1. Introduction

* 1. Context  
       
     In the context of robotics, there has been a significant advancement in the application of robots in various areas. Reinforcement Learning (RL) has emerged as a powerful approach to enable robots to learn movement and perform tasks autonomously. However, controlling all the detailed torque of actuators manually by humans is often impractical and inefficient. To address this challenge, Variational Autoencoders (VAE) come into play as a valuable tool to assist in learning actions and compressing motion data.
  2. Aim and Objectives  
       
     The aim of this study is to apply reinforcement learning in a physics environment using mujoco to enable a virtual swimmer to perform various tasks. The objectives of the study are as follows:

1. Train the virtual swimmer to perform forward and backward swimming, turning right, and dancing using reinforcement learning techniques with Mujoco.
2. Collect motion data of the virtual swimmer and utilize Variational Autoencoders (VAE) to learn the underlying representations of the motion data.
3. Utilize the learned VAE model to compress the motion data into a latent space, which represents a lower-dimensional and meaningful representation of the swimmer's actions.
4. Investigate the characteristics of the latent space, where different motions are associated with specific mean and log\_var values, allowing for the creation of new motions by manipulating these parameters.

By achieving these objectives, the study aims to enhance the capabilities of the virtual swimmer and demonstrate how VAEs can assist in learning complex motions and generating novel actions, contributing to the broader field of robotics and reinforcement learning.

1. **Survey**
   1. **background of methods**
      1. **Reinforcement Learning**
         1. **MDP - Markov Decision Process (Markov Decision Processes-Martin L. Puterman)**

텍스트, 스크린샷, 도표, 라인이(가) 표시된 사진

자동 생성된 설명

Markov Decision Process (MDP) represents the fundamental learning mechanism in Reinforcement Learning. In this MDP environment, there are status(S), actions(a), and rewards(R). The current state reflects the accumulation of all previous history containing status and actions. In the figure, arrows came from block of status and action points to the next status(S), illustrating that the next status is determined by last timestep’s block. And the next action is decided by same timestep status. When performing an action, there is no need to consider previous timestep.  
  
Policy = P\left(s=s\_{i+1} \mid s=s\_{i}, a=a\_{i}\right)

폰트, 화이트, 텍스트, 서예이(가) 표시된 사진

자동 생성된 설명

Return = G\_{t} \doteq R\_{t}+\gamma R\_{t+1}+\gamma^{2} R\_{t+2}+\cdots

폰트, 친필, 서예, 화이트이(가) 표시된 사진

자동 생성된 설명

When the agent chooses the action(a\_t) in circumstance of status\_t, the probability of choosing action(a\_t) is Policy. After agent do action(a\_t), it will gets the Rewards.

* + - 1. State value function and Action value function (Reinforcement Learning: An Introduction – p.58)

폰트, 화이트, 도표, 타이포그래피이(가) 표시된 사진

자동 생성된 설명

G\_{t} \doteq \sum\_{k=t+1}^{T} \gamma^{k-t-1} R\_{k}

G\_t is the Return, representing the total reward that the agent will receive at each timestep. Gamma is the discount factor, aiding in the calculation of the return value as a present value. Gamma determines that the future value is less important than the present value. If I increase gamma, the future value becomes more significant.

State value function

v\_{\pi}(s) \doteq \mathbb{E}\_{\pi}\left[G\_{t} \mid S\_{t}=s\right]=\mathbb{E}\_{\pi}\left[\sum\_{k=0}^{\infty} \gamma^{k} R\_{t+k+1} \mid S\_{t}=s\right] \text {, for all } s \in \mathcal{S}

폰트, 텍스트, 화이트, 라인이(가) 표시된 사진

자동 생성된 설명

The State value function represents the expected return or the expected value of all states from the current state. This method calculates the current value of the agent's state, encompassing all past, present, and future state values.

Action value function

q\_{\pi}(s, a) \doteq \mathbb{E}\_{\pi}\left[G\_{t} \mid S\_{t}=s, A\_{t}=a\right]=\mathbb{E}\_{\pi}\left[\sum\_{k=0}^{\infty} \gamma^{k} R\_{t+k+1} \mid S\_{t}=s, A\_{t}=a\right]

폰트, 텍스트, 친필, 화이트이(가) 표시된 사진

자동 생성된 설명

On the other hand, actions can have their own current values. These must contain both state value and action value. The difference between the state value function and the action value function lies in the inclusion of the action value. The state value function does not consider previous episodes' actions since the state contains all the historical information. Therefore, the state value function considers only the state, while the action value function requires the state in the same episode.

* + - 1. Optimal Policy(Reinforcement Learning: An Introduction – p.73(pdf-p.95))

Finding a policy to maximise state value function is the definite goal of reinforcement learning. This policy is the optimal policy. The policy is an action value function. So, if we find the optimal policy, we need to find out the optimal action-value function.

q\_{\*}(s, a) \doteq \max \_{\pi} q\_{\pi}

폰트, 화이트, 친필, 서예이(가) 표시된 사진

자동 생성된 설명

q\_{\*}(s, a)=\mathbb{E}\left[R\_{t+1}+\gamma v\_{\*}\left(S\_{t+1}\right) \mid S\_{t}=s,\right.

폰트, 텍스트, 친필, 화이트이(가) 표시된 사진

자동 생성된 설명

The equation below is Bellman optimality equation. Bellman Optimality equation can transform V\_t to V\_t+1 or Q\_t to Q\_t+1.

텍스트, 폰트, 친필, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* + - 1. Soft Actor-Critic

Chapter 3

Experiments

1. Software and Envs(Lib and API)
2. **Hardware**  
   ARM architecture (M1 Max)

32GB RAM.

1. **Software and Environment**

Visual Studio Code: 1.80.2

Rl-baseline3-zoo(https://stable-baselines3.readthedocs.io/en/master/index.html)

Python version: Python 3.11.

PyTorch: torch 2.0.1

Numpy: 1.24.3

Pandas: 2.0.2

Stable baseline3: 2.0.0

Gym: 0.26.2

Mujoco\_py: 2.1.2.14

Mujoco:2.3.6

Tensorboard: 2.13.0

1. Reinforcement Learning
2. Env

Reinforcement learning tested with RL-baseline3-zoo. RL-baseline3-zoo is a framework that can embody stable baseline3(SB3). SB3 is the tool for implementing reinforcement learning algorithms to train and evaluate the agent. SB3 can realise parallel training. This is helpful to minimise the resource needed to train agents.  
among gym environments, I picked Swimmer-v3. Swimmer can move with three segments and two joints. This environment has two types of space. (Reinforcement Learning Using Neural Networks, with Applications to Motor Control) It has an Action space and an Observation space. Action space indicates the movement of two joints’ action. And action space produces the observation space that has eight columns. Each column contains speed, direction, and angle.

Reinforcement learning was conducted using the RL-baseline3-zoo framework, a powerful tool built upon the stable baseline3 (SB3) library. SB3 offers a robust platform for implementing reinforcement learning algorithms to train and evaluate agents effectively. One notable feature of SB3 is its ability to facilitate parallel training, which significantly reduces the computational resources required during the training process.

For the experimentation, the Swimmer-v3 environment was selected from the Gym environments. The Swimmer environment comprises three segments and two joints, enabling complex movement patterns. Within this environment, two distinct spaces exist: the Action space and the Observation space. The Action space governs the motion of the two joints, while the resulting actions give rise to an eight-dimensional Observation space. Each dimension of the Observation space encodes information such as speed, direction, and angle. This setup closely follows the description in "Reinforcement Learning Using Neural Networks, with Applications to Motor Control".

[액션 표]

[obs 표]

1. The movement of Swimmer

In this experiment, Swimmer will train four movements.

Chapter. 1

1. Introduction
   1. Context
      1. Robots are spread out in many areas.
      2. Robots learn the movement with RL.
      3. Humans can’t control all the detailed torque of the actuator.
      4. VAE can help to learn the actions.   
         (low dimension의 장점, 이걸로 비정상을 탐지 )
   2. Aim and Objectives
      1. Aim = making a new motion with VAE
      2. Objective
         1. Research gap
         2. Making fundamental motion with reinforcement learning  
            Swimmer swim forward, backward, turn right, and jelly fish.  
            vae model learn from RL  
            Swimmer can make new motion
         3. Conclusion
         4. Futurework
      3. Swimmer swim forward and backward. And turn right and dance.
      4. Collecting swimmer’s motion data and learning VAE.
      5. VAE can compress all the data to latent space.
      6. Many motion has special mu and log\_var
      7. Increasing or changing mu and log\_var allows us to create new motions.

Chapter 2

1. Survey
   1. background of methods
      1. Reinforcement Learning
         1. Q-learning
         2. MDP
         3. State Value function & action value function
         4. Optimal policy
         5. PPO
         6. SAC(Soft Actor-Critic)
         7. Mujoco experiment summary
      2. Variational Autoencoders (vanilla 스토케스틱 그라디언트)
         1. VAE structure
         2. Few-shot learning
   2. Related work

Chapter 3

1. Experiments
   1. Software and env(library and api)
      1. Pytorch, Mujoco, stable-baseline3, rl-baseline3-zoo
   2. Reinforcement Learning(RL-Baseline3-zoo)
      1. Mujoco Swimmer-v3. Action space and observation space
      2. Comparing algorithms by each pose
      3. Output of RL
   3. Variational Autoencoder
      1. Preprocessing Dataset from RL
      2. Model Architecture
      3. Rendering
2. Evaluation
   1. Loss plot
   2. Rendering compare
   3. Action and observation plot between original data and reconstructed data.

Chapter 5

5.1 Discussion

Chapter 5

5.1 Conclusion

Bibliography