Lab report - Mini project 1

In this mini project, we explore Actor-Critic algorithm in a `MazeMDP-v0` environment. Afterward, we use both Grid search and a Bayesian optimization method to tune the key hyperparameters, namely the learning rates for the critic and the actor, using `Optuna` package. Welch's t-test is then used to compare the performances of two hyperparameter-tuning methods and the naive tuning.

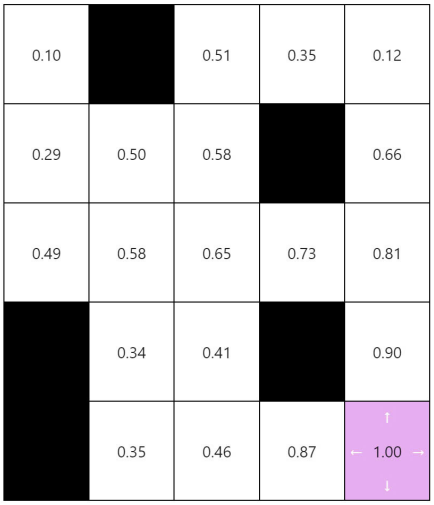
1 Actor - Critic Algorithm

1.1 Setup

- Environment: The environment is a random maze created using `Gymnasium` package. The maze is of size 5x5 with 20% of the grid cells occupied by walls. Each time the environment is reset, the agent's initial state is drawn using a uniform distribution over all the states. This configuration creates an environment that is challenging enough to demonstrate the true effectiveness of the learning algorithm, ensuring that success is not simply due to an easy setting.
- Parameters of the learning algorithm:
  - Training time: The agent is trained over 100 episodes, with each episode having a maximum timeout of 200 steps. These numbers are chosen to provide sufficient time for exploration of different paths in each training episode, while ensuring enough episodes for learning optimal strategies without excessive computational burden.
  - Learning rates: Leaning rates for both the critic and the actor are set to 0.5 each. This is a naive setting, and in Section 2, we will use `Optuna` package to tune these hyperparameters.
  - Initialization: The value function V is initialized to zero for all states, while the policy is initialized to be uniform across all actions, allowing the agent to explore all possible actions at the beginning of training.

1.2 Test result

By performing Actor-Critic algorithm using the presented parameters, we obtained the following results:



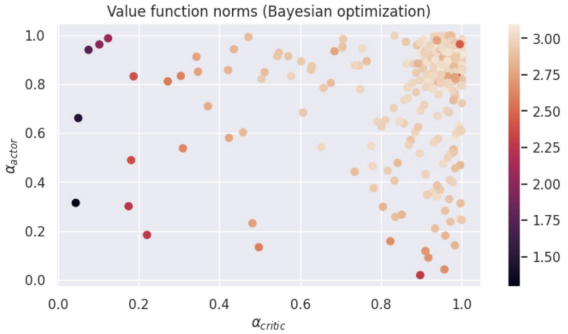
(a) V function

(b) Decrease of episode length through time

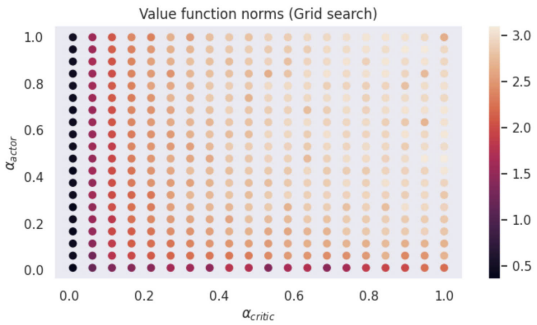
We can observe that, although it is a naive approach of Actor-critic, the algorithm's performance appears to be quite good. The obtained value function is nearly similar to the optimal value function obtained using Dynamic Programming, despite the fact that the agent does not know anything about the environment. Moreover, the number of required steps to find the final reward in each episode also decreases as the training progresses.

2 Hyper-parameters tuning algorithms

Two methods: Bayesian optimization and Grid search are employed over 400 training runs overall for each in order to find the best values of two hyperparameters: the critic and actor learning rates. The heatmap obtained from each method is as followed:



(a) Bayes method

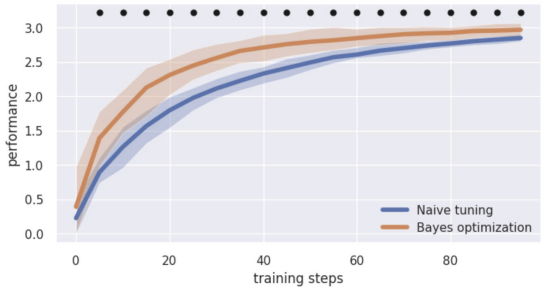


(b) Grid-search method

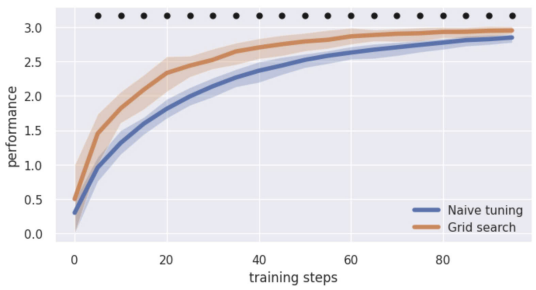
After running the optimization procedures, the best hyperparameters found through Bayesian optimization are `alpha_critic` = 0.9778 and `alpha_actor` = 0.9067, while grid search provides `alpha_critic` = 1.0 and `alpha_actor` = 0.4789. It can be observed that for both

case, the best learning rate for `alpha_critic` is higher than that for `alpha_actor`, which can be explained as the critic is needed to update the actor and thus, needed to be learn faster.

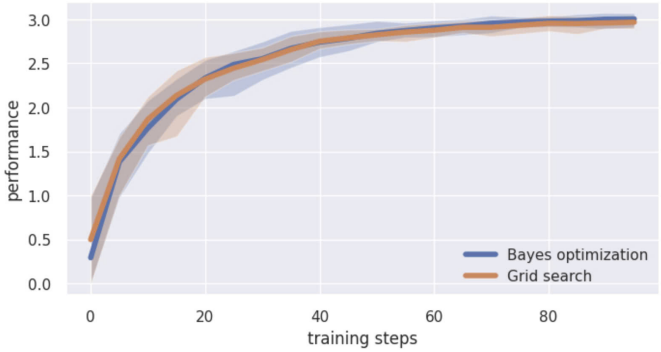
To analyze the statistical significance of the results obtained from the two hyperparameter tuning methods, Welch's t-test was performed. In general, there is no significant difference between the performances of the two methods for this problem. Furthermore, both methods show considerably better performances compared to the naive approach, where the hyperparameters are set without tuning. The improvement is notable across "almost all training steps" (roughly speaking, since we only collect the significant differences every 5 training steps.)



(a) Naive tuning and Bayes method



(b) Naive tuning and Grid-search method



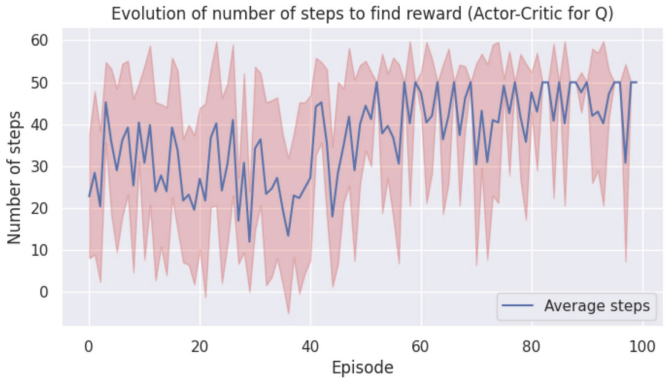
(c) Bayes method and Grid-search method

3 Extension

Beside Actor-critic algorithm using value function V as the critic, we try to implement the algorithm using Q function.

The algorithm successfully finds an optimal policy, as can be seen in the value table. However, the number of steps taken to find the reward fluctuates remarkably throughout the training, rather than showing a clear downward trend. We were unable to identify the underlying reason for this phenomenon, and we would greatly appreciate any comments or suggestions from the readers.

(a) V function and policy



(b) Evolution of episode length through time