Advanced Machine Learning
Chennai Mathematical Institute
December 2024

Assignment 03
Deep Reinforcement Learning for OpenAI Gym

Team Members

Name	Roll Number
Chandranath Bhattacharya	MDS202318
Salokya Deb	MDS202341
Soumyajoy Kundu	MDS202349

1 Acrobot

Figure 1: Acrobot

1.1 Description of Environment

The Acrobot environment consists of a two-link structure, with the joint between the two links actuated. Initially, both links hang downwards.

1.2 Goal

To apply torques on the actuated joint to swing the free end of the outer-link above a line that is a certain height above the base.

1.3 Observation Space

The observation provides information about the two joint angles and their angular velocities.
Where:

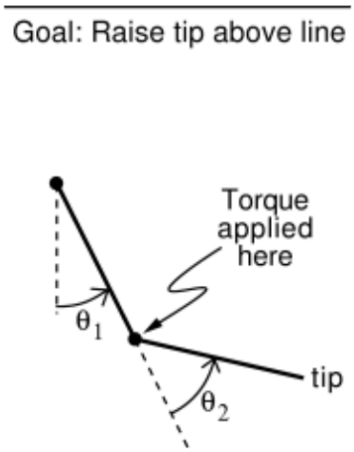

Figure 2: Components

Observation	Min	Max
Cosine of θ_1	-1	1
Sine of θ_1	-1	1
Cosine of θ_2	-1	1
Sine of θ_2	-1	1
Angular velocity of θ_1	~ -12.567 (-4π)	~ 12.567 (4π)
Angular velocity of θ_2	~ -28.274 (-9π)	~ 28.274 (9π)

Table 1: Observation Space for the Acrobot Environment

- θ_1 is the angle of the first joint, where an angle of 0 indicates the first link is pointing directly downwards.
- θ_2 is relative to the angle of the first link. An angle of 0 corresponds to having the same angle between the two links.

This information is from the official Gym documentation. https://gymnasium.farama.org/environments/

1.4 Action Space

The action is the torque applied to the joint connecting the two links.

Action	Result	Unit
0	Apply -1 torque	Torque (N m)
1	Apply 0 torque	Torque (N m)
2	Apply 1 torque	Torque (N m)

Table 3: Action Space for the Acrobot Environment

1.5 Rewards

- All steps that do not reach the goal incur a reward of -1.
- Reaching the goal results in termination with a reward of 0.

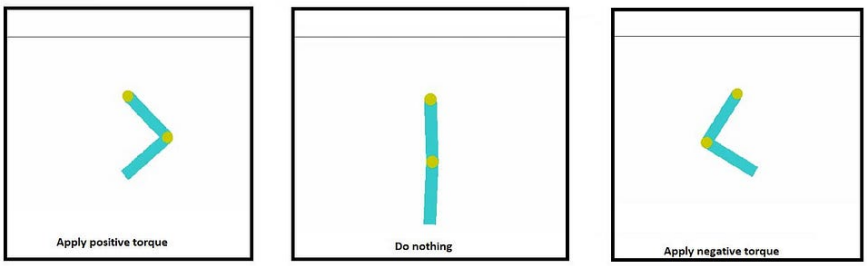

Figure 3: Actions

- The reward threshold is -100.

1.6 Starting State

Each parameter in the state (θ_1, θ_2, and the two angular velocities) is initialized uniformly between -0.1 and 0.1. This implies that both links are pointing downwards with some initial stochasticity.

1.7 Episode Termination

The episode ends if:

1. The free end reaches the target height, which is defined by:

$$-\cos(\theta_1) - \cos(\theta_2 + \theta_1) > 1.0$$

2. The episode length is greater than 500 steps.

2 Dataset – Environment

The Acrobot environment is simulated using OpenAI's `gym` library. The environment provides:

- A 6-dimensional state space representing angular velocities and joint angles.
- A discrete action space of size 3, where the agent can torque the pendulum in one of three directions: positive, negative, or no torque.

No explicit dataset is used; instead, the agent interacts with the environment in real-time.

3 Model Architecture

The task employs a Deep Q-Network (DQN) to solve the control problem. The architecture consists of:

- **Input layer:** A 6-dimensional state vector.
- **Hidden layers:** Two fully connected layers, each with 128 ReLU-activated neurons.
- **Output layer:** A 3-dimensional vector representing the Q-values for each action.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1               [-1, 1, 64]             448
            Linear-2               [-1, 1, 64]           4,160
            Linear-3                [-1, 1, 3]             195
================================================================
Total params: 4,803
Trainable params: 4,803
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.00
Params size (MB): 0.02
Estimated Total Size (MB): 0.02
----------------------------------------------------------------
```

Figure 4: Model Summary

4 Training Results

4.1 Hyperparameters

Hyperparameter	Value	Description
`n_episodes`	2000	Total number of training episodes. Each episode is a complete interaction sequence.
`learning_rate`	0.001	Step size for updating Q-values. Controls the speed and stability of learning.
`gamma`	0.99	Discount factor, balances immediate and future rewards.
`eps_start`	1.0	Initial epsilon value for the epsilon-greedy policy, representing full exploration.
`epsilon_decay`	0.995	Rate of decay for epsilon after each episode. Gradually shifts from exploration to exploitation.
`eps_end`	0.01	Minimum epsilon value, ensuring a small level of exploration even after long training periods.
`batch_size`	64	Number of experiences sampled from the replay buffer during each training step. Larger batches stabilize learning but increase computational cost.
`memory_size`	10000	Size of the replay buffer, which stores past experiences (`state, action, reward, next_state, done`). Larger memory allows learning from a wider range of past experiences.

Table 4: Summary of hyperparameters used in the DQN Agent Training.

4.2 Plotting the training process

The DQN model was trained over 2000 episodes. Key observations include:

- The agent learned to achieve the target height with an average score of -98.29 per episode by the end of training.

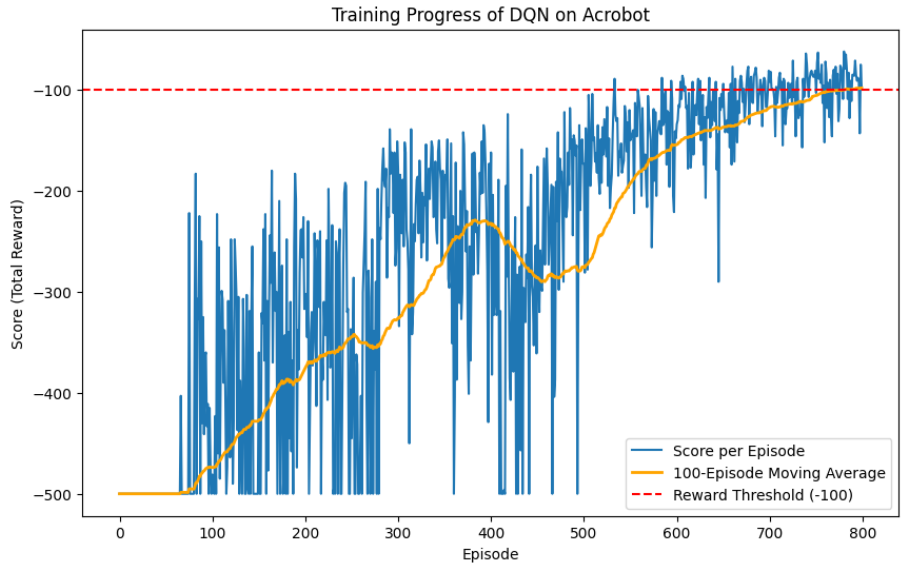

Figure 5: Training Plot

- The reward increased consistently as training progressed, indicating improved policy performance.
- Learning was gradual, with steady progress over 600 - 800 episodes.
- The performance plateaued after the threshold, suggesting the agent's learning stabilized and further improvements became minimal.

4.3 Game played with Trained Agent

4.3.1 Episode 1

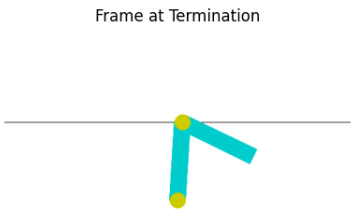

Figure 6: **Total Reward = -87**

4.3.2 Episode 2

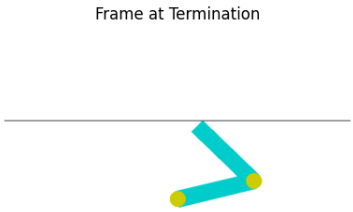

Figure 7: **Total Reward = -72**

4.3.3 Episode 3

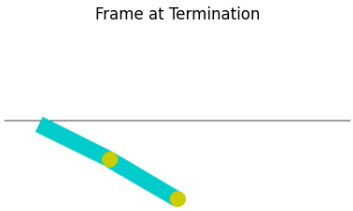

Figure 8: **Total Reward = -96**

5 Comments

- The DQN agent successfully solved the Acrobot environment by achieving rewards better than the threshold of -100 (e.g., reward of $>$ 100 in the above games).
- The 100-episode moving average showed gradual improvement, stabilizing around episode 600 - 800, indicating efficient learning.
- The trained agent consistently reaches termination with rewards above the threshold, demonstrating robustness and reliability.
- The final frame of the games played by trained agent shows the Acrobot reaching its target state with the second link above the line, validating the agent's success in solving the environment.