Reinforcement Learning with Diffusion

Yibin Hu, Daniel Wang, Raymond Liu
Tulane University

Abstract

We replicated and tested the application of diffusion model on reinforcement planning. Specifically, diffusion Model for RL[1] is deployed on Walker2D task to generate future trajectory given the initial state. Our findings reveals a critical issue of the practice that combines generative model and RL, as the generative model with complete expressive capability is combined with RL setting in which real interaction between agent and environment could easily produce novel state-action distribution, resulting in underperformaning outcomes. This project is hosted in https://github.com/yibhu/Generative-RL/tree/main.

1 Introduction

The application of generative methods in reinforcement learning (RL) aligns closely with our intuitive understanding of how agents should interact with their environments. When presented with a current state that encapsulates the environment at a specific moment, an intelligent agent naturally processes this information to predict and plan a sequence of future states and actions. Such a setting matches the core principles of generative modeling, making it a motivated option for combining RL and generative model.

Among various generative models developed in recent years, diffusion model, in particular, is well-suited for this purpose as they can effectively capture the complex representation of state-action distribution. By leveraging their capacity to sample diverse multimodal distribution, diffusion model provides a robust framework for learning agent behavior in various tasks. Specifically, we investigate the Walker2D task to gain insights into how these models can enhance decision-making processes in RL.

2 Background

Given a system with state transition dynamics $s_{t+1} = f(s_t, a_t)$, its trajectory is sequence of state action pair

$$\boldsymbol{\tau} = \begin{bmatrix} \mathbf{s}_0 & \mathbf{s}_1 & \dots & \mathbf{s}_T \\ \mathbf{a}_0 & \mathbf{a}_1 & & \mathbf{a}_T \end{bmatrix}$$

and the actions are chosen to maximize the objective reward function $r(s_t, a_t)$ over the trajectory.

$$\mathbf{a}_{0:T}^* = \underset{\mathbf{a}_{0:T}}{\arg\max} \mathcal{J}\left(\mathbf{s}_0, \mathbf{a}_{0:T}\right) = \underset{\mathbf{a}_{0:T}}{\arg\max} \sum_{t=0}^{T} r\left(\mathbf{s}_t, \mathbf{a}_t\right)$$

The diffusion model[2] is trained by learning to generate data τ^0 from a iterative denoising process $p_\theta\left(\boldsymbol{\tau}^{i-1} \mid \boldsymbol{\tau}^i\right) = \mathcal{N}\left(\boldsymbol{\tau}^{i-1} \mid \mu_\theta\left(\boldsymbol{\tau}^i, i\right), \Sigma^i\right)$, where the upper script index denotes the diffusion time step, and this is reverse of diffusion process $q(\tau^i|\tau^{i-1})$, which is a iterative process that replace information with random noise and eventually converts the data into Gaussian distribution τ^N. The generated data distribution of the model is described as:

$$p_\theta\left(\boldsymbol{\tau}^0\right) = \int p\left(\boldsymbol{\tau}^N\right) \prod_{i=1}^{N} p_\theta\left(\boldsymbol{\tau}^{i-1} \mid \boldsymbol{\tau}^i\right) \mathrm{d}\boldsymbol{\tau}^{1:N}$$

The model parameters θ is optimized by minimizing the variation lower bound of $p_\theta(\tau^0)$ [3], $\theta^* = \arg\min_\theta -E_{\boldsymbol{\tau}^0}\left[\log p_\theta\left(\boldsymbol{\tau}^0\right)\right]$

3 Methods

Given initial condition c, diffusion model is framed as a trajectory generator $p_\theta(\tau|c)$ that generates future trajectory based on c. The condition is flexible to choose to adapt to various problem setting, c is chosen to be the initial state s τ_t in out Walker2D replication, and the planninng can be carried out in an autoregressive as:

$$\tau_t \leftarrow p_\theta(\tau_t|c_t)$$
$$c_{t+1} = \tau_{t,s_0}$$

*

When deployed in environment, the agent generates future trajectory and execute the associated actions from the planned trajectory, resulting in a sequence of real states that is different from the generated states predicted by the model.

In detail, the algorithm consists diffusion training part and denoising inference part. In training, a ground truth trajectory is first sampled from dataset and noised to a random diffusion step associated with a random noise $\boldsymbol{\epsilon}$. The model is tasked to estimate this noise given the noised trajectory and diffusion time step, and the process repeats until convergence. In inference, random noise drawn from Normal distribution is iteratively denoised while conditioning is applied at each step.

Algorithm 1.1 Diffusion: Training of Diffusion $p_\theta(\boldsymbol{\tau}|s)$

$\{\boldsymbol{\tau}\} \leftarrow SampleTrajectories()$
schedule $\{\beta_t\}$
while θ not converge **do**
  $\boldsymbol{\tau^0} \leftarrow sample(\{\boldsymbol{\tau}\})$ ▷ $shape(\boldsymbol{\tau}) = (T+1, dim(s)+dim(a))$
  $t \leftarrow Uniform(\{1, ..., T_d\})$ ▷ t is diffusion time not planning time
  $\boldsymbol{\epsilon} \leftarrow \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
  $\boldsymbol{\tau}^t \leftarrow \sqrt{\bar{\alpha}_t}\boldsymbol{\tau}^0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$ ▷ $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i, \alpha_t = 1 - \beta_t$
  $s_0^t \leftarrow s_0^0$ ▷ condition on noiseless initial state s_0^0
  $\theta \leftarrow \theta - \eta\nabla_\theta \left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\boldsymbol{\tau}^t, t\right)\right\|^2$

Algorithm 1.2 Denoising: Sampling of Diffusion $p_\theta(\boldsymbol{\tau}|s)$

$\boldsymbol{\tau}^{T_d} \leftarrow \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
$s_0^t \leftarrow s$
for $t = T_d, ..., 1$ **do**
  $\mathbf{z} \leftarrow \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ if $t > 1$, else $\mathbf{z} = 0$
  $\boldsymbol{\tau}^{t-1} \leftarrow \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{\tau}^t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{\tau}^t, t)\right) + \sigma_t\mathbf{z}$ ▷ $\sigma_t^2 = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$
  $s_0^{t-1} \leftarrow s$
return $\boldsymbol{\tau}^0$

Although the reward is not used in the above procedure, the diffusion model could generate trajectory that approximate that of dataset distribution. To further shape the trajectory generation to specific goals, a reward function $r_\phi(\tau)$ can be trained on dataset reward or any exteranlly imposed reward, and then the trajectory is steered towards higher rewarding trajectory in inference by classifier guidance as:

$$\tau^t \leftarrow \tau^t + \kappa \nabla_{\tau_t} r_\phi(\tau_t)$$

where κ is guidance scale. This approach is not necessarily compatible with diffusion, as it apply external changes to trajectory generation, and this could exacerbate the distribution shift problem.

4 Experiment

The model is trained with window size being 8, meaning it generate action at current state and future 7 state-action based on the current state. In testin, a effective window size that is less than 7 can be used, and we tested 10 episodes for each effective window size (for window size 7 we also tested with guidance of reward function trained on dataset), and reported the results in following figure.

```
Window Size 1: Mean Value = 4461.9377, V-Rate = 4.8190
Window Size 2: Mean Value = 3660.7400, V-Rate = 4.7716
Window Size 3: Mean Value = 3764.6957, V-Rate = 4.7824
Window Size 4: Mean Value = 4083.3252, V-Rate = 4.8152
Window Size 5: Mean Value = 4295.3235, V-Rate = 4.8278
Window Size 6: Mean Value = 3192.0946, V-Rate = 4.6812
Window Size 7: Mean Value = 3740.4791, V-Rate = 4.7504
```

(a) Raw

Window Size	Normalized Score	Value Rate
1	90.318	0.975
2	74.100	0.966
3	76.204	0.968
4	82.654	0.975
5	86.945	0.977
6	64.614	0.948
7	75.714	0.962
Guided 7	84.065	0.971

(b) Normalized

Figure 1: results of testing for agent trained on walker2d-expert-v2 dataset, where value or total return is normalized to percentage of dataset's mean value by $v_{normalized} = \frac{v}{v_{data}} \times 100$, and this is slightly different from what used in original paper $v_{normalized} = \frac{v - v_{random}}{v_{max} - v_{random}} \times 100$, where v_{random} is value achieved by random action sampler and v_{max} is max value recorded on this dataset.

The figure below shows value achieved by individual runs.

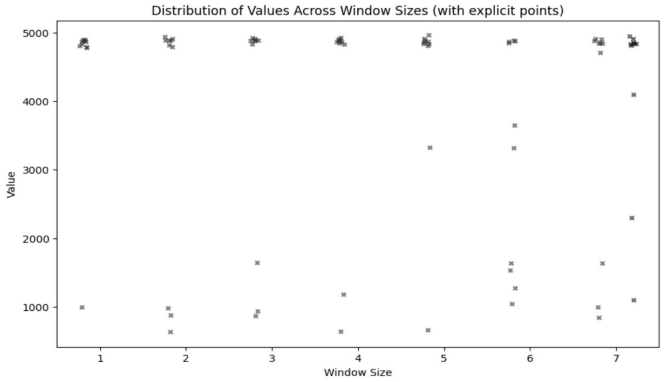

Figure 2: values or total return of test runs, each little cross is one run, and although most of runs clustered near 5000, some run ends early as agent encounter out of data distribution and fall, and this happen more frequently as the window size increases

The first thing to notice is the fact that all normalized scores are below 100% (i.e., less than 1 when not converted to percentage) suggests that the diffusion model does not surpass the performance of the data it was trained on, which is expected. This aligns with the design of diffusion

models, which are intended to sample from the distribution of the training data. Also we found that the value guidance effectively boosted the performance and increases normalized score from 75.7 to 84.1.

The run more often end early due to the encounter of novel scenario that occur more frequently with longer effective window size. More sample points are needed, but the use of larger effective window size could lead to early stop of run in which the walker agent fail to control the pose and fall down, as shown in figure 1, normalized score is max at lowest window size, and also in figure 2, where more data sample appeared in middle part. This also agree with our gained experience during replication of original paper's results. While planning horizon 32 is used, we chose 8 steps as we are unable to train the model using 32 steps, and the trained agent with 32 steps always fall after few initial irregular jumps, suggesting growth of the model's difficulty in approximate the underlying data distribution at longer horizon.

5 Discussion

As seen in this experiment and other examples, the combination of generative models and reinforcement learning can produce novel scenario during generating process, resulting in undesired outcomes. In general, given a data representation $x \in X$ with n normalized entries $x_i \in [0, 1]$. Let $\sigma(x) \in (0, 1)$ be the mean standard deviation of x, the fraction of total volume occupied by this data distribution is approximately $\sigma(x)^n$, which is very small, and for sequence generation of x, the model evolve state frames x_t in this space and need to avoid stepping into invalid regions, and this is the place where reinforcement learning could come into play. Specifically, the generative model $p_\theta(x_t|c_t)$ learns to sample x_t based on the input c_t by minimizing the loss $Loss(x_\theta, x)$, and from x_t actions $a(x_t)$ are extracted and carried out in environment to acquire the next input $c_{t+1} = f_{env}(c_t, a(x_t))$, where f_{env} is a stochastic transition function of the environment dynamics hidden from agent:

$$x_t \leftarrow p_\theta(x_t|c_t)$$
$$c_{t+1} = f_{env}(c_t, a(x_t))$$

this maps the sequence generation task to reinforcment learning setting as:

$$a_t \leftarrow \pi_\theta(a_t|s_t)$$
$$r_t = r(s_t, a_t)$$
$$s_{t+1} = f_{env}(s_t, a_t)$$

where $\pi_\theta(a_t|s_t) = a[p_\theta(x_t|c_t)]$ and $< S, A, f, r, \gamma, \rho_0 >$ forms a complete Markov decision process, though the Markov property may not hold.

In this experiment, we simply applied diffusion model to generate future trajectory and tested the replicated results, while although classifier value guidance is used and shown to boost performance, strictly speaking no reinforcement learning algorithm is incorporated in the generative process. With this formulation, existing RL algorithm could be used to contemplate the generation process and curb the trend to moving to invalid data space by imposing a punishment potential (via policy gradient $\nabla_\theta J(\pi_\theta) = E[\nabla_\theta \log \pi_\theta(a|s) Q_\phi(s, a)]$), thus enabling the consistent generation of sequences that comply with expected physical rules. In the future research, this should be studied and investigated more.

6 Conclusion

In this experiment report, we replicated part of results from the founding paper of using diffusion model to generate trajectory in offline RL. To our best, we managed to train the agent with reduced window under an acceptable performance, which indicates some crucial tricks (ignored in our implementation) are needed to improve the results. Nevertheless, the issue of data distribution

shift during testing is identified and to mitigate or solve this we proposed to formulate the sequence generation as reinforcement learning task. With the quick adaptation of generative models to various fields of application, ensuring and understanding the underlying mechanisms of generation has become more critical, so the study about how RL and generative model interplay will be an important problem in the future of machine learning.

References

[1] Janner, M., Du, Y., Tenenbaum, J. B., Levine, S. (2022). Planning with diffusion for flexible behavior synthesis. arXiv. https://doi.org/10.48550/arXiv.2205.09991

[2] Ho, J., Jain, A., Abbeel, P. (2020). Denoising diffusion probabilistic models. arXiv. https://doi.org/10.48550/arXiv.2006.11239

[3] Luo, C. (2022). Understanding diffusion models: A unified perspective. arXiv preprint arXiv:2208.11970.