Hindsight Value Function for Variance Reduction in Stochastic Dynamic Environment

Jiaming Guo[1,2,3], **Rui Zhang**[1,2], **Xishan Zhang**[1,2], **Shaohui Peng**[1,2,3], **Qi Yi**[1,2,4], **Zidong Du**[1,2], **Xing Hu**[1], **Qi Guo**[1], **Yunji Chen**[1,3,5*]

[1]SKL of Computer Architecture, Institute of Computing Technology, CAS, Beijing, China
[2]Cambricon Technologies
[3]University of Chinese Academy of Sciences, China
[4]University of Science and Technology of China
[5]CAS Center for Excellence in Brain Science and Intelligence Technology, CEBSIT
{guojiaming18s, zhangrui, zhangxishan, pengshaohui18z}@ict.ac.cn, yiqi@mail.ustc.edu.cn, {duzidong, huxing, guoqi, cyj}@ict.ac.cn

Abstract

Policy gradient methods are appealing in deep reinforcement learning but suffer from high variance of gradient estimate. To reduce the variance, the state value function is applied commonly. However, the effect of the state value function becomes limited in stochastic dynamic environments, where the unexpected state dynamics and rewards will increase the variance. In this paper, we propose to replace the state value function with a novel hindsight value function, which leverages the information from the future to reduce the variance of the gradient estimate for stochastic dynamic environments. Particularly, to obtain an ideally unbiased gradient estimate, we propose an information-theoretic approach, which optimizes the embeddings of the future to be independent of previous actions. In our experiments, we apply the proposed hindsight value function in stochastic dynamic environments, including discrete-action environments and continuous-action environments. Compared with the standard state value function, the proposed hindsight value function consistently reduces the variance, stabilizes the training, and improves the eventual policy.

1 Introduction

Deep reinforcement learning has achieved impressive results on solving sequential decision-making tasks, such as video games[Mnih *et al.*, 2015], board games[Silver *et al.*, 2016], and robotics[Levine *et al.*, 2016]. The policy gradient methods[Williams, 1992; Sutton *et al.*, 1999; Kakade, 2001; Schulman *et al.*, 2017] have been widely adopted for reinforcement learning because they can be used with deep neural networks straightforwardly and directly adjust the log probability of actions by estimating gradient based on the future cumulative rewards influenced by these actions. However, the gradient estimate in policy gradient methods suffers from high variance because the effect of an action on the cumulative reward is confounded with the influence of future actions, current state, and stochastic dynamics of the environment. One effective method to reduce variance is substracting a "baseline" from the cumulative reward to exclude the influence of confounding factors. The most common baseline is the state value function, which predicts the average performance starting from the current state[Sutton and Barto, 1998]. Thus, the state value function is able to reduce the high variance of gradient estimate by removing the influence of the current state.

However, the state value function will not work effectively in stochastic dynamic environments. The reason is that the state value function baseline only considers the current state and fails to exclude the influence from unexpected state dynamics and rewards in stochastic dynamic environments. For example, when an archery player learns to shoot at the target in the ever-changing wind, the wind will change the trajectory of the arrow and affect the result. Even the player gets told the relative position of the target, the initial direction and speed of the wind (which can be regarded as "the current state"), improving the skill from the result is difficult as the unexpected changes of wind (which can be regarded as "the stochastic dynamics of the environment") will also affect the result when the arrow is flying. When humans introspect to improve a certain skill, they will take the whole experience including the current state and dynamics of the environment in the future into consideration. Inspired by this, we envision that the environmental dynamics should be considered for variance reduction in addition to the current state, especially when the environment becomes drastically stochastic.

In this paper, we propose a hindsight value function that can work as a new baseline to reduce variance by removing the influence of unexpected state dynamics and rewards on the cumulative rewards. The hindsight value function receives future states and rewards as additional input and is learnt in hindsight, which is similar to the process of human introspection. With the new hindsight value function as a baseline, the policy can be adapted efficiently to optima con-

*Corresponding Author

sequently.

One major challenge of designing the hindsight value function is to keep the gradient estimate unbiased. Directly adopting the future states and rewards to compute the baseline will yield a biased gradient estimate, due to the coupling relationship between future states&rewards and the current action. Therefore, we propose a methodology to decouple current actions from the future information based on the guidance from information theory. The information from the future is provided by the future embeddings, which are supposed to be independent of the current actions. To accomplish the independence, we minimize the upper bound of mutual information between future embeddings and the current action towards zero. The hindsight value function takes embeddings of the future as input and leads to an ideally unbiased gradient estimator.

Experimentally, we compare our hindsight value function to the standard state value function by applying them to the policy gradient methods A2C[Mnih *et al.*, 2016] and PPO[Schulman *et al.*, 2017] for environments including *grid-world* and the robotics control benchmark in the MuJoCo physics simulator[Todorov *et al.*, 2012]. Our results show that the proposed hindsight value function consistently reduces the variance, stabilizes the training, and improves the eventual policies.

2 Related Works

Policy gradient. Policy gradient methods[Sutton *et al.*, 1999] are a subset of policy-based methods that maintain an explicit policy and directly draw actions from the policy. These methods represent the policy using a differentiable parameterized function approximator and use stochastic gradient ascent to update its parameters to achieve more reward. Many researchers have proposed a series of variety of policy gradient methods including Advantage Actor-Critic(A2C), Asynchronous Advantage Actor-Critic(A3C)[Mnih *et al.*, 2016],Trust Region Policy Optimization(TRPO)[Schulman *et al.*, 2015], Proximal Policy Optimization(PPO)[Schulman *et al.*, 2017] and so on. We mainly carry out the experiment based on the commonly used method A2C and PPO.

Variance reduction. The major drawback of policy gradient methods is the high variance in gradient estimate, and a large body of work has investigated variance reduction techniques. Reduce variance through using a baseline has been shown to be effective. [Wu *et al.*, 2018] consider MDPs with multi-variate independent actions and propose the state-action-dependent baseline to improve training efficiency by explicitly factoring out the effect of other actions for each action. [Mao *et al.*, 2019] introduces a kind of environments in which state dynamics depend on the input process. They propose an input-dependent baseline to reduce the variance and use a meta-learning approach to learn the baseline. Q-Prop[Gu *et al.*, 2017] makes use of an action-dependent control variate to reduce the variance. Some works consider the bias-variance tradeoff in policy gradient methods. GAE[Schulman *et al.*, 2016] replaces the Monte Carlo return with a λ-weighted return estimation with value function bootstrap. However, when the state value function provides limited information, the bootstrap will not work. We aim to reduce the variance by providing a stable and informative value function as baseline.

Mutual information. Mutual information (MI) is a fundamental measurement of the dependence between two random variables. It has been applied to a wide range of tasks in machine learning, including generative modeling[Chen *et al.*, 2016], information bottleneck[Tishby *et al.*, 2000], and domain adaptation[Gholami *et al.*, 2020]. Some works applied the MI in reinforcement learning. [Tao *et al.*, 2020] use the mutual information to learn the representation of states to facilitate efficient exploration. DADS[Sharma *et al.*, 2020] use mutual information to discover skills that are identifiable by the transitions of state. In our proposed method, we utilize MI to measure the dependence between future embedding and previous actions. As far as we know, we are the first to use MI to learn a value function.

3 Preliminaries

3.1 Notation

This paper assumes an agent interacts with the environment over discrete timesteps, modeled as a Markov Decision Process(MDP), defined by the 6-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \rho_0, r, \gamma)$. In this setting, $\mathcal{S} \subseteq \mathbb{R}^n$ is a set of n-dimensional states, $\mathcal{A} \subseteq \mathbb{R}^m$ is a set of m-dimensional actions, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the state transition probability distribution, $\rho_0 : \mathcal{S} \to [0, 1]$ is the distribution over initial states, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in (0, 1)$ is the per timestep discount factor. We denote a stochastic policy as $\pi_\theta : \mathcal{S} \times \mathcal{A} \to [0, 1]$ which is parameterized by θ. We aim to optimize the expected return $\eta(\pi) = \mathbb{E}_\tau[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$, where $\tau = (s_0, a_0, \dots)$ is the trajectory following $s_0 \sim \rho_0, a_t \sim \pi(a_t|s_t), s_{t+1} \sim \mathcal{P}(s_{t+1}|s_t, a_t)$. We define the value function as $V(s_t) = \mathbb{E}_{a_t, s_{t+1}, \dots}[\sum_{l=0}^{\infty} \gamma^l r(s_{t+l}, a_{t+l})|s_t]$.

3.2 Policy Gradient

The Policy Gradient Theorem[Sutton *et al.*, 1999] states that

$$\nabla_\theta \eta(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[\sum_{t=0}^{\infty} \nabla_\theta log \pi_\theta(a_t|s_t) \sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t'}]$$

For convenience, we define $R_t(\tau) = \sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t'}$ and $\rho_\pi(s) = \sum_{t=0}^{\infty} \gamma^t p(s_t = s)$ as the state visitation frequency. As using Monte Carlo estimation for $R_t(\tau)$ suffers from high variance, we can subtract off a quantity dependent on s_t from it to reduce the variance of this gradient estimator without introducing bias. The policy gradient estimation with a baseline becomes $\nabla_\theta \eta(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta, \rho_\pi}[\nabla_\theta log \pi_\theta(a|s)(R_t(\tau) - b(s_t))]$. The baseline is usually the state value function $V(s_t)$. In practice, a parametric function v_θ is often optimized to estimate the value. However, the optimization may be difficult in domains with high-dimensional observation spaces. Things become worse when environments or rewards are strongly stochastic.

4 Hindsight Value Function

We introduce a hindsight value function v_h, which represents the expected return conditioned on not only current state s_t

but also future states and rewards. The hindsight value function can be represented as

$$v^h(s_t, s_+, r_+) = \mathbb{E}[R|s_t, r_t, s_{t+1}, r_{t+1} \dots]$$

where s_+ means the future states and r_+ the future rewards. However, the expected return seems a constant $R(\tau)$ when the whole future is known. Then the policy gradient becomes biased as $\nabla_\theta \eta(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta,\rho_\pi}[\nabla_\theta log\pi_\theta(a|s)(R_t(\tau) - R_t(\tau))] = 0$.

4.1 Embeddings of Future

To tackle the problem mentioned above, we have to modify the hindsight value function v^h. It should exploit the information from the future but keep the policy gradient unbiased when employed as a baseline. We introduce a set of hindsight vectors $h^+ : (h_{t+1}, h_{t+2}, \dots)$, the embeddings of the future, which distill information from the future states and rewards. The hindsight value function takes these vectors as input rather than directly exploiting the future state-reward pairs. Then the hindsight value function can be written as:

$$v^h(s_t, h^+) = \mathbb{E}[R|s_t, h_{t+1}, h_{t+2} \dots]. \tag{1}$$

We regard h_t as a hindsight vector that contains the information of a single time step of the future. The hindsight vector should satisfy two properties. First, the hindsight vector should be irrelevant to the actions of the agent to guarantee the gradient estimate unbiased. Second, the hindsight vector is expected to contain enough information of the future for v^h to estimate the expected return. Intuitively, the hindsight vector can be explained as the changing wind in archery, the future traffic when driving, or abstractly, just random factors in some cases.

The hindsight vector h is designed to be a function of the current state-action pair (s, a) and the future state-reward pair (s', r): $h = F((s, a), (s', r))$. The hindsight vector h should satisfy the constraint:

$$I(h; (s, a)) = 0, \tag{2}$$

where $I[\cdot;\cdot]$ is the *Mutual information*(MI) between two random variables. Note that Equation 2 is satisfied if and only if h and (s, a) are independent. We analyze the hindsight vector and claim that under this constraint, the gradient estimate is unbiased with $v^h(s_t, h^+)$ as a baseline. First, we state a lemma to show that under the MDP definition, the hindsight vector $h_{t+1:\infty}$ is independent of previous actions a_t.

Lemma 1. *For a Markov Decision Process, $\forall\ k \in \mathcal{N}_+$, $I(h_{t+k}, a_t) = 0$, i.e., h_{t+k} and a_t are independent.*

Proof. For k = 1, h_{t+1} and a_t are independent according to Equation 2.

For $k > 1$, we have two equivalent forms of expansion for the mutual information: $I(h_{t+k}; (s_{t+k-1}, a_{t+k-1}), a_t)$
$= I(h_{t+k}; (s_{t+k-1}, a_{t+k-1}) + I(h_{t+k}; a_t|(s_{t+k-1}, a_{t+k-1}))$
$= I(h_{t+k}; a_t) + I(h_{t+k}; (s_{t+k-1}, a_{t+k-1})|a_t)$.

As h_{t+k} is a deterministic function of state-action pair (s_{t+k-1}, a_{t+k-1}) and next state-reward pair (s_{t+k}, r_{t+k-1}), according to the property of MDP, $I(h_{t+k}; a_t|(s_{t+k-1}, a_{t+k-1})) = 0$. So we have,

$I(h_{t+k}; (s_{t+k-1}, a_{t+k-1})$
$= I(h_{t+k}; a_t) + I(h_{t+k}; (s_{t+k-1}, a_{t+k-1})|a_t)$
$\Rightarrow I(h_{t+k}; a_t) \le I(h_{t+k}; (s_{t+k-1}, a_{t+k-1}) = 0.$

Then we get the conclusion that h_{t+k} and a_t are independent.

Using Lemma 1, we prove that the gradient estimate keeps unbiased when the hindsight baseline takes hindsight vectors h^+ as input.

Theorem 1. *The hindsight baseline does not bias the gradient estimate.*

Proof. $\nabla_\theta \eta(\pi_\theta)$
$= \mathbb{E}_{\tau\sim\pi_\theta,\rho_\pi}[\nabla_\theta log\pi_\theta(a|s)(R_t(\tau) - v(s_t, h^+))]$
$= \mathbb{E}_{\tau\sim\pi_\theta,\rho_\pi}[\nabla_\theta log\pi_\theta(a|s)(R_t(\tau))]$
$- \mathbb{E}_{\tau\sim\pi_\theta,\rho_\pi}[\nabla_\theta log\pi_\theta(a|s)v(s_t, h^+)]$.

Therefore, we only need to show
$\mathbb{E}_{\tau\sim\pi_\theta,\rho_\pi}[\nabla_\theta log\pi_\theta(a|s)v(s_t, \mathbf{h})] = 0$:

$\mathbb{E}_{\tau\sim\pi_\theta,\rho_\pi}[\nabla_\theta log\pi_\theta(a|s)v(s_t, h^+)]$
$= \mathbb{E}_{s_{0:t},a_{0:t-1}}[\mathbb{E}_{s_{t+1:T},a_{t:T-1}}[\nabla_\theta log\pi_\theta(a|s)v(s_t, h^+)]]$
$= \mathbb{E}_{s_{0:t},a_{0:t-1}}[\mathbb{E}_{s_{t+1:T},a_{t:T-1}}[v(s_t, h^+)]$
$\mathbb{E}_{s_{t+1:T},a_{t:T-1}}[\nabla_\theta log\pi_\theta(a|s)]]$
$= \mathbb{E}_{s_{0:t},a_{0:t-1}}[\mathbb{E}_{s_{t+1:T},a_{t:T-1}}[v(s_t, h^+)]\mathbb{E}_{a_t}[\nabla_\theta log\pi_\theta(a|s)]$
$= \mathbb{E}_{s_{0:t},a_{0:t-1}}[\mathbb{E}_{s_{t+1:T},a_{t:T-1}}[v(s_t, h^+)] \cdot 0] = 0.$

The hindsight vectors are also supposed to contain enough information for v_h to approximate the expected reward. We achieve this by maximizing the mutual information between h and (s', r) conditioned on (s, a):

$$max(I(h; (s', r)|(s, a))). \tag{3}$$

In practice, we employ a neural network F_{θ_f} parameterized by θ_f to obtain the hindsight vectors: $h = F_{\theta_f}((s, a), (s', r))$. However, we can not make sure the constraints talked above holds when F_{θ_f} is a neural network. We transfer these constraints to the following functional:

$$\mathcal{L}(\theta_f) = I(h; (s, a)) - I(h; (s', r)|(s, a)). \tag{4}$$

4.2 Learning the Hindsight Vector

We illustrate the method to minimize the target functional given by Equation 4. We first aim to maximize the mutual information $I(h; (s', r)|(s, a))$ that is related to the information contained in h. Using the entropy form, this term can be represented as:

$$H((s', r)|(s, a)) - H((s', r)|(s, a), h).$$

Thus, we only need to minimize the conditional entropy $H((s', r)|(s, a), h)$. This can be transformed to a prediction task. We employed a neural network P_{θ_p} which takes h and (s, a) as input and predicts the state-reward pair (s', r). The loss function is following:

$$L_P(\theta_f, \theta_p) = \|P_{\theta_p}(h, (s, a)) - (s', r)\|_2^2. \tag{5}$$

Now let us consider the first term of Equation 4. To minimize the mutual information, we employ the *contrastive log-ratio upper bound*(CLUB)[Cheng *et al.*, 2020], which is a method to estimate the upper bound of mutual information. The CLUB method estimates the mutual information by the

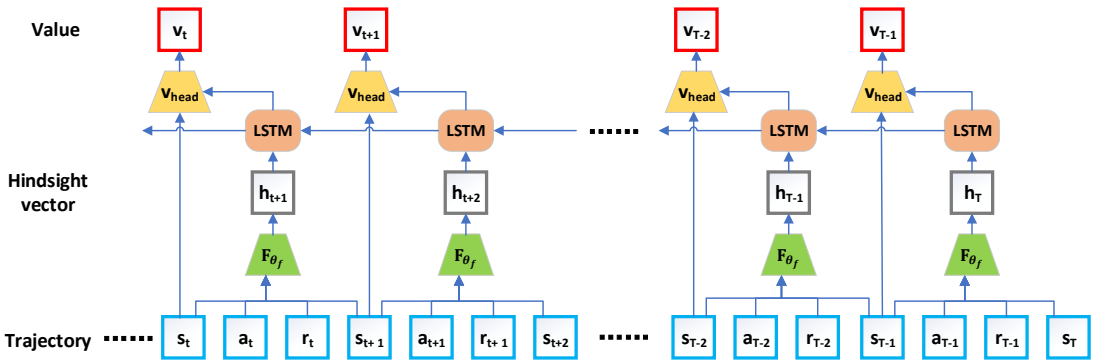

Figure 1: Model architecture of hindsight value function.

difference of conditional probabilities between positive and negative sample pairs. For random variables $\mathbf{x}$ and $\mathbf{y}$, with the conditional distribution $p(\mathbf{y}|\mathbf{x})$, the mutual information contrastive log-ratio upper bound is defined as:

$$I_{CLUB}(\mathbf{x};\mathbf{y}) = \mathbb{E}_{p(\mathbf{y},\mathbf{x})}[logp(\mathbf{y}|\mathbf{x})] - \mathbb{E}_{p(\mathbf{x})}\mathbb{E}_{p(\mathbf{y})}[logp(\mathbf{y}|\mathbf{x})].$$

However, in our case, the conditional probability $p(h|(s,a))$ is unknown. Thus, a variational distribution $C_{\theta_c}(h|(s,a))$ is introduced to approximate the conditional probability and replace $p(h|(s,a))$ in I_{CLUB}. Consequently, a variational CLUB term of h and (s,a) is defined as:

$$\begin{aligned} I_{vCLUB}((s,a);h) = \mathbb{E}_{h,(s,a))}[logC_{\theta_c}(h|(s,a))] \\ -\mathbb{E}_{p((s,a))}\mathbb{E}_{p(h)}[logC_{\theta_c}(h|(s,a))]. \end{aligned} \tag{6}$$

The unbiased estimate for $I_{vCLUB}((s,a);h)$ is obtained by sampling. Then we reduce the mutual information term by optimizing the loss function:

$$L_F(\theta_f) = I_{vCLUB}((s,a);h). \tag{7}$$

We now illustrate how to optimize the hindsight vector encoding function F_{θ_f} according to loss functions given in Equation 5 and Equation 7. When the agent explores in the environment, we store the tuple (s,a,s',r) into a buffer $\mathcal{B}$. At each training iteration, we first sample a batch of tuples $\{(s_i,a_i,s'_i,r_i)\}$ from the buffer and calculate the corresponding h_i by F_{θ_f}. Then we update the variational distribution $C_{\theta_c}(h|(s,a))$, by maximizing the log-likelihood $L(\theta_c) = \frac{1}{N}\sum_{i=1}^{N} logC_{\theta_c}(h_i|(s_i,a_i))$. After $C_{\theta_c}(h|(s,a))$ is updated, we calculate the $L_F(\theta_f)$ and $L_P(\theta_f,\theta_p)$. Finally, the gradient is calculated and back-propagated to θ_f and θ_p. During training, the hindsight vector h will contain more information to facilitate the value estimate and the hindsight value function will update towards bias-free.

4.3 Architecture of the Hindsight Value Function

The hindsight value function is designed to leverage the information of the current state and hindsight vector to estimate the expected return. We employed a neural network $v^h_{\theta_v}$ as the hindsight value function. As the input of this function is sequential values with variable length, we naturally consider LSTM[Hochreiter and Schmidhuber, 1997], a model that operates on sequences. However, inferencing a LSTM for each $v^h(s_t,h^+)$ is computationally expensive. We design the LSTM model to deal with the hindsight vectors in a trajectory at once. As Figure 1 shows, the LSTM accumulates information from the hindsight vectors backward along the trajectory. Then the hidden state of LSTM at time t is supposed to be an aggregation of the distilled information of the future after time t. Finally, the hidden state of LSTM l_t and the current state s_t are used to calculate the value estimate.

Algorithm 1 Learning hindsight vector

Input: Buffer $\mathcal{B}$ containing tuples (s_t,a_t,s',r_t), predict net P_{θ_p} with paremeter θ_p, variational distribution network C_{θ_c} with parameter θ_c, hindsight vector encoding network F_{θ_f} with paremeter θ_f, batch size N.

Sample $\{(s_i,a_i,s'_i,r_i)\}_{i=1}^{N}$ from $\mathcal{B}$
for $i = 1$ to N **do**
  Calculate hindsight vector $h_i = F_{\theta_f}(s_i,a_i,s'_i,r_i)$
end for
Log-likelihood $\mathcal{L}(\theta_c) = \frac{1}{N}\sum_{i=1}^{N} logC_{\theta_c}(h_i|(s_i,a_i))$
Update C_{θ_c} by maximizing $\mathcal{L}(\theta_c)$
for $i = 1$ to N **do**
  Sample k'_i uniformly from $\{1,2,\ldots,N\}$
  $U_i = logC_{\theta_c}(h_i|(s_i,a_i)) - logC_{\theta_c}(h_{k'_i}|(s_i,a_i))$
end for
Variantional contrastive log-ratio upper bound $I_{vCLUB} = \frac{1}{N}\sum_{i=1}^{N} U_i$
for $i = 1$ to N **do**
  $L_i = \|P_{\theta_p}(h_i,(s_i,a_i)) - (s'_i,r_i)\|_2^2$
end for
Prediction loss $\mathcal{L}(\theta_p,\theta_f) = \frac{1}{N}\sum_{i=1}^{N} L_i$
Update P_{θ_p} by minimizing $\mathcal{L}(\theta_p,\theta_f)$
Update F_{θ_f} by minimizing $\mathcal{L}(\theta_p,\theta_f)$ and I_{vCLUB}

5 Experiment

We conduct experiments on both discrete-action environments and continuous-action environments. To demonstrate the generality and scalability of the hindsight value function, we replace the state value function in A2C[Mnih *et al.*, 2016] and PPO[Schulman *et al.*, 2017] with it and show the positive effect. We also carry out experiments to analyze the effect of mutual information constraints.

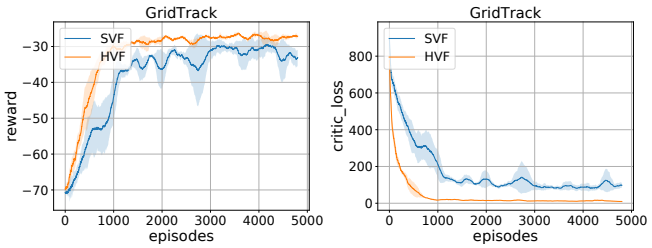

Figure 2: Comparison of hindsight value function(HVF) and state value function(SVF) with A2C on GridTrack environment.The shaded area spans one standard deviation.

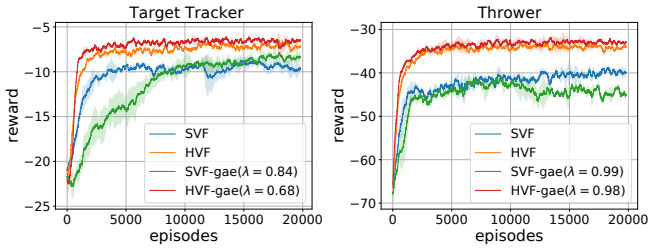

Figure 3: Comparison of hindsight value function(HVF) and state value function(SVF) with PPO on Target Tracker and Thrower environments.

5.1 Environment with Discrete Action Space

We start from two 8×8 versions of the grid-world environment.

Grid track. We consider an open grid-world with only bordering walls. There are an agent and a target. The agent can move up, down, left, and right, and the target will also move randomly in these four directions. We give the agent a dense reward that is the negative squared distance between the agent and the target.

Grid migrate. This environment is similar to Grid Track. The difference is that in this environment, the position of the target is fixed, and the agent will receive a sparse reward when it reaches the target. To make the environment stochastic dynamic, we give the agent a Gaussian random number as a noise reward.

Settings. We set the max length of an episode as 20 for both environments. The discounted factor γ is set as 0.99. And we perform the advantage actor-critic(A2C) algorithm on these environments. The actor and critic are implemented as two independent neural networks that are composed of several fully-connected layers. For the hindsight value function, we directly replace the critic with the architecture in Figure 1. Note that for estimating a single value, the new critic maintains the same network architecture but gets an additional input of the LSTM hidden state. Thus, we exclude the influence of the ability of neural networks. We run all these trials with three random seeds.

Results. Figure 2 shows the learning curves of the episode reward(the sum of rewards in an episode) and the mean square error of training the critic in the Grid Track environment. Compared to the state value function, the proposed hindsight value function achieves lower critic loss through the whole training process. The hindsight value function gets a better value estimation because it considers the unexpected target moving which will affect the cumulative reward. Thus, the variance of the gradient estimate is reduced, and consequently, the A2C algorithm with hindsight value function as baseline learns a better policy as the episode reward curve shows. Figure 5 shows the learning curves in the Grid Migrate environment. We compare three different noise levels that each correspond to adding Gaussian noise centered at zero with the labeled standard deviation to the reward at each time step. From left to right, we have the standard deviation as 0.3, 0.6, 0.9. As the noise level goes up, the learning of policy becomes slower, and when the standard deviation is 0.9, the standard A2C even can not converge. In contrast, A2C with hindsight value function is robust to noise. The loss of critic training converges to a low level under all noise levels. For all noise levels, A2C with hindsight value function adapts the policy to optima more efficiently compared with standard A2C with state value function.

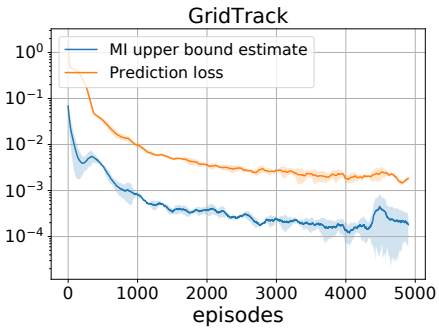

Figure 4: Learning curves of prediction loss and MI upper bound estimate on GridTrack environment.

5.2 Environment with Continuous Action Space

We evaluate the hindsight value function for the continuous-action environment using the MuJoCo robotic simulations in OpenAI Gym[Brockman *et al.*, 2016]. We use the FetchReach and Thrower environments. As the state dynamic is deterministic in the MuJoCo simulator, we modify these environments to be stochastic dynamic.

3-DoF arm target tracker. In this environment which is modified from FetchReach, we train a 3-DoF simulated robotics arm to track a randomly moving target. We give the agent a dense reward that is the negative squared distance between the robot hand and the target.

7-DoF arm thrower. In this environment which is modified from Thrower, we train a 7-DoF simulated robotics arm to throw a ball to a box on the ground. We give the agent a dense reward that is the negative squared distance between the ball and the box. The box will slightly move randomly when the ball in the robot hand and falling, which makes the environment stochastic dynamic.

Settings. We set the max length of an episode as 50 for both environments. The discounted factor γ is set as 0.99. And we

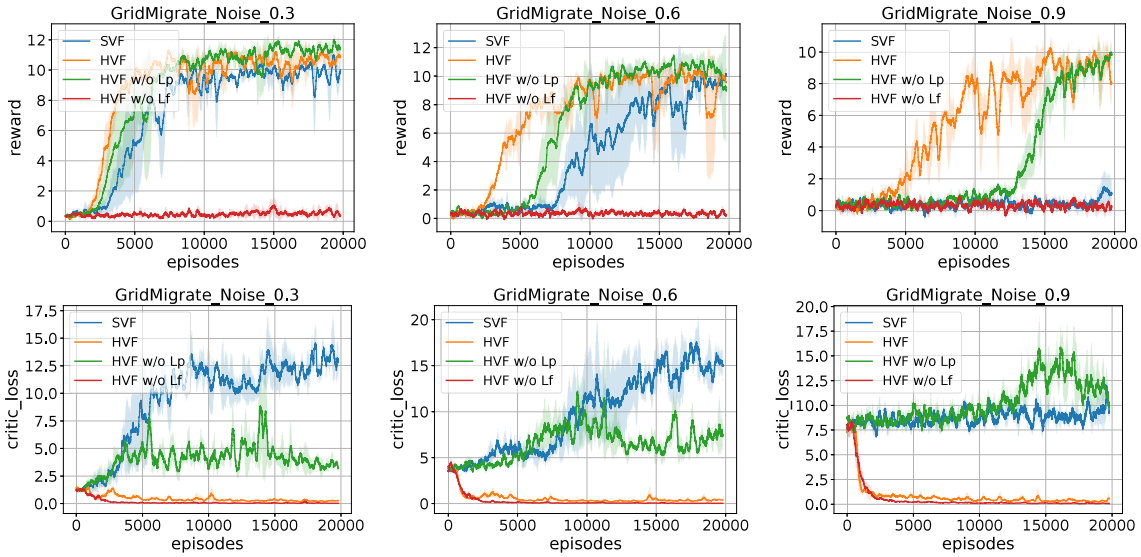

Figure 5: Comparison of hindsight value function(HVF) and state value function(SVF) with A2C on Grid Migrate environment under three different noise level.

perform the PPO algorithm on these environments. The network architecture is the same as A2C except for the number of hidden units. We run all these trials with three random seeds. Generalized advantage estimation (GAE) is a method to reduce the variance of gradient estimate at the cost of some bias which is commonly used in PPO implementations. We first turn off GAE by using $\lambda = 1$. Then we carry out experiments with GAE turned on. For PPO with GAE enhancement, we try $\lambda = [0.99, 0.98, 0.96, 0.92, 0.84, 0.68, 0.36, 0]$ and choose the best one.

Results. Figure 3 show the learning curve of episode reward in Target Tracker and Thrower. We find in both environments, PPO with HVF performs better than PPO with SVF. And GAE brings negligible improvement to PPO with SVF. Because when the value function gives a poor value estimate in a stochastic dynamic environment, weighting return estimation with value function bootstrap is meaningless. In the Target Tracker environment, we find PPO-HVF with GAE performs better, which indicates that the proposed hindsight value function may be combined with previous methods of reducing variance to further bring improvement.

5.3 Analysis of Mutual Information

In this section, we analyze the effect of constraints of mutual information. In the proposed method, $L_F(\theta_f)$ and $L_P(\theta_f, \theta_p)$ are two additional loss to optimize the mutual information and learn the hindsight value function. $L_F(\theta_f)$ is an estimate of the upper bound of mutual information. It is supposed to optimize the gradient estimate towards bias-free. $L_P(\theta_f, \theta_p)$ is a prediction error which maximizes the conditional mutual information $I(h; (s', r)|(s, a))$. It is supposed to optimize the future embedding to contain more information for a better value estimate. Figure 4 shows the learning curve of L_P and L_F in GridTrack. L_P is optimized to 10e-3, which means the future embeddings contain enough information about the unexpected next state. As L_F is optimized to 10e-4, the future embeddings can be regarded as independent of previous actions. Then the gradient estimate is unbiased. We verify the effect of L_F and L_P by removing them during training in GridMigrate. As Figure 5 shows, when L_F is removed, the policy can not be updated because the expectation of gradient is close to zero as talked at the beginning of Section 4. When L_P is removed, the policy converges slower but still faster than SVF. Because the embeddings of the future provide limited information, but it is better than no.

6 Conclusion

In this paper, we propose a hindsight value function to reduce the variance of gradient estimate in a stochastic dynamic environment. We introduce an information-theoretic approach to obtain an ideally unbiased gradient estimate. The hindsight value function is able to reduce the variance, stabilize the training, and improve the eventual policies in several environments. The approach may inspire future works for reinforcement learning in realistic environments which are mostly stochastic dynamic.

Acknowledgments

This work is partially supported by the National Key Research and Development Program of China(under Grant 2018AAA0103300), the NSF of China (under Grants 61925208, 61906179, 62002338, 61732007, 61732002, U19B2019, U20A20227), Beijing Natural Science Foundation (JQ18013), Strategic Priority Research Program of Chinese Academy of Science (XDB32050200, XDC05010300), Beijing Academy of Artificial Intelligence (BAAI) and Beijing Nova Program of Science and Technology (Z191100001119093) ,Youth Innovation Promotion Association CAS and Xplore Prize.

References

[Brockman *et al.*, 2016] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016.

[Chen *et al.*, 2016] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *NeurIPS*, 2016.

[Cheng *et al.*, 2020] Pengyu Cheng, Weituo Hao, Shuyang Dai, Jiachang Liu, Zhe Gan, and Lawrence Carin. CLUB: A contrastive log-ratio upper bound of mutual information. In *ICML*, 2020.

[Gholami *et al.*, 2020] Behnam Gholami, Pritish Sahu, Ognjen Rudovic, Konstantinos Bousmalis, and Vladimir Pavlovic. Unsupervised multi-target domain adaptation: An information theoretic approach. *IEEE Trans. Image Process.*, 2020.

[Gu *et al.*, 2017] Shixiang Gu, Timothy P. Lillicrap, Zoubin Ghahramani, Richard E. Turner, and Sergey Levine. Q-prop: Sample-efficient policy gradient with an off-policy critic. In *ICLR*, 2017.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 1997.

[Kakade, 2001] Sham M. Kakade. A natural policy gradient. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *NeurIPS*, 2001.

[Levine *et al.*, 2016] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *J. Mach. Learn. Res.*, 2016.

[Mao *et al.*, 2019] Hongzi Mao, Shaileshh Bojja Venkatakrishnan, Malte Schwarzkopf, and Mohammad Alizadeh. Variance reduction for reinforcement learning in input-driven environments. In *ICLR*, 2019.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nat.*, 2015.

[Mnih *et al.*, 2016] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *ICML*, 2016.

[Schulman *et al.*, 2015] John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz. Trust region policy optimization. In Francis R. Bach and David M. Blei, editors, *ICML*, 2015.

[Schulman *et al.*, 2016] John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In Yoshua Bengio and Yann LeCun, editors, *ICLR*, 2016.

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[Sharma *et al.*, 2020] Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. In *ICLR*, 2020.

[Silver *et al.*, 2016] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nat.*, 2016.

[Sutton and Barto, 1998] Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. *IEEE Trans. Neural Networks*, 1998.

[Sutton *et al.*, 1999] Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In Sara A. Solla, Todd K. Leen, and Klaus-Robert Müller, editors, *NeurIPS*, 1999.

[Tao *et al.*, 2020] Ruo Yu Tao, Vincent François-Lavet, and Joelle Pineau. Novelty search in representational space for sample efficient exploration. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *NeurIPS*, 2020.

[Tishby *et al.*, 2000] Naftali Tishby, Fernando C. N. Pereira, and William Bialek. The information bottleneck method. *CoRR*, 2000.

[Todorov *et al.*, 2012] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IROS*, 2012.

[Williams, 1992] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 1992.

[Wu *et al.*, 2018] Cathy Wu, Aravind Rajeswaran, Yan Duan, Vikash Kumar, Alexandre M. Bayen, Sham M. Kakade, Igor Mordatch, and Pieter Abbeel. Variance reduction for policy gradient with action-dependent factorized baselines. In *ICLR*, 2018.