Final report

AE4350: Bio-inspired Intelligence and Learning for Aerospace Applications

Zach Kelly (5901405)

Delft University of Technology

TU Delft

Contents

1 | Introduction

The inverted cart pendulum system serves as a classic benchmark problem for studying the control of dynamical systems. The system consists of a cart connected to pole along a track, as seen in Figure 2.1. This system proposes a challenging control problem as it has inherently non-linear, unstable and under-actuated dynamics.

The objective in this report, is to determine the optimal action selection policy using a Q-learning algorithm to maintain upright balance of the pole by forcing the cart to the left or right. The environment developed represents the continuous state space as discrete and is inspired by Gym's cart pole environment [1]. The optimal action selection policy is obtain through careful tuning of the Q-learning parameters described in Table 3.1.

Link to repository: inverted cart pendulum RL

2 | Problem Formulation

2.1. Model

The following model is derived from [2]:

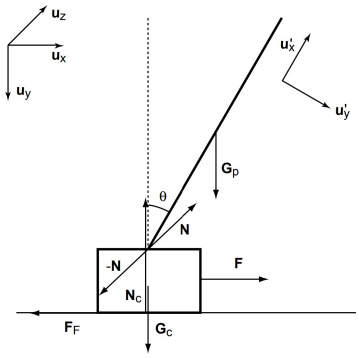

Figure 2.1: Inverted cart-pendulum model diagram [2]

Using Figure 2.1, the following system of differential equations is derived to represent the dynamics of the inverted cart pendulum:

$$\begin{aligned} \ddot{\theta} &= \frac{g(\sin\theta) + \cos\theta(\frac{-F - m_p l\dot{\theta}^2}{m_p + m_c})}{l(\frac{4}{3} - \frac{m_p \cos\theta^2}{m_c m_p})} \\ \ddot{x} &= \frac{F + m_p l(\dot{\theta}^2 \sin\theta - \ddot{\theta}\cos\theta)}{m_c + m_p} \end{aligned} \tag{2.1}$$

where:
m_c [kg] - the mass of the cart
m_p [kg] - the mass of the pole
l [m] - the half length of the pole
g [m/s^2] - the gravitational constant
θ [rads] - the angular displacement of the pole w.r.t the vertical
x [m] - the horizontal displacement of the cart
F [N] - the horizontal force applied to the cart

Equation 2.1 assumes the system is constrained to the x and y directions in Figure 2.1, the cart and pole have uniform density, and the cart moves along a frictionless track.

The following model parameters are sourced from [3]:

Table 2.1: Inverted cart pendulum model parameters

Parameter	Value
m_c	1 kg
m_p	0.1 kg
l	0.5 m
g	9.81 m/s^2
F	$\pm 10N$
x	$\pm 2.4m$
θ	$\pm 12°$
$\dot{x}$	$\pm\infty m/s$
$\dot{\theta}$	$\pm\infty°/s$

After solving for $\ddot{x}$ and $\ddot{\theta}$ in Equation 2.1 by plugging in the values in Table 2.1. Euler's method is used to numerically approximate the solution of the next time step for all states.

$$\begin{aligned} x_{n+1} &= x_n + \tau \dot{x_n} \\ \dot{x_{n+1}} &= \dot{x_n} + \tau \ddot{x_n} \\ \theta_{n+1} &= \theta_n + \tau \dot{\theta_n} \\ \dot{\theta_{n+1}} &= \dot{\theta_n} + \tau \ddot{\theta_n} \end{aligned} \tag{2.2}$$

Equation 2.2 assumes the current state is known and uses a time step τ of 0.02 seconds.

2.2. Environment

The environment simulates the model described in Section 2.1 in discrete space. The discrete space is composed of the action and observation space. The action space consists of 2 discrete actions: move the cart to the left or to the right. These actions control the forces acting on the cart and the balance of the pole. The observation space consists of 4 states: x the position of the cart, θ the angular displacement of the pole, $\dot{x}$ the velocity of the cart, and $\dot{\theta}$ the angular velocity of the pole. These states define the current configuration of the system.

The environment parameters are sourced from [1]:

Table 2.2: Environment size parameters

Parameter	Value
action	0,1
x	$\pm 4.8m$
θ	$\pm 24°$
$\dot{x}$	$\pm 4m/s$
$\dot{\theta}$	$\pm 4°/s$
bins	30

The value of the bins parameter in Table 2.2 determines the number of discrete points equally spaced along each dimension of the observation space. To map the state obtained numerically

from Equation 2.1 to discrete space, the state variables are approximated to the discrete point of closest value. The parameter values are selected to balance approximation error and computation time. (i.e. increasing granularity will decrease approximation error and increase computation time)

The environment resets the state variables at the start of each episode by initializing randomly with a Gaussian normal distribution centered around zero and a small standard deviation. This ensures the cart starts approximately centered and pole upright with approximately zero velocity.

The environment renders graphical display using the Pygame library. The visual display aids the user in understanding the dynamics of the problem and how the agents actions impact the cart and pole's behavior. Figure 2.2 illustrate the environment at the initial state.

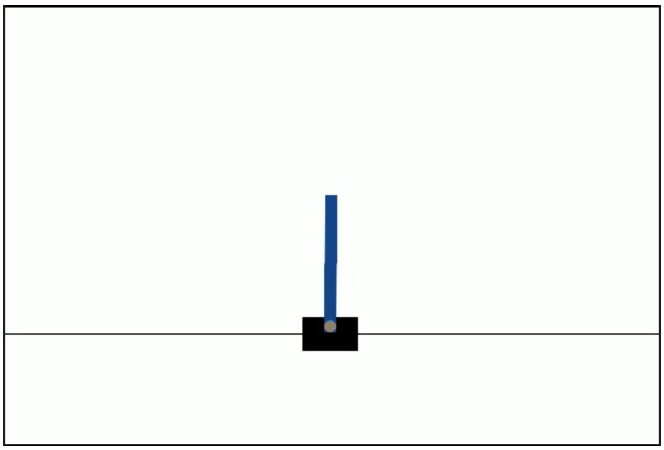

Figure 2.2: Inverted cart-pendulum environment [1]

The frames are rendered at a rate of 50 frames per second to synchronize with the solution time step τ of Equation 2.2.

3 | Q-Learning

3.1. Algorithm

The Q-learning algorithm is a model-free reinforcement learning method that aims to determine the optimal action selection policy for an agent to interact in an environment. This policy can be obtained through a process known as Value Iteration. Value Iteration involves updating the value functions, also known as q-values, in a q-table. The q-table contains a q-value for all state-action pairs that exist in the environment. The Q-learning method implemented to obtain the optimal policy is the following:

1. Initialization: initialize the q-values using a random Gaussian normal distribution centered around 0 with a small standard deviation.
2. Determine Action: at each time step, the agent chooses an action based on the exploration factor policy. If the value of epsilon is greater than the randomly generated threshold the agent exploits its knowledge and chooses the action paired with the largest q-value. Otherwise, the agent chooses a random action.
3. Update State: the next state is computed using the chosen action. Described in Section 2.1.
4. Compute Rewards: the immediate reward is calculated based on the updated state and is the sum of the intermediate angle, intermediate position and time-balanced rewards.
5. Update Q-Value: The q-value of the current state-action pair is computed using the Bellman's equation. The following form of the Bellman's optimality equation is retrieved from [4].

$$Q(s,a) = Q(s,a) + \alpha(R + \gamma \max Q(s',a') - Q(s,a)) \tag{3.1}$$

where:
$Q(s,a)$ - the q-value of the current state-action pair
α - the learning rate: controls the extent that q-values are updated based on new experiences
R - the immediate reward
γ - the discount rate: controls the extent that the agent favors future rewards rather than immediate rewards
$\max Q(s',a')$ - the maximum q-value that can be obtained in the next state given all possible actions

6. Repeat Until Convergence: steps 2 to 5 are repeated for multiple time steps and episodes until the q-values converge to an optimal. After convergence, the optimal policy for each state is the action associated with the largest q-value.

3.2. Parameters

The Q-learning parameters are summarized in the Table 3.1.

Table 3.1: Q-learning parameters

Parameter	Value
$episodes$	1250
$\max steps$	1000
γ	0.95
α_{final}	0.15
α_{intial}	1
ϵ_{final}	0.1
$\epsilon_{initial}$	1
$\max R_{pole}$	10
$\min R_{pole}$	0
$\max R_{cart}$	5
$\min R_{cart}$	0
$\max R_{time}$	2
$\min R_{time}$	0

In Table 3.1, the intermediate angle variable R_{pole} rewards the agent based on the angular displacement between the vertical and the pole. The reward value is a linear interpolation between the minimum and maximum reward for input angles between $0°$ and $12°$. Similarly, the intermediate position variable R_{cart} rewards the agent based on the displacement of the cart from the center position. The reward value is a linear interpolation between the minimum and maximum reward for input positions between $0m$ and $2.4m$. The time-balanced variable R_{time} rewards the agent based on the current number of steps occurred in the episode. The reward value is a linear interpolation between the minimum reward and the reward factor times the maximum number of steps. The learning rate variable α and exploration variable ϵ, described under Equation 3.1, are shown in Figure 3.1.

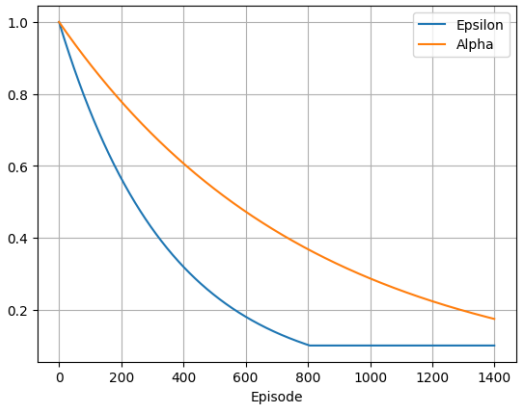

Figure 3.1: Exponential decay of learning rate variable α and exploration factor variable ϵ

The Q-learning algorithm's computation time is limited by the $termination$ and $truncation$ variables. The $termination$ variable ends the current episode if any state variables exceed the range defined in Table 2.1, while the truncation variable ends the current episode if the agent exceeds the maximum number of time steps defined in Table 3.1. The pole is considered balanced once the agents has occurred 500 steps (10 seconds) in a single episode.

4 | Results and Discussion

4.1. Q-Learning Results

The following results are generated to assess the learning performance of the algorithm using the parameters in Table 3.1.

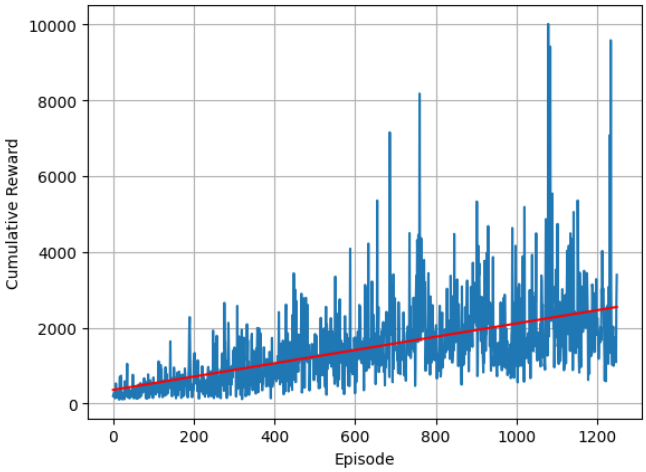

Figure 4.1: Cumulative reward vs episode

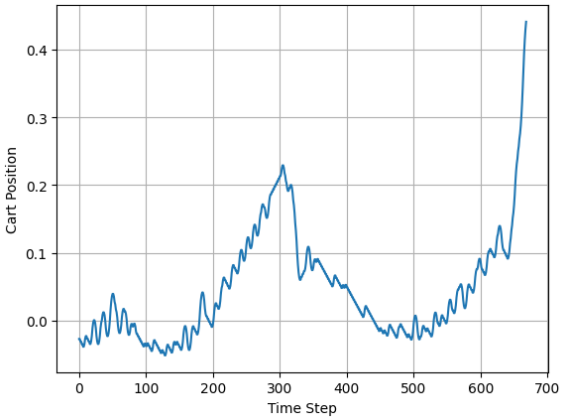

Figure 4.2: Cart position vs steps taken in the environment

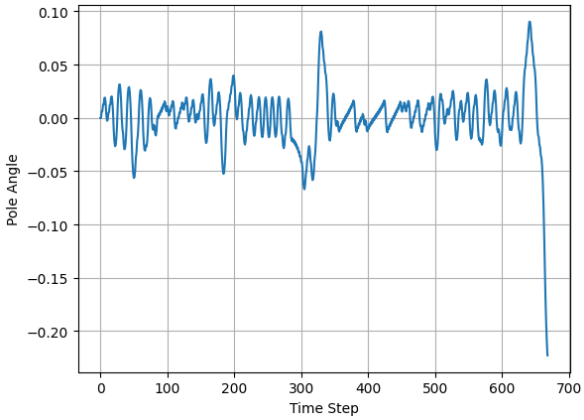

Figure 4.3: Pole angle vs steps taken in the environment

Table 4.1: Q-learning results for varying episode lengths

Episodes	Max Steps	Time Balanced (s)	Max Reward	Computation Time (s)
500	367	7.34	5279	144
1000	494	9.88	7336	333.3
2000	656	13.12	9588	726.1
5000	1000	20	15530	2773

4.2. Sensitivity of Hyperparameters

4.2.1. Reward Shaping

The linear reward variables R_{pole}, R_{cart} and R_{time} from Table 3.1 provide a continuous feedback signal to the agent. Given this signal the agent can generalize its learning over multiple steps towards the target states. In addition, the relative magnitudes of the maximum reward values allows the agent to learn the relative importance of each reward target state. For example, the agent learns to assign more importance to the target state of balancing the pole at $0°$ than the target state of positioning the cart at 0 meters. This is shown in Figure 4.2 and Figure 4.3 the agents has learned to optimize the pole's target state over the cart's, hence the larger deviations from the target state in Figure 4.2.

4.2.2. Exploration Factor

Increasing the value of the exploration factor ϵ increases the agents exploration via random actions, while decreasing ϵ increases the exploitation of the learned policy. Therefore, small values of ϵ may lead to convergence of a sub optimal policy. This is shown in Figure 4.1, the cumulative rewards begins converging to a constant value at approximately the 800th episode where the epsilon value reaches its minimum (see Figure 3.1). The learned policy is sub optimal, because the cumulative reward continues increasing if the minimum value of epsilon is delayed or the number of episodes increases. Figure 4.4 shows that the cumulative reward continues increasing after the 800th episode when epsilon is delayed (see Figure 4.5). To have the optimal policy the agent must maximize cumulative reward over time.

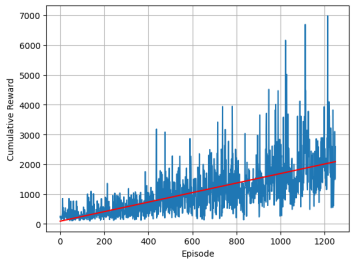

Figure 4.4: Cumulative reward vs episode

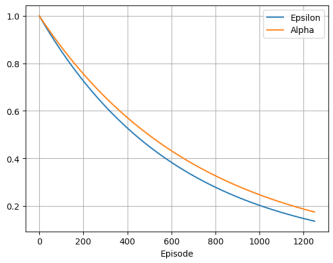

Figure 4.5: Exponential decay of learning rate variable and exploration factor variable

4.2.3. Learning Rate

Increasing the value of the learning rate α results in q-values being updated more frequently based on new information. Similar to Section 4.2.2, decreasing α increases the exploitation of the learned policy. Therefore, small values of α lead to slow convergence.

4.2.4. Discount Factor

Opposite to Section 4.2.2, increasing the value of discount factor γ shifts the importance from immediate rewards to future rewards and increases policy exploitation. Therefore, larger values of γ lead to slow convergence.

5 | Conclusion

The Q-learning agent learned to repeatably balance the inverted cart pendulum for longer than 10 seconds, when ran for 1250 episodes using a discount factor of 0.95 and the learning rate and exploration factor shown in Figure 3.1. The values of the exploration factor, learning rate and discount factor significantly effect the exploitation of the learned policy and the convergence rate towards the optimal policy. All trials in Table 4.1 indicate that the optimal policy is not obtained, hence the maximum cumulative rewards over time improve with more episodes. To obtain the optimal action selection policy, more optimal values of the hyperparameters should be found.

In the future, the hyperparameter values may be optimized using a search method such as grid search, random search or Bayesian optimization. An experience relay buffer may be implemented with the Q-learning algorithm to improve the stability and convergence of the learning process by updating q-values over a set of previous experiences rather than using the consecutive experience. Otherwise, implementation of a Deep Q-Network (DQN) may be advantageous. DQN's are capable of handling the inverted cart pendulum's continuous state space without discretization and more complex policies such as also learning to apply variable forces to the cart.

Bibliography

[1] Rich Sutton et al. *Classic Cart-Pole System*. 2022. URL: https://github.com/openai/gym/blob/master/gym/envs/classic_control/cartpole.py (visited on 08/31/2023).

[2] Razvan V. Florian. *Correct equations for the dynamics of the cart-pole system*. 2007. URL: https://coneural.org/florian/papers/05_cart_pole.pdf/ (visited on 08/31/2023).

[3] Andrew G. Barto, Richard S. Sutton and Charles W. Anderson. *Neuronlike adaptive elements that can solve difficult learning control problems*. 1983. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6313077 (visited on 08/31/2023).

[4] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction (second edition)*. 2020. URL: http://incompleteideas.net/book/RLbook2020.pdf (visited on 08/31/2023).