Lab Report

—— Jinyi WU, Lecheng WANG

Exercise 1

**Process of algorithm:**

1. Initialize  for every steps. For each step, the policy is a uniform distribution.

2 Set the maximum number of episodes. We will terminate the loop when the maximum episodes is reached.

3 For each episode, continue the following steps until either ‘timeout’ occurs or the final state is reached:

1. Randomly chose the start state .
2. Chose an action for state , using random sampling according to the distribution of action.
3. Perform the action, get the next state , the reward , and check if the process terminates or ‘timeout’.
4. Compute the delta, update , and update  to adjust the policy for the next iteration.
5. Use the current state as the start of next loop and skip a).

4 Data collected during the process:

a) **V**: we will return the V value we get.

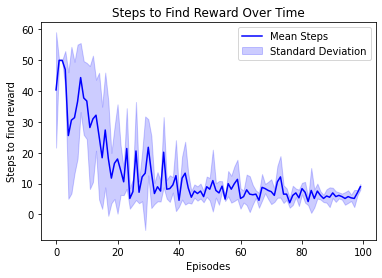
b) **V norms**: Tracks the growth of V values, which should converge over time.

c) **Time list**: Record the number of steps taken in each episode. Initially, it’s hard for the agent to reach the final state, so we need to loop until timeout. As the agent improves, we found that even the start is random, it takes less steps to find the final state. Which means the policy is more and more effective.

1. Plot function: repeatedly call the actor-critic algorithm to show the number of steps the agent takes to find the reward.

**Parameters in the Exercise1:**

****



**Results for Exercise 1:**

The algorithm successfully updated the V values and provided a useful policy for the maze. With the growth of episodes, the steps it takes to find the reward significantly decrease.

Exercise 2

**Grid Search Method:**

* + 1. We generated 20 values for both  and  using . Then we use  to create 400 pairs of parameters for grid search.
    2. For each parameter pair, we evaluated the final V value obtained. The lager the V value, the better the performance. To avoid coincidence, we run each pair for 5 times and get a average value of V.
    3. After running all pairs, we identified the optimal parameters. The results are(each time can have slightly difference):



The value of  is relatively stable, often exceeding 0.9. However, varied more, with values typically between 0.6 and 0.9. The reasons for this variation may include:

-The method itself has uncertainty. The initial state and actions for each state is chosen randomly (depends on distribution of policy) instead of the largest one.

-The evaluation is base on V value, but it is possible that with in 100 episodes, there are a range of alpha can lead the V value get from iteration close enough the true value. In this case, the uncertainty can dominate the difference between different alpha thus hide the actual performance of each pair of parameters.

**Bayesian Optimization Method:**

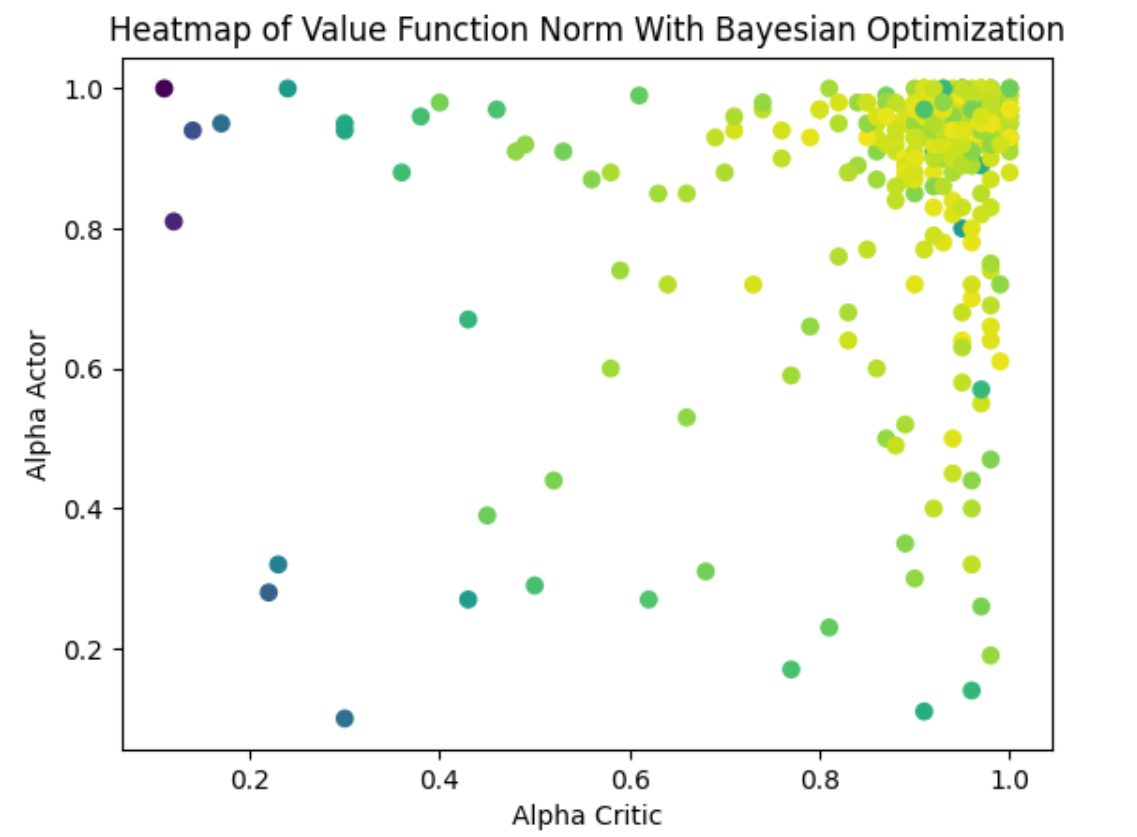
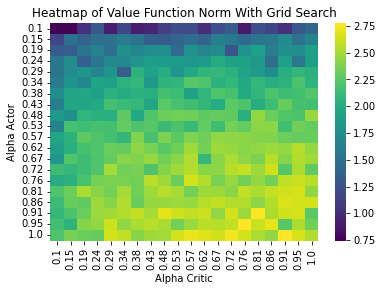
In this method, we apply to conduct Bayesian optimization. The process framework is similar to the grid search, but instead of fixing the parameters in advance, selected them dynamically between 0.1 and 1. The results are:



**Heatmap Generation:**

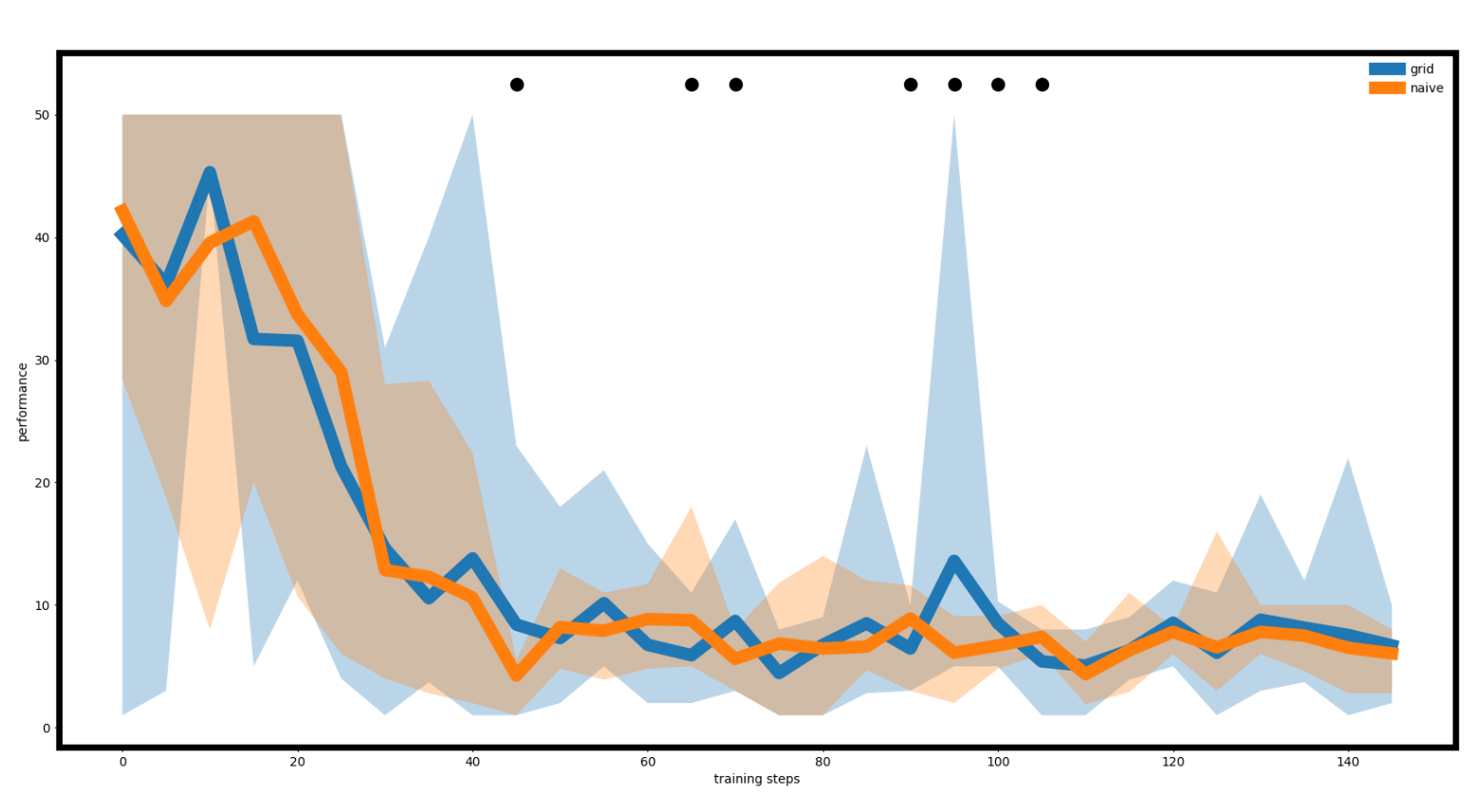
Generating a heatmap for grid search is straightforward because the grid is uniform over the two axes. However, a direct heatmap is not possible for the Bayesian method due to the non-uniform

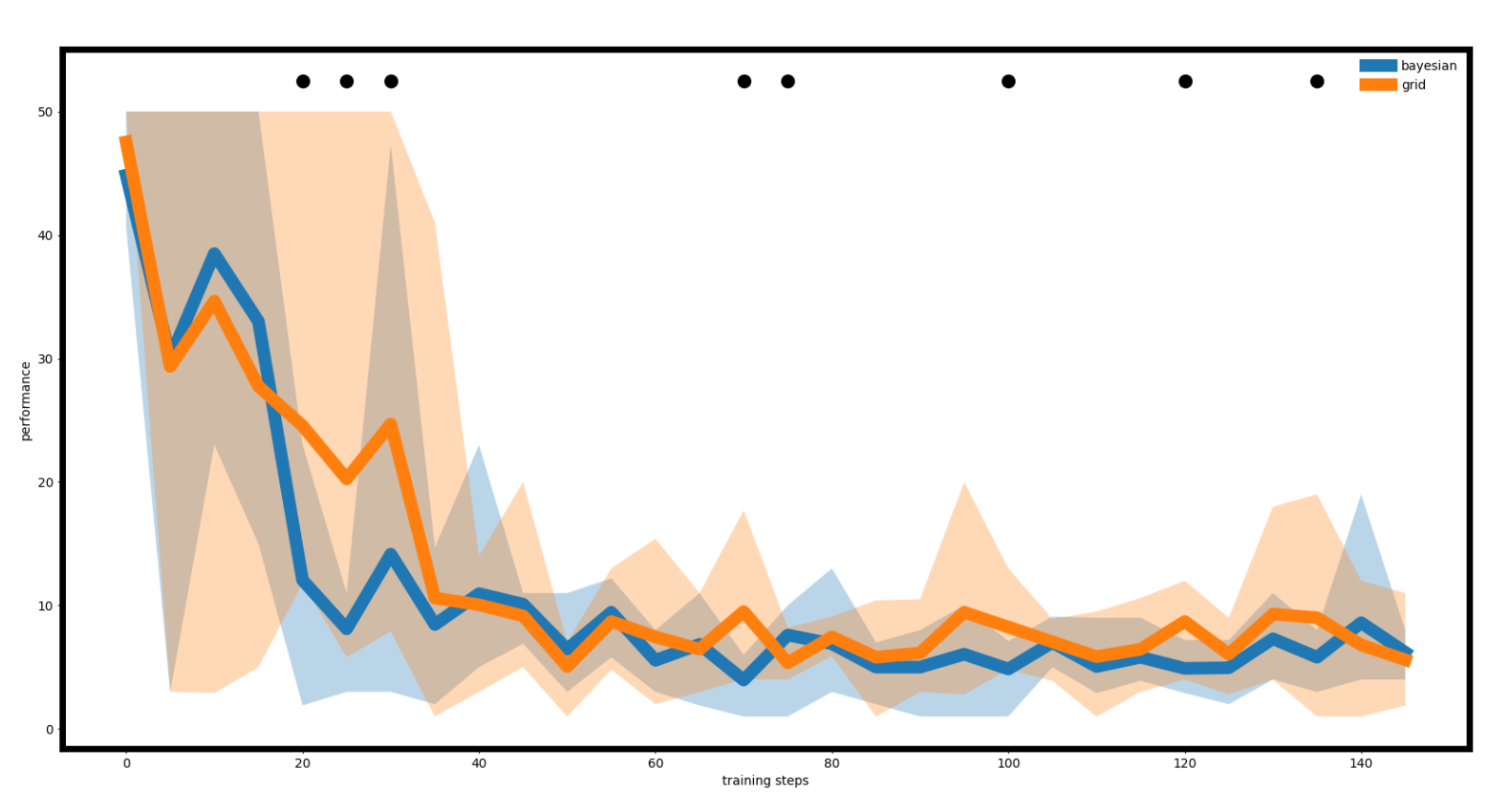
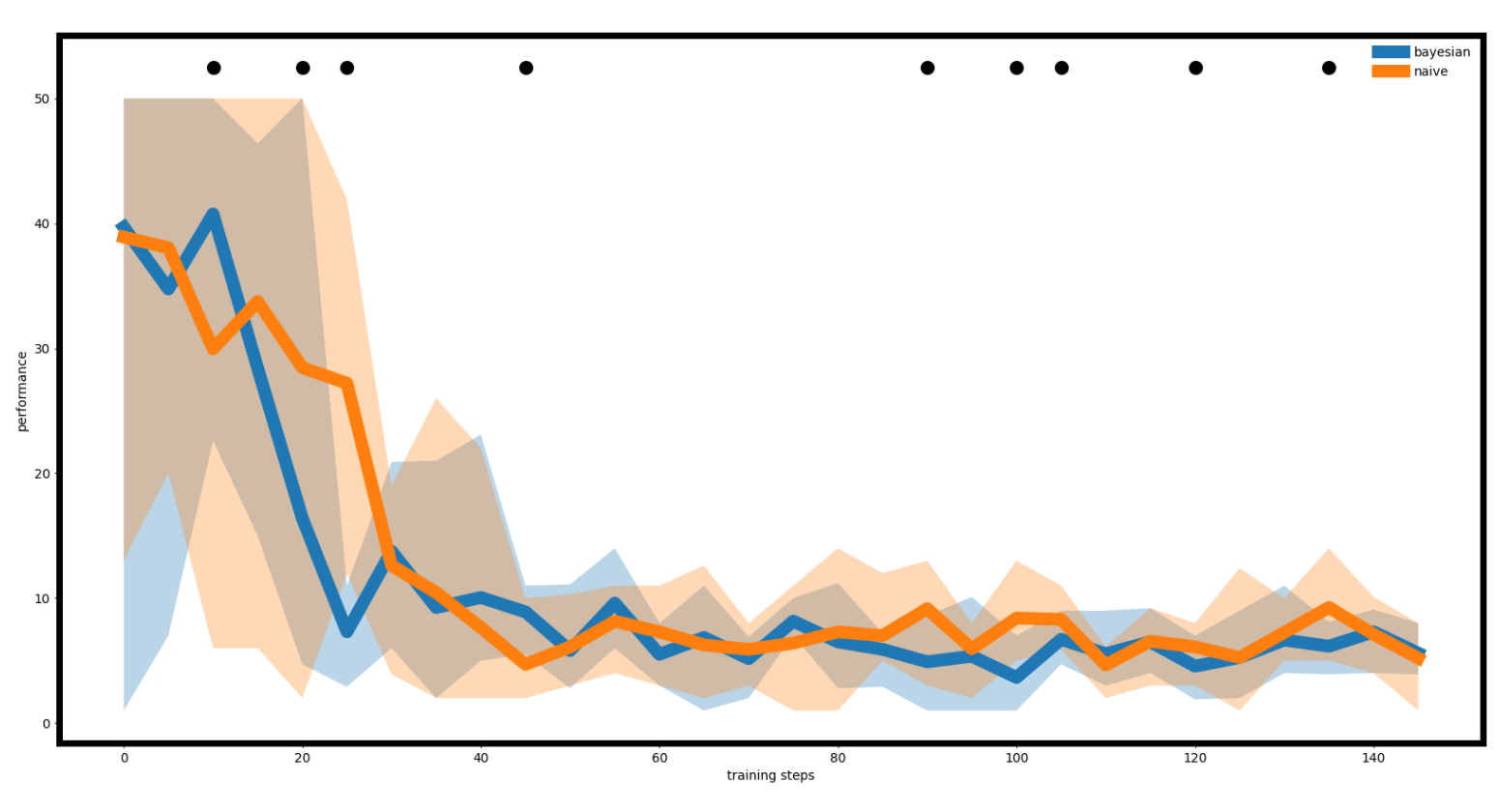
sampling of values. Instead, we used a scatter plot with color to represent the corresponding V values, visualizing the distribution of points and their performance.



Exercise 3

We use the functions provided in the notebook to test the results and evaluate the performance differences between hyperparameters. Below are some representative images.





**Conclusion:**

After 50 episodes, we have to admit that there are no significant difference between the V values obtained from different values of . The optimized parameters obtained from both grid search and Bayesian optimization are slightly better than the naive parameters when the episode is small. We believe that the naive  isn’t that bad in the first place, the grid search backs our point. So, if we compare the medium parameters with the best parameters, considering the uncertainty of the algorithm, the difference become less and less during the iteration is a reasonable conclusion.