Skill Based Model Based Reinforcement Learning on D4RL PointMaze Environment

Bhavya Pranav Tandra* **Javad Moein†** **Dhruv Miyani‡**
Khoury College of Computer Sciences, Northeastern University
tandra.b@northeastern.edu moeinnajafabadi.j@northeastern.edu miyani.d@northeastern.edu

Abstract

Intelligent agents, including humans, often rely on past experiences when learning new tasks. For tasks that span long horizons, humans tend to plan efficiently by breaking down the task into higher-level sub-tasks, using previously acquired high-level skills. This concept is mirrored in recent advancements in reinforcement learning (RL), focusing on hierarchical learning to develop artificial intelligent agents capable of utilizing high-level skill hierarchies. In line with this, our study is centered on training agents to master long-horizon complex tasks by initially learning sub-tasks, high-level behaviors, and skills. In this work we introduce SKIMO paper, an innovative, sample-efficient, model-based hierarchical RL algorithm that leverages task-agnostic offline data. We aim to implement SKIMO on Maze environment. The method involves two primary phases: (1) learning the skill dynamics model and skills from an offline dataset, and (2) applying these learned skills in downstream task learning.

Introduction and Related Work[1]

In addition to using prior experiences for solving complex tasks, a key trait of humans is the ability to plan abstractly. For example, when cooking, we envision the outcomes of high-level skills like washing and cutting vegetables, rather than planning every muscle movement. We aim to apply this principle to artificial intelligence agents, teaching them high-level behaviors and skills from previous experiences, which they can then transfer to learning downstream tasks. The agent could learn these behaviors either by extracting skills from task-agnostic datasets or through environment interactions. To this end, we introduce and implement SKIMO paper, a novel, sample-efficient, model-based hierarchical RL algorithm that utilizes task-agnostic data. This strategy aims to extract a reusable skill set and a skill dynamics model, facilitating efficient and accurate long-term planning for sample-efficient RL. The SkiMo approach involves two phases: (1) learning the skill dynamics model and skills from an offline dataset, and (2) applying this knowledge in downstream task learning. In downstream Model-based RL learning, rollouts are generated in the skill space. We sample a skill, $z_t \sim \pi_\theta(h_t)$, and then predict the H-step future following skill execution, where $h_{t+H} = D_\phi(h_t, z_t)$. The skill dynamics model only requires $1/H$ of the dynamics predictions and action selections compared to conventional flat model-based RL methods, leading to more efficient and precise long-horizon imaginary rollouts.

Online Skill Learning

The concept of leveraging prior experience and inter-task transfer has been extensively explored within the reinforcement learning (RL) community, and is commonly referred to as skill-based RL Hausman et al. 2018; The idea of transferring skills be- tween tasks has been in existence for quite some time, with the SKILLS algorithm Thrun et al. 1994; being one of the earliest exam- ples. Learned skills can be represented as sub-policies in the form of options.Sutton et al. 1999; Konidaris and Barto, Konidariset al. 2010 utilized the options framework to learn transferable options us- ing the so-called agent-space. In addition, meta-learning ap- proaches Rakelly, K. 2019 aim to extract useful priors from previous ex- perience to improve the learning efficiency for unseen tasks. However, such approaches necessitate a defined set of train- ing tasks and online data collection during pre-training and are unable to leverage large offline datasets. While much of the prior work has focused on online settings and assumed access to an environment for interacting to learn high-level behaviors, such approaches are not sample-efficient. There- fore, recent research has shifted focus towards methods sup- porting sample-efficient embodied transfer learning, with a particular emphasis on off-policy learning. Skill-based RL is a class of these approaches that is particularly well-suited for learning long-horizon behaviors, as they enable the extraction of reusable skills from task-agnostic datasets and subsequently learn new tasks by recombining them.

*Worked Completely on learning skill and skill dynamics model from offline dataset (Offline Learning)

†Worked on Downstream RL(High-level task learning + Fine tuning skill dynamics model and state encoder)

‡Worked on MPC with CEM Planning

[1]This part is done by Javad Moein

Offline Skill Learning with demonstrations

The offline reinforcement learning paradigm involves learning a task solely from recorded agent experience, without any interactions with the environment. Recently, several approaches Singh, A et al. 2020, Ajay, A et al. 2020 have proposed data-driven extraction of be- havioral priors, which refer to learned skills. These methods fit a behavioral prior to an offline dataset of demonstrations and then use it to guide the reinforcement learning policy to solve downstream tasks. However, they still require the use of engineered rewards for downstream tasks, which can be difficult to provide for large, real-world datasets. Addi- tionally, these approaches assume access to a clean offline dataset of expert demonstrations that are specifically rele- vant to the downstream tasks. In practical scenarios, it is un- likely that such clean datasets exist.

Offline Skill Learning with Unstructured dataset

As an alternative to costly task-specific demonstrations, numerous recent studies have proposed the acceleration of reinforcement learning by utilizing task-agnostic experience in the form of extensive datasets collected across vari- ous tasks Pertsch, K et al. 2020. Unlike demonstrations, such task-agnostic datasets can be gathered inexpensively from various sources such as autonomous exploration or human tele-operation. However, as the data is not tailored to the downstream task, the learning process is slower than that of demonstrations. Importantly, in offline skill learning, the goal is to exploit unstructured data that lacks annotations of tasks or sub-skills and does not contain reward information.

Offline Skill Learning by using stochastic latent variable models

A promising approach to addressing challenging long- horizon tasks involves the extraction of skills through fitting generative models to large offline datasets. Recently, several works have explored the embedding of skills into a continu- ous skill space via stochastic latent variable models Whitney, W et al. 2019, Hausman, K et al 2018 . These approaches leverage powerful latent variable models to represent a large number of skills in a compact embedding space. However, as the amount of prior experience increases, the number of transferable skills also increases, posing a challenge for exploration of the rich skill embedding space during downstream learning, which can result in inefficient downstream task learning. As a result, recent works such as Pertsch, K et al. 2021, Shi, L. X et al. 2022, Pertsch, K et al. 2021 have sought to guide the exploration of the skill embedding space, enabling efficient learning on a rich skill space.One such work, SPiRL (Skill-Prior RL) Pertsch, K et al. 2021, proposes a deep latent variable model that jointly learns an embedding space of skills and the skill prior from offline agent experience. Skill-based Learning with Demonstrations (SkiLD) Pertsch, K et al. 2021 employs demonstrations to guide the explo- ration of the set of reusable skills learned from large offline datasets, and the authors propose to extend the skill prior- guided approach by leveraging target task demonstrations to additionally learn a task-specific skill distribution. To bet- ter align skill extraction with human intent, introduces Skill Preferences (SkiP), an algorithm that learns a model over human preferences and uses it to extract human-aligned skills from offline data. In Shi, L. X et al. 2022, a Skill-based Model-based RL framework (SkiMo) is proposed, which learns a model and a policy in a high-level skill space, enabling accurate long-term prediction and efficient long-term planning. Un- like traditional model-based RL that learns a flat single-step dynamics model, SkiMo directly predicts the resultant state after skill execution given a state and a skill to execute, with- out needing to model every intermediate step and low-level action,

High-level Skill Learning

Decision-making in environments with continuous state and action spaces in robotics is a challenging task, especially when dealing with long horizons and sparse feedback. To tackle this issue, a common approach is to decompose decision-making into a high-level "what to do" and a low-level "how to do it" strategy. Hierarchical approaches, such as task and motion planning (TAMP), have been proposed to address these challenges by breaking down decision-making into multiple levels. Konidaris et al. 2018 proposed a hierarchy where symbolic AI planning Helmert, M. 2006. is used to sequence together temporally-extended continuous skills Konidaris, G. D et al 2007. Their method mainly focused on learning symbols from known skills. In [19], the authors studied the inverse problem of learning skills from known symbols.

Although the use of high-level skills can significantly improve performance, planning in high-dimensional, continuous state spaces remains a difficult task. The authors of [9] hypothesized that effective planning for complex devices such as robots requires both procedural abstraction, in the form of high-level skills, and state abstraction, in the form of symbolic representations. They proposed a specific relationship between these two types of abstraction, stating that high-level skills induce abstract representations. In this research, we aim to use skill hierarchies to break down large problems into smaller subproblems that can be solved independently, gaining an advantage in decision-making.

Background[2]

Reinforcement Learning

The problem at hand is formulated as a reinforcement learning problem, specifically a Markov Decision Process (MDP). The MDP is defined by the tuple $(S, A, R, P, \rho_0, \gamma)$, where S is the state space, A is the action space, $R(s, a)$ is the reward, P is the transition probability, ρ_0 is the initial state distribution, and γ is the discount factor. The objective is to learn a policy that minimizes the expected sum of cumulative rewards.

Model-based Reinforcement Learning

In Model-based reinforcement learning (MBRL), the agent learns a model of the environment to make decisions and

[2]This part is done by Javad and Pranav

plan actions. Instead of directly interacting with the real environment, the agent uses a learned model to simulate the consequences of its actions. MBRL typically involves the following components:

Model Learning: The agent learns a predictive model of the environment. This model tries to estimate how the environment will behave when the agent takes different actions. Common approaches include neural network models, Gaussian processes, or other machine learning techniques. In this work, the agent learns model using offline task-agnostic data and then does fine tune them in online leanring.

Planning: After learning the model, the agent can use it to perform planning. It simulates different action sequences in its model and evaluates which sequence is likely to yield the best outcome, according to its learned model. In this work, we use TD-MPC and CEM for planning.

Policy Optimization: Once the agent has planned and identified a promising action sequence, it can use various RL algorithms (such as policy gradient methods) to optimize its policy based on the simulated experiences.

Skill-Based Reinforcement Learning

The concept of utilizing prior experience and transferring knowledge between tasks has been extensively explored within the reinforcement learning (RL) com- munity, often referred to as skill-based RL. The idea of transferring skills across tasks has been around for some time, with the SKILLS algorithm serving as one of the early examples. Learned skills can be represented as sub- policies, often in the form of options. Konidaris and Barto applied the options framework to learn transferable options within the agent-space. More- over, meta-learning approaches aim to extract valuable priors from previous experiences to enhance learning efficiency for new tasks.

A skill is defined as a sequence of actions. In the context of this paper, skills are sequences of fixed-length actions, known as the horizon. However, in general, skills may also be sequences of actions with variable lengths. SpiRL is one of the skill-based approaches where a trained skill prior is employed to efficiently sample skills given a condition, such as a state. Utilizing a skill prior for sampling skills in reinforcement learning allows for exploration of a few meaningful actions, as opposed to sampling from an infinite space of continuous actions. In this work, the authors utilize a skill prior trained during offline learning, employing it as a policy to efficiently sample actions during downstream RL.

Offline Reinforcement Learning

Offline reinforcement learning is a paradigm where tasks are learned solely from recorded agent experience, without interactions with the environment. Recently, several approaches have proposed data-driven extraction of be- havioral priors, which essentially refer to learned skills. These methods fit a behavioral prior to an offline dataset of demonstrations and utilize it to guide reinforcement learning policies for solving downstream tasks. However, they still rely on engineered rewards for these tasks, which can be challenging to pro- vide for extensive, real-world datasets. Additionally, these approaches assume access to a pristine offline dataset of expert demonstrations specifically relevant to the downstream tasks. In practical scenarios, such ideal datasets are rarely available.

Variational Auto-Encoder (VAE)

In this project, variational auto-encoders (VAE) are employed to encode skills and states. VAE is widely used in the offline learning of skills and skill-dynamics models from task-agnostic data.

First introduced by Kingma, VAE is a modified version of an auto-encoder that addresses the challenge of encoding input data distribution in its latent space. It achieves this by introducing a Variational Lower Bound (ELBO) and employing a parameterization trick, enabling sampling from the distribution in the latent space while facilitating the back-propagation of gradients through the sampling operation.

PointMaze by D4RL

The Datasets for Deep Data-Driven Reinforcement Learning (D4RL) are widely used for offline learning. This project specifically focuses on the PointMaze environment within D4RL. PointMaze is a maze environment where the agent is a point mass (ball) navigating from a randomly selected start position to a goal position.

Datasets for the PointMaze environment are generated by randomly sampling various small configurations. These datasets, characterized by long horizons and sparse rewards, are utilized to train skills and skill dynamics models using offline learning approaches. The primary downstream task is to solve a more complex and larger maze environment with sparse rewards. Consequently, skills and skill dynamics models are trained on smaller, task-agnostic configurations.

The PointMaze dataset for offline learning includes actions, observations, and dones. The dataset comprises 3046 trajectories, some complete and others incomplete. Trajectories vary in length and consist of actions (representing forces in the x and y directions applied to the ball), observations (x and y coordinates of the ball, as well as its velocities), and dones (indicating whether the trajectory is completed or terminated).

Methods

To address the challenge of long-horizon, sparse-reward tasks in the PointMaze Environment, we employ a Skill-Based Model-Based Reinforcement Learning approach. This method combines the advantages of Skill-Based Reinforcement Learning Approaches and Model-Based Reinforcement Learning Approaches.

We begin by jointly training skills and skill dynamics models using a task-agnostic offline dataset in the pretraining phase. Subsequently, in the downstream reinforcement learning (RL) phase, we fine-tune to learn a task policy that plans within the skill space.

Therefore, the proposed method consists of two key phases: 1) Learning skill dynamics model and skills from an offline dataset. 2) Downstream learning of the task policy.

Pretraining of Skills and Skill Dynamics from Offline Task-Agnostic Data[3]

In the pretraining phase, we simultaneously train a skill model that encodes skills based on state-action trajectories into a tanh-normalized Gaussian normal in the latent space. This latent space is where skills are sampled. Additionally, we train a skill dynamics model, predicting the state in the latent space H steps ahead after executing the sampled skill from the current state in the latent space. Several auxiliary components contribute to this training process.

The Offline Learning Model comprises the following components:

$$\begin{aligned}
&\text{State Encoder: } h_t = E_\psi(s_t)\\
&\text{Observation Decoder: } \hat{s}_t = O_\theta(h_t)\\
&\text{Skill Prior: } \hat{z}_t = p_\theta(s_t)\\
&\text{Skill Encoder: } z_t = q_\theta((s,a)_{t:t+H-1})\\
&\text{Skill Policy: } \hat{a}_t = \pi_\theta^L(s_t, z_t)\\
&\text{Skill Dynamics: } \hat{h}_{t+H} = D_\psi(h_t, z_t)
\end{aligned}$$

To train the offline learning model, all the above components are jointly trained using a single loss function. This involves encoding an H-horizon length of state-action trajectory sequence using a Skill Encoder to produce a Skill Distribution in the latent space. The Skill Policy, acting as a decoder part of a Variational Auto-Encoder (VAE), then samples skills from the encoded skill distribution to reconstruct the H-length sequence of actions representing that skill. The VAE is trained using the following loss:

$$L_{VAE} = E_{(s,a)_{0:H-1}\sim D}\Bigg[\frac{\lambda_{BC}}{H}\sum_{i=1}^{H-1}(\pi_\theta^L(s_i,z)-a_i)^2 + \beta\cdot \mathrm{KL}\big(q_\theta(z|(s,a)_{t:t+H-1})||p(z)\big)\Bigg] \tag{1}$$

The first term is the Behavioral Cloning Loss, acting as a reconstruction loss for actions to skills to actions. The second term serves as a regularizer, ensuring the skill distribution closely resembles a standard normal distribution.

We train the skill dynamics model and the observation auto-encoder (comprising State Encoder and Observation Decoder) using the following loss:

$$L_{\text{REC}} = E_{(s,a)_{0:NH}\sim D}\Bigg[\sum_{i=0}^{N-1}\Bigg[\lambda_O\|s_{iH} - O_\theta(E_\psi(s_{iH}))\|_2^2 + \lambda_L\|D_\psi(\hat{h}_{iH}, z_{iH}) - E_\psi(s_{(i+1)H})\|_2^2\Bigg]\Bigg] \tag{2}$$

The first term trains the observation auto-encoder with a reconstruction loss, while the second term ensures consistent encodings for states, termed latent state consistency.

We also train the Skill Prior over the skill distribution to aid exploration during downstream RL using the following loss:

$$L_{SP} = E_{(s,a)_{0:H-1}\sim D}\Bigg[\lambda_{SP}\mathrm{KL}\big(q_\theta(z|(s,a)_{t:t+H-1})||p_\theta(z|s_0)\big)\Bigg] \tag{3}$$

The final loss enforces the skill distribution to closely match the skill prior distribution.

Now, we train all components using a single loss, which is the sum of all the aforementioned losses:

$$L = L_{VAE} + L_{REC} + L_{SP}$$

Training all components jointly enriches the latent state representation of all models by contrasting them with each other, facilitating collaborative learning.

Downstream Online Task Learning using Learned Skill Dynamics Model and Skill space[4]

In downstream RL using the learned skill model and skill space, we learn a high-level task policy $\pi_\theta(z_t|h_t)$ that outputs a latent skill embedding z_t. We could easily transfer this latent skill z_t into a sequence of H actions using the pre-trained skill policy π^0 to act in the environment.

In this section for generating imaginary rollouts we use the skill dynamics model and the updated task policy. The rollouts are generated in the skill space. We sample a skill, $z_t \sim \pi_\theta(h_t)$, and (2) and predict H-step future after executing the skill, $h_{t+H} = D_\phi(h_t, z_t)$. The skill dynamics model requires only $1/H$ dynamics predictions and action selections of the flat model-based RL approaches, resulting in more efficient and accurate long-horizon imaginary rollouts (see Appendix, Figure 10).

We use these imaginary rollouts both for planning (Algorithm 2) and policy optimization (Equation (7)). By generating these rollouts we significantly reduces the number of environment interactions. During rollout, we perform Model Predictive Control (MPC), which re-plans every step using CEM and executes the first skill of the skill plan. After generating rollouts we need to evaluate them by using V-value and Q-value. So we need to train a reward function $R_\phi(h_t, z_t)$ that predicts the sum of H-step rewards r_t, and a Q-value function $Q_\phi(h_t, z_t)$. During online learning we also finetune the skill dynamics model D_ϕ and state encoder E_θ on the downstream task to improve the model prediction:

$$L_{\text{REC}} = E_{s_t,z_t,s_{t+H},r_t^D}\Bigg[\lambda_{\text{dyn}}\|D_\phi(h_t,z_t) - E_\theta(s_{t+H})\|^2 + \lambda_r\|r_t - R_\phi(h_t,z_t)\|^2\Bigg] \tag{4}$$

[3]This part is done by Bhavya Pranav Tandra

[4]This part is done by Javad Moein

Finally, we train a high-level task policy π_θ to maximize the estimated Q-value while regularizing it to the pre-trained skill prior p_θ, which helps the policy output plausible skills:

$$L_{\text{RL}} = E_{s_t \sim D}\left[-Q_\phi(h_t, \pi_\theta(g(s_t))) + \alpha \cdot KL(\pi_\theta(z_t|g(s_t))||p_\theta(z_t|s_t))\right] \tag{5}$$

We minimize Equation (4) and Equation (5) using back-propagation through time over N consecutive skill-level transitions. It is worthwhile that $\pi_\theta(z_t|h_t)$ is initilaezed first by $p_\theta(z_t|s_t)$ so we will not have random behavior. Here is Downstream RL online learning:

Algorithm 1 : SkiMo (Downstream RL)

Result: Return trained parameters ψ, ϕ
Require: θ, ψ, ψ^- : pre-trained parameters
Initialize replay buffer $B \leftarrow \emptyset$
Randomly initialize ϕ
$\phi^- \leftarrow \phi$
Initialize task policy with skill prior $\pi_\phi \leftarrow \rho_\theta$
while *not converged* **do**
  $t \leftarrow 0$, $s_0 \sim \rho_0$
  while *episode not done* **do**
    Sample skill $z_t \sim CEM(s_t)$
    $s, r_t \leftarrow s_t, 0$
    for *H steps* **do**
      $s, r \leftarrow ENV(s, \pi_\theta^L(E_\psi(s), z_t))$
      $r_t \leftarrow r_t + r$
    end
    $B \leftarrow B \cup \{(s_t, z_t, r_t)\}$
    $t \leftarrow t + H$
    $s_t \leftarrow s$
  end
  Sample mini-batch $B = \{(s, z, r)\}^N \sim B$
  Update ϕ, ψ with
  $\phi, \psi \leftarrow \phi, \psi - \lambda_{\phi,\psi} \nabla_{\phi,\psi} L_{REC}(B)$
  Update ϕ with $\phi_\tau \leftarrow \phi_\tau - \lambda_\phi \nabla_\phi L_{RL}(B)$
  Update target network $\psi^- \leftarrow (1-\tau)\psi^- + \tau\psi$
  Update target network $\phi^- \leftarrow (1-\tau)\phi^- + \tau\phi$
end

Planning in Skill Space[5]

We used a model-based reinforcement learning (RL) approach within the skill space, taking advantage of our skill dynamics model. This skill dynamics model, in conjunction with the task policy, can help generate simulated scenarios in the skill space by doing two key things: (1) randomly selecting a skill, z_t, from the policy $\pi_\phi(h_t)$, and (2) predicting what will happen H steps into the future after executing the skill, $h_{t+H} = D_\psi(h_t, z_t)$. Our skill dynamics model requires only $1/H$ dynamics predictions and action selections, making it more efficient and accurate for long-term simulated scenarios compared to the traditional flat model-based RL methods.

[5]This part is done by Dhruv Miyani

In TD-MPC , we utilize these simulated scenarios for both planning (as outlined in Algorithm) and policy optimization, effectively reducing the number of interactions required with the actual environment. During the simulated scenarios, we employ Model Predictive Control (MPC), which re-plans at each step using Cross-Entropy Method (CEM) and executes the first skill from the plan.

To evaluate these simulated scenarios, we train a reward function $R_\phi(h_t, z_t)$ that predicts the cumulative rewards over H steps, r_t, and a Q-value function $Q_\phi(h_t, z_t)$. Furthermore, we fine-tune the skill dynamics model D_ψ and the state encoder E_ψ on downstream tasks to enhance the accuracy of model predictions.

Algorithm 2 : SkiMo (CEM planning)

Require: θ, ψ, ϕ: learned parameters, s_t: current state

1. Initialize sampling distribution $\mu^0, \sigma^0 \leftarrow 0, 1$
2. for $i = 1, \ldots, N_{CEM}$ do
   (a) Sample N_{sample} trajectories of length N from $\mathcal{N}(\mu^{i-1}, (\sigma^{i-1})^2)$
   (b) Sample N_π trajectories of length N using π_ϕ, D_ψ
   (c) Estimate N-step returns of $N_{sample} + N_\pi$ trajectories using R_ϕ, Q_ϕ
   (d) Compute μ^i, σ^i with top-k return trajectories
3. end for
4. Sample a skill $z \sim \mathcal{N}(\mu_{N_{CEM}}, (\sigma_{N_{CEM}})^2)$
5. return z

- **Planning Horizon Determination (`horizon`):** The planning horizon represents the number of time steps into the future the agent is considering when making a plan. It is calculated using a decay function called `horizon_decay` based on a step counter. The horizon determines how far ahead the agent should plan.
- **State Encoding (`state`):** The current observation (`ob`) is encoded into a state representation using a function called `state_encoder`. This state representation is used to guide the planning process.
- **Policy Trajectory Sampling (`policy_ac`):** Method samples multpolicy trajectories by repeatedly calling the `skill_policy` function. These trajectories represent actions that the agent might take according to its policy.
- **Action Distribution Parameters (`mean` and `std`):** The code initializes the mean and standard deviation for the action distribution. These parameters are adjusted during the CEM optimization to generate better action plans. The mean action trajectory is updated based on the previous one, if available.
- **Action Sampling (`sample_ac`):** Within each CEM optimization step, the code samples action trajectories from the action distribution. These trajectories are subject to exploration noise to encourage exploration during training.

- **Value Estimation (`imagine_return`):** The code estimates the value of the imagined action trajectories using a function called `estimate_value`. This estimation helps assess the quality of different action plans.
- **Elite Action Selection (`elite_action`):** The top-performing action trajectories (elite plans) are selected based on their estimated values. These elite action trajectories are used to update the action distribution.
- **Weighted Aggregation (`new_mean` and `new_std`):** The Method aggregates the elite action plans to update the mean and standard deviation of the action distribution. The aggregation is weighted based on the elite plan's performance.
- **Action Selection (`ac`):** After the CEM optimization process, the Method selects a final action (`ac`) from the action distribution based on the aggregated elite action plans. This action is sampled using a probability distribution.
- **Exploration Noise (Training Mode):** If the agent is in training mode (`is_train=True`), exploration noise is added to the selected action to encourage exploration and improve learning.
- **Action Clamping (`torch.clamp`):** Finally, the code ensures that the selected action falls within valid bounds by clamping it to the range [-0.999, 0.999].

We need thease two parameter which are trained for downstream task using environment interactions R_ϕ and Q_ϕ.

code link:
"https://github.com/DhruvMiyani/Planning$_CEM$"

Experiments

In this section, we evaluate the performance of the methods. In first section we show the results of offline learning which trained on offline data. In the second we show the outputs of the downstream online learning. We implement online learning on Maze.

Results of Pretraining on Task-Agnostic Offline Dataset[6]

The authors of the original paper did not evaluate the offline learning/pre-training step but we assess the performance of our offline learning model, which jointly learns skills and skill dynamics. The evaluation involves analyzing the losses utilized in pre-training.

The PointMaze Offline Dataset comprises task-agnostic reward-free data with 3046 trajectories. The action space is a `Box(-1.0, 1.0, (2,), float32)`, where each action represents the linear force exerted on the green ball in the x and y directions. Additionally, the ball velocity is clipped in a range of $[-5, 5]$ m/s to prevent unbounded growth. The observation space is a dictionary containing information about the robot's position, with each observation being an `ndarray` of shape (4,). It includes kinematic information of the force-actuated ball, providing the x and y coordinates of the ball, ball linear velocity in the x and y directions. The start state and goal of the ball are randomly generated.

The loss of the offline learning model is evaluated on the PointMaze dataset by sampling batches of length 256 (trajectory length) for each training/evaluation iteration, totaling around 750 iterations. This sampling strategy compensates for limited compute resources, where each iteration represents a pass through a sub-episode of length 256 rather than a complete pass through the entire 3046 trajectory dataset.

We employed the Exponential Linear Unit (ELU) nonlinearity for all networks, with a hidden dimension of 256 and 4 hidden layers in each network. The `AdamW` optimizer was chosen with a learning rate of 0.001 and weight decay of 0. Skill horizon was set to 10, skill dimension to 10, encoded state dimension to 50, and the number of skills to 10. Scalar hyperparameters in the loss include Encoder KL Regularization β (set to 0.0001), Reconstruction Loss Coefficient λ_O (set to 1), Consistency Loss Coefficient λ_L (set to 2), Lower-Level Actor Loss Coefficient λ_{BC} (set to 2), and Skill Prior Loss Coefficient λ_{SP} (set to 1).

Upon training the offline model with these hyperparameters, the achieved best loss was 14. Figure 2 illustrates the training and validation loss curves for all components, including the total loss. The total loss consistently decreases, showing a steady improvement. Notably, the loss has not yet reached saturation, as evidenced by a 1-point drop observed every 100 iterations. It's crucial to recognize that our training loop iteration, involving the sampling of a trajectory of length 256 each time, differs from the typical epoch structure, contributing to the illusion of saturation in the loss curve.

Figures 1 to 4 are analysis of all the loss curves.

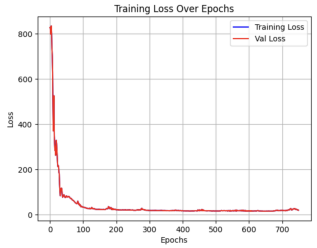

Figure 1: Training and Validation Losses for offline learning

The figure above represents the training and validation loss curves. The loss is sum of three loss components as mentioned in the methods section. We could see that the loss keeps decreasing, and the minimum loss is 14.

The overall loss decreases both for the training set and the validation set, but the Skill-Prior Loss seems to increase and then decrease at a steady rate. VAE Loss and REC Loss as shown in figures 2 and 3 seem to decrease, but the SP loss abnormally increases. As of now we have no clue as to why this anamoly is occurring. The most probable reason could be because of some bug we missed in calculating the SP loss.

The code for reproducing the above results can be found in :https://www.kaggle.com/code/pranavtandra/skimo

[6]This part is done by Bhavya Pranav Tandra

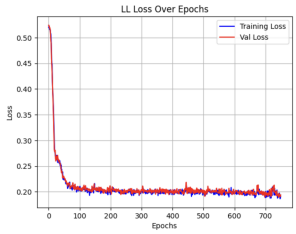

Figure 2: VAE Loss defined as L_{VAE} in the methods section

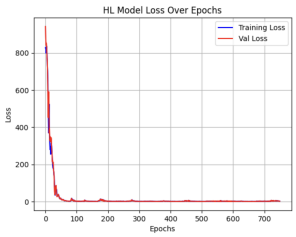

Figure 3: REC Loss defined as L_{REC} in the methods section

Results of Downstream RL[7]

In this section, we present the results of downstream task learning. We executed the downstream reinforcement learning (RL) component of the SKIMO framework within a Maze environment. Due to the necessity of employing a pretrained model for downstream learning and the unavailability of our models at the time of execution, we utilized pretrained models from SKIMO GitHub. For demonstrating some results, they use integrated Weights Biases (Wandb) into the code and we follow them. It should be noted that for downstream task learning, we initialized the task policy with the skill prior.

We adopted the maze navigation task from Pertsch et al. [16], where an agent represented by a point mass. The agent's observable state includes its 2D position and velocity, and it can control its velocity along the x and y axes. A sparse reward of 100 is awarded when the agent arrives at the goal. The maze has dimensions of 40 by 40. The starting state is selected randomly near a specified region (depicted as the green circle in Figure 3a), and the goal location is constant, marked by the red circle in Figure 5. The agent navigates the maze by adjusting its continuous (x, y) velocity. The maximum length of an episode is set to 2,000 steps, although the episode ends prematurely if the agent enters the vicinity of the goal, defined by a radius of 2.

We attempted to learn a high-level task policy through the implementation of Algorithm 1, interacting with a maze environment utilizing the downstream RL segment of the SKIMO code available on GitHub. Regrettably, we were unable to produce any outputs for presentation. This code was executed on three distinct devices, with various Ubuntu versions, and also tested on Kaggle. In each instance, we encountered errors, some of which remain unresolved. The documentation provided was insufficient, and the requirements.txt did not enumerate all necessary dependencies, leading to numerous errors. Even after rectifying some issues, we still faced fundamental conflicts between the packages specified and those utilized within the code. Here are a few reasons why we have yet to generate any output:

1- The downstream reinforcement learning (RL) component is notably complex, and developing the code for this section is quite time-consuming. Given the constraints of this course, it proved impractical to code all aspects of this segment. Consequently, we opted to utilize the downstream RL portion of the SKIMO code. Nonetheless, the SKIMO code references another GitHub repository named 'rolf'[8] for training model-based RL. Unfortunately, there are discrepancies between the package versions listed in SKIMO's requirements.txt and those required by the 'rolf' package. Moreover, using 'rolf' presents its own set of challenges.

2- Further complications arose while working with Mujoco, which had issues syncing properly, and subsequently with the integration of Weights Biases (Wandb).

3- The list of requirements provided in the code was incomplete, necessitating the installation of additional packages not initially specified.

The link to the modified code is provided, but it requires further refinement before it can yield any outputs.

https://github.com/jaavad/skimo

[7]This part is done by Javad Moein

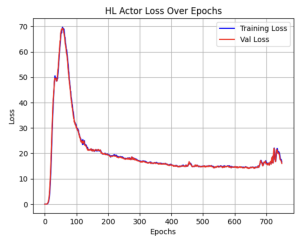

Figure 4: SP Loss defined as L_{SP} in the methods section

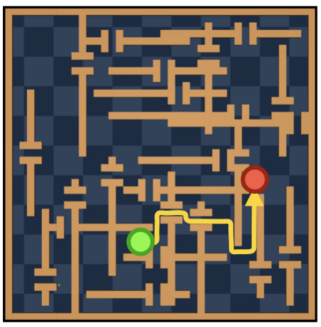

Figure 5: Maze navigation task - Shi, L. X et al. 2022

[8]https://github.com/youngwoon/robot-learning

Below we present some outputs featured in the SKIMO paper, which we aim to replicate.

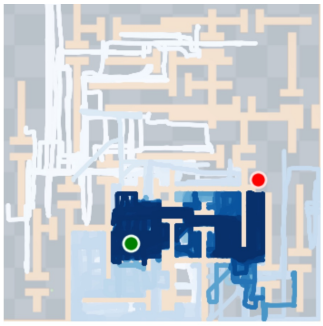

Figure 6: Maze using SKIMO - Shi, L. X et al. 2022

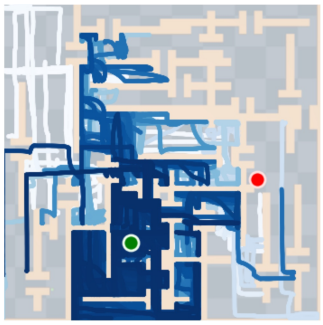

Figure 7: Maze using SKIMO w:o joint learning - Shi, L. X et al. 2022

Conclusion

The Skill-Based Model-Based approach presents a significant advancement in addressing challenges posed by long-horizon sparse-reward tasks, as encountered in maze navigation. This innovative methodology not only builds upon the merits of existing skill-based and model-based approaches but also effectively addresses their limitations. By seamlessly integrating the strengths of both paradigms, it introduces a robust framework that capitalizes on skill learning and model-based planning.

The offline learning model, a cornerstone of this approach, exhibits promising outcomes with a consistently decreasing loss. This trajectory suggests the potential for further improvements with extended training iterations. The observed trend implies that prolonged training could likely yield even more favorable results, reinforcing the effectiveness of the joint training approach employed by the authors. The model's capacity to refine its latent representation over time underscores the significance of this pioneering joint training strategy, marking a noteworthy step forward in enhancing latent space encoding for improved task performance.

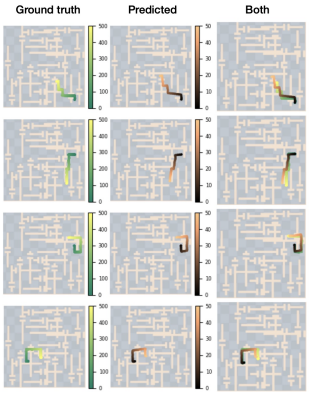

Figure 8: Maze using SKIMO comparing with ground truth - Shi, L. X et al. 2022

Our Contribution[9]

This paper's objective is to enhance the proficiency of artificial intelligence agents by teaching them to acquire high-level behaviors and skills from prior knowledge and to apply these learnings to downstream tasks. After a thorough examination and implementation of the paper, we believe there are areas that could be further developed:

1- The paper characterizes skills as sequences of ten actions of fixed length. There is potential to explore skills of variable lengths and to incorporate symmetry considerations into the skill-learning process.

2- The task policy in the study is initially set by a skill prior and consistently regularized by this skill prior with a fixed alpha, as described in Equation 6. This alpha could be dynamically adjusted over time, decreasing in importance with increased interaction with the environment and the acquisition of new data, thus new skills. This approach would allow for greater adaptation to new environments and skills, enhancing the generality of the learning process.

3- The current method involves learning and training skills and policies in latent spaces, which lack interpretability. We propose to transform these latent representations into a more interpretable form, akin to human language, to enhance explainability. This knowledge could then be transferred to downstream learning, making the processes more transparent and understandable.

References

Zhang, Z., Jin, J., Jagersand, M., Luo, J., Schuurmans, D. (2022). A Simple Decentralized Cross-Entropy Method. In Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Shi, L. X., Lim, J. J., Lee, Y.(2022). Skill-based Model-based Reinforcement Learning. In Proceedings of the 36th

[9]This part is done by Javad Moein

Conference on Neural Information Processing Systems (NeurIPS 2022).

Hausman, K.; Springenberg, J. T.; Wang, Z.; Heess, N.; and Riedmiller, M. 2018. Learning an Embedding Space for Transferable Robot Skills. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Thrun, S., Schwartz, A. (1994). Finding structure in reinforcement learning. In Advances in Neural Information Processing Systems (Vol. 7)

Sutton, R. S., Precup, D., Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. Artificial Intelligence, 112(1-2), 181-211.

Konidaris, G. D., Barto, A. G. (2007). Building Portable Options: Skill Transfer in Reinforcement Learning. In Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI), Vol. 7, pp. 895-900.

Rakelly, K., Zhou, A., Finn, C., Levine, S., Quillen, D. (2019). Efficient off-policy metareinforcement learning via probabilistic context variables. In International.

Singh, A.; Liu, H.; Zhou, G.; Yu, A.; Rhinehart, N.; and Levine, S. 2020. Parrot: Data-driven behavioral priors for reinforcement learning. arXiv preprint arXiv:2011.10024.

Ajay, A.; Kumar, A.; Agrawal, P.; Levine, S.; and Nachum, O. 2020. Opal: Offline primitive discovery for accelerating offline reinforcement learning. arXiv preprint arXiv:2010.13611.

Pertsch, K.; Lee, Y.; and Lim, J. 2021. Accelerating Reinforcement Learning with Learned Skill Priors. In Kober, J.; Ramos, F.; and Tomlin, C., eds., Proceedings of the 2020 Conference on Robot Learning, volume 155 of Proceedings of Machine Learning Research, 188–204. PMLR.

Whitney, W.; Agarwal, R.; Cho, K.; and Gupta, A. 2019. Dynamics-aware embeddings. arXiv preprint arXiv:1908.09357.

Hausman, K.; Springenberg, J. T.; Wang, Z.; Heess, N.; and Riedmiller, M. 2018. Learning an embedding space for transferable robot skills. In International Conference on Learning Representations.

Pertsch, K.; Lee, Y.; Wu, Y.; and Lim, J. J. 2021. Guided reinforcement learning with learned skills. arXiv preprint arXiv:2107.10253.

Pertsch, K.; Lee, Y.; and Lim, J. 2021. Accelerating Reinforcement Learning with Learned Skill Priors. In Kober, J.; Ramos, F.; and Tomlin, C., eds., Proceedings of the 2020 Conference on Robot Learning, volume 155 of Proceedings of Machine Learning Research, 188–204. PMLR.

Helmert, M. 2006. The fast downward planning system. Journal of Artificial Intelligence Research, 26: 191–246.

Kaelbling, L. P.; and Lozano-Perez, T. 2012. Integrated ´ robot task and motion planning in the now. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE COMPUTER SCIENCE AND ARTIFICIAL . . . .

Konidaris, G.; Kaelbling, L. P.; and Lozano-Perez, T. 2018. From skills to symbols: Learning symbolic representations for abstract high-level planning. Journal of Artificial Intelligence Research, 61: 215–289.

Konidaris, G. D.; and Barto, A. G. 2007. Building Portable Options: Skill Transfer in Reinforcement Learning. In IJCAI, volume 7, 895–900.