Homework 4 Submission

Use this template to record your answers for Homework 4. Add your answers using LaTeXand then save your document as a PDF to upload to Gradescope. You are required to use this template to submit your answers. **You should not alter this template in any way** other than to insert your solutions. You must submit all 9 pages of this template to Gradescope. Do not remove the instructions page(s). Altering this template or including your solutions outside of the provided boxes can result in your assignment being graded incorrectly.

You should also export your code as a .py file and upload it to the **separate** Gradescope coding assignment. Remember to mark all teammates on **both** assignment uploads through Gradescope.

Instructions for Specific Problem Types

On this homework, you must fill in blanks for each problem. Please make sure your final answer is fully included in the given space. **Do not change the size of the box provided.** For short answer questions you should **not** include your work in your solution. Only provide an explanation or proof if specifically asked.

Fill in the blank: What is the course number?

10-703

Problem 0: Collaborators

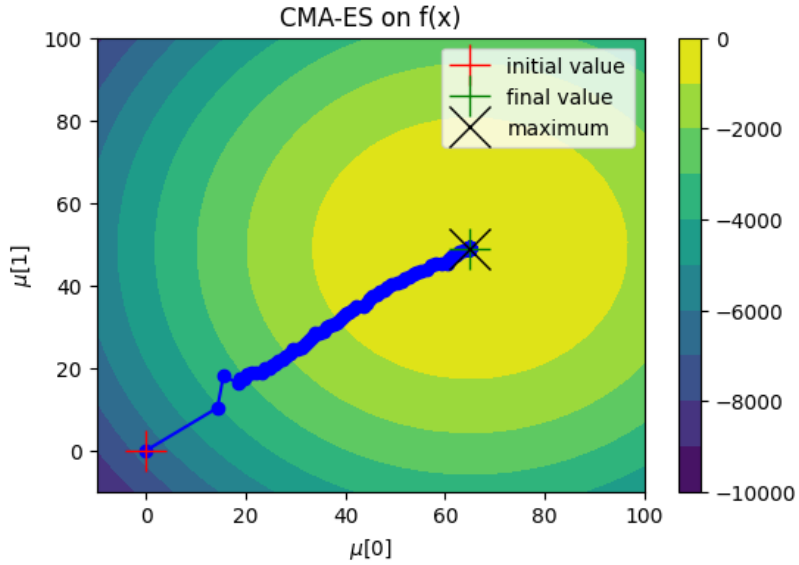
Enter your team members' names and Andrew IDs in the boxes below. If you worked in a team with fewer than three people, leave the extra boxes blank.

Name 1:	Manyung Hon	Andrew ID 1:	mehon
Name 2:	Boxiang Fu	Andrew ID 2:	boxiangf
Name 3:	Amy Jiang	Andrew ID 3:	amyjiang

Problem 1: CMA-ES (24 pts)

1.1 Plot of CMA-ES on simple objective function (10 pts)

Solution



1.2 RL reward of fixed policies (4 pts)

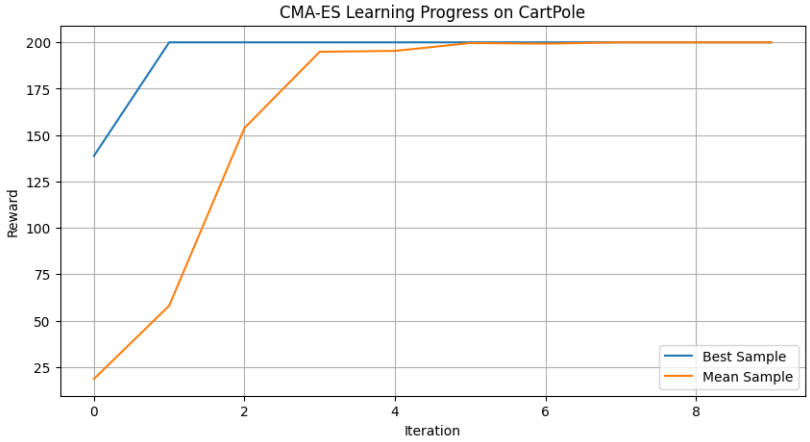
$x = (-1, -1, -1, -1, -1)$: 15.6

$x = (1, 0, 1, 0, 1)$: 14.5

$x = (0, 1, 2, 3, 4)$: 9.4

1.3 Plot of CMA-ES on Cartpole (10 pts)

Solution

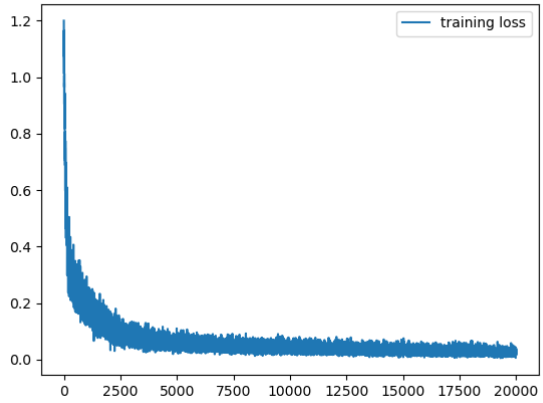


Problem 2: Imitation Learning (62 pts)

Problem 2.1: BC (14 pts)

2.1.1 Loss Plot + Final Loss Value BC (6 pts)

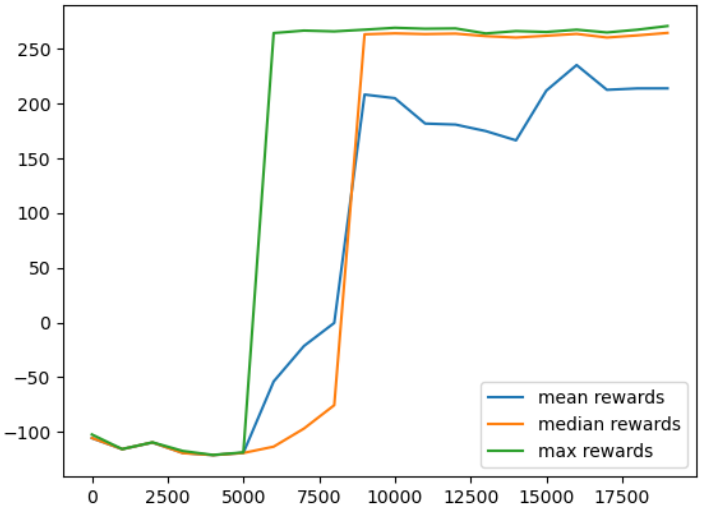
Solution



0.03293710574507713

2.1.2 Rewards Plot BC (6 pts)

Solution



2.1.3 GIF link BC (2 pts)

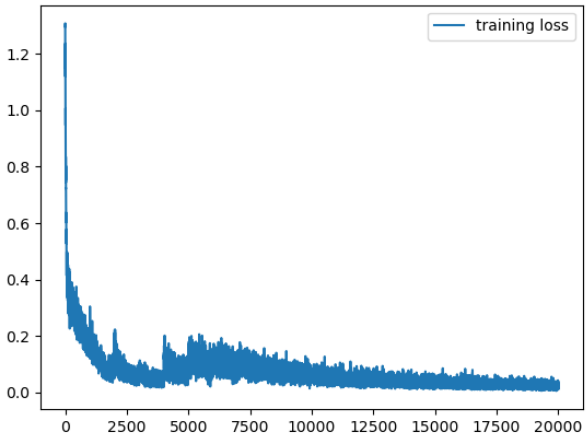
Solution

Find in code submission here.

Problem 2.2: DAgger (18 pts)

2.2.1 Loss Plot DAgger (6 pts)

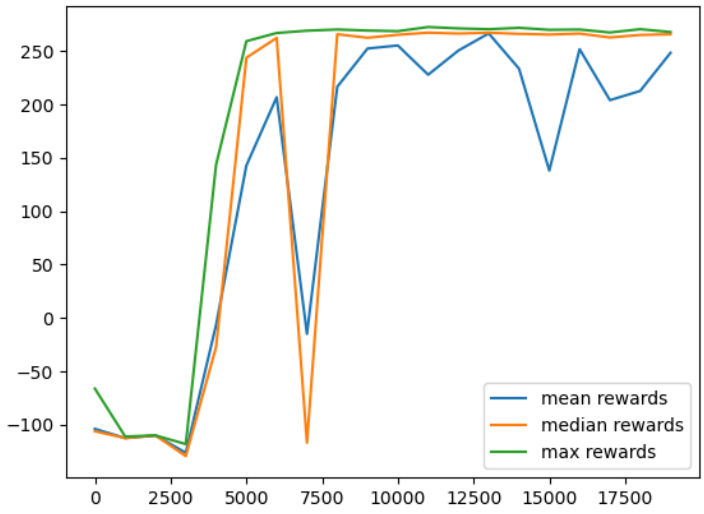
Solution



0.036606065928936005

2.2.2 Rewards Plot DAgger (6 pts)

Solution



2.2.3 GIF link DAgger (2 pts)

Solution

Find in code submission here.

2.2.4 Compare DAgger training with BC (written, 4 pts)

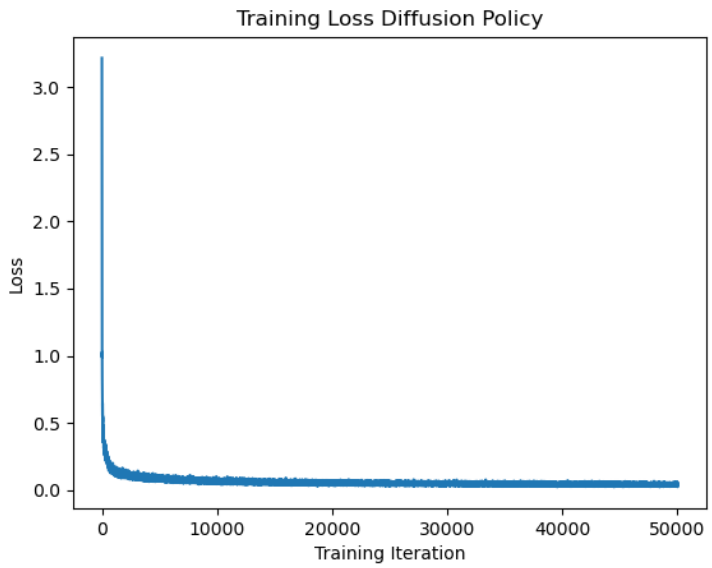
Solution

DAgger worked significantly better than BC, achieving both higher mean performance and dramatically reduced variance compared to BC. This is because DAgger has a higher ability to handle distributional shift through iterative data collection and aggregation, making it far more robust and reliable, while BC trains exclusively on states from expert demonstrations. Once the agent enters unfamiliar states, it has no learned behaviors to guide it, and errors compound rapidly, as seen in the walker GIF comparison, where BC is not able to recover from the fall.

Problem 2.3: Diffusion Policy (30 pts)

2.3.1 Loss Plot + Final Loss value Diffusion Policy (6 pts)

Solution



0.04196654260158539

2.3.2 Rewards Diffusion Policy (15 pts)

2.3.2.1; 3 actions evaluated in a row (5 pts)

avg trajectory time: 77.79 mean: 224.57 median: 262.46 max: 266.71

2.3.2.2; 2 actions evaluated in a row (5 pts)

avg trajectory time: 123.96 mean: 254.34 median: 261.19 max: 264.39

2.3.2.3; 1 action evaluated in a row (5 pts)

avg trajectory time: 269.00 mean: 259.90 median: 260.58 max: 263.56

2.3.3 GIF link Diffusion Policy (2 pts)

Solution

Find in code submission here.

2.3.4 Compare diffusion policy and simple model runtime (written, 4 pts)

Solution

Diffusion policy takes longer because it generates actions through an iterative denoising process rather than direct predictions. For each action, the diffusion policy must perform 30 sequential forward passes through a 6 layer transformer network, starting from random noises and gradually refining it into a coherent action. Opposite to that, BC and DAgger generate actions with a single forward pass through a simple 3 layer MLP. This means the diffusion policy performs approximately 30 times more neural network evaluations per action, which is shown in our results as well, and each evaluation is more expensive due to the transformer's attention mechanisms.

2.3.5 Compare diffusion policy with different actions in a row runtime (written, 3 pts)

Solution

The difference in average time is due to the frequency of diffusion sampling calls required for trajectory generation. When $num\_actions = 1$, the model needs to perform a complete diffusion sampling process for every single action, resulting in approximately 900 diffusion calls for a typical trajectory and the highest average time. When $num\_actions = 2$, the model predicts 2 actions per diffusion call, reducing the number of calls by half. When $num\_action = 3$, only around 300 diffusion calls are needed, resulting in the fastest time. Since each diffusion sampling call involves 30 forward passes through the transformer regardless of the actions, the time saving scale approximately linear with the reduction in sampling calls. This explains the observed pattern where $num\_actions = 3$ is 3.5 times faster than $num\_action = 1$, matching the 3 times reduction in the number of diffusion sampling calls.