Control-Tutored Q-Learning

Report submitted by

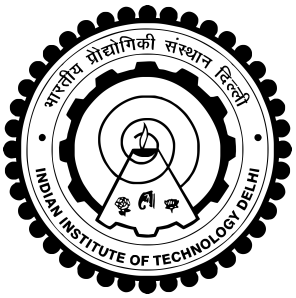
Sarthak Gupta, Akshat Gadhwal
2019EE30296, 2019EE30551

under the supervision of

Prof. Indra Narayan Kar

for End-term report of
ELD-431 B.tech. Project-I

Bachelor of Technology



Department Of Electrical Engineering
INDIAN INSTITUTE OF TECHNOLOGY DELHI

November 2022

Contents

ABSTRACT

The use of Machine learning algorithms is very useful when we do not have the complete model of the system. Q-Learning is a type of reinforcement learning which is a model-free approach to training the agent. In some control problems were designing a complete model of the system is either very complex or expensive, then such model-free approaches are used. In this paper, we implemented Control Tutored Q-Learning, which combines the partial model with the classical model-free Q-Learning approach and derived a hybrid Q-Learning approach (CTQL). We used the benchmark problem of an Inverted Pendulum on a moving cart to compare the result of classical Q-Learning and Control Tutored Q-Learning. We also discussed a simpler problem of Amazon Warehouse and solved it using CTQL. And designed a controller for the Inverted Pendulum with a Moving Cart system. We linearized the model and calculated the linear controller. Then we used this controller to simulate the balancing of non-linear system.

Chapter 1

Introduction

1.1 Motivation

Reinforcement-learning is a great approach for solving a control problem when system dynamics are not known, or the system is very complex to design a complete model. Or even when designing the system is expensive, in those cases, model-free Reinforcement-learning is helpful.

Q-Learning is a tabular form of Reinforcement learning which stores the learnt policy in form of Q-table. It can be used when the state and action space are discrete. But this algorithm has some negative aspects as well if we use it in control applications. Lengthy training periods are frequently unacceptable in control applications, and mistakes made while learning could result in dangerous circumstances.

In order to get around these restrictions,a mathematical model of the plant can be used to partially address some of the issues raised above. For this Control Tutored Q-Learning is developed which uses the mathematical model of plant to address these issues as in [2], [3] to control fixed inverted pendulum.

1.2 Problem Statement

We want to compare the performance of conventional Q-Learning and Control Tutored Q-Learning in stabilizing the Cart-pole. The Cart-pole problem has an inverted pendulum attached to a moving cart. This problem is suitable for our purpose for two reasons:

1. The upward position of the pendulum is unstable, and the system dynamics are non-linear. This problem is used for testing new control strategies
2. We used a linear controller in the Control-Tutored Q-Learning, which is not a complete model as the linear controller is not designed for large deviation from equilibrium condition and it does not account for the non-idealities of the system

1.3 Cart-pole System

Take into account the inverted-pendulum control scheme shown in Figure below. In this illustration, the only motions that matter are the pendulum's and the cart's motion in the

plane of the page.As far as feasible, the inverted pendulum should remain upright while the cart's location is under control.

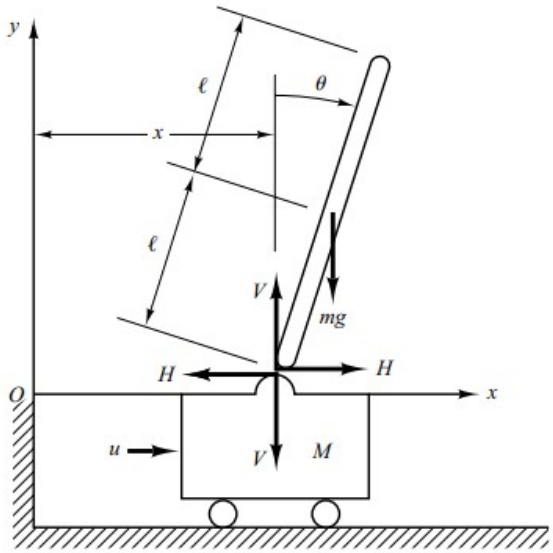


Figure 1.1: Diagram of Inverted Pendulum on cart

The system parameters are as follows:

- Mass of Cart $M = 2$ kg
- Mass of Pole $m = 0.1$ kg
- Length of Pole = 1 m
- Distance of centre of mass of Pole $l = 0.5$ m

If we apply control force u as shown in the figure above the Dynamics of system are governed by following equation:

$$\ddot{x} = \frac{u + ml\ (sin\theta)\ \dot{\theta}^2 - mg\ cos\theta\ sin\theta}{M + m - m\ cos^2\theta}$$

$$\ddot{\theta} = \frac{u\ cos\theta - (M+m)g\ sin\theta + ml(cos\theta\ sin\theta)\ \dot{\theta}^2}{ml\ cos^2\theta - (M+m)l}$$

1.3.1 Controller Design

We used pole placement method to design control as shown in [4].We can linearize above system with respect to angular position at 0 radian. Near the equilibrium point i.e. near 0

radian we can approximate $\sin\theta$ to θ and $\cos\theta$ to 1. After doing this we got the following equations:

$$Ml\ddot{\theta} = (M+m)g\theta - u$$

$$M\ddot{x} = u - mg\theta$$

We can produce a linear system that can be expressed as follows in state space representation if we linearize our system around the upper equilibrium, where our states are

$$X = \begin{bmatrix} \theta \\ \dot{\theta} \\ x \\ \dot{x} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

we obtain a linear system that can be written in state space representation as:

$$\dot{X} = \begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \\ \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{M+m}{Ml}g & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{m}{M}g & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{1}{Ml} \\ 0 \\ \frac{1}{M} \end{bmatrix} u$$

To design controller for the system we use pole placement method to the find state feedback gain matrix using **Ackermann's Formula**. To obtain reasonable damping and speed of response we choose following closed-loop poles at:

$$\mu_1 = -1 + j\sqrt{3}, \quad \mu_2 = -1 - j\sqrt{3}, \quad \mu_3 = -5, \quad \mu_4 = -5, \quad \mu_5 = -5$$

After using the Ackermann's Formula we got our state feedback gain matrix which is given by: $K = [-157.6336 - 35.3733 - 56.0652 - 36.7466]$

1.4 Simulating non-linear System using linear Controller

Since we have dynamics of the system in form of non-linear differential equation therefore we don't have any closed form equation to obtain state of the system at given time. Therefore we used the **ODE45** Differential Equation solver in MATLAB to obtain the state of the system at given time. We implemented the Servo-system given in Example 10.5 in [**?**] To verify that our controller is working fine we test it under following circumstances:

- Varying the initial condition of the system
- Introducing $\pm 5\%$ error in the state measurements
- Introducing second disturbance after the initial disturbance got settled

The results of the above are shown in the figure below in which we added ±5% error to state observation after 150 steps and introduce angle disturbance of 10 degrees after the initial disturbance of 24 degree is settled by the controller. From the figures we can see that our controller is working correctly in above situations and it is also immune to the measurement error in the state observations

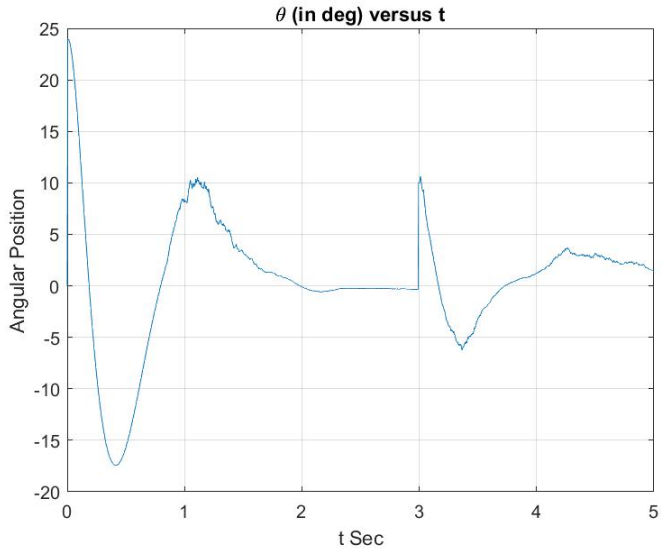


Figure 1.2: Error in angular position decreases with time and it also converges after second disturbance

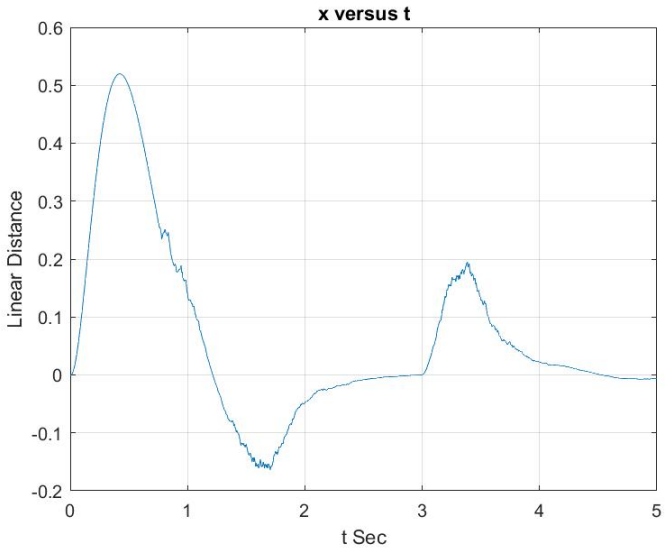


Figure 1.3: Error in Linear position decreases with time and it converges to 0 even after second disturbance

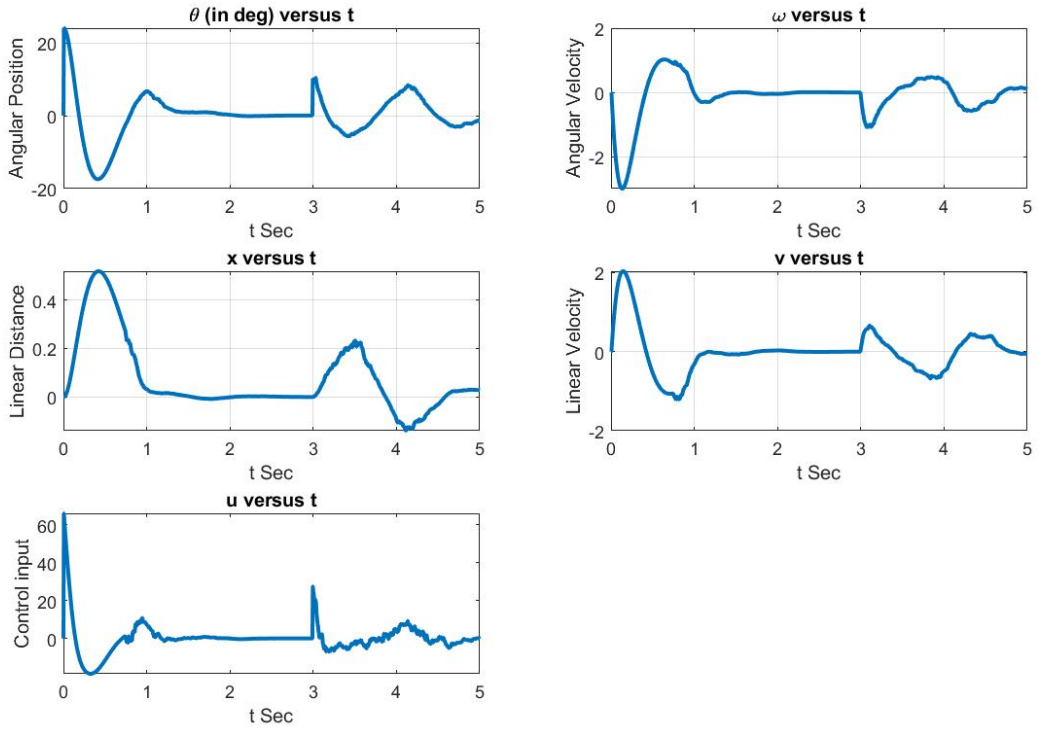


Figure 1.4: Error in all state variable decreases with time and it also converges after second disturbance

1.5 Q-Learning

1.5.1 Q-Learning Introduction

Q-learning is a model free approach to learn the optimal action in each state introduced in [5]. It select the best action based on the expected future reward. To select an action it uses Bellman equation. A model-free reinforcement learning algorithm called Q-learning is used to determine the worth of a given action in a given situation. The term "model-free" refers to the fact that it is not dependent on a model of the environment and that it is capable of handling stochastic transition and reward issues without the need for modifications.

Q-learning identifies an optimal policy for any finite Markov decision process (FMDP) by maximising the expected value of the total reward across any and all subsequent steps, beginning from the current state.

Given limitless exploration time and a somewhat random policy, Q-learning can find the best action-selection strategy for any FMDP. The term "Q" refers to the function that the algorithm computes, or the anticipated benefits of a certain action in a particular state.

1.5.2 Exploration Vs Exploitation

Exploration

In exploration we choose random actions to explore all the possible ways to reach the target. Initially when the the agent do not know the environment, it does the exploration.

Exploitation

In exploitation we use the the existing knowledge to predict the best action and take that action. As the agent spend more and more, the rate of exploitation will increase and rate of exploration will decrease.

In starting rate of exploration is more and in the end of training the rate of exploitation is more.

Policy

- Policy is a function made up of tune-able parameters
- There is a set of parameters that produces the optimal policy
- For Q-learning the final policy is the Q-table after training
- Our final goal is to find the optimal policy that gives us an optimal solution

Q Table and Q-value

- Stores the Q-value i.e.$Q(s_t, a_t)$
- $Q(s_t, a_t)$ of state s' and action 'a' is expected long-term reward obtained by taking action 'a' on state 's' at time 't'

Q-Table Updation

We try to calculate the new Q-value for state s and action taken a while doing exploration and exploitation during training phase. we update the Q-table using Bellman Equation :

$$newQ(s,a) = Q(s,a) + \alpha[R(s,a) + \gamma * maxQ^{'}(s^{'}, a^{'}) - Q(s,a)]$$

where α is learning rate which governs how much the new value should change after updation and γ is discount factor which takes account the fact that future reward is less valuable than current reward

1.5.3 Epsilon-Greedy Policy

The Epsilon-Greedy Policy used in control is given below:

$$\pi^{rl}(x) = \begin{cases} \arg\max_{u \in U} Q_k(x,u), & \text{with probability } 1-\epsilon \\ u \sim Rand(U) & \text{with probability } \epsilon \end{cases}$$

where $\pi^{rl}(x)$ is action which the agent will take in current state x, $\epsilon < 1$ is exploration rate and U is action space

We have a diminishing exploration rate which will make our agent more exploration at initial phase of training and more exploitation in later stages.

1.5.4 Issues with Control Using Classical Q-Learning

- Longer training time
- Probability of system failure due to exploration in unsafe situation

1.6 Control Tutor Q-Learning

1.6.1 CTQL Introduction

The idea here is to use the partially accurate model for the prediction of next action. Even though they might not be realistic enough to allow for a totally model-based solution of the control problem, some equation-based models of the environment are frequently available in many control applications. When using classical RL, these rough or incomplete models of the environment are frequently abandoned in favour of a wholly model-free strategy. In this study, we explore the potential for integrating a feedback control law that is created using a partial environment model to support learning.

1.6.2 Control Tutor

From the state gain matrix obtained in section 1.3.1 we can compute the control input force which is given by $u = -K.x$, where K is state feedback gain matrix and x is current state of system since the state output from the environment is continuous we can compute control input precisely but since we can only apply discrete force value to the system our control tutor will suggest the nearest available action to the calculated control input value

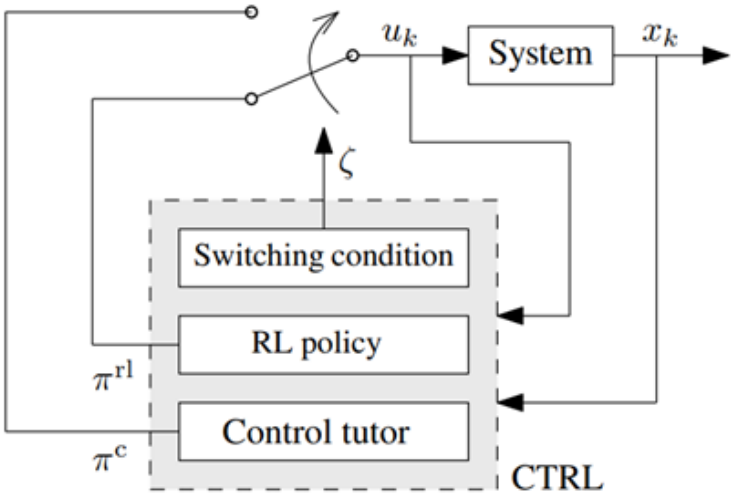


Figure 1.5: CTQL Policy

1.6.3 CTQL Epsilon-Greedy Policy

In CTQL we modify our Epsilon-Greedy Policy to incorporate the tutor. The modify policy uses a switching condition to toggle between RL policy and Control tutor. This switching condition is given by:

$$\pi(x) = \begin{cases} \pi^{rl}, & \text{with probability } 1-\zeta \\ \pi^{c}, & \text{with probability } \zeta \end{cases}$$

where ζ is switching rate, $\pi^{rl}(x)$ and $\pi^{c}(x)$ is given by:

$$\pi^{rl}(x) = \begin{cases} \arg\max\limits_{u \in U} Q_k(x,u), & \text{with probability } 1-\epsilon^{rl} \\ u \sim Rand(U) & \text{with probability } \epsilon^{rl} \end{cases}$$

$$\pi^{c}(x) = \begin{cases} \arg\min\limits_{u \in U} ||\mathrm{v(x)} - \mathrm{u}||, & \text{with probability } 1-\epsilon^{c} \\ u \sim Rand(U) & \text{with probability } \epsilon^{c} \end{cases}$$

where $\pi^{rl}(x)$ is action suggested according to RL policy, $\pi^{c}(x)$ is action suggested by control tutor, ϵ^{rl} is exploration rate for RL policy, ϵ^{c} is exploration rate for control policy U is action space, and $v(x)$ is continuous control input generated by the mathematical model.

Both exploration rate is diminishing in nature which will make our agent more exploration at initial phase of training and more exploitation in later stage.

Figure 1.5 Explains the CTQL Policy.

Chapter 2

Simulation

2.1 Performance Comparison Parameters

2.1.1 Average Reward

In the initial Phase of training out agent does not know anything so its Q-table is just random values so it tries to explore the system to gain knowledge of it. This random exploration may lead to undesirable state where out agent will gain lesser reward. This behaviour is true for the initial phase of training later it will have learnt enough to avoid that state.

So if we take reward of every episode of initial phase of training and take its average we can quantify how frequently our agent encounter this undesirable state and failures. Higher the average reward, less frequent those encounters. To make sure we only take average of the initial training phase we take average of reward of first 150 episodes of training. Since this reward can vary with initial conditions we take its mean over 10 independent training session each having different initial condition.

2.1.2 Average Converging Episode

Our algorithm will converge if it balances the pole continuously for 120 episodes. So converging episode is number of episode after which this condition is satisfied. Less converging episode indicate that our agent learn faster. This parameter will also vary with the initial condition so we take its mean over 10 independent training session each having different initial condition.

2.2 Environment and Reward used for Simulation

OpenAI gym is a well known platform which provides various environments to compare and develop Reinforcement learning algorithms.[1] known as Cart-pole system is an environment of classical control environment library.
It has a cart that travels down a friction-less track that has a pole linked to it by a joint that cannot be moved. The upright position of the pendulum on the cart is intended to balance the pole by exerting pressures to the left and right of the cart.

We modified the environment parameter to simulate our system using this environment.The details of environment, constraints, episode termination condition, action space, state space and reward of modified system is given below:

Physical Parameters of system:

- Mass of Cart $M = 2$ kg
- Mass of Pole $m = 0.1$ kg
- Length of Pole = 1 m
- Distance of centre of mass of Pole $l = 0.5$ m

State space:

- Angular Position : -24 degrees to +24 degrees divided equally in 6 buckets
- Angular Velocity: -Inf to +Inf divided in 3 buckets
- Linear Position = -4.8 to +4.8 meters
- Linear Velocity = -Inf to +Inf

Action space: 5 control forces [-10N, -5N, 0N, 5N, 10N]

Episode End condition:

- Termination: Angular position goes out of ±5 degree bound
- Termination: Linear position goes out of ±2.4 meter bound
- Truncation: Episode length greater than 200

Reward: The pole must remain upright for as long as feasible, thus each step taken including the ending step receives a reward of 1

2.3 Result and Inference

2.3.1 Rewards in QL and CTQL

	Conventional Q-Learning	Control Tutored Q-Learning
Average Initial Rewards	39.815	46.951
Converging Episodes	299.1	293.1

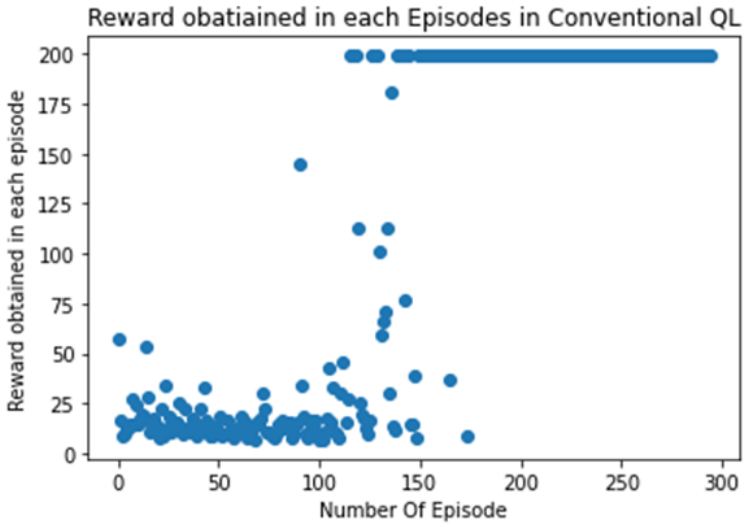


Figure 2.1: Rewards in QL

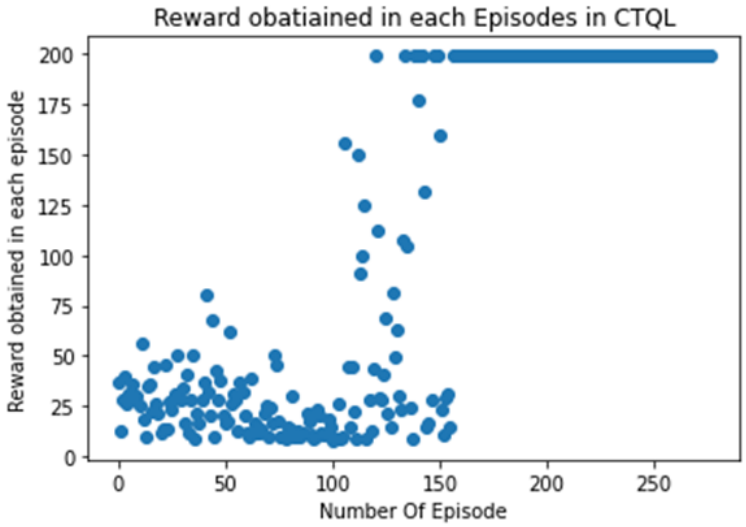


Figure 2.2: Rewards in CTQL

1. The figures 2.1 and 2.2 show the reward obtained in each episode. When we compare the the rewards in the initial phase, we see that reward in control tutored q-learning is much higher than classical q-learning

2. After around 150 episodes the reward in both the algorithms is much higher and reaching the maximum value

3. The number of episodes required to train the agent is less in control tutored q-learning

2.3.2 Inference from result

- Higher average reward in starting phase means that control tutor is suggesting better action as compared to reinforcement learning policy in starting phase. The control

tutored q-learning is able to balance the pendulum for more time-stamps as compared to classical q-learning algorithm.

- After around 150 episodes the reward in both the algorithms is much higher and reaching the maximum value. This shows the convergence in both the algorithms. The CTQL converges faster as compared to q-learning as the number of episodes required is less in CTQL as compared to classical QL.

From above discussion we can draw the inference that the control tutored q-learning is more safe and faster.

Chapter 3

Summary

In this paper we tested control tutored Q-learning and classical Q-learning on Inverted Pendulum problem as defined in OpenAI Gym [1]. We designed controller for the linearized inverted pendulum system and used this as the partial model in balancing non-linear system. From the result we can conclude that control tutored Q-learning (CTQL) has performed better than classical Q-learning. Control tutored Q-Learning give higher average reward as compared to classical Q-learning. Therefore, when we have a partial model of the system then we can use this partial model with classical reinforcement learning for more safer and faster training of agent.

Bibliography

[1] openai/gym, November 2022. original-date: 2016-04-27T14:59:16Z.

[2] Francesco De Lellis, Giovanni Russo, and Mario Di Bernardo. *Tutoring Reinforcement Learning via Feedback Control*. December 2020.

[3] Francesco De Lellis, Marco Coraggio, Giovanni Russo, Mirco Musolesi, and Mario di Bernardo. Control-Tutored Reinforcement Learning: Towards the Integration of Data-Driven and Model-Based Control. In *Proceedings of The 4th Annual Learning for Dynamics and Control Conference*, pages 1048–1059. PMLR, May 2022. ISSN: 2640-3498.

[4] Katsuhiko Ogata. *Modern control engineering*. Prentice-Hall electrical engineering series. Instrumentation and controls series. Prentice-Hall, Boston, 5th ed edition, 2010.

[5] Christopher Watkins and Peter Dayan. Technical Note: Q-Learning. *Machine Learning*, 8:279–292, May 1992.