Solving the Inverted Pendulum Control Problem using Reinforcement Learning

Aalap Doshi
Electrical and Computer Engineering
Arizona State University

Poojan Patel
Electrical and Computer Engineering
Arizona State University

Varadaraya Ganesh Shenoy
Electrical and Computer Engineering
Arizona State University

***Abstract*—The inverted pendulum is a typical, intriguing control issue that includes numerous essential components of control hypothesis. It is a classical benchmark control problem because its dynamics resembles with that of numerous real world systems of interest like missile launchers, human walking, segways and many more. The control of this system is challenging as it is exceptionally unsteady, profoundly non-straight, non-minimum phase system and under actuated. Also, there are physical constraints on the track position control voltage complexity in its control design. We need a controller to balance the pendulum such that the cart on the track can be controlled quickly and accurately so that the cart remains in the center and the pendulum is always straight in its inverted position during its motion. Reinforcement learning methods are discussed which follows with the idea of punishing for bad actions and rewarding for good ones.**

***Index Terms*—Inverted Pendulum, Reinforcement Learning, Temporal Difference**

I. Problem Statement

The inverted pendulum is a popular research problem as it is an unstable system (ie) the pendulum will fall over if the movement of the cart doesn't balance it. Thus, the primary objective of the control design is to apply a force to the cart in order to balance the pendulum.

However, orthodox design techniques are successful given knowledge of the system including its dynamics, and an objective function expresses the desired behavior of the system. But, how can control be achieved if this knowledge is not available? This question is solved by learning a function which selects control actions given the current state of the pendulum, through experience of trying various actions and examining the results, with no history pertaining to the correct action to begin with.

This type of controller design where the objective function which evaluate states and actions is absent, can be modified only on basis of occurrence of failure signals. This results in an assignment-of-credit problem, which is necessary to trace the sequence of actions leading to failure.[3]

In this project, a 2D problem in which the pendulum movement is restricted along the vertical plane. The control input for the system is the force responsible for horizontal movement of the cart while outputs are the angular position and the cart position.

The project implements machine learning methods that help realize successful action sequences by generating two functions, namely action and evaluation functions respectively. Current state of the system is translated to the corresponding control using action function while the mapping of the current state into its evaluation is carried out by the latter. These functions are learnt using two networks referred to as : Action Network and Evaluation Network.

II. Inverted Pendulum

In this case we will consider a two-dimensional issue where the pendulum is compelled to move in the vertical plane appeared as depicted in the figure. For this framework, the control input is the force F_t that moves the cart horizontally on the level plane and the outputs are the angular position of the pendulum θ_t and the horizontal position of the cart h_t.[4]

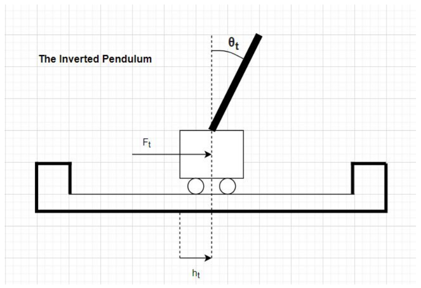

Fig. 1: The Inverted Pendulum

The θ_t and its state is used to balance the pendulum while the cart is moving in the horizontal direction both in right and left direction.

Initially we observe a lot of change in the angle because of the disturbance and constant movement of cart. As time moves forward we see that the error in the angle reduces as the systems starts learning from the failure signal.

Now, what is failure signal? Here, failure signal can be described in two parts: first being the change in the angles in degree and second the horizontal movement of cart in both the direction. We can always assign a threshold for the system for both θ_t and h_t so that when the system starts balancing the inverted pendulum with each motion of pendulum and cart the system learns from its failure to balance it perfectly with each attempt and manages to reduce the error every time.

For example, the pendulum falling past +15 degree or the cart hitting the bounds of the track at +3.0m. The goal of the inverted pendulum task is to apply a sequence of right and left forces of fixed magnitude to the cart such that the pendulum is balanced and the cart does not hit the edge of the track.

The objective as simply expressed makes this errand exceptionally troublesome; the failure signal is a delayed and rare performance measure. Before portraying an answer for this plan of the altered pendulum task, we quickly examine different methodologies that assume the existence of additional task-specific knowledge.

Here, we try the approach with state equation and state variables.

III. Force Equations and System Analysis

State variable of the system are those parameters of interest which can be utilized to locate every single other parameter of the system and whose information permit us to think about the present or future condition of the general system. And State Equation shows the relationship between the system's actual state and its input, and the future state of the system. The Output Equation shows the relationship between the system state and its input, and the output.

We tested the approach which used just the state variables and state equation along with some motion equations to control motion of inverted pendulum.

According to the free-body diagrams of the two elements of the system as illustrated below **[9]** :

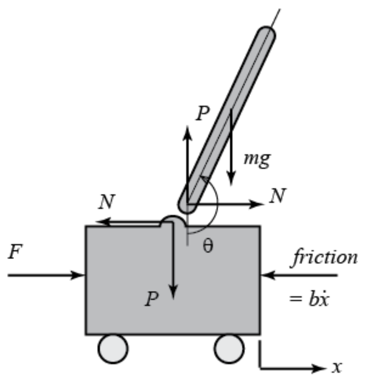

Fig. 2: Free-Body Diagram

Summing the forces in horizontal and vertical directions of the cart, the equation obtained for the horizontal direction is,

$$M\ddot{x} + b\dot{x} + N = F \tag{1}$$

The following equation is got when the forces in horizontal direction of the pendulum is summed,

$$N = m\ddot{x} + ml\ddot{\theta}cos(\theta) - ml\dot{\theta}^2 sin(\theta) \tag{2}$$

Putting (2) in (1),

$$(M+m)\ddot{x} + b\dot{x} + ml\ddot{\theta}cos(\theta) - ml\dot{\theta}^2 sin(\theta) = F \tag{3}$$

When the forces perpendicular to the pendulum are added , the second equation of motion is derived,

$$Psin(\theta) + Ncos(\theta) - mgsin(\theta) = ml\ddot{\theta} + m\ddot{x}cos(\theta) \tag{4}$$

In order to eliminate P and N in the above equation, the centroids about the pendulum is added,

$$-Plsin(\theta) - Nlcos(\theta) = I\ddot{\theta} \tag{5}$$

Combining (4) and (5) we get,

$$(I + ml^2)\ddot{\theta} + mglsin(\theta) = -ml\ddot{x}cos(\theta) \tag{6}$$

It is observed that eq(6) represents a non-linear equation, which calls for linearization of the system. For the inverted pendulum system, it is realistic to linearize the system along the vertical axis.Thereby, $\theta = \pi$ is the equilibrium chosen assuming the system will always be in the neighborhood of this criteria.Thus, resulting in θ becoming $\pi + \phi$.

The traditional approach for the control and modeling of any system is when we know all the dynamics of the system and all the motion equations of the system.

For inverted pendulum problem the equations are assumed that the control Force F t is a linear function of the four state variables (θ_t, $\dot{\theta}_t$, h_t, $\dot{h}_t$) and in an equation form it would have constant coefficients multiplied with it. From the exercise carried out and also when we read about how state equations and state variables can be used for dynamic control of inverted pendulum and also cart motion. It was found that this approach was tested by **[1]** in which they used linear feedback in three out of four variables to test stable control of a ball-balancing experiment.The experiment's success heavily depends on state equations and the linearized approximation.

However, we need a compact system which is able to control in any dynamical situation and it does not have any limitation regarding any environmental conditions for balance. We found that the first use of neural network was performed using state variables and equations and then after seeing the input-output behavior of the linear control systems mimic the same system by training the neural network. This caused many different approaches to arise with different limitations.

As per our analysis multilayer networks can also be used with nonlinear control laws but here as well the system dynamics should be known. What if we do not know the system dynamics but we still want the desired result? Here, we would need some adaptive or learning approach to obtain a stable control. Even a novel approach of completely eliminating state variables and using image processing to derive visual image of the pendulum location and giving a real time feedback to the system can be used. But, all this approach cannot give us an optimal solution.

One of the interesting solutions we found was that if neither a designed controller nor a human expert is available we should make learning based on the actual parameters and keep the feedback loop closed so that each and every time an error is generated it learns from that and reduces it in the next go. A reference states can be assigned for example (0, 0, 0, 0) and based on difference from the current state to reference state each action can be taken to reduce this difference. All the above discussion is effective but has some or other limitation which eliminates its use.

Following we discuss neural network approach using networks, state variables and machine learning technique to solve the problem.

IV. Methods

A. *Inverted Pendulum using Widrow-Hoff LMS Algorithm*

According to **[2]** , the force required to stabilize the inverted pendulum system shown in Fig1 at a time instant t is -

$$F = U * sgn[(k_1 + k_2)x_t + k_2 v_{t-1} + (k_3 + k_4)\theta_t - k_4 \omega_{t-1}] \quad (7)$$

where : the position of the cart (x), the velocity of the cart (v), the angle of the pendulum (θ), and the angular velocity of the pendulum (ω). The cart and pendulum velocities can be estimated from the instantaneous cart and pendulum velocities which can be obtained from the current states and its history. For the network to balance the pendulum, in some sense it needed to implement Equation 10, estimating the pendulum and cart positions multiplied by the coefficients k_1,k_2,k_3,and k_4. Finding a set of weights requires solving a system of M linear equations with N unknowns where M is the number of images and N is the number of weights. Minimum mean-squared-error can be found iteratively by using the Widrow-Hoff LMS algorithm.

Although, the above work used visual images to train the network.It is important to note that training is limited, therefore, machine should be capable enough to perform the task independent of training, responding correctly to almost all situations.

B. *Single Layer Approach*

In this section we will discuss about a different approach in which we learn about neural network approach and learning methods such as supervised learning, unsupervised learning. Supervised learning methods are most commonly used learning method in neural network but it requires a training data consisting of input data and output data. For the inverted pendulum problem we are not aware about the output data since it varies with environment so we cannot predict the correct action. Here, Reinforcement learning comes into the picture because it learns from the effects measured by changes in an evaluation signal.

[1] While we do not know the output vectors we can complete the loop by checking each failure signal and reduce the error by each iteration. The method which we used has two single layer network with one named action network and other being evaluation network. Action network penalizes each control action of the pendulum for maintaining the angle and for cart to push left and right according the magnitude of force. Output of this unit is probabilistic which means the initial weights are zero and then the action units learn via a reinforcement learning method. With each movement it learns and assigns different weights to the system.

So, even with a slight movement learning action takes place but it is very slow because of the delayed signal. Second network is called the evaluation network, which also consists of single unit. This unit specially calculates the Temporal Difference (TD) which measures the sum of future failure signals by prediction. The prediction is based on the output of the inverted pendulum state vectors and failure signal. Through learning, the output of the evaluation network comes to predict the failure signal, with the quality of the expectation showing how soon disappointment can be relied upon to happen. With each step of prediction of network's input and the change in the value of new prediction (the difference), based on the current state of the inverted pendulum, and the previous prediction, based on the previous state the predictions are adjusted accordingly. This sequence of event goes on and changes ends with occurrence of failure signal. The final prediction is based on failure signal and the previous prediction. This is the reason why evaluation network is very important. Inner product of the input vector and unit's weight vector gives output. Input vectors are the states of inverted pendulum and weights are developed by the TD method. With each ranking to the states the difference in the units output on the transition from one state to another is used to judge effectiveness of previous action. A decrease in failure signal proves that the evaluation is effective and probability of prediction is increased. In this way reinforcement learning is performed. However, one problem which we found is that learning is not sufficient in initial stage without much experience so the weights updated at that time will be very uncertain and that causes disturbance in the system initially.

The four state variables are adequate to represent action unit since the output for it is linear. The failure signal is given value -1 and 0 to other states. Failure occurs when the value of θ goes less than -12 or goes higher than 12 degrees. Evaluation unit is

basically function of just θ (pendulum's angle) and $\dot{\theta}$ (angular velocity). With the pendulum moving towards balanced position, evaluation is closer to 0 which means a weaker prediction of failure. But we need to develop a different representation of function for evaluation unit.

Parameters	Values
Mass Cart	1.0
Mass Pole	0.1
Pole Length	0.5
Force	10
δt	0.02
$\theta_{threshold}$	12(degrees)
δt	2.4

TABLE I: Used Parameters in Code

The controller takes in the system states, and outputs a fixed force on the cart to either left or right. The controller needs to be designed so that within 4 seconds, the pole angle does not exceed 12 degrees, and the cart displacement does not exceed 2.4 unit lengths. A trial will be terminated if the system is balanced for more than 4 seconds, or any of the constraints are violated. We need to use the above parameters on controller side so that the system can stay in the stable form.

C. *The Optimization Problem*

[10]Optimal Control is used to train the model and Reinforcement learning plays the vital role to train the model with neural network. We need to assign a learning algorithm and in our method controller is learned without understanding of the dynamical system.

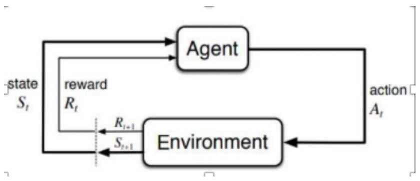

Fig. 3: Typical Reinforcement Learning

We have used Markov Decision Process and MDP contains a state space, an action space, a transition function, a reward function, and a decay parameter. In our case, the state space contains all possible combinations of (θ_t, $\dot{\theta}_t$, h_t , $\dot{h}_t$). The action space contains the force to the left, and the force to the right. The transition function computes the next state based on the current state and the action. In our case, the transition is given by the system equations. The reward function defines an instantaneous reward. In our case, we define reward as 1 when the system does not fail, or 0 otherwise. The decay parameter defines a long term value of the controller: describes how important future rewards are to the current control decision: larger decay (small) leads to more greedy decisions.

Reinforcement Learning is a type of Machine Learning. It allows machines and software agents to automatically decide the perfect conduct inside a particular setting, so as to boost its exhibition. Simple reward feedback is required for the agent to learn its behavior; this is known as the reinforcement signal.

There are a wide range of calculations that handle this issue. Truly, Reinforcement Learning is characterized by a particular sort of issue, and every one of its answers are classed as Reinforcement Learning calculations. In the problem, an agent is supposed to decide the best action to select based on his current state. When this step is repeated, the problem is known as a Markov Decision Process.**[10]**

A simple Markov Decision Process model contains: **[10]**

- A set of possible world states S.
- A set of Models.
- A set of possible actions A.
- A real valued reward function R(s, a).
- A policy the solution of Markov Decision Process.

The agent receives rewards each time step:-

- Small reward each step (can be negative when can also be term as punishment, in the above example entering the Fire can have a reward of -1).
- Big rewards come at the end (good or bad).
- The goal is to maximize sum of rewards.

D. *Bellman's Equation*

[7]As written in the book by Sutton and Barto, the Bellman equation is an approach towards solving the term of "optimal control". Which is done through the creation of a functional equation that describes the problem of designing a controller to minimize a measure of a dynamical system's behavior over time.. This approach of Bellman utilizes the concepts of a state and a value function as shown below.

$$V^{\pi}(s) = E_{pi}[G_t \mid s_t = s] = E_{\pi}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s] \quad (8)$$

$$Q_{\pi}(s,a) = E[G_t \mid s_t = s, a_t = a] = E_{\pi}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a] \quad (9)$$

Equations 7 and 8 represent State-Value Function and Action-Value Function respectively.

The main purpose of the Bellman's Equation is to represent State-Value and Action-Value Functions as a recursive relationship between the value of a state and the value of its successor states.

V. Implementation

This project has been conducted on Python3.7 using the OpenAI gym package to learn a control algorithm to balance a pole with a learning rate of 0.1 and a decay parameter of 0.95.

Markov Decision Process a discrete time stochastic control process contains a state space, an action space, a transition function, a reward function, and a decay parameter . Since we perform optimization using Markov Decision Process, the state space contains all possible combinations of position of pole. The action space has the left and right forces, the transition function is given by the system equations. Reward is defined as 1 if system does not fail otherwise 0.

A. *Learning Algorithm*

[8]Given a trial run with time steps based on the current controller, instantaneous rewards, actions and the controller outputs. Thus, allowing us to minimize the loss function given in below equation.

$$f(w) = -\sum_{k=1}^{K}(\sum_{j=k}^{K} \gamma^{j-k} r_j)(a_k log(\pi_k) + (1 - a_k log(1 - \pi_k)) \quad (10)$$

In this loss function, the first term -

$$\boxed{\sum_{j=k}^{K} \gamma^{j-k} r_j} \quad (11)$$

represents the value at time step, the second term -

$$\boxed{(a_k log(\pi_k) + (1 - a_k log(1 - \pi_k))} \quad (12)$$

is close to when the favored action is chosen, and when the unfavoured action is chosen.

In short, minimizing the loss when all chosen controls are at a high value. Therefore, tuning helps correct the shortcomings in the trial (i.e., high value from wrong move, or low value right move).The values are normalized using Adaptive Normalization using Pop-Art.**[5]**

The controller modeled using a single layer as it is found to be efficient and more effective in this setting.

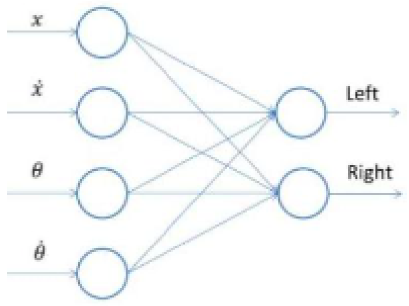

Fig. 4: Controller Model Single-Layer Neural Network

B. Training Algorithm

[8]The averaged loss over number of trials has been minimized because of the value function being probabilistic nature. The gradient is stochastic as we use random samples for approximation. The gradient descent is realized using Adaptive Moment Estimation (ADAM) and Momentum optimizer.

VI. Results

The figure below shows an instance when the pendulum is balanced for a physical time fo 4 seconds.

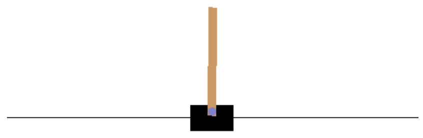

Fig. 5: Instance of Balanced Pendulum

In the default environment, the following convergence is obtained for ADAM and MOMENTUM. **[8]**From the above

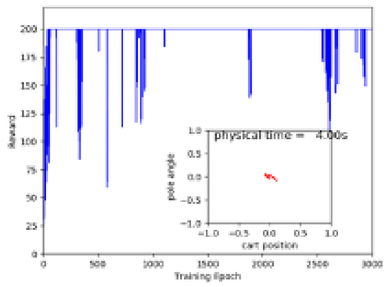

Fig. 6: Convergence using ADAM optimizer

figures, it is observed that as the training epoch increases the

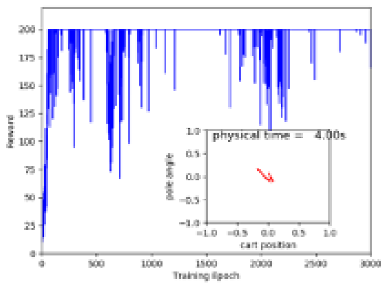

Fig. 7: Convergence using Momentum Optimizer

better control is achieved in both cases although the control is more stable in case of ADAM optimizer compared to that of Momentum optimizer, because it incorporates adaptive estimation of moments.

From the smaller plot which shows the normalized trajectory of cart position versus the pole angle, ADAM optimizer results a better pole balancing with minimal displacement of cart when compared to that of the momentum optimizer which needs move left and right in order to achieve the right balancing position.

As the controller takes in 4 states to determine two actions, a single layer is more sufficient. With 2 hidden layers, the controller is not reliable as a result of which controller failure is occurring more than the successful balancing the pole. On comparing Figure (5) with Figure (7), it is pretty obvious that having single layer is optimal to learn control of the cart-pole system.

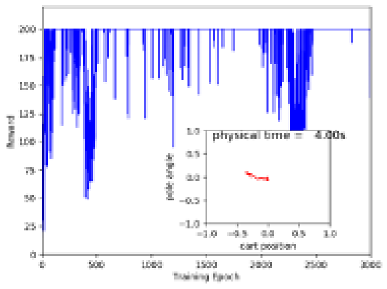

Fig. 8: Convergence using 2 hidden Layers

VII. Future Work

A. Two Layer Network

In concluding how to isolate the state space, one must find some kind of harmony between an all-inclusive statement, furthermore, learning speed. An extremely fine quantization with numerous districts licenses exact estimate of complex capacities, yet learning the right yield for every one of the numerous locales requires a lot of understanding. Learning can be quicker with a coarse quantization on the grounds that learning for one state in a district is moved to all states in the district, yet just capacities whose yield remains generally consistent over locales can be spoken to. Obviously, a versatile portrayal that learns a quantization or other type of highlight set dependent on experience is required. It ought to figure out how to make fine segregations among certain states and coarse speculations among others, as fitting for guaranteed task. **[3]** The first units are known as the yield units of the

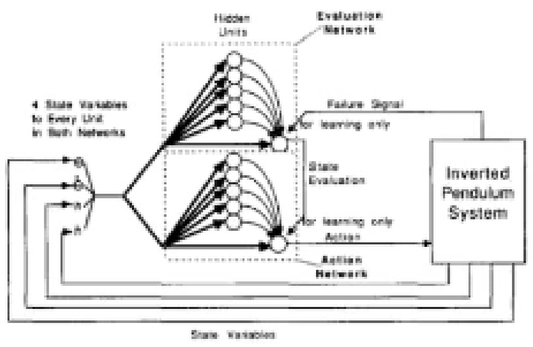

Fig. 9: Two Layer Network

systems. The new units are called concealed units since their yields do not directly affect the system's condition. Though yield unit learning can be founded on assessment contrasts, there is no like a flag on which to base leading in the shrouded units. Subsequent to doling out credit to an individual activity, there remains the issue in a multilayer system of conveying this credit among the concealed units that impacted the choice of that activity by the yield unit. This is one of the serious issues that eased back advancements in versatile systems.When given the state factors as input, couldn't figure out how to adjust the pendulum, really improving than the nonearning methodology of picking activities arbitrarily with equivalent likelihood. Indeed, even though a decent control law can be spoken to by the single-layer activity arrange, the reality that the direct assessment arrange can't structure valuable assessments forestalled the control law from being scholarly.

B. Bellman's Equation and Q-Learning

Other popular methods used to learn control include using Bellman's Equation for value and policy iteration.In addition, this can be extended to the implementation of Q-Learning and Deep Q-Networks depicted in figure 9 to learn the controller better and faster with more precise estimates.

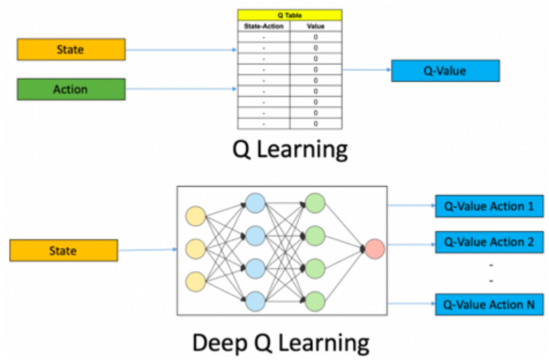

Fig. 10: Q-Learning and Deep Q-Networks **[6]**

References

[1] K. C. Cheok and N. K. Loh. "A Ball-Balancing Demonstration of Optimal and Disturbance-Accommodating Control". In: *IEEE Control Systems Magazine* 7.1 (1987). DOI: 10.1109/MCS.1987.1105235.

[2] V.V.Tolat and Bernard Widrow. "An adaptive 'broom balancer' with visual inputs". In: *IEEE 1988 International Conference on Neural Networks* (1988). DOI: 10.1109/ICNN.1988.23982.

[3] Charles W Anderson. "Learning to control an inverted pendulum using neural networks". In: *IEEE Control Systems Magazine* 9.3 (1989), pp. 31–37. DOI: 10.1109/37.24809.

[4] Valeri Mladenov. "Application of Neural Networks for Control of Inverted Pendulum". In: *WSEAS Transactions on Circuits and Systems* (2011).

[5] Matteo Hessel Volodymyr Mnih Hado van Hasselt Arthur Guez and David Silver. "Learning values across many orders of magnitude". In: *NIPS 2016* (2016).

[6] Ankit Choudhary. *A Hands-On Introduction to Deep Q-Learning using OpenAI Gym in Python*. URL: https://www.analyticsvidhya.com/blog/2019/04/introduction-deep-q-learning-python/.

[7] Xavier Geerinck. *Bellman Equations*. URL: https://xaviergeerinck.com/post/ai/rl/bellman-equations/.

[8] Transport Design Informatics Lab — School for Engineering of Matter and Arizona State University (2020) Energy. *Tutorial on Reinforcement Learning for Controller Design*. URL: https://designinformaticslab.github.io/mechdesign_lecture/2018/04/11/reinforcementlearning.html.

[9] University of Michigan. *Inverted Pendulum: System Modeling*. URL: http://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum&section=SystemModeling.

[10] Abhishek Sharma. *Markov Decision Process*. URL: https://www.geeksforgeeks.org/markov-decision-process/.