AML Assignment 3

Aman, Ankan Kar, Utpalraj Kemprai

December 2024

1 Introduction

In this assignment, we explore the application of Deep Q-Networks (DQN) to the Acrobot game. DQN is a reinforcement learning algorithm that uses a neural network to approximate the Q-values for each action in a given state. We evaluate the performance of a trained DQN agent compared to a random agent.

2 Deep Q-Network (DQN) Architecture

The Deep Q-Network (DQN) is a fully connected neural network designed to approximate Q values for reinforcement learning tasks such as Acrobot. Below is a concise description of its architecture.

2.1 Architecture

- **Input Layer:** 6-dimensional state vector (Acrobot state space).
- **Hidden Layers:** Two layers, each with 128 neurons and ReLU activation.
- **Output Layer:** 3 Q-values corresponding to actions (left torque, neutral torque, right torque).

2.2 Forward Pass

The forward pass processes the input state through the network as follows:

$$\begin{aligned} x_1 &= \text{ReLU}(\text{Layer1}(s)) \\ x_2 &= \text{ReLU}(\text{Layer2}(x_1)) \\ Q(s) &= \text{Layer3}(x_2) \end{aligned}$$

Here, s represents the input state, and $Q(s)$ represents the Q-values for all possible actions.

3 Training process for DQN

3.1 Initialization

Hyperparameter	**Value**
Batch Size	128
Discount Factor (γ)	0.99
Initial Exploration Rate (ϵ_{start})	0.9
Minimum Exploration Rate (ϵ_{end})	0.05
Exploration Rate Decay (ϵ_{decay})	1000
Learning Rate (LR)	1×10^{-4}
Replay Memory Capacity	10,000 transitions
Soft Update Rate (τ)	0.005

Table 1: Hyperparameters for DQN

- Initialize the policy and target networks.
- Initialize replay memory.

3.2 Training Loop

For each episode (total number of episodes for training : 500)

1. **Reset Environment:** Obtain the initial state.
2. **Action Selection:** Use an ϵ-greedy policy to choose an action.
3. **Interaction:** Perform the selected action, observe the next state, reward, and termination signal.
4. **Store Transition:** Save $(state, action, reward, next\_state)$ in replay memory.
5. **Optimization:**
   - If the replay memory contains at least Batch Size many transitions:
     (a) Sample a minibatch of transitions.
     (b) Compute Q-values: $Q(s, a)$ using the policy network.
     (c) Compute target Q-values: $r + \gamma \max_{a'} Q(s', a')$ using the target network.
     (d) Calculate the Huber loss and update the policy network using backpropagation.
6. **Target Network Update:** Perform a soft update of target network weights:

$$\theta' \leftarrow \tau\theta + (1 - \tau)\theta'$$

4 Model Performance

Agent	Avg. Reward	Median	Std. Dev.	Max	Min
Trained Agent	-80.83	-77.0	14.10	-68.0	-133.0
Random Agent	-499.00	-500.0	6.75	-448.0	-500.0

Table 2: Performance Comparison Between Trained Agent and Random Agent for Acrobot game calculated after playing 100 games with each of them

4.1 Observations

The Random Agent performs poorly across all metrics, which is expected since it selects actions randomly without any learning mechanism. This highlights the importance of training in reinforcement learning tasks.

5 Conclusion

The trained DQN agent outperforms the random agent across all metrics, demonstrating the effectiveness of training in reinforcement learning tasks.