Generative Reinforcement Learning via Denoising Diffusion

Yibin Hu
Tulane University

October 2, 2024

Abstract

Diffusion Model for RL is deployed on Walker2D task to gain insights and findings. Out of distribution issue is found and is to be resolved. (Project under Construction)

1 Motivation

Traditional model-based RL approaches often separate the process of learning the environment's dynamics from the process of planning. This split can result in inefficient or inaccurate plans because the models aren't directly built for the kinds of decisions planners need to make.

For example, many learned models focus on predicting individual state transitions, which can lead to compounding errors over time. These errors grow as time, making it harder to generate reliable long-term plans. Additionally, classical trajectory optimizers tend to exploit these models, producing paths that might look optimal on paper but fail when applied to the real-world environment, like adversarial examples.

It is motivated to combine the learning and planning processes into a unified framework. Instead of relying on separate trajectory optimizers, the trajectory generation is embedded within the model itself. By using a diffusion-based model, which iteratively refines a noisy guess of the entire trajectory, this creates a system where planning and model sampling are nearly identical. This allows us to generate smooth, long-term trajectories that can adapt to different tasks and constraints, improving flexibility and robustness.

2 Algorithm for Guided Generative Diffusion

Generative Reinforcement Learning with Denoising Diffusion. A trajectory is sequence of state action pair.

1, Train a base diffusion generative model $p_\theta(\tau|s)$, which given s generateS the trajectory that start with this state s

2, Train a value estimation function $V(\tau)$ that estimates the value of trajectory

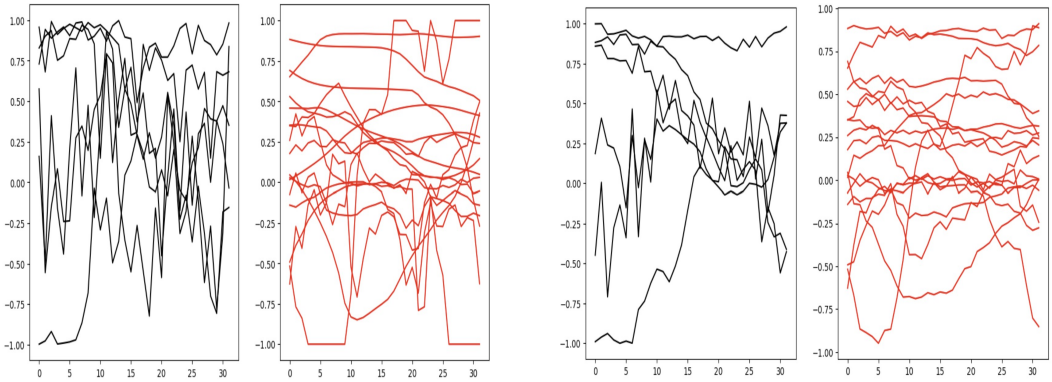
Algorithm 1.1 Diffusion: Training of Prior Diffusion $p_\theta(\boldsymbol{\tau}|s)$

$\{\boldsymbol{\tau}\} \leftarrow SampleTrajectories()$
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 Prior 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 $\mathbf{z} = 0$
  $\boldsymbol{\tau}^{t-1} \leftarrow \frac{1}{\sqrt{\alpha_t}} \left(\boldsymbol{\tau}^t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon_\theta}(\boldsymbol{\tau}^t, t) \right) + \sigma_t \mathbf{z}$ ▷ $\sigma_t^2 = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$
  $s_0^{t-1} \leftarrow s$
return $\boldsymbol{\tau}^0$

An agent is composed of a diffusion model and value function. With Algorithm 1.1 and 1.2, a diffusion model can be trained on a trajectory dataset to generate similar trajectories. A value function can also be trained at the same time. However, the agent trained in this way face data shift problem when deployed in real environment. Namely, although sampled trajectory has plausible state series that mimic the dataset's, the generated action series don't achieve such state series, and there is misalignment between generated states and actions. The agent knows what good state is but don't really know the actions to reach it, and the agent fail in real environmentt with many compounding error. Some sampled trajectory series trained in offline dataset is plotted below, a video showing the state series (rendered by directly imposing state and ignoring action) autoregresively generated is also available on project page.



(a) Dataset Trajectory (b) Sampled Trajectory

Figure 1: real action series (black), state series (red) over 32 timesteps, and sampled action series , state series from sampled trajectory conditioned on true initial state.

To resolve this data shift problem, an algorithm is designed, and it puts agent online and interact environment, during this process, diffusion model (pretrained on offline dataset) and value function are trained iteratively to generate trajectory with aligned states and actions that are compatible.

Algorithm 2: Align Generated States and Actions

$env \leftarrow environment$
while Training Episode not End **do**
  $s \leftarrow env.reset()$
  $\boldsymbol{\tau}_{pred} \leftarrow Sample(p_\theta(\boldsymbol{\tau}|s))$
  extract plan $\{a_t\}$ from $\boldsymbol{\tau_{pred}}$
  $\boldsymbol{\tau}_{true} \leftarrow$ real trajectory from carrying $\{a_t\}$
  $V_{true} \leftarrow$ real value from $\boldsymbol{\tau}_{true}$
  $\boldsymbol{\tau}^t_{true} \leftarrow$ diffuse and condition $\boldsymbol{\tau}_{true}$
  $\theta \leftarrow \theta - \eta\nabla_\theta \left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\boldsymbol{\tau}^t, t\right)\right\|^2$
  $V_{pred} \leftarrow V_\phi(\boldsymbol{\tau}^t_{true}, t)$
  $\phi \leftarrow \phi - \eta\nabla_\phi \left\|V_{true} - V_{pred}\right\|^2$

This way, the agent may learns to generate trajectory with aligned states and actions with relations corresponding to real dynamics. With a good diffuser and value function, the guided diffusion aims to generate trajectory that is steered towards higher value by value function.

Algorithm 3 Guided Denoising: Guided Sampling $p_\theta(\boldsymbol{\tau}|s)$

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