Reinforcement Learning in Unity: A Deep Dive into Observations and Decision Frequencies

Abstract: This paper delves into the fundamental components of reinforcement learning using the Unity ML-Agents toolkit, particularly focusing on observation collection and decision-making frequencies. Through a detailed analysis, I aim to provide insights that can guide the effective training and operation of agents in a simulated environment.

1. Introduction

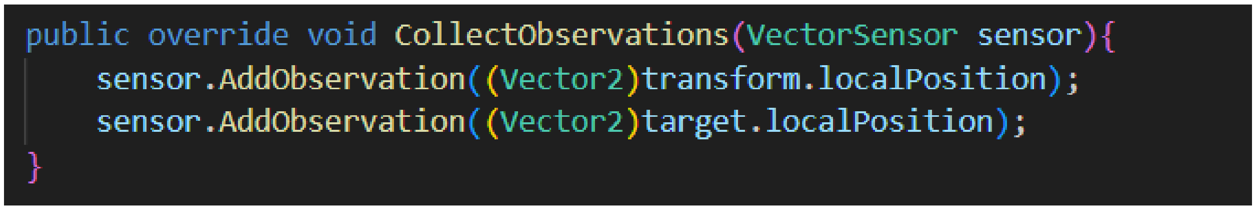
Reinforcement Learning (RL) in Unity, facilitated by the ML-Agents toolkit, enables the creation and training of agents capable of interacting with and learning from their environment. Crucial to this process are the agent's observations of its surroundings and the frequency with which it makes decisions. Both aspects profoundly impact the agent's learning efficacy and operational performance.

2. Observations in RL
2.1 The Role of Observations

In RL, an agent's understanding of its environment is governed by observations. These are specific pieces of information the agent receives about its surroundings, enabling it to make informed decisions.

2.2 Collecting Observations in Unity

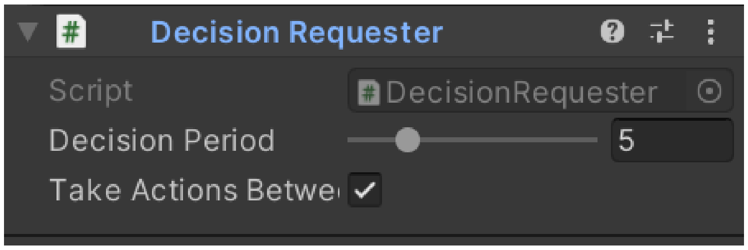
Within the scope of this project, the CollectObservations method facilitates this. This method, called once per step, allows the agent to gather current state data about its own position as well as the target's position. In other projects this may be expanded to include positions, velocities, or any relevant scalar and vector quantities. The composition and nature of these observations can significantly influence an agent's learning trajectory.



```
public override void CollectObservations(VectorSensor sensor){
    sensor.AddObservation((Vector2)transform.localPosition);
    sensor.AddObservation((Vector2)target.localPosition);
}
```

3. Decision-Making in Unity: The Decision Requester
3.1 Overview

The Decision Requester is a pivotal component in Unity's ML-Agents toolkit. It controls how often an agent decides on an action, thus regulating the agent's interaction frequency with its environment.


Decision Requester

Script: DecisionRequester

Decision Period: 5

Take Actions Betwe ✓

Above, you will notice that our decision period is 5, meaning that the agent will make a decision every 5 steps. For the steps in between, the agent will repeat its last action. For example, if the agent decides to move forward at step 0, it will keep moving forward for steps 1, 2, 3, and 4. Then, at step 5, it will make a new decision

3.2 Key Components

Decision Period: Dictates the number of simulation steps between consecutive decisions. A higher period can improve performance but may reduce the agent's responsiveness to rapid environmental changes.

Take Actions Between Decisions: A toggle that determines if an agent should continuously enact its most recent decision or remain passive between decision periods.

4. Implications and Best Practices

4.1 Balancing Performance and Responsiveness

A higher decision period can accelerate training due to reduced computational overhead. However, this might come at the cost of the agent's ability to promptly react to its environment.

4.2 Relevance of Observations

To enhance the agent's learning efficiency, only pertinent information should be included as observations. Extraneous data can slow down or even misdirect the learning process.

5. Conclusion

The success of reinforcement learning in Unity is interwoven with the agent's observational data and its decision-making frequency. While the CollectObservations method offers a window into the environment, the Decision Requester modulates the agent's interactions with it. Striking a balance between computational efficiency and optimal learning is key, necessitating a thorough understanding and apt utilization of these components.

Keywords: Unity, ML-Agents, Reinforcement Learning, Observations, Decision Requester