Learning to Self-Modify Rewards with Implicit Gradients

Aiden Boyd, Shibani and Will Callaghan

June 12, 2022

Learning to Self-Modify Rewards with Implicit Gradients

Aiden Boyd

Shibani

Will Callaghan

Abstract

Reward shaping is a powerful technique for efficient learning of optimal policies in sequential decision-making. However, it is challenging to design auxiliary rewards to help the agent, and often needs considerable time and effort by domain experts. In this paper, we build on the optimal rewards methodology to adapt a given reward function. This problem can be naturally formulated as a meta-learning problem and solved in a bi-level optimization framework. However, standard approaches used in literature for these problems are not scalable. Hence we propose to use an implicit-gradient technique to solve this problem. We demonstrate the effectiveness of our method in both a) learning optimal rewards and b) adaptive reward shaping.

1 Introduction

The nature of reward function is an important factor in the learning success of any RL algorithm. Providing dense and useful rewards can dramatically improve sample efficiency of algorithms (Dorigo & Colombetti, 1994). Early work in RL often used hand-crafted reward function for tasks like bicycle control (Randløv & Alstrøm, 1998).Recently, reward shaping has also been successfully applied in more complex tasks such as solving the video game Doom (Lample & Chaplot, 2017; Song et al., 2019).

Designing rewards that provide frequent useful feedback is non-trivial. Even small intuitive tweaks to a reward function can often result in sub-optimal or undesired behavior (Ho et al., 2019; Asmuth et al., 2008). As such a lot of work in reward shaping has focused on potential based rewards (PBRS); as one can guarantee policy invariance (Ng et al., 1999; Laud & DeJong, 2003; Marthi, 2007; Harutyunyan et al., 2015)

Even with potential based methods, it can be challenging to translate human intuition about the problem into actual rewards. For example, in most games an agent would need to keep an eye on various resources like health and mana and promote building of skills and inventory items (OpenAI, 2018). However it is difficult to translate these into specific numerical rewards. Furthermore even if converted to numerical rewards, their relative combinations can be very dynamic, and designers usually have to experiment with many different functions to see which work best. Finally even after all such considerations the new rewards may be not be very reliable or in some cases counter-productive. This raises the main challenge we try to address here, a method that allows for a) easy specification of rewards while helping learning and b) robustness to ill-specified rewards.

To address these challenges we consider how to utilize a given auxiliary reward function. The term auxiliary here is used to distinguish between two different rewards. The first is primary reward whose optimization is guaranteed to solve the given task. The second rewards is the auxiliary or helping reward which is additional rewards provided by the user to aid learning. We aim to utilize the guidance of the given reward function while ignoring the mis-specified or unbeneficial ones.

This naturally produces an online bi-level optimization problem. The inner-level constitutes of policy optimization under an auxiliary shaping reward function, and the outer-level optimizes the parameterized auxiliary reward function for maximization of the true primary reward. Changing auxiliary rewards dynamically shapes the inner policy optimization procedure and allows ensuring that the policy converges to the one that exhibits the desired behavior. We use the method of implicit gradients(Krantz & Parks, 2002) to do the outer optimization in a scalable fashion. We then conduct experiments on different environment with our proposed method for both a) optimal reward learning and b) adaptive reward shaping.

2 Preliminaries

We first describe the notations followed in the paper followed by a description of and . We also summarize the implicit gradient method which is used in this work for the outer optimization via gradient based methods.

2.1 Reinforcement Learning with Learnt Rewards

In this section we provide a brief summary of using dynamic objectives for reinforcement learning. Unlike structure prediction there is a larger body of work which uses some form of meta-learning in reinforcement learning(Sorg et al., 2010; Singh et al., 2010; Zheng et al., 2018). Our formulation of reward learning follows from the work of Lewis et al. (2010); Singh et al. (2010) and recently Zheng et al. (2018)

Notation A Markov Decision Process (MDP) is specified by a tuple $(\mathcal{S}, \mathcal{A}, P, r_p, \gamma, d_0)$. $\mathcal{S}$ is the state space, $\mathcal{A}$ is the set of actions, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ are the transition probabilities, $r_p : \mathcal{S} \times \mathcal{A} \to [-R_{\max}, R_{\max}]$ is the primary reward function, $\gamma \in [0, 1]$ is the discount factor and d_0 is the inital state distribution. A policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ takes as input a state action pair (s, a) , and provides with probability with which the agent takes the action a when observing state s. Let S_t, A_t, R_t be the state, action and reward at time t. The goal in this case is to optimize the performance $J(\theta, r_p, \gamma) = \mathbb{E}_{\pi_\theta}[\sum\limits_{t=0}^{T} \gamma^t r_p(S_t, A_t)]$ (the cumulative discounted primary rewards) of a policy π_θ parameterized by θ. A popular method to obtain the optimal parameter $\theta^* = \operatorname{argmax} J(\theta)$ is by using gradient based optimization with gradient estimates obtain using the policy gradient method (Sutton et al., 2000) :

$$\nabla J(\theta, .) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \partial_\theta \log(\pi_\theta(s, a)) \sum_{j=t}^{T} \gamma^{j-t} r_p(S_j, A_j)\right] \tag{1}$$

Auxiliary Rewards In many cases the objective $J(\theta, r_p)$ can be difficult to optimize due to lack of strong supervision (Ng et al., 1999). A natural method to help training is to modify the original reward function with an auxiliary reward function which incorporates domain knowledge. Formally we include a new reward function r_{aux} such that the objective to optimize is now $J(\theta, r_p + r_{aux})$. However adding such an auxiliary reward can change the optima of the objective and lead to the agent learning a different policy than the one desired(Ng et al., 1999). Ng et al. (1999) provides an approach which guarantees the invariance of policy optimality. Their key idea is that invariance can be maintained by a potential function based auxiliary reward viz. $r_{aux} = \gamma\Phi(s') - \Phi(s)$, where $\Phi : \mathcal{S} \to \mathbb{R}$ is a potential function. Wiewiora et al. (2003) have shown that Q-learning with PBRS is equivalent to Q-learning with with a different initialization of Q-values, which limits the extent of potential based methods.

2.2 Implicit Gradient Method

Implicit gradient technique allows one to analytically obtain the gradient from the resultant of an optimization process directly. A particular instance of this technique relevant for the current work is that for a optimization problem of the form:

$$\begin{aligned}&\phi^* = \underset{\phi}{\operatorname{argmin}}\, J_{\text{Prim}}(\theta^*(\phi), \phi)\\ &\text{such that}\\ &\theta^*(\phi) = \underset{\theta}{\operatorname{argmin}}\, J_{\text{Aux}}(\theta, \phi)\end{aligned} \tag{2}$$

one can find the 'meta-gradient' of ϕ as:

$$\begin{aligned}&\left[\frac{\partial}{\partial\theta}\frac{\partial}{\partial\theta}(J_{\text{Aux}}(\theta,\phi))\right]^{-1}\left[\frac{\partial}{\partial\phi}\frac{\partial}{\partial\theta}(J_{\text{Aux}}(\theta,\phi))\right]\frac{\partial(J_{\text{Prim}}(\theta^*(\phi),\phi)}{\partial\theta}\\ &+\frac{\partial(J_{\text{Prim}}(\theta^*(\phi),\phi))}{\partial\phi}\end{aligned} \tag{3}$$

This result is obtained by using the characteristic equation for the inner optimization viz. $\partial_\theta J_{\text{Aux}}(\theta,\phi)|_{\theta=\theta*} = 0$ This result is particularly helpful as one simply requires the value of the optimal $\theta*$ to obtain the gradient wrt ϕ, and does not depend on how the inner optimization is conducted. In contrast, if the inner optimization is conducted via gradient descent, one can use the automatic differentiation to backpropagate through the optimization. It can be proven that the two results are identical (Krantz & Parks, 2002). If the required hessians multiplications can be done efficiently, then the implicit method is more scalable. Moreover if the inner optimization is not perfect then the backprop through optimization route is dependent on choice of optimization, number of steps and other choices.

3 Implicitly adjusting reward functions

3.1 Learning Auxiliary Rewards

When the reward function r_p is sparse the objective $J(\theta, r_p)$ can be difficult to optimize due to lack of strong supervision (Ng et al., 1999). A natural method to help training is to add auxiliary rewards r_{aux} such that the objective to optimize is now $J(\theta, r_p + r_{aux})$. However adding such an auxiliary reward can change the optima of the objective and lead to the agent learning a different policy than the one desired(Ng et al., 1999). However as mentioned earlier, this technique is neither very satisfactory (due to (Wiewiora, 2003; Laud & DeJong, 2003) limitation) but still requires some expertise in creating numerical rewards from intuitive ideas (Song et al., 2019; Jaderberg et al., 2019). One way around this issue is the intrinsic optimal rewards framework (Singh et al., 2004; Sorg et al., 2010) where a new rewards is learnt while simultaneously optimizing a policy under those rewards. Building on this idea of learning intrinsic rewards, we want to change the auxiliary rewards r_{aux} in such a way that learning a policy π_θ using them, improves the primary the primary objective $J(\theta, r_p)$. If we parameterize the auxiliary rewards as r_ϕ, this can be written as the following bi-level objective:

$$\phi^* = \underset{\phi}{\operatorname{argmax}}\, J(\theta(\phi), r_p) \quad \text{s.t.} \quad \theta(\phi) = \underset{\theta}{\operatorname{argmax}}\, J(\theta, r_\phi)$$

Intuitively θ is learnt to improve via the rewards r_ϕ Zheng et al. (2018); Bechtle et al. (2019) solve this problem via alternate maximization of θ and ϕ, where θ is updated o improve $J(\theta, r_\phi)$, while ϕ is updated to improve $J(\theta(\phi), r_p)$. However their method unrolls the inner optimization only a few steps. We instead are going to use the implicit gradient method to directly obtain the "meta-gradients" of ϕ.

Reward Shaping with Auxiliary Rewards Often experts can design useful helper rewards which can be helpful for the agent by providing rewards for the agent to visit a certain state or perform an action sequence (Niekum et al., 2010). Such helper rewards can be made available to r_ϕ. Similarly the primary rewards can also be used in r_ϕ. In fact, Hu et al. (2020) explicitly consider learning state based weights of auxililary helper rewards; and Zheng et al. (2018) optimize a combination of intrinsic/auxiliary rewards and extrinsic/primary rewards. In this work we allow for externally specified helper rewards r_h and give a generic form for r_ϕ and as:

$$r_\phi = (z_\phi(s,a) r_h(s,a) + f_\phi(s,a))$$

Objective regularization An ideal reward function is not only dense, but also instructive and instantaneous. To favour learning dense instantaneous rewards, we modify the the outer objective to have a small penalty for sparse additive rewards f_ϕ and allow the method to adjust its discount function $\gamma_\phi \in (0, 1)$ as well. The latter allows our method to modify temporal distribution of auxiliary rewards as well. Putting it all together we can now specify the exact objective we use for learning auxiliary rewards.

$$r_\phi(s, a) = r_p(s, a) + z_\phi(s, a) r_h(s, a) + f_\phi(s, a) \tag{4}$$

$$\phi^* = \underset{\phi}{\operatorname{argmax}}\, J(\theta(\phi), r_p, \gamma) - \lambda_\gamma |\gamma_\phi|^2 + \lambda_r \sigma(|f_\phi|^2) \tag{5}$$

$$\text{s.t.} \quad \theta(\phi) = \underset{\theta}{\operatorname{argmax}}\, J(\theta, r_\phi, \gamma_\phi) \tag{6}$$

where λ_γ and λ_ϕ are hyperparameters and σ is the sigmoid function.

3.2 Estimation of Implicit Gradient

By Equation 3, the implicit gradient of ϕ is given by:

$$\partial_\phi J(\theta, \phi) = -\partial_\theta J(\theta, \phi) \left(\underbrace{\partial^2_{\theta,\theta} J(\theta, \phi)}_{H_\theta} \right)^{-1} \underbrace{\partial^2_{\phi,\theta} J(\theta, \phi))}_{H_\phi} \tag{7}$$

where $\partial^2_{\phi,\theta}$ and $\partial^2_{\theta,\theta}$ refers to the second order derivatives.

For deterministic computation (such as in supervised learning), all the partial derivatives in Equation equation 7 are exactly computable by automatic differentiation. However this is not so for the reinforcement learning case. As such we need to find efficient ways to compute these terms, ideally in the same way as the standard policy gradient is computed. Notice that the first term in the above expression is just the partial gradient for J with respect to policy parameters. As such this can be replaced by the usual policy gradient expression (Equation 1). Next one also observes that the same policy gradient terms also appears inside the second order derivatives. Hence if one can compute the partials of the the policy gradient with respect to the parameters ϕ, θ, one can use the corresponding expressions to compute the "meta-gradient".

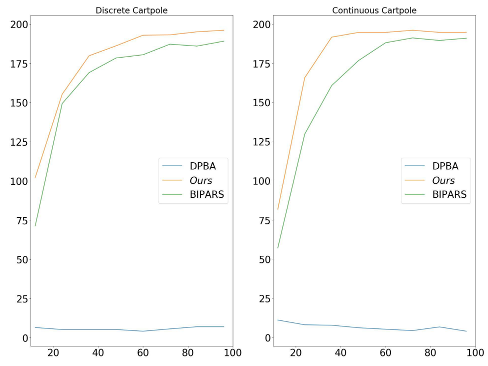
Remark 1. *Note that the gradients with respect to ϕ are to be computed from the optimized value of θ. As such an inefficiency arises from the need to rerun the new policy $\pi_{\theta^*}(\phi)$ to compute the required gradients. One can improve the efficiency by reusing the data generated for estimating the inner optimization gradient (i.e using π_θ and do off-policy correction for estimation of the gradients at π_{θ^*}.*

4 Experiments

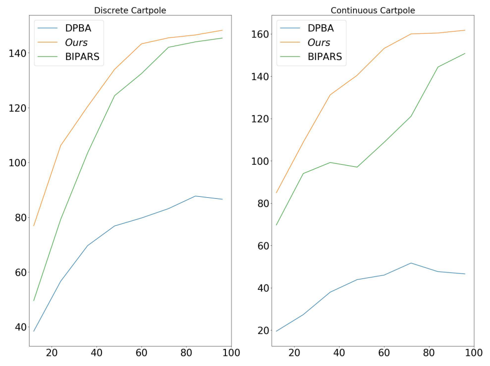
In this section, we present the results from our experiments. We conducted three groups of experiments. First we do exploratory experiments on a simple environment (*cartpole*) to explore how our model works in presence of beneficial, irrelevant and harmful auxiliary rewards. Next we test our method on Mujoco continuous control tasks (Duan et al., 2016) to show how our method can use auxililary rewards. Finally, we deploy our method for learning intrinsic rewards where we follow the procedure espoused in Zheng et al. (2018). We conduct both the second and third experiment on a few Mujoco continuous control tasks (Duan et al., 2016) viz Hopper, HalfCheetah, and Walker2d. Finally

4.1 Exploratory Experiments

In these experiments how and whether our method is robust to random and actively harmful helper rewards. For this we conduct experiments in the cartpole environment. For the harmful rewards case, we give the external reward of -0.1 when the deviation angle of the pole becomes small. In the second experiment, we choose a random reward function with values in $[-1, 1]$.



(a) Experiment with harmful rewards



(b) Experiment with random rewards

Figure 1: Return curves for all methods on discrete ad continuous cartpole with a) Harmful rewards and b) Random rewards. The x-axis is time steps durin training. The y-axis is the moving window of average return.

The goal of the two tests is to show that our methods can identify the harmful shaping rewards and useless shaping rewards and downweigh or negate them. In Figure 1a, we plot the performance of the different algorithms with the actively negative reward on both discrete as well as continuous cartpole. Similarly in Figure 1b, we plot the performance of the different algorithms with the random reward on both environments. Since it is meaningless to provide the result of a standard algorithm such as PPO with these rewards, we plot the methods which use these rewards as advice viz DPBA (Harutyunyan et al., 2015), BIPARS(Hu et al., 2020) and our method.

We can see from the figures that DPBA does not work at all with random and harmful rewards. This is not unexpected as it is designed with the idea of the rewards being helpful. The other two methods i.e. ours and BIPARS (Hu et al., 2020) are able to adjust to the reward function. In the first experiment, both are able to figure out that the rewards are actually harmful and are able to flip them around to create useful rewards. Similarly for the random rewards they are able to learn when the random rewards are helpful and adjust accordingly. Note further that our method has a small advantage over the BIPARS method. We believe this is due to the advantage which implicit gradient methods have over differentiable optimizers when the inner optimization is not perfect. Implicit methods are able to better characterize the optima better, leading to better gradients and faster learning.

4.2 Learning to Shape Rewards

In each task, the agent is a robot composed of several joints and should decide the amount of torque to apply to each joint in each step. For these experiments, we follow the procedure of Hu et al. (2020). The primary reward is the predefined reward function in Gym. We use the reward function from Tessler et al. (2019) as the shaping reward. These shaping rewards are designed to constrain the average torque applied at the joints to be below 0.25. More specifically, $f(s,a) = w(0.25 - \frac{1}{L}\sum_{i=1}^{L}|a_i|)$, where L is the number of joints, a_i is the torque applied to joint i, and w is a task-specific weight. As baselines we used the BIPARS method of Hu et al. (2020), the potential advice method DPBA (Harutyunyan et al., 2015) and dynamic potential method PBRS of Devlin & Kudenko (2012). Our result is presented in Figure 2

4.3 Learning Optimal Rewards

In this experiment, we use our method for learning intrinsic rewards for the task i.e. $r_h = 0$, and only f_ϕ is being learnt. Following Zheng et al. (2018) in this experiment, the environments are tested in a delayed reward setting. Specifically, the primary rewards are changed from the default reward to one which aggregated over a window of N=20 steps before providing them to the agent. This sparsifies the reward and makes credit assignment difficult; and learning helpful auxiliary rewards can be very useful. We also run the experiment with purely the the learned auxiliary rewards i.e. $r_p = 0$ to see how useful they are for solving the task. As

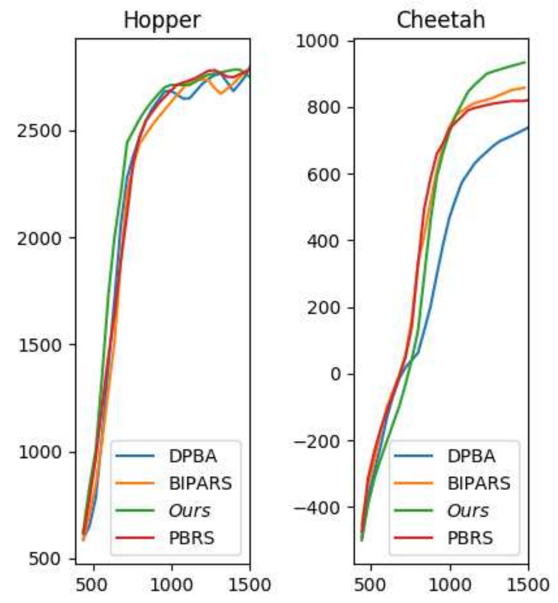


Figure 2: Return curves for all methods on MujoCo domains. The y-axis is the moving window of average return and the x-axis is number of time steps.

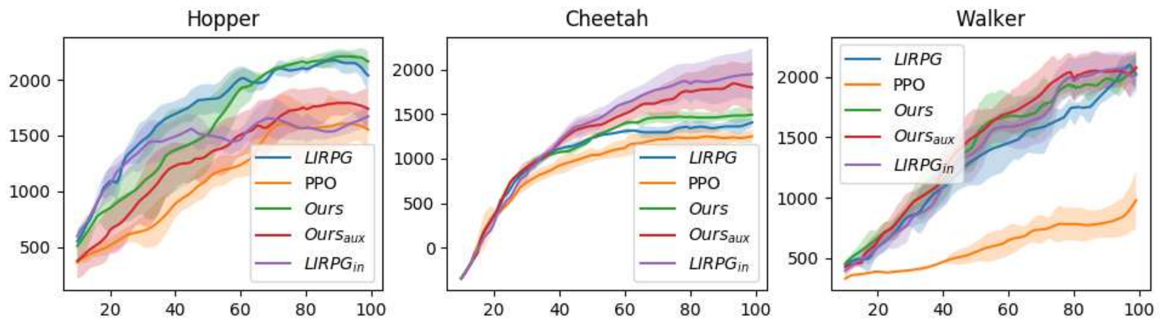


Figure 3: Return curves for all methods on MujoCo domains. The x-axis is time steps durin training. The y-axis is the moving window of average return. We have indicated the standard error of the trials on the chart

baselines we used the LIRPG method of Zheng et al. (2018) along with a PPO (Schulman et al., 2017) based agent [1]. Our result is presented in Figure 3

The red and green curves are for our method. Similar to Zheng et al. (2018) we run experiments via using both auxiliary plus primary rewards ans well as purely auxiliary reward ($Ours_{aux}$). The corresponding versions of LIRPG are labeled $LIRPG$ and $LIRPG_{In}$. From the results its clear that our method usually is able to match LIRPG and sometimes performs better than it. Another interesting observation is that our method can sometimes lag behind direct gradient based learning. We believe that this is due to the inner optimizer not reaching its fixed point during early stages. In such a case the gradient produced by our method can be biased. This is not surprising as works have found that it is usually better to use backpropagated gradient for truncated inference.

[1]Our experiments were done directly by modifying https://github.com/Hwhitetooth/lirpg

5 Related Work

Relation to Meta Learning Meta-learning broadly refers to the idea of learning to learn and has been for tasks like multi-task and structural transfer (Metz et al., 2019; Gupta et al., 2018), adaptive learning (Mendonca et al., 2019; Meier et al., 2018) and other RL applications (Xu et al., 2020; Oh et al., 2020). The key idea in meta-learning in all these applications is to make the model 'aware' of the 'learning process' and choose suitable parameters to improve learning (Franceschi et al., 2017; Thrun & Pratt, 2012; Kirsch et al., 2019). As such the ideas in this work are closely related to meta-learning. Specifically, the idea of learning parameterized objectives (in our case ϕ) has been explored in supervised learning (Sung et al., 2017).

Implicit Gradients The method of implicit gradients (Dontchev & Rockafellar, 2009; Krantz & Parks, 2002) naturally occurs whenever one deals with hierarchical optimization. They provide an analytical technique for differentiable optimization (Vlastelica et al., 2019), and so have been used for building such layers in neural-networks (Amos & Kolter, 2017; Agrawal et al., 2019). They have also been used for few-shot learning (Rajesvaran et al., 2019; Lee et al., 2019) and automatic hyper-parameter optimization (Lorraine et al., 2020).

Learning Training Losses Earlier works such as that of Wu et al. (2018); Huang et al. (2019) have attempted to learn training losses for supervised learning with neural networks. Since the auxiliary and primary rewards combined to create the 'inner' policy optimization objective, our works can be considered similar as trying to 'meta-learn' an objective loss for increasing a model performance. However, there are multiple vital differences between them. First, these works rely on 'unrolling' one/few-step gradient updates in the inner optimization and then backpropagate through those updates. This leads to poor scalability especially when the inner optimization takes hundreds of steps. Secondly, our goal is not to create a new objective but instead learn to combine various user specified rewards.

Learning Rewards Recent literature in reinforcement learning have followed the basic meta-learning framework for learning functions for trajectory returns (Xu et al., 2018), TD learning targets (Xu et al., 2020), and update rules(Oh et al., 2020; Kirsch et al., 2019). Bechtle et al. (2019) have also used meta-learning to learn new rewards for multi-task RL application. Our work is based on the optimal reward framework (Singh et al., 2004; 2010; Lewis et al., 2010; Sorg et al., 2011) The basic framework we considered in this work is very similar to that of Zheng et al. (2018) and Hu et al. (2020). However, the key difference lies in the update rule we develop. Similar to the literature on learnable losses (Bechtle et al., 2019; Wu et al., 2018), these works treat the inner and outer optimization step as computations in a single graph and backpropagate through truncated optimization process. They suffer from improper characterization of the model/optimizee parameters as, unlike supervised learning, the gradients are highly stochastic, needing unrolling for many more time steps.

Sparse Rewards and Reward Shaping Early work of reward shaping Dorigo & Colombetti (1994); Randløv & Alstrøm (1998) focuses on designing the shaping reward function F, but ignores that the shaping rewards may change the optimal policy. Ng et al. (1999) in their seminal works proved that potential-based reward shaping retains policy optimality. Dynamic potential-based advice (DPBA) Harutyunyan et al. (2015) directly incorporates any external auxiliary reward by transforming said rewards into potentials. While variants of PBRS have been succesfully used in many applications Devlin & Kudenko (2012); Marthi (2007); Grzes & Kudenko (2008); the results of Wiewiora (2003); Laud & DeJong (2003) in some sense gives limits to the extent of potential based shaping.

Other works have used the key ideas of reward shaping for multi-agent reward shaping Devlin & Kudenko (2011); Sun et al. (2018), belief shaping Marom & Rosman (2018), ethics shaping Wu & Lin (2018), and meta learning rewards Zou et al. (2019); Zheng et al. (2018). Similar to this work, Jaderberg et al. (2019) propose a population based optimization process for learning intrinsic rewards and achieve spectacular performance on Quake III.

References

Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and Zico Kolter. Differentiable convex optimization layers. *arXiv preprint arXiv:1910.12430*, 2019.

Brandon Amos and Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, 2017.

John Asmuth, Michael L Littman, and Robert Zinkov. Potential-based shaping in model-based reinforcement learning. In *AAAI*, pp. 604–609, 2008.

Sarah Bechtle, Yevgen Chebotar, Artem Molchanov, Ludovic Righetti, Franziska Meier, and Gaurav S Sukhatme. Meta-learning via learned loss. 2019.

Sam Devlin and Daniel Kudenko. Theoretical considerations of potential-based reward shaping for multi-agent systems. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'11)*, pp. 225–232, 2011.

Sam Michael Devlin and Daniel Kudenko. Dynamic potential-based reward shaping. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'12)*, pp. 433–440, 2012.

Asen L Dontchev and R Tyrrell Rockafellar. *Implicit functions and solution mappings*, volume 543. Springer, 2009.

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

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International conference on machine learning*, pp. 1329–1338. PMLR, 2016.

Luca Franceschi, Michele Donini, Paolo Frasconi, and Massimiliano Pontil. Forward and reverse gradient-based hyperparameter optimization. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1165–1173. JMLR. org, 2017.

Marek Grzes and Daniel Kudenko. Learning potential for reward shaping in reinforcement learning with tile coding. In *Proceedings of the AAMAS 2008 Workshop on Adaptive and Learning Agents and Multi-Agent Systems (ALAMAS-ALAg 2008)*, pp. 17–23, 2008.

Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Meta-reinforcement learning of structured exploration strategies. In *Advances in Neural Information Processing Systems*, pp. 5302–5311, 2018.

Anna Harutyunyan, Sam Devlin, Peter Vrancx, and Ann Nowe. Expressing arbitrary reward functions as potential-based advice. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI'15)*, pp. 2652–2658, 2015.

Mark K Ho, Fiery Cushman, Michael L Littman, and Joseph L Austerweil. People teach with rewards and punishments as communication, not reinforcements. *Journal of Experimental Psychology: General*, 148(3): 520, 2019.

Yujing Hu, Weixun Wang, Hangtian Jia, Yixiang Wang, Yingfeng Chen, Jianye Hao, Feng Wu, and Changjie Fan. Learning to utilize shaping rewards: A new approach of reward shaping. *Advances in Neural Information Processing Systems*, 2020.

Chen Huang, Shuangfei Zhai, Walter Talbott, Miguel Bautista Martin, Shih-Yu Sun, Carlos Guestrin, and Josh Susskind. Addressing the loss-metric mismatch with adaptive loss alignment. In *International Conference on Machine Learning*. PMLR, 2019.

Max Jaderberg et al. Human-level performance in multiplayer games with population-based rl. 2019.

Louis Kirsch, Sjoerd van Steenkiste, and Jürgen Schmidhuber. Improving generalization in meta reinforcement learning using learned objectives. *arXiv preprint arXiv:1910.04098*, 2019.

Steven George Krantz and Harold R Parks. *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media, 2002.

Guillaume Lample and Devendra Singh Chaplot. Playing fps games with deep reinforcement learning. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI'17)*, pp. 2140–2146, 2017.

Adam Laud and Gerald DeJong. The influence of reward on the speed of reinforcement learning: An analysis of shaping. In *Proceedings of the 20th International Conference on Machine Learning (ICML'03)*, pp. 440–447, 2003.

Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10657–10665, 2019.

Richard L Lewis, Satinder Singh, and Andrew G Barto. Where do rewards come from? In *Proceedings of the International Symposium on AI-Inspired Biology*, pp. 2601–2606, 2010.

Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In *International Conference on Artificial Intelligence and Statistics*, pp. 1540–1552. PMLR, 2020.

Ofir Marom and Benjamin Rosman. Belief reward shaping in reinforcement learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI'18)*, pp. 3762–3769, 2018.

Bhaskara Marthi. Automatic shaping and decomposition of reward functions. In *Proceedings of the 24th International Conference on Machine Learning (ICML'07)*, pp. 601–608, 2007.

Franziska Meier, Daniel Kappler, and Stefan Schaal. Online learning of a memory for learning rates. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2425–2432. IEEE, 2018.

Russell Mendonca, Abhishek Gupta, Rosen Kralev, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Guided meta-policy search. *arXiv preprint arXiv:1904.00956*, 2019.

Luke Metz, Niru Maheswaranathan, Brian Cheung, and Jascha Sohl-Dickstein. Learning unsupervised learning rules. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HkNDsiC9KQ.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, 1999.

Scott Niekum, Andrew G Barto, and Lee Spector. Genetic programming for reward function search. *IEEE Transactions on Autonomous Mental Development*, 2010.

Junhyuk Oh, Matteo Hessel, Wojciech M Czarnecki, Zhongwen Xu, Hado van Hasselt, Satinder Singh, and David Silver. Discovering reinforcement learning algorithms. *arXiv preprint arXiv:2007.08794*, 2020.

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

Arvind Rajesvaran, Chelea Finn, Sham Kakade, and Sergey Levin. Meta-learning with implicit gradients. 2019.

Jette Randløv and Preben Alstrøm. Learning to drive a bicycle using reinforcement learning and shaping. In *Proceedings of the 15th International Conference on Machine Learning (ICML'98)*, pp. 463–471, 1998.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv:1707.06347*, 2017.

Satinder Singh, Richard L Lewis, Andrew G Barto, and Jonathan Sorg. Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Transactions on Autonomous Mental Development*, 2(2): 70–82, 2010.

Satinder P. Singh, Andrew G. Barto, and Nuttapong Chentanez. Intrinsically motivated reinforcement learning. In *Advances in Neural Information Processing Systems 17 (NIPS'04)*, pp. 1281–1288, 2004.

Shihong Song, Jiayi Weng, Hang Su, Dong Yan, Haosheng Zou, and Jun Zhu. Playing fps games with environment-aware hierarchical reinforcement learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (AAAI'19)*, pp. 3475–3482, 2019.

Jonathan Sorg, Richard L Lewis, and Satinder Singh. Reward design via online gradient ascent. *Advances in Neural Information Processing Systems*, 2010.

Jonathan Sorg, Satinder Singh, and Richard L Lewis. Optimal rewards versus leaf-evaluation heuristics in planning agents. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

Fan-Yun Sun, Yen-Yu Chang, Yueh-Hua Wu, and Shou-De Lin. Designing non-greedy reinforcement learning agents with diminishing reward shaping. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society (AIES 2018)*, pp. 297–302, 2018.

Flood Sung, Li Zhang, Tao Xiang, Timothy Hospedales, and Yongxin Yang. Learning to learn: Meta-critic networks for sample efficient learning. *arXiv preprint arXiv:1706.09529*, 2017.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neurips*, 2000.

Chen Tessler, Daniel J. Mankowitz, and Shie Mannor. Reward constrained policy optimization. In *Proceedings of the 7th International Conference on Learning Representations (ICLR'19)*, 2019.

Sebastian Thrun and Lorien Pratt. *Learning to learn*. Springer Science & Business Media, 2012.

Marin Vlastelica, Anselm Paulus, Vít Musil, Georg Martius, and Michal Rolínek. Differentiation of blackbox combinatorial solvers. *arXiv:1912.02175*, 2019.

Eric Wiewiora. Potential-based shaping and q-value initialization are equivalent. *J. Artif. Intell. Res.*, 19: 205–208, 2003.

Eric Wiewiora, Garrison W Cottrell, and Charles Elkan. Principled methods for advising reinforcement learning agents. In *Proceedings of the 20th International Conference on Machine Learning (ICML'03)*, pp. 792–799, 2003.

Lijun Wu, Fei Tian, Yingce Xia, Yang Fan, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. Learning to teach with dynamic loss functions. *arXiv:1810.12081*, 2018.

Yueh-Hua Wu and Shou-De Lin. A low-cost ethics shaping approach for designing reinforcement learning agents. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI'18)*, pp. 1687–1694, 2018.

Zhongen Xu, Hado Hasselt, Matteo Hessel, Junyuk Oh, Satinder Singh, and David Silver. Metagradient reinforcement learning with objective discovered online, 2020.

Zhongwen Xu, Hado van Hasselt, and David Silver. Meta-gradient reinforcement learning. *arXiv preprint arXiv:1805.09801*, 2018.

Zeyu Zheng, Junhyuk Oh, and Satinder Singh. On learning intrinsic rewards for policy gradient methods, 2018.

Haosheng Zou, Tongzheng Ren, Dong Yan, Hang Su, and Jun Zhu. Reward shaping via meta-learning. *arXiv preprint*, arXiv:1901.09330, 2019.