EE488D Introduction to Reinforcement Learning Report

Team number: 22

Team member names: Seungjoo Lee, Yewon Kim

Section 1. [Problem 1] hovering task

Subsection 1. Approach

When implementing the hovering task, we used two approaches: Soft Actor Critic and Curiosity-driven Experience Replay.

1. Soft Actor Critic (SAC)

   We implemented the Soft Actor Critic (SAC) algorithm for discrete action space[2]. Key ideas of SAC are using entropy regularization, and a temperature parameter. Entropy represents the randomness or uncertainty in a probability distribution. Through policy entropy maximization, SAC promotes exploration and prevents premature convergence. During training, the policy's entropy is adjusted according to the predefined target entropy. SAC introduces a temperature parameter, α (alpha), to manage exploration and exploitation trade-offs. Higher values of α result in more exploration, while lower values prioritize exploitation. We made this temperature parameter as a learnable parameter.

   In addition, SAC utilizes two critic networks and gradient clipping to prevent overestimation and stabilize the training process. As part of our efforts, we also attempted to incorporate the concept of 'Delayed policy update' from TD3 into the final model, but it was not able to work well within SAC settings, so it was not included.

2. Curiosity-driven Experience Replay (CER)

   In conventional reinforcement learning paradigms, random sampling strategies are typically employed for replay buffer management. Since the random sampling strategy does not explicitly encourage an agent to explore novel experiences, in order to maximize the information gain and training speed, we instead implemented the Curiosity-driven Experience Replay (CER) algorithm.

   This approach was motivated by various curiosity-driven exploration mechanisms [3] that aim to incentivize an agent to seek novel experiences, thereby enhancing learning efficiency. Instead of selecting samples randomly, the CER algorithm leverages an intrinsic reward system to prioritize samples that provide the agent with the most novel information. As such, this strategy could lead to more effective learning, especially in complex, non-stationary environments. Specifically, we reward the agent for taking actions that lead to states which the agent cannot predict well, gauging this by the error of a specifically trained model of the environment. We approached this by training a simple predictor model that predicts the next state given the current state and action, and used the prediction error as an intrinsic reward. We thus promote an environment where the agent is incentivized to maximize the predictor's per-sample error on state transitions. Assuming the predictor is more accurate on transitions it frequently encounters, the agent will prioritize infrequent state transitions, thereby enhancing the agent's learning efficiency.

We further accelerate the training process by utilizing softmax probability and temperature scaling [1]. Specifically, upon computation of prediction errors from the predictor, these values in the memory buffer are normalized into probabilities via a softmax function. We further ensure that the agent more frequently explores novel samples by using a low temperature value, i.e., 0.3, in order to sharpen the probability distribution so that samples with higher prediction errors get more chance to be selected.

In addition, in order to balance exploration and exploitation, we increase the temperature value in the later stage of the training phase. This approach ensures that, in the latter stages of the training process, the selection paradigm gradually approximates uniform sampling, as opposed to primarily curiosity-driven exploration. This methodology imparts the model with the ability to transition from a highly exploratory initial phase—during which it is incentivized to learn about and navigate unfamiliar states—to a subsequent phase where it capitalizes on its acquired knowledge for efficient problem-solving. Specifically, after 1200 episodes, we set the temperature value as 2.0.

We also tune the hyperparameters and select the best result. We list the hyperparameter spaces we explored below.

- temperature: [0.3, 0.5, 0.7, 1.0, 1.3, 1.5]
- memory buffer size: [10000 50000 100000 500000]

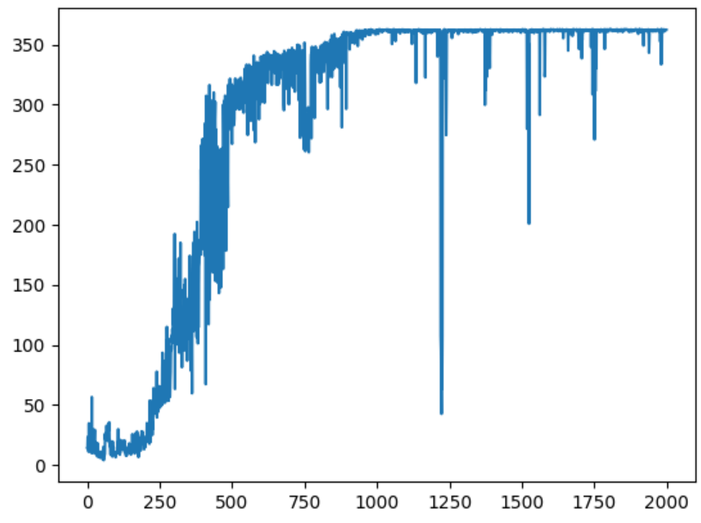
- gradient clipping: [0.1, 0.5, 1.0, 5.0, 10.0]

The hyperparameter set we selected:

- temperature: 0.3
- memory buffer size: 50000
- gradient clipping: 10.0

[1] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On calibration of modern neural networks. In Proceedings of the 34th International Conference on Machine Learning - Volume 70 (ICML'17). JMLR.org, 1321–1330.
[2] Christodoulou, P. (2019). Soft actor-critic for discrete action settings. arXiv preprint arXiv:1910.07207.
[3] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell (2017). "Curiosity-driven exploration by self-supervised prediction". In: International Conference on Machine Learning (ICML). Vol. 2017.

Subsection 2. Return plot



Subsection 3. Performance ratio

$$ratio = \frac{Everage\ of\ G_0\ of\ your\ learned\ policy\ over\ 100\ episode}{17} = \frac{27.847}{17} \simeq 1.638$$

Section 2. [Problem 2] landing task

Subsection 1. Approach

In the landing task, we used two approaches (Same as problem 1: hovering task): Soft Actor Critic and Curiosity-driven Experience Replay.

1. Soft Actor Critic (SAC; Same as problem 1)

   We implemented the Soft Actor Critic (SAC) algorithm for discrete action space[2]. Key ideas of SAC are using entropy regularization, and a temperature parameter. Entropy represents the randomness or uncertainty in a probability distribution. Through policy entropy maximization, SAC promotes exploration and prevents premature convergence. During training, the policy's entropy is adjusted according to the predefined target entropy. SAC introduces a temperature parameter, α (alpha), to manage exploration and exploitation trade-offs. Higher values of α result in more exploration, while lower values prioritize exploitation. We made this temperature parameter as a learnable parameter.

   In addition, SAC utilizes two critic networks and gradient clipping to prevent overestimation and stabilize the training process. As part of our efforts, we also attempted to incorporate the concept of 'Delayed policy update' from TD3 into the final model, but it was not able to work well within SAC settings, so it was not included.

2. Curiosity-driven Experience Replay (CER; Same as problem 1)

   In conventional reinforcement learning paradigms, random sampling strategies are typically employed for replay buffer management. Since the random sampling strategy does not explicitly encourage an agent to explore novel experiences, in order to maximize the information gain and training speed, we instead implemented the Curiosity-driven Experience Replay (CER) algorithm.

   This approach was motivated by various curiosity-driven exploration mechanisms [3] that aim to incentivize an agent to seek novel experiences, thereby enhancing learning efficiency. Instead of selecting samples randomly, the CER algorithm leverages an intrinsic reward system to prioritize samples that provide the agent with the most novel information. As such, this strategy could lead to more effective learning, especially in complex, non-stationary environments. Specifically, we reward the agent for taking actions that lead to states which the agent cannot predict well, gauging this by the error of a specifically trained model of the environment. We approached this by training a simple predictor model that predicts the next state given the current state and action, and used the prediction error as an intrinsic reward. We thus promote an environment where the agent is incentivized to maximize the predictor's per-sample error on state transitions. Assuming the predictor is more accurate on transitions it frequently encounters, the agent will prioritize infrequent state transitions, thereby enhancing the agent's learning efficiency.

We further accelerate the training process by utilizing softmax probability and temperature scaling [1]. Specifically, upon computation of prediction errors from the predictor, these values in the memory buffer are normalized into probabilities via a softmax function. We further ensure that the agent more frequently explores novel samples by using a low temperature value, i.e., 0.7, in order to sharpen the probability distribution so that samples with higher prediction errors get more chance to be selected.

In addition, in order to balance exploration and exploitation, we increase the temperature value in the later stage of the training phase. This approach ensures that, in the latter stages of the training process, the selection paradigm gradually approximates uniform sampling, as opposed to primarily curiosity-driven exploration. This methodology imparts the model with the ability to transition from a highly exploratory initial phase—during which it is incentivized to learn about and navigate unfamiliar states—to a subsequent phase where it capitalizes on its acquired knowledge for efficient problem-solving. Specifically, after 1200 episodes, we set the temperature value as 2.0.

We also tune the hyperparameters and select the best result. We list the hyperparameter spaces we explored below.

- temperature: [0.3, 0.5, 0.7, 1.0, 1.3, 1.5]

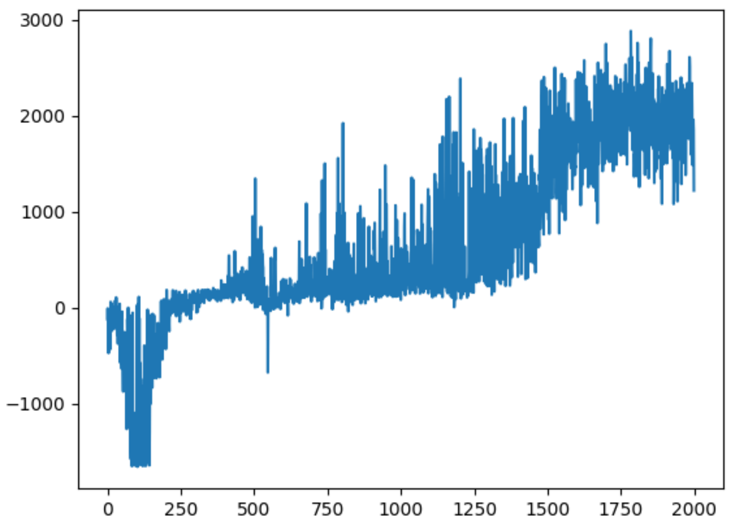
- memory buffer size: [10000 50000 100000 500000]
- gradient clipping: [0.1, 0.5, 1.0, 5.0, 10.0]

The hyperparameter set we selected:

- temperature: 0.7
- memory buffer size: 500000
- gradient clipping: 5.0

[1] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On calibration of modern neural networks. In Proceedings of the 34th International Conference on Machine Learning - Volume 70 (ICML'17). JMLR.org, 1321–1330.
[2] Christodoulou, P. (2019). Soft actor-critic for discrete action settings. arXiv preprint arXiv:1910.07207.
[3] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell (2017). "Curiosity-driven exploration by self-supervised prediction". In: International Conference on Machine Learning (ICML). Vol. 2017.

Subsection 2. Return plot



Subsection 3. Performance ratio

$$ratio = \frac{Everage\ of\ G_0\ of\ your\ learned\ policy\ over\ 100\ episode}{120} = \frac{172.450}{120} \simeq 1.437$$