ddlGAN: a Deep Dual-LSTM Generative Adversarial Network for Inverse Reinforcement Learning with Multi-task

Yutong Bai
School of Software Engineering and Microelectronics Engineering
Northwestern Polytechnical University
`hippo@cs.cranberry-lemon.edu`

Xiaoyang Han

Abstract

Inverse reinforcement learning (IRL) is the problem of recovering the underlying reward function from the behavior of an expert. Usually, traditional IRL only imitates the expert demonstrations, yet not taking the causality into considerations, which traps agent into repeated states, especially in the multitask sequential scene. It is hard for agent to get rewards, wasting a long time. We propose a novel deep dual-LSTM generative adversarial network framework to improve the decision in imitation learning under complex multitask circumstances. Further, considering when the number of the demonstrations are limited, it is hard to evaluate the results, we put forward an Adaboost-like discriminative model to solve this. In addition, this paper builds an end-to-end deep neural network, which is model-free, adaptable to more complex and larger issues and have significant improvements on speed.

1 Introduction

Although Reinforcement Learning has achieved applausive performs these years in many domains including robot navigation, urban rote planning, adaptive control etc., how to define reward functions for complex systems becomes increasingly vital. Thus, Ng et al. proposed the concept of Imitation Learning [1, 2], that is to approximate the reward function from the expert's demonstrations in a Markov Decision Process (MDP).

Not only is agent required to learn how to conduct analogous behaviors to expert's demonstrations, but also to understand them as a set of sequential decisions. Traditional IRL algorithms [3,4,5] merely imitate expert's demonstrations, yet neglecting the causality and timing relationship among state-action sequences. As a result, the agent is often trapped in some states while repeatedly performing certain actions, especially in the process of multi-task decisions. When the agent is about to complete a sequence of actions, such as clearing the vicinity of the target object, picking up an object, and placing an object in a specified place etc., it is hard to transit to rewarding situations for overlooking the causality. Therefore, Long-Short Term Memory (LSTM) algorithm [6] is applied to recording the order between action sequences to enable the agent to understand the intention of expert's decisions and choose actions based on that. Generative adversarial networks (GANs) [7] are a recently proposed class of generative models where a generator is trained to optimize a cost function that is being simultaneously learned by a discriminator. Recently, some work has recognized that IRL combined with GAN can learn strategy from expert route data in real time. Finn et al [8] prove that IRL methods are in fact mathematically equivalent to GANs, and Ho et al [9] proposed a real-time learning method which can extract a policy from observations. However, when a limited number of expert's demonstrations are only provided such that only a few states are visited in the state space, it is hard to distinguish the distances between the trajectories the expert demonstrated and the policy obtained through the approximated reward function by IRL.

Preprint. Work in progress.

To tackle this problem, an Adaboost-like Discriminative model is proposed in this paper. Adaboost is a Boosting-based classification algorithm. An Adaboost classifier is a strong classifier composed of multiple weak classifiers with different adaptive weight [10]. It adjusts every weak classifier weight depending on the error rate, to reach an adaptive classification for a limited class of samples. We adopt this idea to improve the discriminator in GAN. In training discriminators, each step we obtain a 'weak' discriminator according to Ada-weight, which updates based on the previous discriminator's classification performance, increasing the wrong classifiers' weights and decreasing the right classifiers' weights. Eventually, better classification results are achieved via limited demonstrations.

2 Background

Preliminaries We model the environment as a Markov decision process $< S, A, P, \gamma, R >$, where S denotes the state space; A denotes the set of actions; $P(s'|s,a)$ denotes the state transition probability of changing to state s' from state s when action a is token; $\gamma \in [0,1)$ denotes the discounted factor; R denotes the reward function, bounded in absolute value by R_{max}.

A policy is defined as map $\pi : S \rightarrow A$. The value function of policy π for each state s is computed by:

$$V^{\pi}(s) = R(s) + \gamma \sum_{s'} P(s'|s,\pi(s))V^{\pi}(s')\,. \tag{1}$$

Similarly, the Q-function is defined as:

$$Q^{\pi}(s,a) = R(s) + \gamma \sum_{s'} P(s'|s,a)V^{\pi}(s')\,. \tag{2}$$

We use the expected discounted sum of rewards to denote the value of policy π :

$$\mathbb{E}_{\pi}[R(s)] = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^{t} R(s_t)|\pi]\,. \tag{3}$$

Particularly, we will refer to the expert policy as π_E .

Inverse Reinforcement Learning RL aims to find the optimal policy π by Bellman optimality :

$$\pi(s) \in \arg\max_{a\in\mathcal{A}} Q^{\pi}(s,a)\,. \tag{4}$$

Given an expert policy π_E, and assuming that the π_E is better than any policy, which means:

$$Q^{\pi} <= Q^{\pi_E}\,. \tag{5}$$

Therefore, RL aims to

$$\pi(s) \in \arg\min_{\pi\in\Pi} \mathbb{E}_{\pi_E}[R(s)] - \mathbb{E}_{\pi}[R(s)]\,. \tag{6}$$

IRL looks for a reward function R that assigns high reward to the expert policy and low reward to other policies, which means:

$$maximize_R \min_{\pi\in\Pi} \mathbb{E}_{\pi_E}[R(s)] - \mathbb{E}_{\pi}[R(s)]\,. \tag{7}$$

GAN and IRL Ho et al[9]has proved the equivalence between IRL and GAN, which means,

$$\min_{\pi_G} \max_{D} \mathbb{E}_{\pi_E}[\log D(s,a)] + \mathbb{E}_{\pi_G}[\log(1 - D(s,a))]\,, \tag{8}$$

where $D : S \times A \rightarrow (0,1)$ is a discriminative classifier.

3 Dual LSTM Structure

$LSTM_E$ We adopt an LSTM, named $LSTM_E$ to train the expert's demonstrations from different states. Therefore, when an agent initially starts from a state, it will activate the trained $LSTM_E$ to recall a trajectory τ^*. Each state s^* on this trajectory can be seen as a periodical target of the agent in the process of the sequential decisions.

$LSTM_R$ We use another cascade LSTM, called $LSTM_R$ in parallel to $LSTM_E$ for the discriminator of agent side.

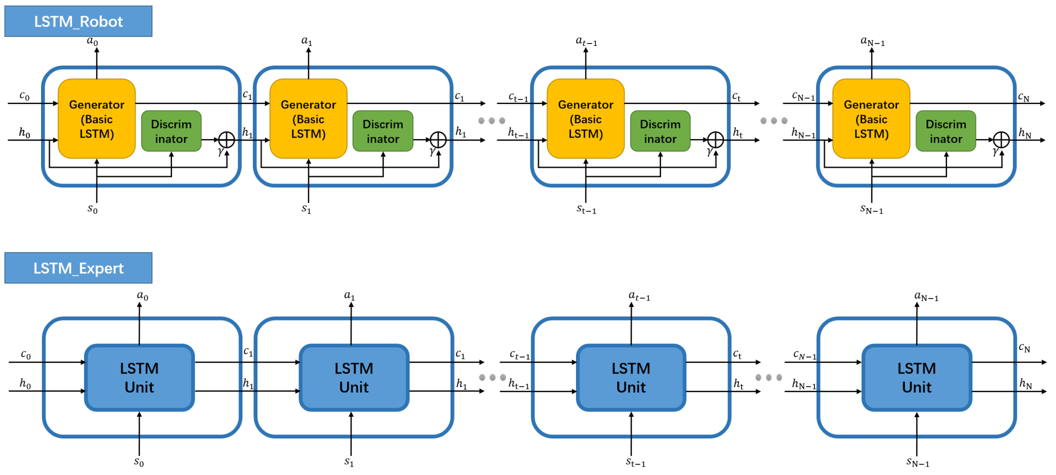


Figure 1: The dual LSTM structure: The upper one is the LSTM for the robot and the other is for expert.

The accumulated reward extracted from the last hidden layer output of $LSTM_R$ can be boost the generative model, which is an analogous policy network in terms of reinforcement learning. The generative model is trained by policy gradient.

The discriminative model, which produces immediate reward for a state-pair, is designed to assign high reward to sequential state-action pairs of $\tau^*$and low reward to other state-action pairs. We propose Adaboost-like discriminative model in Section 4, which performs as discriminator better when demonstrations are few.

4 Adaboost-like Discriminative Model

Adaboost Classifier An Adaboost classifier C is a strong classifier consists of a set of weak classifiers C_m with different adaptive weight D_m. It adjusts every weak classifier weight α_m depending on the error rate ϵ_m. For a weak classifier $C_m(x) : x \rightarrow y$, the error rate ϵ_m can be computed as:

$$\epsilon_m = P(C_m(x_i) \neq y_i) = \sum_{i=1}^{N} w_{m,i} I(C_m(x_i) \neq y_i)\,. \tag{9}$$

Where,

$$D_m = (w_{m,1}, w_{m,2}, ...w_{m,i}, ...w_{m,N})\,. \tag{10}$$

α_m represents the classifier weight $C_m(x)$ in Adaboost classifier C, where:

$$\alpha_m = \frac{1}{2}\ln(\frac{1-\epsilon_m}{\epsilon_m})\,. \tag{11}$$

We update the adaptive weight as follows:

$$w_{m+1,i} = \frac{w_{m,i}}{Z_m} e^{-\alpha_m I(\mu_{E,i}=\mu_{m,i})}\,, \tag{12}$$

$$Z_m = \sum w_{m,i} e^{-\alpha_m I(\mu_{E,i}=\mu_{m,i})}\,. \tag{13}$$

When the error rate ϵ_m of a weak classifier C_m is high, it means this weak classifier is unreliable. So we decrease its classifier weight and increase its adaptive weight. On the contrary, if the error rate of the weak classifier is high, we will increase its classifier weight and decrease adaptive weight.

For a binary Adaboost classifier $D(x) \in \{0, 1\}$, the upper bound of its error is represented as:

$$\frac{1}{N} \sum_{i=1}^{N} I(D(x_i) \neq y_i) \leq e^{-2M\gamma^2}\,. \tag{14}$$

Adaboost-like Discriminative Model Obviously, when $R(s) \in (0, 1)$, the reward function can be regarded as a discrimintator, represented as:

$$R(s) = r^T \phi(s)\,, \tag{15}$$

where $r \in \mathbb{R}^k$ is the reward weight, k the number of features and $\phi : s \to [0, 1]^k$ the Feature Function.

Therefore,

$$\begin{aligned} \mathbb{E}_\pi[R(s)] &= \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t)|\pi] && (16) \\ &= \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r^T \phi(s_t)|\pi] && (17) \\ &= r^T \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \phi(s_t)|\pi] && (18) \\ &= r^T \mu(\pi)\,, && (19) \end{aligned}$$

where,

$$\mu(\pi) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \phi(s_t)|\pi] \in \mathbb{R}^N \tag{20}$$

is the feature expectation. Thus the discriminative model becomes:

$$\arg\max_{\omega} \min_{\pi \in \Pi} \omega^T (\mu(\pi_E) - \mu(\pi))\,. \tag{21}$$

We use $\Delta\mu_m = \mu_E - \mu_m$ denoting searching direction and propose a Adaboost-like discriminative model. The reward weight updating equation is shown as:

$$r_{m+1} = r_m + D_{m+1} \cdot \Delta\mu_m\,, \tag{22}$$

where D_{m+1} is the adaptive weight.

Since we know that the optimal weight r^* of reward weight could be a linear combination of a set of adaptive weight. Adaptive weight will change according to the error rate of classification result.

Adaptive weight is updated as follows:

$$w_{m+1,i} = \frac{w_{m,i}}{Z_m} e^{-\alpha_m I(\mu_{E,i}=\mu_{m,i})}\,, \tag{23}$$

where,

$$Z_m = \sum w_{m,i} e^{-\alpha_m I(\mu_{E,i} = \mu_{m,i})}\,, \tag{24}$$

$$\epsilon_m = \sum_{k:\mu_E(k) \neq \mu_i(k)} D_m(k)\,, \tag{25}$$

$$\alpha_m = \ln(\frac{1-\epsilon_m}{\epsilon_m})\,. \tag{26}$$

5 ddlGAN

A $LSTM_R$ Unit is both generator and discriminator.

The Generative model generates action a based on current state s, long time memory c and short time memory h considered as reward/cost.

$$a_{t-1} = G_\theta(s_{t-1}, h_{t-1})\,, \tag{27}$$

where θ denotes the parameters for generator.

The Discriminative model produces immediate reward/cost for current state-action pair.

$$h_t = D_w(s_t, \tau^*) + \gamma h_{t-1} \tag{28}$$

where w denotes the parameters for discriminator.

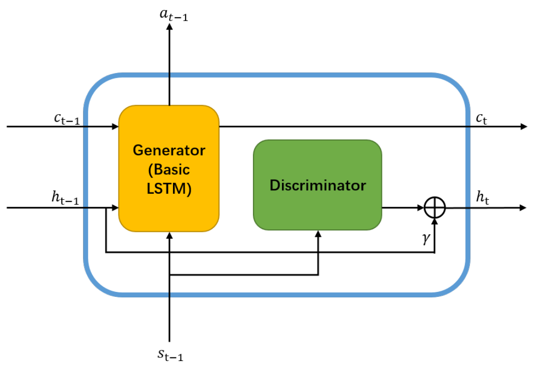


Figure 2: $LSTM_R$ Unit

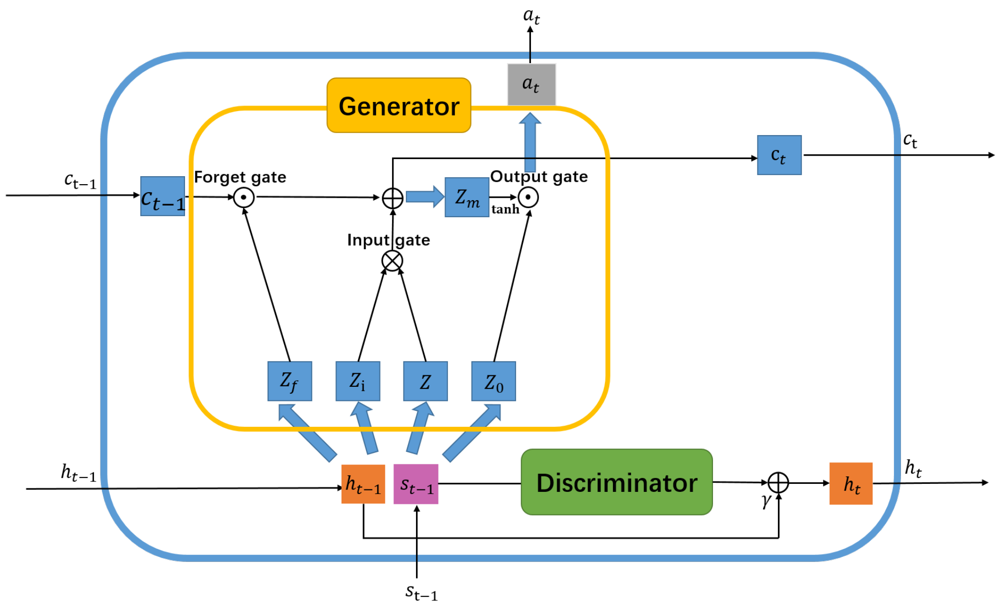


Figure 3: Details of $LSTM_R$ Unit

$LSTM_E$ is used to train the expert's demonstrations τ_E starting from different states via supervised learning method. Hence, when an agent initially starts from a state s_0, we can get an approximate expert trajectory τ^*, even if there is no state s_0 covered by expert's demonstrations. Further, every state in τ^* can be regarded as the periodical target of the agent in sequential decisions.

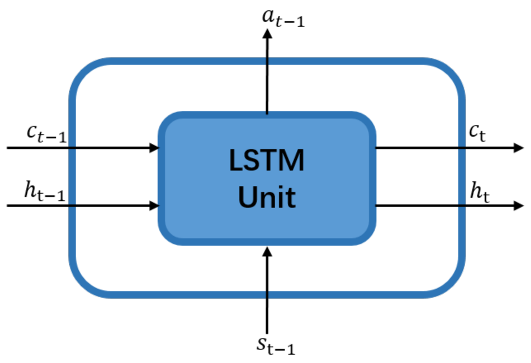


Figure 4: $LSTM_E$ Unit

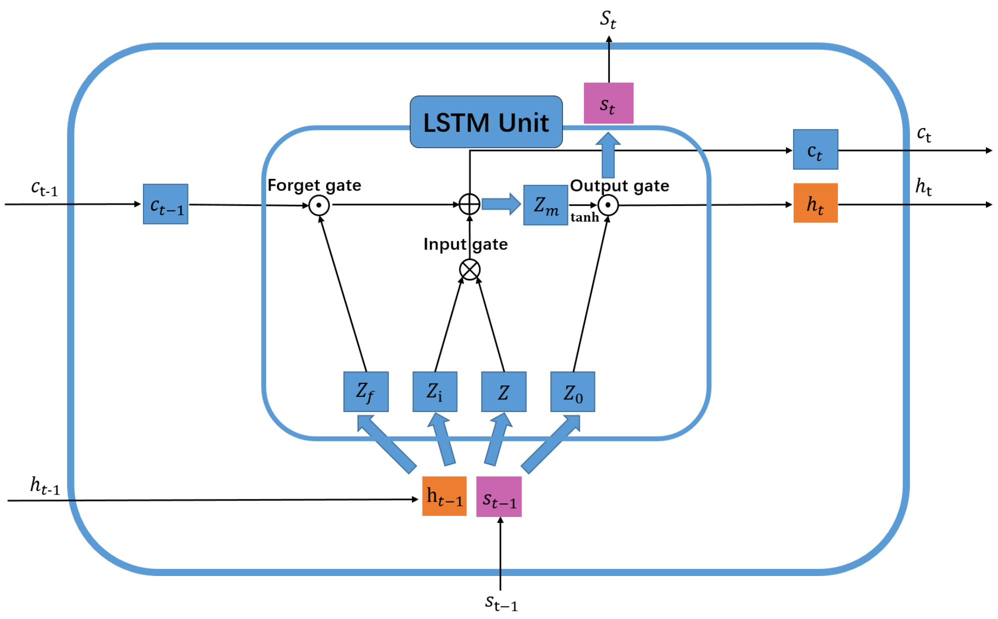


Figure 5: Details of $LSTM_E$ Unit

Algorithm Using $LSTM_E$ to train the expert policy starting from different state by supervised learning method. Thus when an agent start from a state s_0, the trained $LSTM_E$ will recall a trajectory as:

$$\tau^* = LSTM_E(s_0)\,. \tag{29}$$

$LSTM_R$ computes a mapping from an input state sequence $s_1, ..., s_t$ to output action sequence $a_1, ..., a_t$ by calculating the networks activations using the following equations iteratively form $t = 1$ to T:

$$\begin{aligned}
z_f &= \tanh(W_f \times (s_t \circ h_t) + b_f) && (30)\\
z_i &= \tanh(W_i \times (s_t \circ h_t) + b_i) && (31)\\
z &= \tanh(W \times (s_t \circ h_t) + b) && (32)\\
z_o &= \tanh(W_o \times (s_t \circ h_t) + b_o) && (33)\\
c_{t+1} &= z_f \odot c_t + z_i \odot z && (34)\\
a_{t+1} &= \sigma(W' \times (z_o \odot \tanh(c_{t+1})))\,, && (35)
\end{aligned}$$

where the W terms denote weight matrices, the b terms denote bias vector, c denotes long time memory vector, h denotes short time memory, $\circ$ is the concatenation operator to connect two vector, $\odot$ is the element-wise product of vectors.

The discriminative model produces for the current state the immediate reward r, which thinks of the τ^* as trajectory sampled from the expert policy.

$$r_t = D_w(s_t, \tau^*)\,. \tag{36}$$

Meanwhile we update the short time memory h_{t+1} according to the immediate reward for the current state as:

$$h_{t+1} = r + \gamma h_t\,. \tag{37}$$

In conclusion, the generative model generates action a based on current state s, and accumulated reward h,

$$a_{t-1} = G_\theta(s_{t-1}, h_{t-1})\,. \tag{38}$$

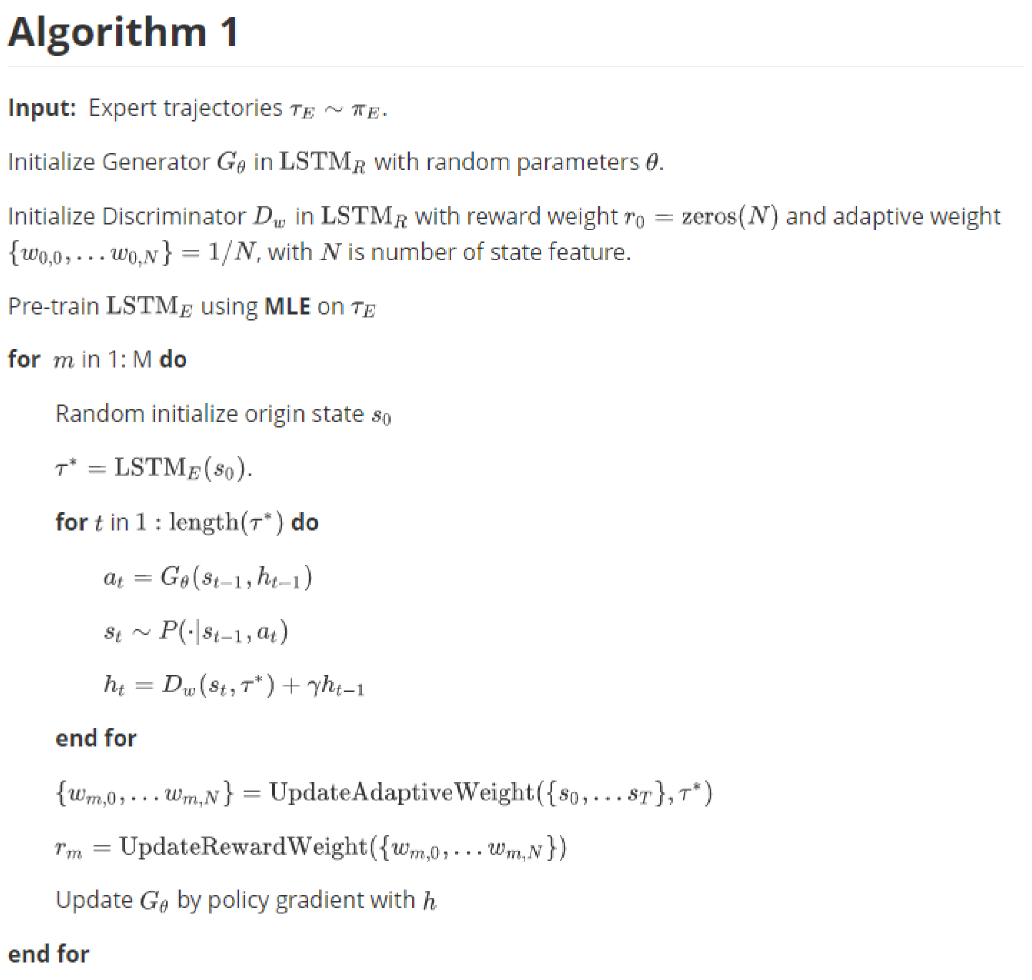
where θ denotes the parameters for generator.

The discriminative model produces immediate reward for current state-action pair.

$$h_t = D_w(s_t, \tau^*) + \gamma h_{t-1}\,, \tag{39}$$

where w denotes the parameters for discriminator.

We update the generator by policy gradient with the accumulated reward extracted from the last step of $LSTM_R$ and improve the discriminator by adaboost method.



```
Algorithm 1

Input: Expert trajectories $\tau_E \sim \pi_E$.
Initialize Generator $G_\theta$ in $\mathrm{LSTM}_R$ with random parameters $\theta$.
Initialize Discriminator $D_w$ in $\mathrm{LSTM}_R$ with reward weight $r_0 = \mathrm{zeros}(N)$ and adaptive weight
$\{w_{0,0}, \ldots w_{0,N}\} = 1/N$, with $N$ is number of state feature.
Pre-train $\mathrm{LSTM}_E$ using MLE on $\tau_E$
for $m$ in 1: M do
    Random initialize origin state $s_0$
    $\tau^* = \mathrm{LSTM}_E(s_0)$.
    for $t$ in $1 : \mathrm{length}(\tau^*)$ do
        $a_t = G_\theta(s_{t-1}, h_{t-1})$
        $s_t \sim P(\cdot|s_{t-1}, a_t)$
        $h_t = D_w(s_t, \tau^*) + \gamma h_{t-1}$
    end for
    $\{w_{m,0}, \ldots w_{m,N}\} = \mathrm{UpdateAdaptiveWeight}(\{s_0, \ldots s_T\}, \tau^*)$
    $r_m = \mathrm{UpdateRewardWeight}(\{w_{m,0}, \ldots w_{m,N}\})$
    Update $G_\theta$ by policy gradient with $h$
end for
```

Figure 6: Algorithm

References