Model-Based Soft Actor-Critic

Jen-Tzung Chien and Shu-Hsiang Yang
Department of Electrical and Computer Engineering, National Yang Ming Chiao Tung University, Hsinchu, Taiwan

***Abstract*—Deep reinforcement learning has been successfully developed for many challenging applications. However, collecting new data in actual environment requires a lot of costs which make the agent to learn slowly for high-dimensional states and actions. It is crucial to enhance the sample efficiency and learn with long-term planning. To tackle these issues, this study presents a stochastic agent driven by a new model-based soft actor-critic (MSAC). The dynamics of the environment as well as the reward function are represented by a learnable world model which allows the agent to explore latent representation of environment which conducts stochastic prediction and foresight planning. An off-policy method is proposed by combining with an online learning for world model. The actor, critic and world model are jointly trained to fulfill multi-step foresight imagination. To further enhance the performance, an overshooting scheme is incorporated for long-term planning, and the multi-step rollout is applied for stochastic prediction. The experiments on various tasks with continuous actions show the merit of the proposed MSAC for data efficiency in reinforcement learning.**

I. Introduction

Deep reinforcement learning (RL) has been successfully developed to learn complex behaviors in many challenging tasks. Model-free RL algorithm relies on real samples from environment and never uses the generated predictions of next state and reward to alter the behavior. The transition samples from interaction with environment provide the training data to learn the agent without knowing the state dynamic. Nevertheless, this algorithm suffers from the lack of data efficiency. A large number of steps or trials are required to interact with environment. On the other hand, model-based algorithm captures the probability of state transition and reward between two time steps under a Markov decision process. However, the ground-truth model is difficult to obtain in most of applications. This paper deals with the issue of data efficiency by boosting the merits of model-free and model-based algorithms. In the literature, deep Q-network (DQN) [1], [2], [3] is seen as a popular model-free RL method which is implemented for the tasks operated in discrete action space. Deep Q-learning simply implements an implicit policy by choosing the action with the largest Q-value. The estimated Q-value can be improved according to the temporal difference learning. The learning process can be conducted by reusing the experience data since the learning policy is different from the behavior policy. Nevertheless, the capability of DQN is bounded because its extension to a continuous or large-dimensional action space is difficult. On the other hand, the policy-based methods [4], [5] are feasible to perform actions in continuous action space. The realization of policy can be either deterministic or stochastic. Since the policy is directly parameterized, the decisions are then shaped by the resulting action space. However, these methods continuously change the policy during learning procedure. This situation causes the difficulty of directly reusing the past experience data in on-policy methods [6]. Data efficiency in policy learning is degraded. To improve data efficiency, the proximal policy optimization (PPO) [7], [8], [9], [10] was proposed. Basically, PPO updated the current policy parameter by using the past data according to the distribution distance from the previous policies. A penalty was applied during policy updating when the current policy far from the previous policy. Nevertheless, PPO was still on-policy method which required new data samples to be collected from the recent policy. An alternative strategy to strengthen data efficiency is based on the actor-critic methods which combine the value-based and policy-based methods. A successful solution to continuous control problem is based on the deep deterministic policy gradient (DDPG) [11]. DDPG is advantageous of adopting the off-policy updating with the replay data according to the gradient of Q-value function with respect to the deterministic policy parameters [12]. Nevertheless, DDPG is still insufficient in exploration and is sensitive to hyperparameter selection. There is an extra effort required to adjust system parameters [13].

More recently, the soft actor-critic (SAC) [14] was proposed as a model-free off-policy approach to learn a stochastic actor driven by the best behavior modes with the maximum expected reward under the maximum entropy framework. This framework offered a new structure of energy-based value [15] which explored high reward states by using the energy-based model. However, since such a model-free method only handled the reward signal relation, the sample efficiency in the environment was disregarded. In general, model-based RL [16], [17], [18] maintains the information of environment states which is beneficial to improve sample efficiency. The properties of individual states can be directly acquired similar to the scenario that the agent knows the environment characteristics. But, in real-world situations, it is difficult to identify the underlying model of an environment. The methods we use to train a world model from the past experience data can help the agent to capture the dynamic of environment. However, the environment representation considerably introduces the bias in predicting the dynamic. It is crucial to reduce the model bias for decision making in model-based methods.

This study aims to improve the data efficiency in model-free RL where the advanced solution based on SAC is strengthened. The sophisticated mapping from states to actions is developed. In particular, we boost the advantages of model-free and model-based methods to build an agent where the state dynamic model is exploited to make a desirable long-term prediction.

Basically, the model-free agent may adopt the information in future horizon to improve learning efficiency. The challenge in model-free method is the bottleneck for the quality of the learned agent. Also, the cumulative bias is likely increased during the rollout from current stage. This paper resolves these difficulties by twofold approaches. First, the agent is learned to rely on the experience data from real environment. A world model is trained and combined with SAC so as to generate samples which are adopted for future estimation. The generated data in current stage are utilized in a way similar to a scenario that the future agent is foreseeing. The agent is benefited from real experience using the model-based SAC. Second, a long-term regularization is imposed for model prediction. The model bias is compensated in the rollout of states. The samples of states from future horizon are merged to improve the robustness of an agent. The agent is learned to act for long sight. The experiments on continuous control tasks are conducted to illustrate the performance of model-based soft actor-critic.

II. Background Survey

Reinforcement learning aims to learn an intelligent agent who takes actions in an environment by maximizing the cumulative reward. RL is basically modeled by an Markov decision process with the notations $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ where $s_t \in \mathcal{S}$ and $a_t \in \mathcal{A}$ denote the state and action, respectively, P denotes the conditional distributions of state transitions $p(s_{t+1}|s_t, a_t)$ due to an action a_t, $r_t = r(s_t, a_t) \in R$ denotes the reward given by the environment, and $\gamma \in (0, 1)$ denotes the discount factor in calculation of cumulative reward $R_t = \sum_{i=t}^{T} \gamma^{i-t} r_i$ at time t. The goal of RL is to learn a policy which maximizes the expected return from the start distribution $J(\theta) = \mathbb{E}[R_1|\pi_\theta]$. In general, RL agent based on the proximal policy optimization (PPO) [7] aims to increase the training stability of policy $\pi_\theta(a_t|s_t)$ with parameter θ by reusing the experience data. This agent optimizes a surrogate objective function [19] which can approximate the data distribution for current policy with old parameter θ_{old}. To improve the training stability, PPO further uses the clipped objective function to limit the updating of policy parameters. The clipping of probability distribution ratio $\omega_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ in an interval $[1-\epsilon, 1+\epsilon]$ with a penalty hyperparameter ϵ is performed to constrain the policy learning in accordance with the learning objective $J(\theta)$ in a form of [7]

$$\mathbb{E}_{(s_t,a_t)\sim\rho^\pi}[\min(\omega_t(\theta)\widehat{A}_t, \text{clip}(\omega_t(\theta), 1-\epsilon, 1+\epsilon)\widehat{A}_t)] \quad (1)$$

where ρ^π denotes the state and action distribution given by policy π_θ. The ratio $\omega_t(\theta)$ is computed as a clipped weight for importance sampling over the estimator of advantage function $\widehat{A}_t = A^{\pi_{\theta_{\text{old}}}}(s_t|a_t) = Q^{\pi_{\theta_{\text{old}}}}(s_t, a_t) - V^{\pi_{\theta_{\text{old}}}}(s_t)$ with the actions sampled from the current policy $\pi_{\theta_{\text{old}}}$. $V^\pi(\cdot)$ and $Q^\pi(\cdot)$ denote the state and the state-action value functions of a policy π, respectively. Nevertheless, PPO still limits the learning efficiency in a new environment although the approximation of learning objective via clipping scheme does improve the stability in policy gradient methods.

A. Actor-critic reinforcement learning

To enhance the learning efficiency, the actor-critic RL based on deep deterministic policy gradient (DDPG) algorithm [11] was developed by extending the deterministic policy gradient algorithm [12] where the policy-based and value-based methods were combined under an actor-critic framework. The actor is updated by following the policy gradient theorem where the gradient of learning objective given by Q-value function is given by

$$\begin{aligned}\nabla_\theta J(\theta) &= \int_s \rho^\mu(s) \nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s,a)\Big|_{a=\mu_\theta(s)} ds \\ &= \mathbb{E}_{s\sim\rho^\mu}\left[\nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s,a)\Big|_{a=\mu_\theta(s)}\right].\end{aligned} \quad (2)$$

In Eq. (2), $\rho^\mu(s)$ denotes the discounted state distribution and μ_θ denotes the deterministic policy with parameter θ which maps an input state s into a deterministic action a. On the other hand, the critic in DDPG comparably performs the Q-learning which uses the off-policy experience data to update critic parameters. The Q-value function $Q^\nu(\cdot)$ of the critic with parameter ν is estimated according to the temporal difference learning where the temporal difference error is calculated by $\delta_t = r_t + \gamma Q^\nu(s_{t+1}, a_{t+1}) - Q^\nu(s_t, a_t)$. The updating of critic parameter ν is performed by using the gradient

$$\nabla_\nu J(\nu) = \delta_t \nabla_\nu Q^\nu(s, a). \quad (3)$$

Using this deterministic actor-critic algorithm, the critic is learned to approximate true action-value function. $Q^\nu(s, a) \approx Q^\mu(s, a)$ is used in the implementation.

Basically, DDPG uses the off-policy Q-function to avoid the importance sampling for policy gradient. This off-policy method is feasible to increase data efficiency. However, DDPG adopts the deterministic policy which still suffers from low exploration in continuous action space. An explicit noise to action could alleviate this issue. In addition, DDPG is also sensitive to the variations of hyperparameters which likely degrade the stability in training procedure. In [20], the twin delayed deep deterministic policy gradient (TD3) was proposed as an extension of DDPG. In particular, TD3 dealt with the overestimation problem of Q-values in actor-critic RL which led to a policy to receive the correct indication of states. The estimation error was handled by finding the unbiased estimation of value function based on the clipped double Q-learning. In the implementation, two independent value functions were approximated and learned to act as the critic in TD3. The Q-values were clipped by using the minimum Q-value function. The estimation bias of value function was reduced for continuous control tasks. The delayed updating for target neural network was also applied to improve the stability in training this off-policy method.

B. Stochastic policy

To strengthen the exploratory behavior in RL, an explicit noise was added to choose actions. The noise was also injected to learn the parameters of neural networks of an agent so that a more consistent exploration and a richer set of behaviors were

attained. The environment dependent scalar in an adaptive noise scaling scheme was chosen to reflect real exploration in state space of an agent [21]. In [22], a learnable parametric function was introduced to estimate the noise which controlled the action with noise robustness. The noise parameter controlled the policy for taking action which was affected by the corresponding probability distribution where the distribution could be selected to reflect physical meaning with mathematical interpretation. Considering the stochastic policy in presence of continuous actions, the Gaussian policy is popularly used to realize most of the actions which are close to the mean and control the actions which are spread out from mean with the variance. This is different from the deterministic policy in DDPG using Eq. (2) where only the certain point of Q-value is considered to search the maximum position for the policy. Using stochastic policy, the uncertainty of the action samples is represented by the variance of Gaussian. The exploration is enhanced by considering the stochastic actions which are around the mean of the policy. The policy function $f(\cdot)$ is then driven by a fixed state-independent parameter ξ which is characterized by a Gaussian noise for exploring the action $a_t = f(s_t; \xi)$ where $\xi \sim \mathcal{N}(\mu, \sigma^2)$. In addition, the mean and variance of Gaussian policy can be determined by a neural network controlled by the input of a given state. As a result, the policy is not only learned to handle the complicated mapping between states and actions but also determine the degree of exploration to the next state with different actions. Such an exploration is adjusted by the variance of the distribution which is learned from the experience data. The agent could automatically and flexibly explore the unknowns in environment with large variance. The stochastic policy can be implemented by using the encoder network of variational autoencoder [23] with the sampler using the reparameterization trick. The sampler generates the action from the policy distribution which is trained by maximizing the expected total reward from the state-action samples $\{s_t, a_t\}$. Correspondingly, the policy is learned to search over the landscape of Q-value function, and is evaluated to focus on some important states. Besides, Gaussian policy also smooths the landscape so as to avoid trapping in local minimum during learning procedure with Gaussian sampling [24]. In this procedure, the policy distribution with convergence is expected to shrink to a certain mode. Namely, the policy is learned to concentrate around the optimal action for each state. In case that the Gaussian policy converges to a sharp distribution, the agent behaves similar to the deterministic policy which means that the confidence of mapping the input state to the target action is high. Such a policy collapses to the mode in accordance with the Gaussian mean which finds the optimal behavior in the action structure. This stochastic policy is implemented in soft actor-critic.

C. Soft actor-critic

Soft actor-critic (SAC) [14] was proposed as an alternative extension of DDPG where the style of stochastic policy was implemented. SAC particularly carries out the entropy-regularized reinforcement learning for maximum entropy (ME) exploration based on soft Q-learning via

$$\pi_{\mathrm{ME}} = \arg\max_{\pi} \ \mathbb{E}_{(s_t,a_t)\sim\rho^{\pi}}\Big[r(s_t,a_t) + \alpha\mathcal{H}(\pi(a_t|s_t))\Big] \quad (4)$$

where α is a temperature parameter to adjust the relative importance of the entropy term $\mathcal{H}(\cdot)$ against the reward r_t. Maximizing the entropy term in SAC is beneficial to increase the diversity of the actions taken by the policy in training procedure. The distribution of policy is implemented by the reparameterization trick which is used to control the variance of the behavior. By deriving from Eq. (4), the policy parameter θ of a stochastic actor in SAC is equivalently estimated by minimizing the expected Kullback-Leiblier (KL) divergence

$$\mathbb{E}_{(s_t,a_t)\sim\mathcal{D}}\left[D_{\mathrm{KL}}\left(\pi_\theta(\cdot|s_t)\middle\|\exp\left(Q^{\phi}_{\mathrm{soft}}(s_t,\cdot) - V^{\psi}_{\mathrm{soft}}(s_t)\right)\right)\right]$$

to push the estimated policy $\pi_\theta(a|s)$ which is close to the ME policy $\pi_{\mathrm{ME}}(a|s) = \frac{\exp\{Q^{\phi}_{\mathrm{soft}}(s,a)\}}{\exp\{V^{\psi}_{\mathrm{soft}}(s)\}}$. In this KL divergence, $\alpha = 1$ is specified and $\mathcal{D}$ denotes the set of previously sampled states and actions which are stored in a replay buffer. The energy-based policy is implemented by merging the energy functions using the soft Q and value functions $Q^{\phi}_{\mathrm{soft}}(\cdot)$ and $V^{\psi}_{\mathrm{soft}}(\cdot)$ with critic parameters ϕ and ψ, respectively. The critic parameters ϕ and ψ are estimated by minimizing the squared residual errors for soft Q and soft value functions, respectively. SAC has been successfully developed to behave in continuous action space. But, the performance was still constrained since the issue of sample efficiency was neglected. To assure the performance of RL in continuous action space, the policy should be capable of accurately mapping the observed states into continuous variables. However, the exploration space of continuous actions is significantly larger than that of discrete actions. The improvement of a learned policy heavily depends on the complicated structure of state space [25]. A desirable policy needs to handle high-dimensional states as well as actions. The issue of data efficiency is crucial to guarantee a convergent training procedure and an effective system performance for reinforcement learning.

III. Model-Based Soft Actor-Critic

This study aims to improve data efficiency in actor-critic reinforcement learning. The model-based RL is incorporated into a maximum entropy framework. The uncertainty of policy is compensated to predict the state transitions and observations based on stochastic actions according to the soft Q-learning as addressed below.

A. Soft Q-learning

The policy in RL is usually learned from random initialization without knowing the state structure beforehand. Under the complicated environments, the agent could not precisely decide if the policy exploration should be continued or not. This issue often yields the distribution of policy prematurely collapsing to a position where some possible action sequences are ignored. It is a trade-off between fast trapping to a suboptimal model or slow convergence to a

desirable solution. To handle this issue, the agent needs to acquire useful information to allow policy to determine the state-action mapping and the environment exploration by itself. The entropy term in Eq. (4) measures the uncertainty of a policy which is used for information maximizing exploration and gradient updating optimization [26]. In [27], the on-policy RL using the asynchronous advantage actor-critic also merged the entropy of current policy in policy optimization. After receiving the value signals to find advantage estimate $\widehat{A}_t$, the entropy bonus $\mathcal{H}(\cdot)$ is incorporated by adding it to calculate the policy gradient as expressed by

$$\begin{aligned}\nabla_\theta J(\theta) =& \mathbb{E}_{(s_t,a_t)\in\rho^\pi}\Big[\Big(Q(s_t,a_t) - V(s_t) \\ &+ \beta\mathcal{H}(\pi_\theta(a_t|s_t))\Big)\nabla_\theta \log \pi_\theta(a_t|s_t)\Big]\end{aligned} \tag{5}$$

where β is a control parameter. Higher entropy corresponds to a larger updating. However, single-step entropy is insufficient to encourage policy to explore informative actions in sequential prediction. A meaningful alternative is to consider the objective of maximizing the long-term entropy. The soft state-action value of a maximum entropy policy is constructed with a discount factor in a form of

$$\begin{aligned}Q^\pi_{\text{soft}}(s_t,a_t) = r(s_t,a_t) + \mathbb{E}_{(s_{t+1},a_{t+1},\ldots)\sim\rho^\pi}\Big[\sum_{i=1}^{\infty}\gamma^i \\ \times\Big(r(s_{t+i},a_{t+i}) + \alpha\mathcal{H}(\pi(a_{t+i}|s_{t+i}))\Big)\Big]\end{aligned} \tag{6}$$

which is obtained by extending Eq. (4). This soft Q-value is naturally related to the energy-based policy model [28] for stochastic continuous action according to

$$\begin{aligned}\pi(a|s) &= \frac{\exp\{\frac{1}{\alpha}Q^\pi_{\text{soft}}(s,a)\}}{\int_{\tilde{a}\in\mathcal{A}}\exp\{\frac{1}{\alpha}Q^\pi_{\text{soft}}(s,\tilde{a})\}} \\ &= \exp\left\{\frac{1}{\alpha}(Q^\pi_{\text{soft}}(s,a) - V^\pi_{\text{soft}}(s))\right\}\end{aligned} \tag{7}$$

driven by energy function $\mathcal{E}(s_t,a_t) = -\frac{1}{\alpha}Q^\pi_{\text{soft}}(s_t,a_t)$. The energy gap between soft Q-value $Q^\pi_{\text{soft}}(\cdot)$ and soft state value $V^\pi_{\text{soft}}(\cdot)$ with the interaction sample is increased. The policy is seen as a form of softmax distribution of Q-value. From an alternative perspective, the derived policy for taking an action is also considered as a kind of softmax function or exponential of advantage function enhanced by the long-term entropy value. This soft Q-learning is fulfilled to implement the model-based RL for soft actor-critic.

B. Model-based reinforcement learning

Model-based RL builds a model to represent the environment behavior which can be characterized by Markov decision process (MDP). Using MDP, the data samples are continuously observed by following the state transition probabilities according to the expected reward functions under a given policy. Importantly, both of the state transitions and reward functions can be explicitly represented by a dynamic model which is trained to simulate the environment. Figure 1(a) shows how the dynamic and reward model is merged to simulate

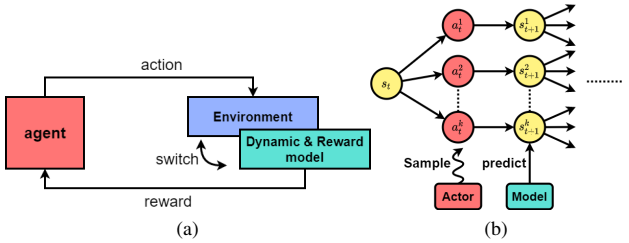


Fig. 1: Illustrations for (a) model-based versus model-free reinforcement learning and (b) on-policy rollout with stochastic policy. The model can predict future steps with current policy. The branches of trajectory are produced for experience augmentation.

and switch with environment for model-based RL. Model-free RL is implemented without this MDP model. With the model of environment, the optimal action can be efficiently searched for decision making and planning. Sample efficiency is partially handled by the model-based RL. In [29], the variational information maximizing exploration was proposed for policy learning by providing an additional intrinsic reward based on maximization of information gain as an exploration strategy. In [30], the world model was developed by building the generative neural network model to represent the environment. This model was applied to infer the action with a controller. In [31], the PlaNet was presented to mimic the dynamic model for environment in presence of a latent representation which was designed to act for planning. Basically, the aforementioned methods are performed to explore the environment by training the off-policy agent from random initialization so as to increase the diversity of information using experience data. Figure 2 depicts the working flow for model-based soft actor-critic (MSAC) where action is selected with an actor and a critic which are trained by using the states and rewards from the environment as well as the world model M_φ with parameter φ. The critic provides the values to actor and is delayed when updating the model for environment so that the stability for learning the policy can be improved. Data efficiency is compensated by the samples from world model. Multi-step bootstrap and long-term overshooting are applied. Actor, critic, target model and world model are trained.

C. World model in learning an actor-critic

Basically, the policy information is learned to help the agent to receive states from environment where the resulting values are used to update policy by itself. The agent is trained to oversee if additional information could be further acquired from the existing experience data. Although the real-world interaction brings faithful information to train agent, the collected data are sparse and expensive. Hence, a world model is incorporated to represent and simulate the data from environment. In particular, this study extends the actor-critic framework by merging the Markov decision process. The world model is simultaneously learned with actor-critic. The agent is trained to interact with

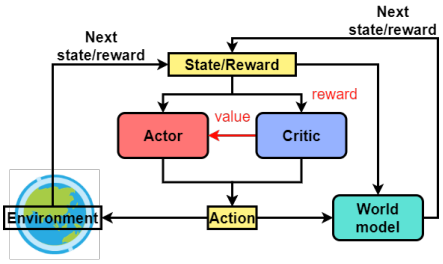


Fig. 2: Computational architecture for training the model-based soft actor-critic. World model for environment is continuously learned from the actor guided experience.

environment so that the actor-guided experience data are collected to figure out the future trajectory. The dynamic model is learned to predict the state and reward in a way like what true environment may provide at each time step t. The approximate MDP model for environment is feasible to promptly provide the prediction in future steps to guide a training procedure for agent to make decision from policy. The future states $s_{t:t+T} = \{s_i\}_{i=t}^{t+T}$ within a trajectory of length T are predicted by the dynamic model M_φ using the decomposed distributions of state transitions at individual time steps $t \le i \le t+T$ via

$$p(s_{t:t+T}|a_{t-1:t+T-1}, s_{t-1}) = \prod_{i=t}^{t+T} p(s_i|s_{i-1}, a_{i-1}). \quad (8)$$

The improvement of policy is based on the scenario when the learning of world model is controllable under the predicted dynamics of state and reward. A desirable agent is therefore learned with the increased accuracy of behavior of world model. The training of world model is self contained and planned for pursuing future information without interacting with real environment. The experience data collected from rollout with current policy are seen as a kind of augmented data. Figure 1(b) illustrates the rollout of trajectory branches by using the stochastic actor which provides the action samples for predicting next states by using the world model. The rollout data used for policy updating can be continuously generated to reflect the environment. The stability of training is sufficiently increased. The sample efficiency can be improved by using rollout data from future trajectory which is imagined from the current policy. The stochastic policy draws the action samples which contain multiple branches of state transitions. Owing to these branches and uncertainties, this policy is more likely to search a sequence of actions which reach a higher return in a trajectory during interaction and exploration. As a result, world model helps the policy which tends to re-examine the optimal decisions in actual environment without getting stuck in local optimum in training procedure. Such an approach is designed to carry out a policy for long sight from any time step in a trajectory, and act to utilize the parallel multi-step experience data. The Q-value function during soft Q-learning is driven by the accumulated reward and the policy entropy. The multi-step updating of experience data can speed up convergence for policy optimization.

D. Implementation procedure and algorithm

In policy evaluation, the Q-value of critic network with a discount factor γ is estimated with N time steps as the bootstrap for state value function at next step $N+1$ via

$$Q_w(s_t, a_t) = \sum_{i=0}^{N} \gamma^i r(s_{t+i}, a_{t+i}) + \gamma^{N+1} V(s_{t+N+1}).$$

This multi-step bootstrap is used to improve the approximate value function so as to calculate the temporal difference loss

$$J_Q(w) = \mathbb{E}_{(s_t,a_t)\sim\mathcal{D}}\Big[\Big(Q_w(s_t, a_t) - \Big(\sum_{i=0}^{N} \gamma^i r(s_{t+i}, a_{t+i}) + \gamma^{N+1}\widetilde{V}(s_{t+N+1})\Big)\Big)^2\Big] \quad (9)$$

where

$$\widetilde{V}(s_{t+T+1}) = Q_{w^-}(s_{t+T+1}, a_{t+T+1}) - \log \pi_\theta(a_{t+T+1})$$

by introducing the target network $Q_{w^-}(\cdot)$ which is continuously updated by critic or behavior network $Q_w(\cdot)$. This loss is minimized to find critic parameter w. The policy improvement in actor-critic is affected by these Q-value functions. Policy updating is performed in an off-policy way similar to that in DDPG. Different from DDPG, this model-based soft actor-critic considers the whole policy with stochastic actor using the energy-based value function. The policy π is learned by minimizing the criterion

$$\begin{aligned} J_\pi(\theta) &= \mathbb{E}_{s_t\sim\mathcal{D}}[D_{\mathrm{KL}}\left(\pi_\theta(\cdot|s_t)\| \exp(Q^{\pi_{\mathrm{old}}}(s_t, \cdot))\right)] \\ &= \mathbb{E}_{s_t\sim\mathcal{D}}[\log \pi_\theta(\cdot|s_t) - Q^{\pi_{\mathrm{old}}}(s_t, \cdot)]. \end{aligned} \quad (10)$$

MSAC does not simply take the action with the largest Q-value. The online learning over the world model in MSAC is guided with the maximum entropy policy where the diversity of experience is increased to capture the structure of states. In the implementation, the issue of overestimation of critic parameter w likely happens due to the noise in calculation of temporal difference error which may incur bias information from interaction with environment. Such a bias is here reduced by adopting twin networks $\{Q_w^{(1)}, Q_w^{(2)}\}$ for critic and the corresponding networks $\{Q_{w^-}^{(1)}, Q_{w^-}^{(2)}\}$ for target where $w^- \leftarrow \tau w^- + (1-\tau)w$ is used with a parameter τ. The training stability and robustness is improved by selecting the smaller Q-value via $Q(s,a) = \min(Q^{(1)}(s,a), Q^{(2)}(s,a))$.

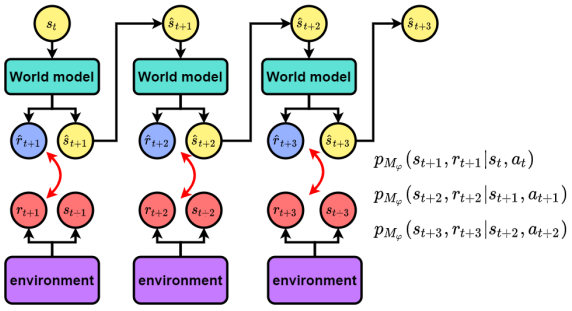


Fig. 3: Illustration for world model M_φ by using overshooting.

Importantly, MSAC builds the world model in a form of deep neural network which predicts the complicated dynamics of state and reward from time t to $t+1$ expressed by the function $\{s_{t+1}, r_{t+1}\} = M_\varphi(s_t, r_t)$. Given the world model, MSAC is able to plan by choosing the action with maximum reward. Agent can plan for future horizon and take action to achieve the maximum total reward via Monte Carlo tree search at each time step t along an interval with length H via

$$\widehat{a}_t = \arg\max_{a_t} \sum_{i=t}^{t+H} r(s_i, a_i) \tag{11}$$

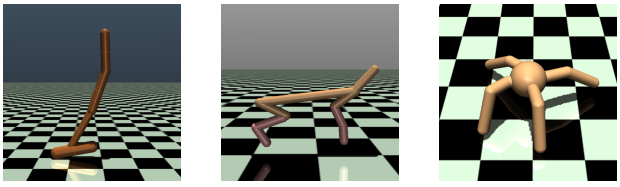
where $\{s_{t+1}, r_{t+1}\}$ are obtained from M_φ by using $\{s_t, r_t\}$ and are applied to find action a_{t+1} via maximum total reward through Monte Carlo policy optimization. If the world model M_φ predicts accurately, the dynamics of state and reward in multiple steps can be reliably calculated in a recursive way. The multi-step backup is useful to improve the estimation of value function. In this study, model predictive control is performed to allow the agent to adapt its plan based on the newest observations. The real experience data are given from the environment. These data are recycled in replay buffer which are used to update parameters φ for model M_φ. Model capacity is considered in this work. The larger model is more likely accurate but the computation cost is larger and the online learning is harder for policy. The world model with suitable size is implemented. In the implementation, the autoregressive overshooting method as illustrated in Figure 3 is merged for three-step prediction ($L = 3$) from s_t to s_{t+1}, s_{t+2} and s_{t+3}. Therefore, the probabilistic cost function for the world model $J_M(\varphi)$ is formed for minimization over

$$-\mathbb{E}_{(s_t,a_t)\sim\mathcal{D}}\Big[\sum_{i=0}^{L} \log p_{M_\varphi}(s_{t+i+1}, r_{t+i+1}|s_{t+i}, a_{t+i})\Big].$$

Algorithm 1 shows the implementation for MSAC. Parameters φ, θ and w are sequentially updated by minimizing J_M, J_π and J_w, respectively. The lengths for planning M, overshooting H and bootstrapping N are hyperparameters.

IV. Experiments

A series of experiments on MuJoCo continuous control tasks [32] were conducted and compared to evaluate the agents which were trained by using PPO [7], DDPG [11], SAC [14] and the proposed model-based SAC.



(a) (b) (c)

Fig. 4: Examples of MuJoCo environments including (a) Hopper-v2, (b) HalfCheetah-v2, and (c) Ant-v2.

A. Experimental setup

MuJoCo tasks were simulated through the environments based on the standard interaction protocol using OpenAI Gym [33] which has been widely evaluated for reinforcement learning. Figure 4 shows three examples of environments in this study. The standard settings of reward functions and state constraints were adopted. The standard interaction loop was performed to facilitate the reproducible implementation and consistent comparison with the related methods. Using MSAC, the energy-based policy based on maximum entropy framework was implemented. The actor network consisted of two hidden layers with two sets of outputs which corresponded to mean and standard deviation of a Gaussian policy. The critic network was composed of two hidden layers to find Q-value function. Each layer had 256 hidden units. The architecture of world model or MDP model was specified according to the complexity of environmental dynamics. Generally, the model with the increased complexity provided more accurate prediction of fine state structure but also required a larger amount of calculation. This study would like to investigate how a compact model was used to build the policy to learn effectively. World model adopted a similar architecture as critic network. The networks of actor, critic and world model were trained from random initialization. The parameter updating was performed by using Adam optimizer. The critic network was configured by the clipped double Q-network [34] which handled the overestimation problem in value function. This twin network independently measured the Q-value so as to efficiently predict the dynamics of state and reward in learning process. The delayed updating was applied for critic network. The target network with parameter w^- was implemented as a delayed and interpolated copy of critic network with parameter

Algorithm 1 Model-based Soft Actor-Critic

Initialize actor network $\pi_\theta(a|s)$, critic networks $Q_w^{(i)}(s,a), i \in \{1,2\}$, and world model $M_\varphi(s,a)$
Initialize replay buffer $\mathcal{D}$ and target networks $Q_{w^-}^{(i)}(s,a) \leftarrow Q_w^{(i)}(s,a), i \in \{1,2\}$
for each episode **do**
  receive initial observation state s_0
  for each time step t **do**
    select action a_t with planning by (11)
    observe reward r_t and new state s_{t+1}
    store transition (s_t, a_t, r_t, s_{t+1}) in replay buffer $\mathcal{D}$
    update world model with overshooting $\varphi \leftarrow \arg\max_\varphi \mathbb{E}[\log p_{M_\varphi}(s', r'|s, a)]$
    sample transition (s_t, a_t, r_t, s_{t+1}) from $\mathcal{D}$
    rollout multiple steps and collect on-policy data
    update actor network θ by (10) and critic network w with multi-step bootstrap (9)
    update target network w^- via $w_i^- \leftarrow \tau w_i^- + (1-\tau)w_i, i \in \{1,2\}$
  end for
end for

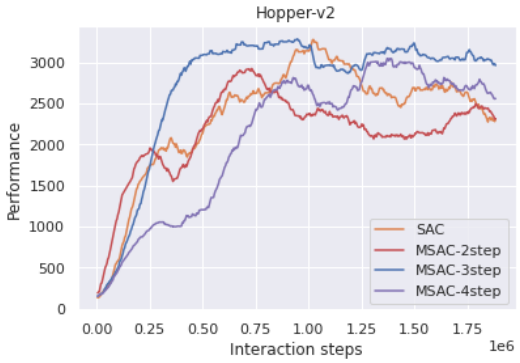


Fig. 5: SAC versus MSAC with different rollout steps for policy updating. Overshooting method is *disregarded*.

w. The delayed updating in accordance with the temporal difference error was performed to reduce the variance of the estimated value function which could avoid the sub-optimal improvement for policy. There were two interaction loops when training the agent. At the beginning, the agent was trained by only relying on the data collection from policy without knowing the ground-truth state transitions. World model acted as the virtual environment to make prediction. The delayed updating was applied to deal with the robustness issue in the predicted state transitions by world model. The schedule of delayed updating was basically controlled to predict state transitions which was slower than that of applying replay buffer. Linear replay buffer was used. The performance was evaluated in terms of total return from the environment. At the end of each training epoch, the next 10 episodes using current policy were evaluated to measure the expected return. The results on the expected return were averaged over the policy initialization using five different random seeds. RL algorithms using PPO, DDPG, SAC and MSAC were included for comparison. Each policy used a purely random exploratory for 50K steps to remove the dependencies caused by the initial parameters.

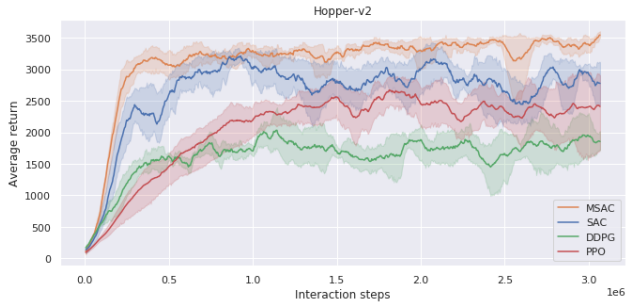


Fig. 6: Comparison of learning curves using different methods. Overshooting method is *applied*.

B. Experimental result

Different from SAC, the proposed MSAC implements the model-based reinforcement learning using the stochastic policy

TABLE I: Comparison of average returns using different methods under different tasks.

Agent	Hopper	HalfCheetah	Ant
PPO	2494.15	2947.84	2166.44
DDPG	2040.11	8902.48	2302.92
SAC	3137.44	9002.84	6197.56
MSAC	**3446.54**	**9010.74**	**6443.08**

where multi-step rollout is merged. The rollout data are viewed as an augmentation for experience replay. Figure 5 shows the effect of rollout steps in the learning procedure using MSAC. Notably, the rollout steps are performed without adopting the overshooting method. The increase rollout steps does not improve the learning stability in presence of world model. The information from world model could provide the hint to avoid policy to forget the knowledge from data examples which were learned before. Rollout steps likely improve learning efficiency. However, the bias of the rollout from world model is possibly increased to degrade the performance. Accordingly, this situation highlights the importance of overshooting method in planning for model-based SAC. The case of three rollout steps is better than other cases. In the following, the overshooting scheme is applied in MSAC. Figure 6 and Table I compares the learning curves and the average return over episodes by using different methods, respectively. The experience result shows that the proposed MSAC is faster and uses smaller number of interactions to reach the desirable performance. MSAC receives higher average return than other methods. Data efficiency is significantly improved by using the model-based SAC. The interaction steps from the environment can be reduced to achieve high-performance policy after model training. PPO performs better than DDPG in Hopper task but gets worse in other tasks. MSAC consistently obtains the best results in various tasks in the comparison.

V. Conclusions

This paper has presented the stochastic policy and world model for continuous control tasks which were jointly trained to fulfill data efficient reinforcement learning. The model-based soft actor-critic was proposed and learned from random initialization without knowing the ground-truth dynamics of the environment. Online learning of world model was performed to mimic the environment without any pretrained model. The hybrid training of soft policy and virtual environment was better than the individual training of them. From the experiments on high-dimensional reinforcement learning, sample efficiency was significantly improved when compared with the strong baseline method using the soft actor-critic. With the benefits from stochastic policy, multi-step rollout, long-term planning and overshooting for prediction, the proposed model-based SAC only took one third of interaction steps to achieve the competitive performance as obtained by the best baseline method using SAC. Curse of dimensionality was eased for efficient exploration on the states and actions. The issue of data efficiency was handled. The usefulness of reinforcement learning was improved.

REFERENCES

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[2] Jen-Tzung Chien and Po-Yen Hung, "Multiple target prediction for deep reinforcement learning," in *Proc. of Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, 2020, pp. 1611–1616.

[3] Huan-Hsin Tseng, Yi Luo, Sunan Cui, Jen-Tzung Chien, Randall K Ten Haken, and Issam El Naqa, "Deep reinforcement learning for automated radiation adaptation in lung cancer," *Medical Physics*, vol. 44, no. 12, pp. 6690–6705, 2017.

[4] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems*, 2000, pp. 1057–1063.

[5] Jen-Tzung Chien, "Deep Bayesian multimedia learning," in *Proc. of ACM International Conference on Multimedia*, 2020, pp. 4791–4793.

[6] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *Proc. of the International Conference on Learning Representations*, 2016.

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

[8] Mahdin Rohmatillah and Jen-Tzung Chien, "Causal confusion reduction for robust multi-domain dialogue policy," *Proc. of Annual Conference of International Speech Communication Association*, pp. 3221–3225, 2021.

[9] Mahdin Rohmatillah and Jen-Tzung Chien, "Corrective guidance and learning for dialogue management," in *Proc. of ACM International Conference on Information & Knowledge Management*, 2021.

[10] Chuan-En Hsu, Mahdin Rohmatillah, and Jen-Tzung Chien, "Multitask generative adversarial imitation learning for multi-domain dialogue system," in *Proc. of IEEE Automatic Speech Recognition and Understanding Workshop*, 2021.

[11] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra, "Continuous control with deep reinforcement learning.," in *Proc. of International Conference on Learning Representations*, 2016.

[12] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller, "Deterministic policy gradient algorithms," in *Proc. of International Conference on Machine Learning*, 2014.

[13] Jen-Tzung Chien and Wei Xiang Lieow, "Meta learning for hyperparameter optimization in dialogue system.," in *Proc. of Annual Conference of International Speech Communication Association*, 2019, pp. 839–843.

[14] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. of International Conference on Machine Learning*, 2018, pp. 1861–1870.

[15] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine, "Reinforcement learning with deep energy-based policies," in *Proc. of International Conference on Machine Learning*, 2017, pp. 1352–1361.

[16] Jen-Tzung Chien and Po-Chien Hsu, "Stochastic curiosity maximizing exploration," in *Proc. of International Joint Conference on Neural Networks*, 2020, pp. 1–8.

[17] Jen-Tzung Chien and Po-Chien Hsu, "Stochastic curiosity exploration for dialogue systems.," in *Proc. of Annual Conference of International Speech Communication Association*, 2020, pp. 3885–3889.

[18] Jen-Tzung Chien, Wei-Lin Liao, and Issam El Naqa, "Exploring state transition uncertainty in variational reinforcement learning," in *Proc. of European Signal Processing Conference*, 2021, pp. 1527–1531.

[19] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz, "Trust region policy optimization," in *Proc. of International Conference on Machine Learning*, 2015, pp. 1889–1897.

[20] Scott Fujimoto, Herke Hoof, and David Meger, "Addressing function approximation error in actor-critic methods," in *Proc. of International Conference on Machine Learning*, 2018, pp. 1587–1596.

[21] Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz, "Parameter space noise for exploration," in *Proc. of International Conference on Learning Representations*, 2018.

[22] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Matteo Hessel, Ian Osband, Alex Graves, Volodymyr Mnih, Remi Munos, Demis Hassabis, et al., "Noisy networks for exploration," in *Proc. of International Conference on Learning Representations*, 2018.

[23] Diederik P Kingma and Max Welling, "Auto-encoding variational Bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[24] Ofir Nachum, Mohammad Norouzi, George Tucker, and Dale Schuurmans, "Smoothed action value functions for learning gaussian policies," in *Proc. of International Conference on Machine Learning*, 2018, pp. 3692–3700.

[25] William Whitney, Rajat Agarwal, Kyunghyun Cho, and Abhinav Gupta, "Dynamics-aware embeddings," in *Proc. of International Conference on Learning Representations*, 2019.

[26] Zafarali Ahmed, Nicolas Le Roux, Mohammad Norouzi, and Dale Schuurmans, "Understanding the impact of entropy on policy optimization," in *Proc. of International Conference on Machine Learning*, 2019, pp. 151–160.

[27] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. of International Conference on Machine Learning*, 2016, pp. 1928–1937.

[28] Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu Jie Huang, "A tutorial on energy-based learning," *Predicting Structured Data*, 2006.

[29] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel, "VIME: Variational information maximizing exploration," in *Advances in Neural Information Processing Systems*, 2016, pp. 1109–1117.

[30] David Ha and Jürgen Schmidhuber, "World models," *arXiv preprint arXiv:1803.10122*, 2018.

[31] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson, "Learning latent dynamics for planning from pixels," in *Proc. of International Conference on Machine Learning*, 2019, pp. 2555–2565.

[32] Emanuel Todorov, Tom Erez, and Yuval Tassa, "MuJoCo: A physics engine for model-based control," in *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5026–5033.

[33] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba, "OpenAI gym," *arXiv preprint arXiv:1606.01540*, 2016.

[34] Hado van Hasselt, Arthur Guez, and David Silver, "Deep reinforcement learning with double Q-learning," in *Proc. of AAAI Conference on Artificial Intelligence*, 2016, pp. 2094–2100.