Learning Notes

DAVID

Reinforcement Learning Notes

1 Part1

1.1 Q3

Markov Decision Process. Markov Decision Process (MDP) [1] $\mathcal{M}_{\text{Env}} := (\mathcal{S}, \mathcal{A}, P_0, P, R)$ with states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, initial state distribution P_0, transition probabilities P, and reward R.

At each timestep t, the agent (e.g., robot) observes the state $s_t \in \mathcal{S}$, takes an action $a_t \sim \pi\left(a_t \mid s_t\right) \in \mathcal{A}$, transitions to the next state s_{t+1} according to $s_{t+1} \sim P\left(s_{t+1} \mid s_t, a_t\right)$ while receiving the reward $R\left(s_t, a_t\right)^2$. Fixing the MDP $\mathcal{M}_{\text{ENV}}$, we let $\mathbb{E}^\pi$ (resp. $\mathbb{P}^\pi$) denote the expectation (resp. probability distribution) over trajectories, $\tau = (s_0, a_0, \ldots, s_T, a_T)$ with length $T+1$, with initial state distribution $s_0 \sim P_0$ and transition operator P. We aim to train a policy to optimize the cumulative reward, discounted by a function $\gamma(\cdot)$, such that the agent receives:

$$\mathcal{J}\left(\pi_\theta\right) = \mathbb{E}^{\pi_\theta, P_0}\left[\sum_{t \geq 0} \gamma(t) R\left(s_t, a_t\right)\right].$$

Policy optimization. The policy gradient method (e.g., REINFORCE [85]) allows for improving policy performance by approximating the gradient of this objective w.r.t. the policy parameters:

$$\nabla_\theta \mathcal{J}\left(\pi_\theta\right) = \mathbb{E}^{\pi_\theta, P_0}\left[\sum_{t \geq 0} \nabla_\theta \log \pi_\theta\left(a_t \mid s_t\right) r_t\left(s_t, a_t\right)\right], \quad r_t\left(s_t, a_t\right) := \sum_{\tau \geq t} \gamma(\tau) R\left(s_\tau, a_\tau\right),$$

where r_t is the discounted cumulative future reward from time t (more generally, r_t can be replaced by a Q-function estimator [76]), γ is the discount factor that depends on the time-step, and $\nabla_\theta \log \pi_\theta\left(a_t \mid s_t\right)$ denotes the gradient of the logarithm of the likelihood of $a_t \mid s_t$. To reduce the variance of the gradient estimation, a state-value function $\hat{V}^{\pi_\theta}\left(s_t\right)$ can be learned to approximate $\mathbb{E}\left[r_t\right]$. The estimated advantage function $\hat{A}^{\pi_\theta}\left(s_t, a_t\right) := r_t\left(s_t, a_t\right) - \hat{V}^{\pi_\theta}\left(s_t\right)$ substitutes $r_t\left(s_t, a_t\right)$.
Diffusion models. A denoising diffusion probabilistic model (DDPM) [49, 29, 71] represents a continuousvalued data distribution $p(\cdot) = p\left(x^0\right)$ as the reverse denoising process of a forward noising process $q\left(x^k \mid x^{k-1}\right)$ that iteratively adds Gaussian noise to the data. The reverse process is parameterized by a neural network

[1] More generally, we can view our environment as a Partially Observed Markov Decision Process (POMDP) where the agent's actions depend on observations o of the states s (e.g., action from pixels). Our implementation applies in this setting, but we omit additional observations from the formalism to avoid notional clutter.

Author's Contact Information: DAVID.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

Table 1. Notations Table.

Notations	Descriptions
t	timestamps
$\mathcal{M}_{\mathrm{Env}}$	MDP
$s \in \mathcal{S}$	states
$a \in \mathcal{A}$	actions
P_0	initial state distribution
P	transition probabilities
R	reward

[2] For simplicity, we overload $R(\cdot, \cdot)$ to denote both the random variable reward and its distribution.

$\varepsilon_\theta\left(x_k, k\right)$, predicting the added noise ε that converts x_0 to x_k [29]. Sampling starts with a random sample $x^K \sim \mathcal{N}(0, \mathrm{I})$ and iteratively generates the denoised sample:

$$x^{k-1} \sim p_\theta\left(x^{k-1} \mid x^k\right) := \mathcal{N}\left(x^{k-1} ; \mu_k\left(x^k, \varepsilon_\theta\left(x^k, k\right)\right), \sigma_k^2 \mathrm{I}\right).$$

Above, $\mu_k(\cdot)$ is a fixed function, independent of θ, that maps x^k and predicted ε_θ to the next mean, and σ_k^2 is a variance term that abides by a fixed schedule from $k=1, \ldots, K$. We refer the reader to Chan [10] for an in-depth survey.

Diffusion models as policies. Diffusion Policy (DP; see [15]) is a policy π_θ parameterized by a DDPM which takes in s as a conditioning argument, and parameterizes $p_\theta\left(a^{k-1} \mid a^k, s\right)$ as in (3.3). DPs can be trained via behavior cloning by fitting the conditional noise prediction $\varepsilon_\theta\left(a^k, s, k\right)$ to predict added noise. Notice that unlike more standard policy parameterizations such as unimodal Gaussian policies, DPs do not maintain an explicit likelihood of $p_\theta\left(a^0 \mid s\right)$. In this work, we adopt the common practice of training DPs to predict an action chunk - a sequence of actions a few time steps (denoted T_a) into the future - to promote temporal consistency. For fair comparison, our non-diffusion baselines use the same chunk size.

As observed in [16, 6] and [59], a denoising process can be represented as a multi-step MDP in which policy likelihoods can be obtained directly. We extend this formalism by embedding the Diffusion MDP into the environmental MDP, obtaining a larger "Diffusion Policy MDP" denoted $\mathcal{M}_{\mathrm{DP}}$, visualized in Fig. 3. Below, we use the notation δ to denote a Dirac distribution and $\otimes$ to denote a product distribution.

Recall the environment MDP $\mathcal{M}_{\mathrm{Env}} := \left(\mathcal{S}, \mathcal{A}, P_0, P, R\right)$ in Section 3. The Diffusion MDP $\mathcal{M}_{\mathrm{DP}}$ uses indices $\bar{t}(t, k)=t K+(K-k-1)$ corresponding to (t, k), which increases in t but (to keep the indexing conventions of diffusion) decreases lexicographically with $K-1 \geq k \geq 0$. We write states, actions and rewards as,

$$\bar{s}_{\bar{t}(t, k)}=\left(s_t, a_t^{k+1}\right), \quad \bar{a}_{\bar{t}(t, k)}=a_t^k, \quad \bar{R}_{\bar{t}(t, k)}\left(\bar{s}_{\bar{t}(t, k)}, \bar{a}_{\bar{t}(t, k)}\right)=\left\{\begin{array}{ll} 0 & k>0 \\ R\left(s_t, a_t^0\right) & k=0 \end{array},\right.$$

where the bar-action at $\bar{t}(t, k)$ is the action a_t^k after one denoising step. Reward is only given at times corresponding to when a_t^0 is taken. The initial state distribution is $\bar{P}^0=P_0 \otimes \mathcal{N}(0, \mathbf{I})$, corresponding to $s_0 \sim P_0$ is the initial distribution from the environmental MDP and $a_0^K \sim \mathcal{N}(0, \mathbf{I})$ independently. Finally, the transitions are

$$\bar{P}\left(\bar{s}_{\bar{t}+1} \mid \bar{s}_{\bar{t}}, \bar{a}_{\bar{t}}\right)=\left\{\begin{array}{ll} \left(s_t, a_t^k\right) \sim \delta_{\left(s_t, a_t^k\right)} & \bar{t}=\bar{t}(t, k), k>0 \\ \left(s_{t+1}, a_{t+1}^K\right) \sim P\left(s_{t+1} \mid s_t, a_t^0\right) \otimes \mathcal{N}(0, \mathbf{I}) & \bar{t}=\bar{t}(t, k), k=0 \end{array}.\right.$$

That is, the transition moves the denoised action a_t^k at step $\bar{t}(t,k)$ into the next state when $k > 0$, or otherwise progresses the environment MDP dynamics with $k = 0$. The pure noise a_t^K is considered part of the environment when transitioning at $k = 0$. In light of (3.3), the policy in $\mathcal{M}_{\mathrm{DP}}$ takes the form

$$\bar{\pi}_\theta\left(\bar{a}_{\bar{t}(t,k)} \mid \bar{s}_{\bar{t}(t,k)}\right) = \pi_\theta\left(a_t^k \mid a_t^{k+1}, s_t\right) = \mathcal{N}\left(a_t^k; \mu\left(a_t^{k+1}, \varepsilon_\theta\left(a_t^{k+1}, k+1, s_t\right)\right), \sigma_{k+1}^2 \mathrm{I}\right).$$

Fortunately, (4.1) is a Gaussian likelihood, which can be evaluated analytically and is amenable to the policy gradient updates (see also [59] for an alternative derivation):

$$\nabla_\theta \overline{\mathcal{J}}\left(\bar{\pi}_\theta\right) = \mathbb{E}^{\bar{\pi}_\theta, \bar{P}, \bar{P}^0}\left[\sum_{\bar{t} \geq 0} \nabla_\theta \log \bar{\pi}_\theta\left(\bar{a}_{\bar{t}} \mid \bar{s}_{\bar{t}}\right) \bar{r}\left(\bar{s}_{\bar{t}}, \bar{a}_{\bar{t}}\right)\right], \quad \bar{r}\left(\bar{s}_{\bar{t}}, \bar{a}_{\bar{t}}\right) := \sum_{\tau \geq \bar{t}} \gamma(\tau) \bar{R}\left(\bar{s}_\tau, \bar{a}_\tau\right).$$

Evaluating the above involves sampling through the denoising process, which is the usual "forward pass" that samples actions in Diffusion Policy; as noted above, the inital state can be sampled from the enviroment via $\bar{P}^0 = P_0 \otimes \mathcal{N}(0, \mathbf{I})$, where P_0 is from the environment MDP. 4.2 Instantiating DPPO with Proximal Policy Optimization

We apply Proximal Policy Optimization (PPO) [70, 18, 32, 1], a popular improvement of the vanilla policy gradient update.

Definition 4.1 (Generalized PPO). Consider a general MDP. Given an advantage estimator $\hat{A}(s, a)$, the PPO update is given by [70] the sample approximation to where ε, the clipping ratio, controls the maximum magnitude of policy change from the previous policy. We instantiate PPO in our diffusion MDP with $(s, a, t) \leftarrow (\bar{s}, \bar{a}, \bar{t})$. Our advantage estimator takes a specific form that respects the two-level nature of the MDP: let $\gamma_{\mathrm{ENV}} \in (0,1)$ be the environment discount and $\gamma_{\mathrm{DENOISE}} \in (0,1)$ be the denoising discount. Consider the environment-discounted return:

$$\bar{r}\left(\bar{s}_{\bar{t}}, \bar{a}_{\bar{t}}\right) := \sum_{t' \geq t} \gamma_{\mathrm{ENV}}^t \bar{r}\left(\bar{s}_{\bar{t}(t',0)}, \bar{a}_{\bar{t}(t',0)}\right), \quad \bar{t} = \bar{t}(t,k),$$

1.2 Q4

1.3 Q6

References