yet know how effective the method will be when combined with a broader range of agent architectures, although in principle it should be possible to use it with any kind of policy gradient method. The Atari experiments provide some evidence for this in their use of the A2C actor-critic architecture. We also do not yet know how the effectiveness of the method depends on the extent of the difference between match and practice environments. Because the possible benefits of practice are limited by the environment used for practice, an important direction for future work is to understand which environments are well suited for practice and how to construct them, possibly automatically.

More broadly, our results provide additional evidence for the perhaps surprising effectiveness of meta-gradient approaches in reinforcement learning, and more specifically for the effectiveness of methods for adapting rewards. But like any meta-gradient method that depends on a signal from a primary task gradient, very delayed/sparse and difficult-to-obtain rewards remain significant challenges. These challenges suggest important directions for future research.

Acknowledgements

References

Barto, A. G.; Singh, S.; and Chentanez, N. 2004. Intrinsically motivated learning of hierarchical collections of skills. In *Proceedings of the 3rd International Conference on Development and Learning*.

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

Eysenbach, B.; Gupta, A.; Ibarz, J.; and Levine, S. 2018. Diversity is all you need: Learning skills without a reward function. *International Conference on Learning Representations*.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*.

Finn, C.; Yu, T.; Zhang, T.; Abbeel, P.; and Levine, S. 2017. One-shot visual imitation learning via meta-learning. In *Proceedings of the 1st Annual Conference on Robot Learning*.

Florensa, C.; Held, D.; Geng, X.; and Abbeel, P. 2018. Automatic goal generation for reinforcement learning agents. In *Proceedings of the 35th International Conference on Machine Learning*.

Goyal, A.; Islam, R.; Strouse, D.; Ahmed, Z.; Botvinick, M.; Larochelle, H.; Levine, S.; and Bengio, Y. 2019. Infobot: Transfer and exploration via the information bottleneck. *International Conference on Learning Representations*.

Guo, X.; Singh, S.; Lewis, R.; and Lee, H. 2016. Deep learning for reward design to improve monte carlo tree search in atari games. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*.

Gupta, A.; Eysenbach, B.; Finn, C.; and Levine, S. 2018. Unsupervised meta-learning for reinforcement learning. *ArXiv* abs/1806.04640.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*.

Ng, A. Y.; Harada, D.; and Russell, S. J. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*.

Oudeyer, P.-Y., and Kaplan, F. 2009. What is intrinsic motivation? a typology of computational approaches. *Frontiers in Neurorobotics*.

Rothfuss, J.; Lee, D.; Clavera, I.; Asfour, T.; and Abbeel, P. 2018. Promp: Proximal meta-policy search. *International Conference on Learning Representations*.

Schmidhuber, J. 2010. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*.

Singh, S.; Lewis, R. L.; Barto, A. G.; and Sorg, J. 2010. Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Transactions on Autonomous Mental Development*.

Sorg, J.; Lewis, R. L.; and Singh, S. 2010. Reward design via online gradient ascent. In *Advances in Neural Information Processing Systems*.

Sutton, R. S.; McAllester, D. A.; Singh, S.; and Mansour, Y. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*.

Xu, Z.; van Hasselt, H. P.; and Silver, D. 2018. Meta-gradient reinforcement learning. In *Advances in Neural Information Processing Systems*.

Zheng, Z.; Oh, J.; and Singh, S. 2018. On learning intrinsic rewards for policy gradient methods. In *Advances in Neural Information Processing Systems*.