2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Policy Optimization by Looking Ahead for Model-based Offline Reinforcement Learning

Yang Liu, Marius Hofert
liuyang1123@connect.hku.hk mhofert@hku.hk

***Abstract*— Offline reinforcement learning (RL) aims to optimize a policy, based on pre-collected data, to maximize the cumulative rewards after performing a sequence of actions. Existing approaches learn a value function from historical data and then guide the updating of the policy parameters by maximizing the value function at a single time. Driven by the gap between maximizing the cumulative rewards of RL and the greedy strategy of existing methods, we propose an approach of policy optimization by looking ahead (POLA) to mitigate the gap. Concretely, we optimize the policy on both current and future states where the future states are predicted by a transition model. A trajectory contains numerous actions before the task is done. Performing the best action at each time does not mean an optimal trajectory in the end. We need to allow sub-optimal or negative actions occasionally. But existing methods focus on generating the optimal action at each time according to the maximizing Q-value principle. This motivates our looking ahead approach. Besides, hidden confounding factors may affect the decision making process. To that end, we incorporate the correlations among dimensions of the state into the policy, providing more information about the environment for the policy to make decisions. Empirical results on the Mujoco dataset show the effectiveness of the proposed approach.**

I. Introduction

Reinforcement learning (RL) has achieved tremendous success in a number of domains like autonomous driving [1], health [2], [3], robots [4], [5], computer games [6], [7], [8] and so on. These applications rely on extensive interactive trajectories to learn the behavior of the offline data that might be expensive, dangerous or unethical to obtain, such as in medical environments. This problem is further exacerbated when the action space is continuous and high dimensional where offline data only covers a minority of the feature space. In this case, since many aspects are not recorded, the model easily suffers from over-generalization and performs poorly in regions outside the support of the offline data. Recently, a number of offline RL methods [9], [10], [11] have been proposed to alleviate the distribution shift problem. The main paradigm is to incorporate conservatism to the offline RL algorithms to keep the learned policy within the support of the data. Model-free methods constrain the target policy to be close to the behavior policy by a metric such as KL or MMD distance, thus leading the target policy to take actions close to the data and mitigating the adverse effect of out-of-distribution (OOD) actions. Another strategy is to add a regularization item to the Q function and assign small Q-values towards the OOD actions. Although these type of methods address the distribution shift problem effectively, they favor conservative actions and suppress OOD actions, which, as a consequence, weakens the exploration ability of the agent and prevents the policy from generalizing beyond the data. What's worse, offline data are usually sub-optimal or even random, imitating such data would not result in a good policy. Recent works [12], [13] have tried to quantify the uncertainty with ensemble Q functions, then use the uncertainty to penalize corresponding Q-values to improve the reliability of Q functions when being evaluated on OOD actions. But the uncertainty estimation relies on the generalization of each Q function, which is highly unreliable on the OOD region.

In contrast, model-based methods like Dyna [14] learn a transition (dynamics) model of the environment in a supervised way, thus providing extra information about the environment. Compared with model-free methods, model-based methods augment the offline data and are a natural choice for generalization since the transition model can generate the next state based on the current state and action. The transition model is dedicated to fully reflect the dynamics of the environment. When the data are heterogeneous and contain different trajectories, the transition model can generate new rollouts to composite multiple different trajectories while model-free methods cannot provide such connections and can easily get stuck in isolated trajectories [15].

Although model-based methods, with extra state-action pairs, learn a better Q function with more interactions (thus better guide the optimization of the policy), existing model-based methods optimize the policy parameters at a single time step that focuses on searching the best action in each state. Although according to the Bellman equation [16], the Q function is expected to reflect the cumulative rewards, the policy is evolving. The best action derived from the maximizing Q-value principle only holds for the policy at the current time, without considering the future. Intuitively, we argue that this greedy search strategy may result in a sub-optimal trajectory at the end. Instead, we alleviate the greediness by looking a few time steps ahead. The looking-ahead policy is not only optimized at the current time step, but also in future time steps where the future states are predicted by the transition model. This compels the policy to maximize both the current and future returns. POLA stands out in the sparse-reward setting where the reward signal is sparse and the Q function has difficulties converging. In this case, the policy receives weak guidance from the feedback of actions. The POLA objective contains multiple Q functions that can provide more guidance for the policy. The generated

Department of Statistics and Actuarial Science, The University of Hong Kong, Hong Kong, China.

rollouts might be radical and deviate far from the ground truth when the transition model looks ahead far into the future. The countermeasure is to limit the rollout length of looking ahead. Empirically, we investigate the length of looking ahead in subsection VI-C.

Besides, we observe the correlation among the dimensions of the state and incorporate it into the diffusion model policy. Traditional policies model the mapping relation from states to actions by an uncorrelated Gaussian distribution, i.e., the state vector is mapped into a mean vector $\boldsymbol{\mu}$ and variance vector $\boldsymbol{\sigma}$, then an action is sampled from this distribution. But multivariate Gaussian distributions can have an unstructured, non-diagonal covariance matrix Σ. For brevity, existing policies assume different dimensions of the state are independent and identically distributed (i.i.d.). But this assumption violates the fact that different dimensions are in fact correlated. Take the `halfcheetah-medium-expert` environment as an example, as is shown in Figure 2, some dimensions of the state space are strongly correlated. The 3rd and 6th, and the 12th and 15th dimension have a strong negative relationship, and the 3rd and 5th dimension have a strong positive relationship. The correlations among dimensions are non-negligible when building a model for the policy. To incorporate the information of dependence among dimensions, we concatenate the product X_iX_j of different dimensions to the state vector and then feed it to the diffusion policy. We will show the effect of correlation in subsection VI-D.

In summary, our contributions are: We propose to optimize the policy by looking ahead to make the policy less greedy and to allow sub-optimal actions. And we incorporate the correlations among the dimensions of the state space into the diffusion policy to provide more information of the environment for the policy to make decisions.

II. Related Work

A number of methods have been proposed to address the offline RL [17] problem under limited data coverage. These methods can be primarily categorized as model-free and model-based approaches.

Model-free Offline RL: The typical routine of model-free offline RL methods is to constrain the policy or regularize the value function to limit the target policy within the offline data manifold. BCQ [9] models the policy as a VAE model and uses a MSE loss to constrain the generated actions to be close to actions in the data. BEAR [10] uses MMD and KL divergence to constrain the target policy. IQL [18] directly clones the behavior of the data by learning a weight for every state-action pair without querying OOD actions to clone high-quality actions and suppress low-quality actions. Some works [9], [19], [20] adopt VAEs for the policy. In LAPO [19], the VAE loss is weighted by the advantage of actions. Behavior cloning (BC) methods [21], [22] constrain the learned policy by directly cloning the relationship between the states and actions.

Model-based Offline RL: Due to the conservatism of model-free methods, many model-based methods have been proposed to improve the data efficiency. Although the offline data are augmented by the transition model, the compounding prediction bias of the next state may accumulate with increasing rollout length. Some works [23], [24] try to improve the robustness of transition models by limiting the rollout length to mitigate the compounding error. MOPO [25] and MORel [26] build a pessimistic MDP for the transition model where the reward is penalized by the uncertainty of the predicted states. Subsequently, COMBO [27] learns a lower-bound of the true Q function to implement a conservative policy. PMDB [28] proposes an alternating Markov game to promote the coverage of Q-function, then maximizing the MLE of the generated actions in a reference policy.

Diffusion Models in RL: Inspired by the success of text-to-image generation, the RL community starts to adopt diffusion models. [29] combines the DDPM [30] policy with the Q function to maximize the action values and has achieved remarkable performance in the D4RL datasets. [31] propose to imitate human behavior via a diffusion model with reliable sampling schemes. Diffusier [32] builds the the transition model as a diffusion model and uses it to generate new trajectories. AdaptDiffuser [33] proposes a diffusion planer to generate expert trajectories that are filtered by a reward-guided discriminator, so as to improve the data diversity. Different from the above diffusion policy, we incorporate the correlation information into the diffusion model and optimize the policy with extra look-ahead time steps, aiming to make the policy less greedy.

III. Preliminaries

Diffusion Model: The diffusion model [30] contains a forward process where the original data $\boldsymbol{x_0}$ is progressively diffused by standard Gaussian noise until it almost follows $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and a backward process where $\boldsymbol{x}_0$ is reconstructed by a denoising process. Assuming there are T time steps to diffuse noise into the data $\boldsymbol{x}_0$, it produces a sequence of latent variables $\boldsymbol{x}_{1:T}$. Let $\beta_{1:T}, 0 < \beta_1 < \beta_2 < \cdots < \beta_T < 1$ be the variance schedule and $\alpha_t = 1 - \beta_t$, $\bar{\alpha}_t = \alpha_1\alpha_2...\alpha_t \to 0, \boldsymbol{z}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. The diffusion process constructs the relation between $\boldsymbol{x}_{t-1}$ and $\boldsymbol{x}_t$ by $\boldsymbol{x}_t = \sqrt{1-\beta_t}\boldsymbol{x}_{t-1} + \beta_t\boldsymbol{z}_t$. Repeat above process $T-1$ times, then the relation between $\boldsymbol{x}_t$ and $\boldsymbol{x}_0$ is $\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{z}_t$. Since $\bar{\alpha}_t \to 0$, $\boldsymbol{x}_t$ almost follows $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$. In the reverse process, based on the Bayesian equation,

$$p\left(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t\right) = q\left(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}\right)\frac{q\left(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0\right)}{q\left(\boldsymbol{x}_t|\boldsymbol{x}_0\right)} \sim \mathcal{N}(\tilde{\mu}_t,\ \tilde{\beta}_t)$$
$$\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\varepsilon}_\theta\right), \quad \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t \quad (1)$$

where $\boldsymbol{\varepsilon}_\theta$ is a neural network predictor whose input is $\boldsymbol{x}_t$ and $\boldsymbol{\varepsilon}$. We will sample $\boldsymbol{x}_{t-1}$ based on $\boldsymbol{x}_t$ according to Equation 1 and finally the $\boldsymbol{x}_0$ is the generated sample. The training objective is to predict the noise added to the input data. The surrogate loss [30] is

$$\mathcal{L}(\theta) = \mathbb{E}_{x_0,\varepsilon}\left[\|\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta\left(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\varepsilon}, t\right)\|^2\right] \quad (2)$$

where $\boldsymbol{\varepsilon}$ is a sample of Gaussian noise and t is randomly sampled from the set $1, 2, \cdots, T$.

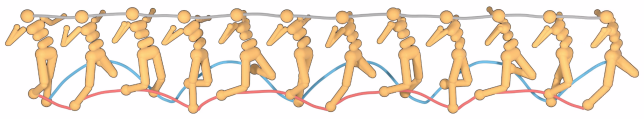

Fig. 1: The trajectory of a human walking, it contains both optimal and sub-optimal actions

IV. Policy Optimization by Looking Ahead

The policy training objective of existing methods tries to generate an optimal action to maximize the Q-value at a single time step:

$$\mathcal{L}(\theta) = -Q_\phi(s_t, \hat{a}_t), \quad \hat{a}_t = \pi_\theta(s_t).$$

We argue that this objective is greedy and can be improved by looking ahead into the future. A trajectory usually contains numerous actions before reaching the terminal state. Performing the best action at each time step doesn't necessarily lead to the best trajectory in the end. Just like in Fig. 1, the trajectory is optimal but it contains both good and bad actions, i.e., lifting up or putting down a leg. But finally, the man can walk normally and the trajectory is optimal. The example of a human walk demonstrates that the agent needs to perform suboptimal or even negative actions occasionally to obtain a maximum cumulative reward. The greedy optimization strategy violates the real target. Therefore, we propose a policy optimization looking ahead (POLA) to alleviate the greediness and narrow the gap between the training objective and the cumulative goal of RL. However, in future time points, the states are unknown. We use a transition model to predict the future state $\hat{s}_{t+1}$ based on the current state s_t and action a_t. Then we get the next action $\hat{a}_{t+1} = \pi_\theta(\hat{s}_{t+1})$. Based on $\hat{s}_{t+1}$ and $\hat{a}_{t+1}$, we can again predict $\hat{s}_{t+2}$ by p_ψ and $\hat{a}_{t+2}$ by policy π_θ:

$$\begin{aligned} &\text{for} \;\; t \;\; \text{from} \;\; 1 \;\; \text{to} \;\; T: \\ &\quad \hat{s}_{t+1}, \;\; \hat{r}_t = p_\psi(\hat{s}_t, \hat{a}_t) \\ &\quad \hat{a}_{t+1} = \pi_\theta(\hat{s}_{t+1}) \\ &\quad \hat{s}_t, \hat{a}_t \leftarrow \hat{s}_{t+1}, \hat{a}_{t+1} \end{aligned}$$

The transition model $p_\psi(s_{t+1}|s_t, a_t)$ is trained in a supervised way, where (s_t, a_t) denotes the input and s_{t+1} the output. The transition model can be an MLP or a diffusion model; in this work, we use the former. Theoretically, we can predict the state and action at any future time step by p_ψ and π_θ. But with the rollout length T increasing, the prediction error will accumulate and the predicted states deviate from real states further and further, so the predicted dynamics become more and more unreliable. So T should be limited to a rather small value.

To improve upon the behavior policy, we maximize the Q-values to optimize the policy parameters by looking ahead for T time steps via

$$L(\theta) = -Q_\phi(s_t, \hat{a}_t) - \sum_{i=1}^{T} \gamma^i Q_\phi\left(\hat{s}_{t+i}, \hat{a}_{t+i}\right) \qquad (3)$$

where $\hat{a}_{t+i} = \pi_\theta(\hat{s}_{t+i}), \hat{s}_{t+i} = p_\psi(\hat{s}_{t+i-1}, \hat{a}_{t+i-1}), 1 \leq i \leq T$; in Eq. (3), the second item means looking ahead T time steps. As a result, there will be $T+1$ Q functions to jointly guide the policy optimization. This joint optimization will enforce the policy to not only maximize the reward at the current time step, but also maximize the reward at future T time steps. Looking ahead narrows the gap between the cumulative rewards and the greedy strategy, the latter resulting in a policy that might easily get trapped in a local optimum. This is important when the offline data exhibit multi-modality where there are multiple sub-optimal solutions. To reach the optimal solution, the policy should not be too greedy and should pass on from these sub-optimal solutions. Eq. (3) helps avoids the problem of short-sightedness, which only performs the best action based on current situation without considering the future. By maximizing the Q-values across multiple time steps, the agent can consider not only the current but also the future. By incorporating the unknown states generated from the transition model into the policy objective, the agent has a better generalization ability since the agent has seen states that never appear in the offline data.

Pessimistic Q-Learning: The Q function is used to guide the policy parameters updating. In order to improve the accuracy of the Q function, we adopt the pessimistic mechanism to alleviate the adverse effect of OOD actions with high uncertainty. The Q function needs to satisfy the Bellman equation $Q(s_t, a_t) = r_t + \gamma Q(s_{t+1}, a_{t+1})$, so we use the temporal difference (TD) loss

$$\mathcal{L}(\phi) = \left[Q_{\phi_k}(s_t, a_t) - r_t - \gamma \min_{1 \leq j \leq N} Q_{\phi_j'}(s_{t+1}, \hat{a}_{t+1})\right]^2$$

where $Q_{\phi_j'}$ models the target Q-value and ϕ_j' is updated by a moving average via $\phi' = (1-\tau)\,\phi' + \tau\phi$ where τ is a small value like 0.005. The final target Q-value is the minimum of N ensemble values. We optimize the parameters ϕ to enforce them to be close to the pessimistic target Q-value. In the implementation, we use N ensemble neural networks (NNs) to estimate the Q-values. Assuming each Q-value follows $\mathcal{N}(\mu, \sigma)$, the expectation of the minimum of N Q-values [34] is

$$\mathbb{E}[\min_{1 \leq j \leq N} Q_{\phi_j'}(s, a)] \approx \mu - \Phi^{-1}\Big(\frac{N - \pi/8}{N - \pi/4 + 1}\Big)\sigma$$

As we can see, the expectation of the minimum is penalized by the uncertainty σ. It results in small Q-values for highly uncertain actions, thus alleviating the over-estimation problem for OOD actions. This penalization will result in a policy that favours in-distribution actions and suppresses highly uncertain OOD actions. So the pessimistic mechanism is a conservative mechanism. Although we use ensemble NNs to estimate the Q-value, it doesn't take much extra training time. In fact, the ensemble Q functions take roughly the same training time as a single Q function. In the implementation, we set the ensemble size to be 10 and combine multiple NNs into one. For a single NN, the weight's shape of each layer is (input_dim, output_dim). For ensemble NNs, the shape is

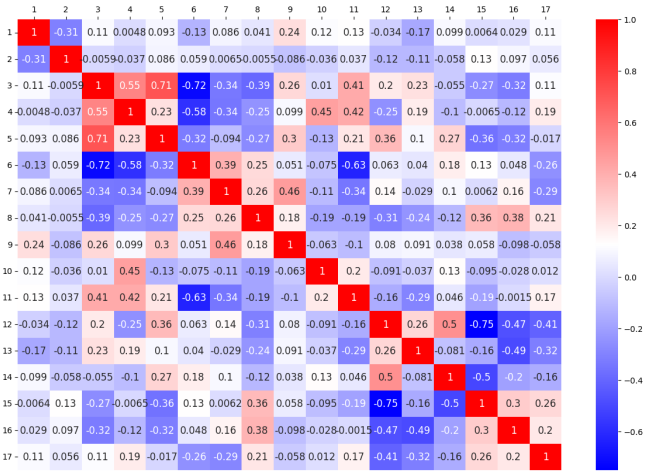

Fig. 2: Heatmap of correlations among all dimensions of the state space. The correlation is computed for the whole `halfcheetah-medium-expert` data in D4RL.

(ensemble_size, input_dim, output_dim), so for the input, we only need to expand the 0-th dimension of the input to turn its shape into (ensemble_size, batch_size, input_dim).

In addition, we use the K smallest (topK) values of the ensemble Q-values to stabilize the Q-learning and mitigate the sparseness of the minimum Q-value where only one Q function can be updated while the rest keeps fixed. The final policy objective is

$$\begin{aligned}\mathcal{L}_q(\theta) = & -\frac{1}{K} topK\left\{Q_{\phi_j}(s_t, \hat{a}_t)\right\}_{j=1}^{N} \\ & -\sum_{i=1}^{T}\gamma^i \frac{1}{K} topK\left\{Q_{\phi_j}(\hat{s}_{t+i}, \hat{a}_{t+i})\right\}_{j=1}^{N}, \end{aligned} \tag{4}$$

where $\hat{a}_{t+i}$ and $\hat{s}_{t+i}$ are generated from the policy and transition model and N is the ensemble size. When the ensemble size N is large and we still just use the minimum of the ensemble Q-values, then the policy parameters in the remaining $N-1$ Q functions won't be updated, causing the learning process to be unstable. As a result, the reward curve is unstable. In a real environment, the agent needs to act in a stable manner; in some environments such as medical experiments and autonomous driving, stability can even be more important than performance. For the topK loss approach, we update K Q functions instead of a just a single one, thus avoiding the sparseness of the minimum Q-value and improving the stabilization of the learning process.

V. Incorporate Correlation into Diffusion Policy

Figure 2 shows the correlation heatmap of the states for the `halfcheetah-medium-expert` environment. A "half cheetah" consists of a front leg and a back leg. Each leg consists of a thigh, shin and foot. Intuitively, if a part of the cheetah moves, other parts will be affected. For the half cheetah environment, the state and action dimensions are 6 and 17, respectively. The 6D action stands for the torques applied on the back/front rotor of thigh, shin and foot. When one of the rotor moves, the other rotors will also move. As a result, the state of different parts are correlated. As is shown in Figure 2, the 3rd dimension is strongly correlated with the 5th and 6th dimensions. For half cheetah, the 3rd, 5th and 6th dimensions model the angle of the second rotor in the back thigh and the velocity of the tip along the x-axis and y-axis. Other dimensions are also correlated as is shown in the heatmap.

Existing methods treat different dimensions as i.i.d. and use a multivariate Gaussian distribution whose covariance matrix was diagonal to model the mapping from states to actions. If we consider all pairwise correlations among the dimension of the state and incorporate this information into the state vector, then the state vector can better represent the environment and provide more information for the agent to make a decision. In order to utilize the unstructured correlation matrix, we adopt the idea of building products of different dimensions to reflect the hidden confounding factors. Assume the state is $(X_1, X_2, \cdots, X_n)$, in order to utilize the correlation between X_i and X_j, we incorporate all terms of the form X_iX_j $(i \neq j)$ at the end of the state vector:

$$\underbrace{(X_1, X_2, ..., X_n)}_{[b,\ n]} \cup \underbrace{(X_1X_2, X_1X_3, ..., X_{n-1}X_n)}_{[b,\ n(n-1)/2]} \ := \ s_{new}$$

The shape of s_{new} is $[b,\ n+n(n-1)/2]$ where b is the batch size. After the concatenation, s_{new} contains more information about the environment, thereby the policy $\pi(a|s_{new})$ can make better decisions. In order to generate an action conditioned on the new state s_{new}, we feed s_{new} into the diffusion model. According to Eq. 1, $\boldsymbol{x}_t$ models the action sampled at time step t, which is done via the reverse process

$$\boldsymbol{a}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{a}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\varepsilon}_\theta(\boldsymbol{a}_t, s_{new}, t)\right) + \sqrt{\tilde{\beta}_t}\boldsymbol{z}, \tag{5}$$

where $\tilde{\beta}_t$ is as in Eq. 1 and $\boldsymbol{z}$ is a sample of $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$. $\boldsymbol{\varepsilon}_\theta$ is a noise-prediction model implemented as an MLP in our work. The action is sampled step by step. If the number of denoising steps is T, then, at the beginning, $\boldsymbol{a}_T$ is sampled from $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and, thereafter, $\boldsymbol{a}_{T-1}$ is sampled according to Eq. 5. Iterate this procedure $T-1$ steps to obtain the final action $\boldsymbol{a}_0$ being generated from the diffusion policy π_θ.

The loss is similar to Eq. 2, but with an extra state s_{new}, via

$$\mathcal{L}_d(\theta) = \mathop{\mathbb{E}}_{\boldsymbol{a}\sim\mathcal{D}}\left[\|\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\boldsymbol{a} + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\varepsilon}, s_{new}, t\right)\|^2\right], \tag{6}$$

where t is sampled from a discrete uniform distribution over $1, 2, \cdots, T$. This loss guides the diffusion policy to clone the behavior of the data. In order to improve upon the behavior policy, we combine the maximizing Q-value loss in Eq. 4 with the diffusion loss in Eq. 6, the final policy loss is then

$$\mathcal{L}(\theta) = \mathcal{L}_d(\theta) + \mathcal{L}_q(\theta)$$

This new loss will not only force the policy to generate an action to maximize the cumulative reward, but also to capture

the multi-modality of the data. Compared with the traditional uncorrelated Gaussian policy, the diversity of the generated actions of the diffusion policy helps to improve the agent's generalization and exploratory abilities.

VI. Experiments

A. Datasets

We use the MuJoCo dataset of D4RL [35] to evaluate our POLA approach. MuJoCo contains the `halfcheetah`, `hopper`, `walker2d` environments. In each type, we use the `medium-expert`, `medium-replay`, `medium`, `random` environments. `Medium-expert` means an equal mixture of expert and medium data, generated from the SAC policy. `Medium-replay` means that the whole replay buffer was recorded before the policy reaches a medium level performance. `Random` means the actions are generated by unrolling a randomly initialized policy. The targets of all three environments are to make the agent run as fast as possible by applying torques on the joints.

B. Results

From the results of our experiments, we want to answer following questions:
(1) How does looking ahead affect the performance and what is the best rollout length for each environment?
(2) How does the correlation information affect the convergence speed and the performance overall?

Table I shows the results of our POLA approach on the MuJoCo dataset. For a fair comparison with existing publications, the values are normalized via $100 * \frac{\texttt{score}-\texttt{random\_score}}{\texttt{expert\_score}-\texttt{random\_score}}$. Each value is the average taken over 10 replications As usual, the value after the "$\pm$" symobl is the sample standard deviation computed from the 10 episodes. We compare our POLA approach with several other approaches of recent years, including CQL [11], RAMBO [36], COMBO [27], IQL [18], MOPO [25], EDAC [12] and two diffusion-based methods, namely Diffusion-QL [29] and AdaptDiffuser [33]. MOPO, RAMBO and COMBO are model-based methods without the looking ahead feature. By comparison with these methods, it should be easy to see the effect of looking ahead. EDAC is a regularization method of the value function in the model-free scope, it also uses ensemble Q functions. IQL is a typical behavior cloning method. AdaptDiffuser uses a diffusion model as the transition model to incorporate the future and extract high quality trajectories to augment the offline data. Diffusion-QL uses a diffusion policy to improve the expressiveness of the policy. Comparing POLA with MOPO, MORel and COMBO, we can see there is a large performance gap. We attribute this gap to the ensemble Q functions and the looking ahead. When comparing POLA with EDAC that also uses ensemble Q functions, we can see that POLA outperforms EDAC consistently in most of the environments which corroborates the effectiveness of our looking ahead approach. Finally, the comparison with diffusion-based methods mainly shows the importance of incorporating the correlation information. In subsection VI-C, we will investigate the effect of specific rollout lengths on the performance.

C. Ablation Results about Rollout Length

We want to investigate how far we should look ahead into the future. Figure 3 shows the performance according to different rollout lengths. We set the rollout length T to be 0, 1, 3, 5 where $T = 0$ means the agent doesn't look ahead. From Figure 3, we can see that, for most environments, the performance of $T > 0$ is better than that of $T = 0$. Only in the `halfcheetah-medium-replay` and `halfcheetah-medium` environments, $T > 0$ performs similar to $T = 0$. Therefore, the results demonstrate that looking ahead brings benefits for the performance. From Figure 3, we also find that convergence gets faster when T increases. In `walker2d-medium-expert` and `walker2d-medium` environments, the convergence for $T = 5$ is much faster than that for $T = 0, 1$. We attribute the fast convergence to the looking ahead strategy that optimizes the policy as a whole in the beginning. For the `walker2d-medium-replay` environment, the training of $T = 0$ suffers from serious over-fitting while the looking ahead strategy doesn't suffer from it.

The rollout length T is a hyper parameter that needs to be carefully tuned. When T increases to a large value, the performance starts to decrease. For the `halfcheetah-medium-expert` environment, the curve suddenly collapses when $T = 5$. An interpretation is that the transition model becomes more and more unreliable and the predicted state deviates far from true state, which leads to an optimization of the policy along a biased trajectory. When T is large, this adverse effect becomes considerable and the performance deteriorates seriously.

D. Ablation Results about the Correlation

We compare the performance of the policy with and without the information of the terms X_iX_j $(1 \leqslant i, j \leqslant N)$ to investigate the effect of incorporating correlations. Figure 4 shows the results on the `halfcheetah-medium-expert` environment. In Figure 4, the policy with correlation information (red line) converges the faster than the other policies, meaning that a policy with correlation information can be advantages concerning speed of convergence. Because the item X_iX_j carries extra information underlying the state $s = (X_1, X_2, ..., X_N)$, taking the original state and X_iX_j as input can better represent the dynamics of the environment, providing more information for the policy to make better decisions. From the figure, we also see that correlations shouldn't be added to the Q functions, the performance of the purple line degrades seriously when adding X_iX_j to the state of Q functions. We also tried to add the covariance matrix of the state space to the input of the policy, but it turns out that the performance degrades seriously like the yellow line. The interpretation is that we calculate the covariance matrix for each batch of data (so the covariance matrix doesn't contain the correlation of X_i and X_j explicitly).

TABLE I: Comparison of our POLA approach against other approaches on the MuJoCo v2 environments.

	Task Name	CQL	RAMBO	COMBO	IQL	MOPO	Diffusion-QL	AdaptDiffuser	EDAC	**POLA**
halfcheetah	medium-expert	62.4	79.3	90.0	86.7	86.9	96.8	89.6	106.3	**111.2± 2.4**
	medium-replay	46.2	67.0	55.1	44.2	52.2	47.8	38.3	61.3	**62.1± 1.3**
	medium	44.4	71.0	54.2	47.4	58.4	51.1	44.2	65.9	**71.8± 0.4**
	random	35.4	33.5	**38.8**	-	-	-	-	28.4	33.7 ± 1.9
hopper	medium-expert	111.0	89.5	111.1	91.5	99.3	111.1	**111.6**	110.7	111.3± 1.7
	medium-replay	48.6	97.6	73.1	94.7	100.2	101.3	92.2	101.0	**102.1± 1.8**
	medium	58.0	91.2	94.9	66.2	86.0	90.5	96.6	**101.6**	92.3± 4.2
	random	10.8	15.5	17.9	-	-	-	-	25.3	**31.6± 0.8**
walker2d	medium-expert	98.7	63.1	96.1	109.6	112.0	110.1	108.2	114.7	**118.7± 0.2**
	medium-replay	32.6	88.5	56.0	73.8	**91.6**	95.5	84.7	87.1	89.2± 1.3
	medium	79.2	89.1	75.5	78.3	86.4	87.0	84.4	92.5	**104.3± 0.6**
	random	7.1	0.2	7.0	-	-	-	-	16.6	**21.6± 1.4**

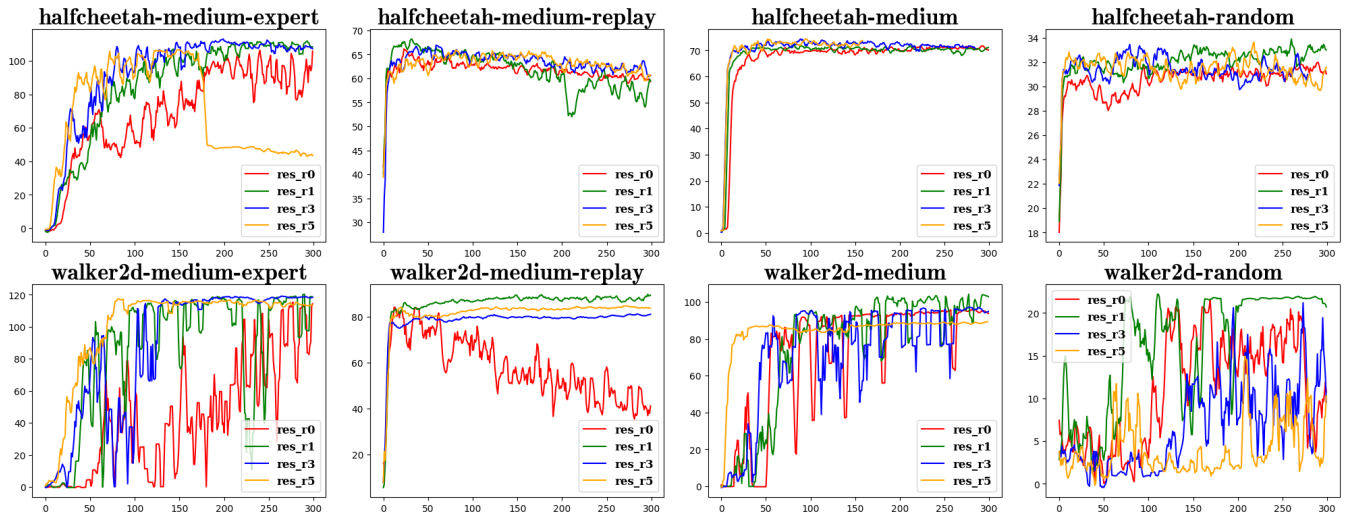

Fig. 3: The impact of different rollout lengths on the performance. "r = 0" means there is no rollout, so the policy degrades to the traditional SAC policy. The x-axis shows the training iteration and the y-axis the corresponding normalized reward.

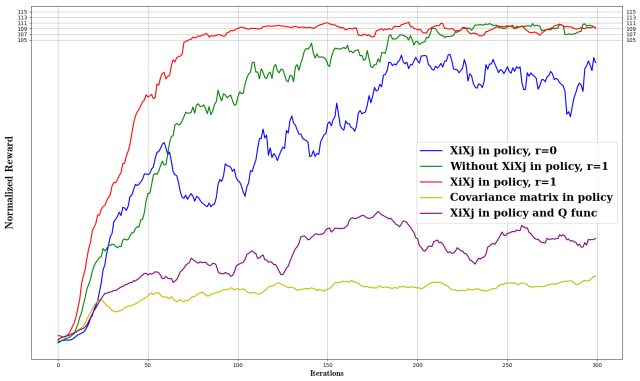

Fig. 4: The effect of correlation. The policy incorporating all correlations converges faster than those without doing so. "r=0" means there is no rollout.

VII. Conclusion

We propose to optimize the policy by looking ahead (POLA) and incorporating correlation. Different from traditional policies that conduct optimization at a single time step, our approach additionally optimizes the policy at T future time steps, which alleviates the greediness of traditional policies. We use a transition model $p(s_{t+1}|s_t, a_t)$ to predict future states, and generate future actions by the policy π_θ. After generating $(\hat{s}_t, \hat{a}_t), 1 \leqslant t \leqslant T$, the policy is optimized by maximizing the $T+1$ Q functions as a whole. Empirical results demonstrate that the looking ahead strategy helps to improve the performance in most MuJoCo environments. However, due to the unreliability of the transition model, the rollout length T shouldn't be too large. When T increases, the predicted states deviate from the true states further and further, leading the optimization procedure in the wrong direction. Besides, the looking ahead approach also helps to speed-up the convergence of the reward curve. We attribute it to the less greedy optimization which helps finding the fastest way to the optimal parameters by optimizing the $T+1$ Q functions as a whole. Besides, we incorporate the correlation of the state space into the diffusion policy, which provides more information about the environment and helps the reward curve to converge faster. The rollout length can be longer if the transition model is more accurate in predicting future states, so the transition model can look ahead further.

REFERENCES

[1] J. Chen, S. E. Li, and M. Tomizuka, "Interpretable end-to-end urban autonomous driving with latent deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 5068–5078, 2021.

[2] A. W. Senior, R. Evans, J. Jumper, J. Kirkpatrick, L. Sifre, T. Green, C. Qin, A. Žídek, A. W. Nelson, A. Bridgland, *et al.*, "Improved protein structure prediction using potentials from deep learning," *Nature*, vol. 577, no. 7792, pp. 706–710, 2020.

[3] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, *et al.*, "Highly accurate protein structure prediction with alphafold," *Nature*, vol. 596, no. 7873, pp. 583–589, 2021.

[4] S. Dasari, F. Ebert, S. Tian, S. Nair, B. Bucher, K. Schmeckpeper, S. Singh, S. Levine, and C. Finn, "Robonet: Large-scale multi-robot learning," in *Proceedings of the Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 100. PMLR, 2020, pp. 885–897.

[5] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson, "Implicit behavioral cloning," in *Proceedings of the 5th Conference on Robot Learning*, 2022, pp. 158–168.

[6] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, *et al.*, "Mastering atari, go, chess and shogi by planning with a learned model," *Nature*, vol. 588, no. 7839, pp. 604–609, 2020.

[7] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[8] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[9] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in *International conference on machine learning*. PMLR, 2019, pp. 2052–2062.

[10] A. Kumar, J. Fu, M. Soh, G. Tucker, and S. Levine, "Stabilizing off-policy q-learning via bootstrapping error reduction," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[11] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 1179–1191.

[12] G. An, S. Moon, J. H. Kim, and H. O. Song, "Uncertainty-based offline reinforcement learning with diversified q-ensemble," in *Proceedings of the 34th Advances in neural information processing systems*, 2021, pp. 7436–7447.

[13] C. Bai, L. Wang, Z. Yang, Z.-H. Deng, A. Garg, P. Liu, and Z. Wang, "Pessimistic bootstrapping for uncertainty-driven offline reinforcement learning," in *International Conference on Learning Representations*, 2022.

[14] R. S. Sutton, "Dyna, an integrated architecture for learning, planning, and reacting," *ACM Sigart Bulletin*, vol. 2, no. 4, pp. 160–163, 1991.

[15] J. Wang, W. Li, H. Jiang, G. Zhu, S. Li, and C. Zhang, "Offline reinforcement learning with reverse model-based imagination," in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 29 420–29 432.

[16] E. Barron and H. Ishii, "The bellman equation for minimizing the maximum cost." *NONLINEAR ANAL. THEORY METHODS APPLIC.*, vol. 13, no. 9, pp. 1067–1090, 1989.

[17] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," in *arXiv preprint arXiv:2005.01643*, 2020.

[18] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit q-learning," in *International Conference on Learning Representations*, 2022.

[19] X. Chen, A. Ghadirzadeh, T. Yu, J. Wang, Y. Gao, W. Li, L. Bin, C. Finn, and C. Zhang, "LAPO: Latent-variable advantage-weighted policy optimization for offline reinforcement learning," in *Advances in Neural Information Processing Systems*, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds., vol. 35, 2022.

[20] J. Wu, H. Wu, Z. Qiu, J. Wang, and M. Long, "Supported policy optimization for offline reinforcement learning," in *Advances in Neural Information Processing Systems*, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds., vol. 35, 2022.

[21] J. Lee, W. Jeon, B. Lee, J. Pineau, and K.-E. Kim, "Optidice: Offline policy optimization via stationary distribution correction estimation," in *International Conference on Machine Learning*. PMLR, 2021, pp. 6120–6130.

[22] J. Lee, C. Paduraru, D. J. Mankowitz, N. Heess, D. Precup, K.-E. Kim, and A. Guez, "Coptidice: Offline constrained reinforcement learning via stationary distribution correction estimation," in *International Conference on Learning Representations*, 2022.

[23] B.-J. Lee, J. Lee, and K.-E. Kim, "Representation balancing offline model-based reinforcement learning," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=QpNz8r_Ri2Y

[24] T. Hishinuma and K. Senda, "Weighted model estimation for offline model-based reinforcement learning," in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 17 789–17 800.

[25] T. Yu, G. Thomas, L. Yu, S. Ermon, J. Y. Zou, S. Levine, C. Finn, and T. Ma, "Mopo: Model-based offline policy optimization," *Advances in Neural Information Processing Systems*, vol. 33, pp. 14 129–14 142, 2020.

[26] R. Kidambi, A. Rajeswaran, P. Netrapalli, and T. Joachims, "Morel: Model-based offline reinforcement learning," *Advances in neural information processing systems*, vol. 33, pp. 21 810–21 823, 2020.

[27] T. Yu, A. Kumar, R. Rafailov, A. Rajeswaran, S. Levine, and C. Finn, "Combo: Conservative offline model-based policy optimization," *Advances in neural information processing systems*, vol. 34, pp. 28 954–28 967, 2021.

[28] K. Guo, S. Yunfeng, and Y. Geng, "Model-based offline reinforcement learning with pessimism-modulated dynamics belief," *Advances in Neural Information Processing Systems*, vol. 35, pp. 449–461, 2022.

[29] Z. Wang, J. J. Hunt, and M. Zhou, "Diffusion policies as an expressive policy class for offline reinforcement learning," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=AHvFDPi-FA

[30] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," vol. 33, 2020, pp. 6840–6851.

[31] T. Pearce, T. Rashid, A. Kanervisto, D. Bignell, M. Sun, R. Georgescu, S. V. Macua, S. Z. Tan, I. Momennejad, K. Hofmann, and S. Devlin, "Imitating human behaviour with diffusion models," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=Pv1GPQzRrC8

[32] M. Janner, Y. Du, J. Tenenbaum, and S. Levine, "Planning with diffusion for flexible behavior synthesis," in *Proceedings of the 39th International Conference on Machine Learning*, vol. 162, 2022, pp. 9902–9915.

[33] Z. Liang, Y. Mu, M. Ding, F. Ni, M. Tomizuka, and P. Luo, "AdaptDiffuser: Diffusion models as adaptive self-evolving planners," in *Proceedings of the 40th International Conference on Machine Learning*, vol. 202. PMLR, 2023, pp. 20 725–20 745.

[34] J. Royston *et al.*, "Expected normal order statistics (exact and approximate)," *Journal of the Royal Statistical Society Series C (Applied Statistics)*, vol. 31, no. 2, pp. 161–165, 1982.

[35] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4rl: Datasets for deep data-driven reinforcement learning," in *arXiv preprint arXiv:2004.07219*, 2020.

[36] M. Rigter, B. Lacerda, and N. Hawes, "Rambo-rl: Robust adversarial model-based offline reinforcement learning," *Advances in neural information processing systems*, vol. 35, pp. 16 082–16 097, 2022.