Homework 3 Submission

Use this template to record your answers for Homework 3. 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 11 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

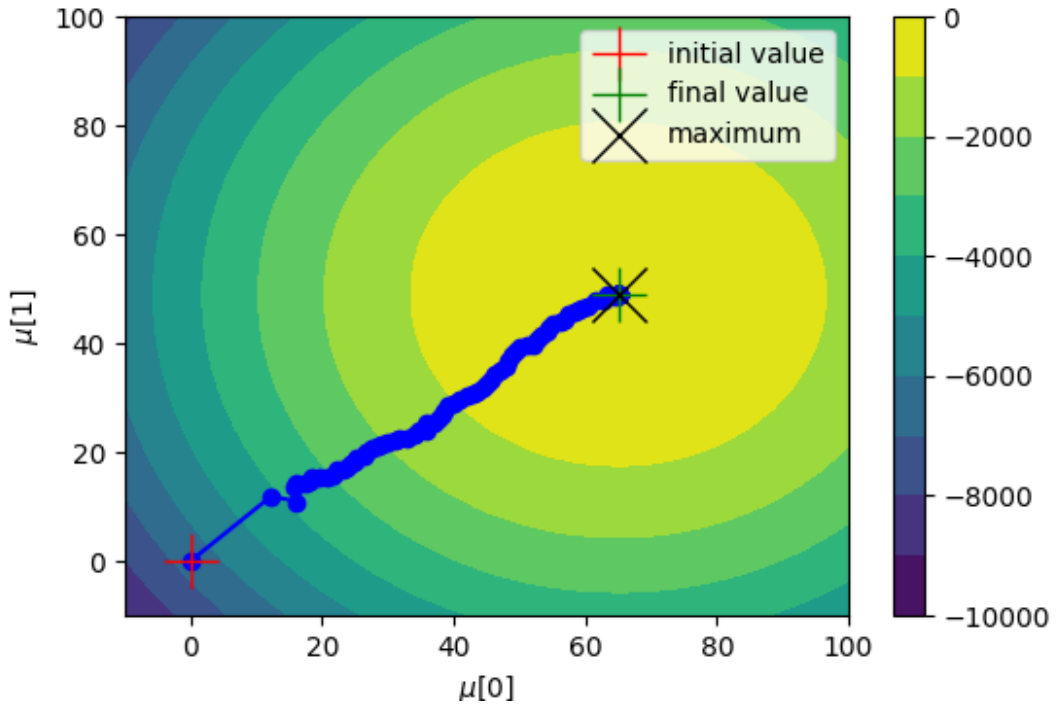
Enter your team members' names and Andrew IDs in the boxes below. If you worked in a team with fewer than three people, leave the extra boxes blank.

Name 1:	Shrudhi Ramesh Shanthi	Andrew ID 1:	srameshs
Name 2:	Madhusha Goonesekera	Andrew ID 2:	mgoonese
Name 3:	Siddharth Ghodasara	Andrew ID 3:	sghodasa

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)

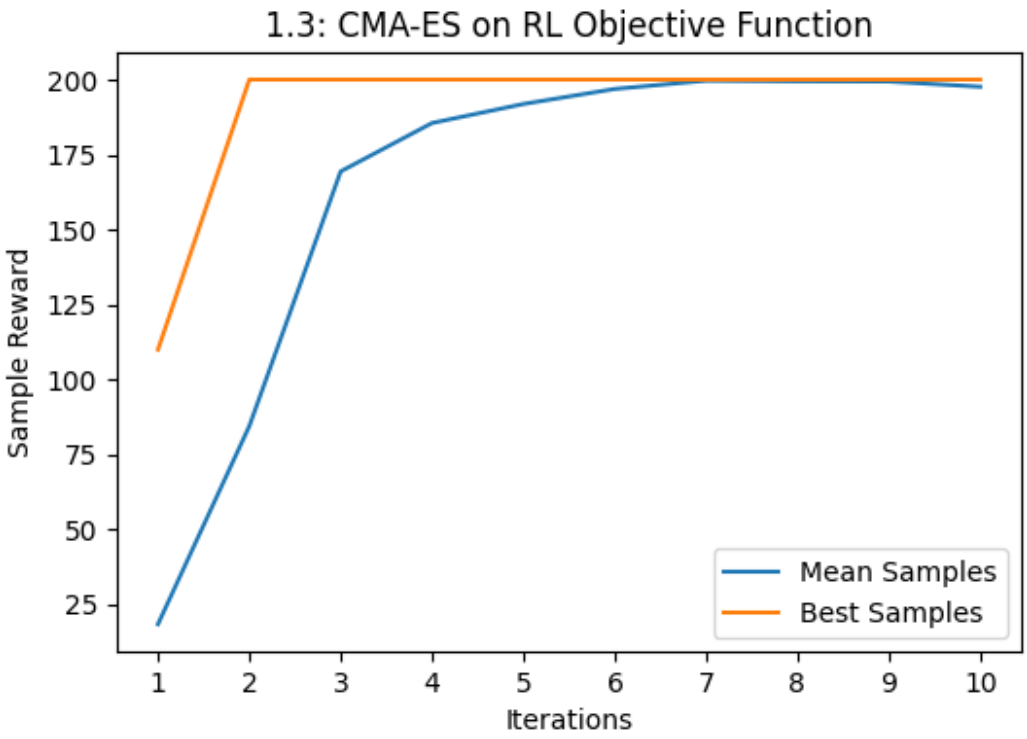
$x = (-1, -1, -1, -1, -1)$: 15.75

$x = (1, 0, 1, 0, 1)$: 14.61

$x = (0, 1, 2, 3, 4)$: 9.44

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

Solution

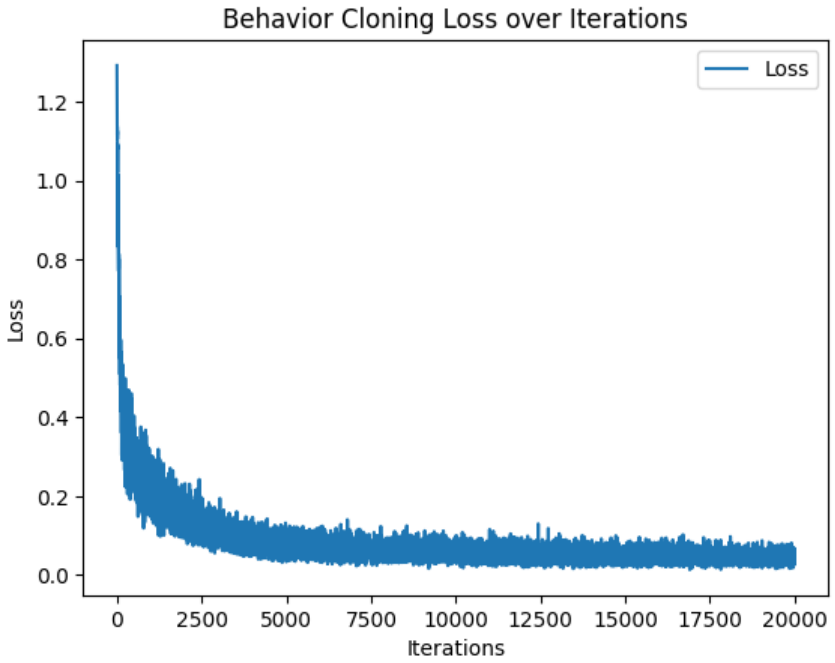


Problem 2: Imitation Learning (62 pts)

Problem 2.1: BC (14 pts)

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

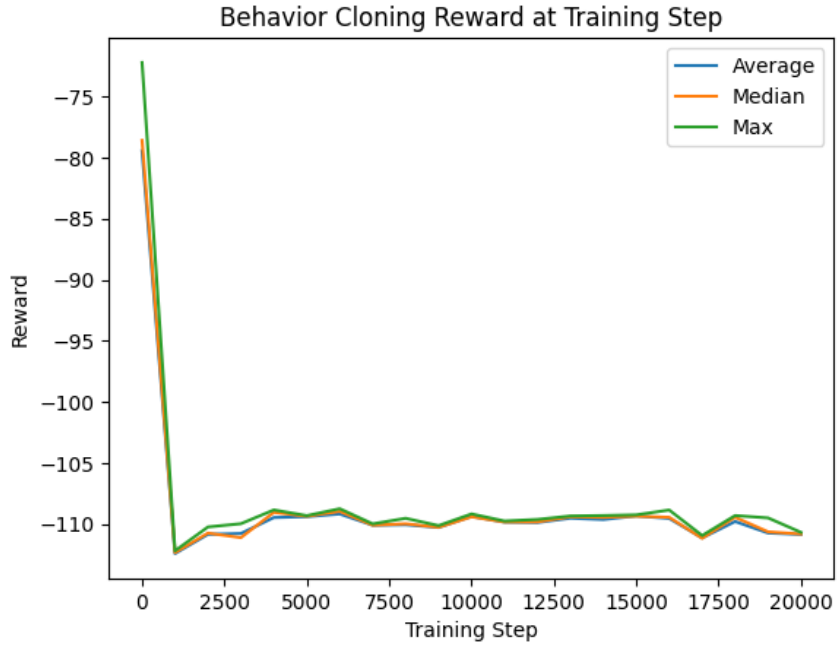
Solution



Final Loss: 0.0439

2.1.2 Rewards Plot BC (6 pts)

Solution



2.1.3 GIF link BC (2 pts)

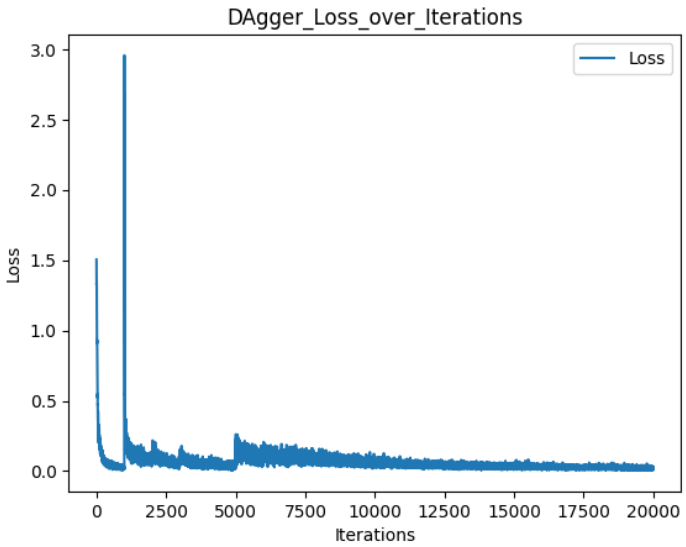
Solution

Link to GIFs

Problem 2.2: DAgger (18 pts)

2.2.1 Loss Plot DAgger (6 pts)

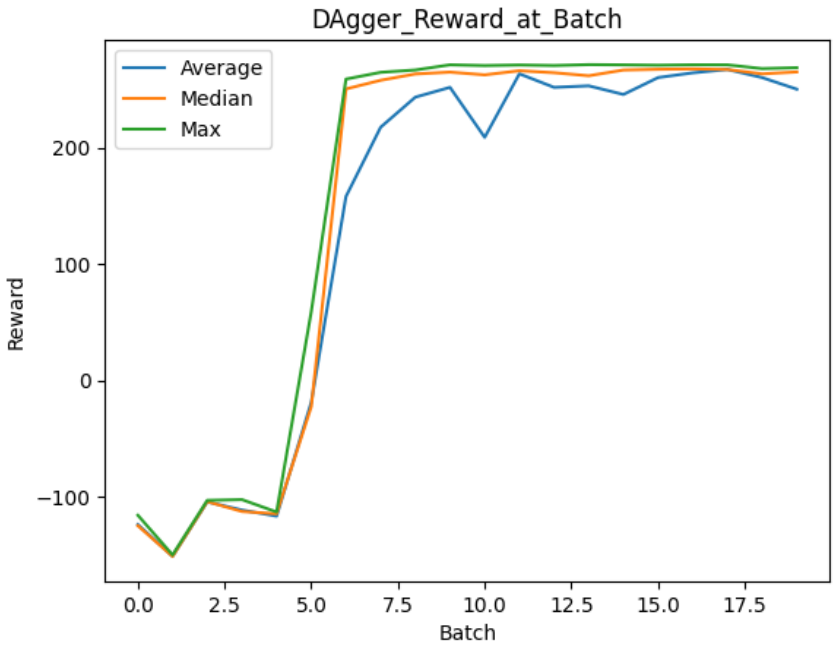
Solution



Final Loss: 0.013279941

2.2.2 Rewards Plot DAgger (6 pts)

Solution



2.2.3 GIF link DAgger (2 pts)

Solution

Link to GIFs

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

Solution

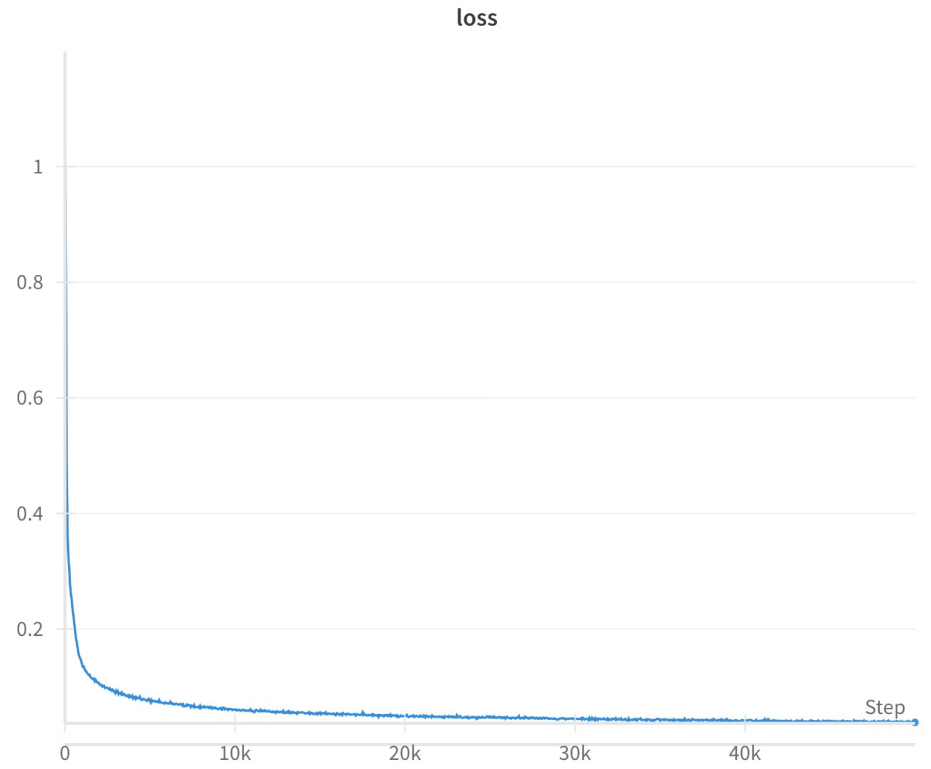
From the data above, DAgger performed significantly better than BC for the following reasons:

- *Error Recovery and Robustness*: DAgger enables the agent to recover from errors by including corrective actions from the expert during its own execution. On the other hand, BC suffers from error accumulation - errors made early in the trajectory can cause cascading failures and lead to poor overall performance
- *Continuous Policy Improvement*: DAgger continuously adapts and improves by learning from its own mistakes & updating the policy with fresh data from the environment
- *Improved Exploration and Generalization*: DAgger allows the agent to experience a broader range of states during training, supporting generalization of unseen scenarios. Whereas BC suffers from an inability to handle outside distribution states, leading to poor performance in novel situations (i.e not encountered during training)

Problem 2.3: Diffusion Policy (30 pts)

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

Solution



Final Loss: 0.0366118

2.3.2 Rewards Diffusion Policy (15 pts)

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

avg trajectory time: 73.278 mean: 200.968 median: 259.834 max: 269.736

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

avg trajectory time: 74.031 mean: 246.873 median: 262.546 max: 266.632

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

avg trajectory time: 286.587 mean: 221.624 median: 260.811 max: 267.795

2.3.3 GIF link Diffusion Policy (2 pts)

Solution

Link to GIFs

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

Solution

- **Multiple Steps**: Diffusion policies generate actions through many denoising steps (often 50+), while DAgger or BC models produce actions in a single step
- **Model Complexity**: Diffusion models are larger and more complex, taking longer to compute each step
- **Sequential Sampling**: Each step in a diffusion model depends on the previous one, which makes it slower and harder to parallelize
- **Stochastic Process**: Diffusion models sample from probability distributions at each step, adding extra computation, unlike the direct mapping used in DAgger or BC

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

Solution

As the number of actions in a row runtime increases, we expect the increase of noise to result in a reduced average reward but with the benefit of a shorter computation-time. This largely holds true with some deviation. The results from our data collection show that the 2 action evaluation performed best with a negligible increase to time for computation. Comparatively, the 1 action evaluation took a lot longer, roughly 3x longer to be precise. This kind of makes sense since it is using the model roughly 3x more times than the 3-action evaluation.

Problem 4: Conceptual Questions (4 pts)

Solution

1. The goal relabeling trick improves data efficiency by creating additional training data from expert trajectories. It relabels intermediate states as new goals, helping the policy learn to reach various goals, not just those directly demonstrated, leading to better generalization, but more importantly, faster learning.
2. In a maze navigation task, a policy without goal relabeling might fail to reach new locations not covered by expert demonstrations. With relabeling, intermediate states along the trajectory become goals, teaching the policy to navigate to a broader set of locations, enabling it to succeed at reaching more diverse goals.