 doi:10.20944/preprints202005.0181.v1

Modeling an Inverted Pendulum via Differential Equations and Reinforcement Learning Techniques

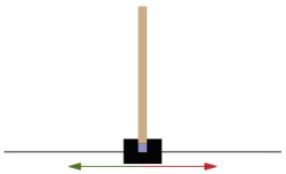
Siddharth Sharma, BASIS Independent Silicon Valley
1st period, Differential Equations,
Supervisor: Dr. Gerges

Abstract -**The prevalence of differential equations as a mathematical technique has refined the fields of control theory and constrained optimization due to the newfound ability to accurately model chaotic, unbalanced systems. However, in recent research, systems are increasingly more nonlinear and difficult to model using Differential Equations only. Thus, a newer technique is to use policy iteration and Reinforcement Learning, techniques that center around an action and reward sequence for a controller. Reinforcement Learning (RL) can be applied to control theory problems since a system can robustly apply RL in a dynamic environment such as the cartpole system (an inverted pendulum). This solution successfully avoids use of PID or other dynamics optimization systems, in favor of a more robust, reward-based control mechanism. This paper applies RL and Q-Learning to the classic cartpole problem, while also discussing the mathematical background and differential equations which are used to model the aforementioned system.**

I. Introduction to the Cartpole problem

The cart pole problem is a famous problem in dynamics and control theory, with a pendulum whose center of gravity is above the pivot point. This naturally creates an unstable system and the pendulum will typically remain vertically downward without any force of swing or dynamic control being applied. The cartpole has one degree of freedom upon its axis, and the system has no vertical movement. The goal of most cartpole systems is to effectively keep the cartpole balanced through applying various forces on the pivot point and its axis of movement (the horizontal direction).

Figure I. Typical Setup for Cartpole problem



Cartpole schematic drawing

II. Differential Equations for modeling the Cartpole System

We may derive the equation for controlling the cartpole system with cart mass (M), pole mass (m), angle (θ), length (l), and force (f). We begin by utilizing Newton's Second Law of Motion. This is preferable since it avoids the mathematics involved with Lagrange's equations while also providing reaction forces between the pendulum and cart at its joint:

$$F - R_x = M\ddot{x}$$
$$F_N - R_y - Mg = 0$$

R_x and R_y represent the reaction forces while F_N represents the normal force applied to the cart. The above equations are in the x and y axes respectively. We may use the first equation to solve for the horizontal reaction force. However, we must first note the acceleration of the point mass of the system. The positional vector can be described as:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 10 May 2020 doi:10.20944/preprints202005.0181.v1

$$\vec{r}_P = (x - \ell \sin\theta)\hat{x}_I + \ell \cos\theta \hat{y}_I$$

Now, we take the derivative twice to obtain the acceleration vector in the Inertial reference frame:

$$\vec{a}_{P/I} = (\ddot{x} + \ell\dot{\theta}^2 \sin\theta - \ell\ddot{\theta}\cos\theta)\hat{x}_I + (-\ell\dot{\theta}^2 \cos\theta - \ell\ddot{\theta}\sin\theta)\hat{y}_I$$

We may now solve for the reaction forces independently via Newton's Second Law:

$$R_x = m(\ddot{x} + \ell\dot{\theta}^2 \sin\theta - \ell\ddot{\theta}\cos\theta)$$
$$R_y - mg = m(-\ell\dot{\theta}^2 \cos\theta - \ell\ddot{\theta}\sin\theta)$$

In the event that the applied force on the system is unknown, we can use the first equation (R_x) to solve for it. We substitute:

$$F - R_x = M\ddot{x} \text{ into } R_x = m(\ddot{x} + \ell\dot{\theta}^2 \sin\theta - \ell\ddot{\theta}\cos\theta)$$

Which yields:

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

By observation, we know that this equation is equal to the output of the Lagrange's system of equations for an inverted pendulum. To obtain the second equation that governs the system, we must dot the pendulum equation with an orthonormal unit vector. We use 2-D translation:

$$\hat{x}_B = \cos\theta\hat{x}_I + \sin\theta\hat{y}_I$$

The equation of motion for the pendulum is written

$$\sum \vec{F} = m\vec{a}_{P/I}.$$

Performing the transpose (dotting) of the system is done by dotting both sides of the pendulum equation with x_B:

Left-Hand Side (a.) : $(\hat{x}_B)^T \sum \vec{F}$

Left Hand Side (b.) :
$(\hat{x}_B)^T(R_x\hat{x}_I + R_y\hat{y}_I - mg\hat{y}_I) = (\hat{x}_B)^T(R_p\hat{y}_B - mg\hat{y}_I) = -mg\sin\theta$

These equations assume that the bar connecting the rod to the cart is massless. The equation utilizes the relationship between the inertial frame components and body frame components of the reaction forces to compute the above relationship between the LHS and RHS. To solve for the tension in the rod, we derive the following equation:

$$R_p = \sqrt{R_x^2 + R_y^2}$$

We may now solve for the Right-Hand Side of the equation via dotting x_B with the acceleration of the pendulum:

Right-Hand Side (a.) :

$$m(\hat{x}_B)^T(\vec{a}_{P/I}) = m(\ddot{x}\cos\theta - \ell\ddot{\theta})$$

We can later combine the equations for the LHS (left side) and RHS (righthand side) to obtain the solution (upon dividing by m):

$$\ell\ddot{\theta} - g\sin\theta = \ddot{x}\cos\theta$$

The above equation is able to model the controlling of the cartpole system for angle θ and length l and displacement x. The result is the same that may be applied from the Lagrangian system of equations.

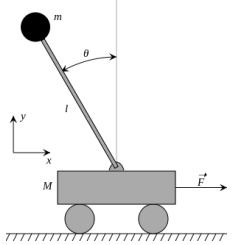


Diagram of cartpole mechanics

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 10 May 2020 doi:10.20944/preprints202005.0181.v1

III. Analysis of Cartpole Differential Equations

The earlier derived equation represents the mechanics and variables which control the cartpole system:

$$\ell\ddot{\theta} - g\sin\theta = \ddot{x}\cos\theta$$

Therefore, it can be reasoned that the cartpole system's balance and net force depends primarily by the length *l*, the acceleration due to gravity *g*, and displacement *x*. In order to keep the system balanced, it is necessary to satisfy this linear equation. In a nonlinear system, we may model the dynamics of the cartpole system as:

$$\ddot{\theta} = \frac{(M+m)g\sin\theta - \cos\theta[F+ml\dot{\theta}^2\sin\theta]}{\left(\frac{4}{3}\right)(M+m)l - ml\cos^2\theta}$$

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

In this case, *F* is the input force on the nonlinear system. For the purposes of the classic cartpole problem, we will only discuss the further applications of the linear dynamics.

IV. Control Theory Techniques

In solving the physical cartpole problem, it is common to utilize a feedback control loop in conjunction with both proportional-integral-derivative (PID) and linear-quadratic-regulator (LQR) controllers. These mechanisms are typically adapted for industrial applications and may be similarly applied to a nonlinear cartpole dynamics system. Both LQR and PID controllers seek to find the optimal control mechanism. However, the complexity of systems has rapidly increased, requiring more sophisticated controllers which can accurately respond to parameter variations such as increased noise and time and space nonlinearities. Thus, we may need more advanced algorithms to robustly handle the various dynamics of the traditional cartpole system. We will now look into Reinforcement algorithms which may effectively manipulate the dynamics of a cartpole system successfully.

V. Reinforcement Learning

Reinforcement Learning (RL) is a growing subset of Machine Learning which involves software agents attempting to take actions or make moves in hopes of maximizing some prioritized reward. There are several different forms of feedback which may govern the methods of a RL system. Compared to Supervised Learning algorithms which map functions from input to output, RL algorithms typically do not involve the target outputs (only inputs are given). There are 3 elements of a basic RL algorithm: the agent (which can choose to commit to actions in its current state), the environment (responds to action and provides new input to agent), and the reward (incentive or cumulative mechanism returned by environment).

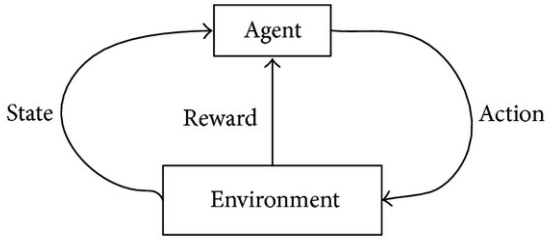


Figure II. Reinforcement Learning Feedback Loop

The broad goal of most RL algorithms is to achieve balance between exploration (training on new data points) and exploitation (use of previously captured data). The immediate goal is to maximize the reward with trials alternating between the aforementioned exploitation and

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 10 May 2020 doi:10.20944/preprints202005.0181.v1

exploration. It is important to note that are three types of RL implementations: policy-based, value-based, and model-based. Policy-based RL involves coming up with a policy or deterministic/stochastic strategy to maximize the cumulative reward. Value-based RL attempts to maximize an arbitrary value function, *V(s)*. Model-based RL is based on creating a virtual model for a certain environment and the agent learns to perform within the constraints of the environment.

VI. Reinforcement Learning for the Cartpole System

Since RL is a form of learning characterized by trial and error response to actions and its effect on the environment, it makes sense to model the cartpole system via RL, since the cartpole system is heavily subject to various parameter changes while having a clearly defined agent-action-environment-reward schema. The agent is the controller or algorithm which controls the movement of the cart. The action is the physical movement of the cartpole in response to various forces and torques following the swing-up phase. The environment is the physical setting of the cartpole in regard to the constrained area of the system. The reward is the ability of the cartpole to achieve sustained balance in its current state. We will now identify the specific actions and states of the cartpole problem:

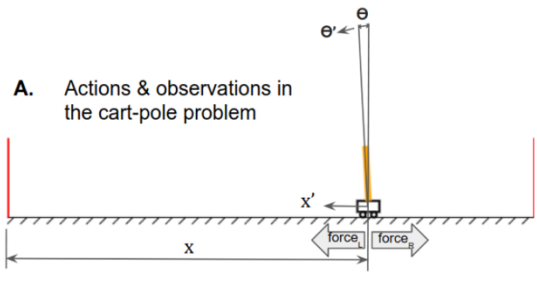


Figure III. Actions for Cartpole problem

The cartpole agent is limited to two possible actions: (1) exert a rightward constant force on the cart. (2) exert a leftward constant force on the cart. As seen in the diagram, these two forces are directed in the horizontal direction. These actions that may be taken by the agent will change the position of the cart and environment accordingly. The state of the cartpole is solely determined by the velocity and position of the cart, angle θ, and pole velocity at the tip. All of these parameters were earlier identified as the basis for the differential equation which addresses the properties necessary to adjust and control the cartpole system.

Each time a force is applied by the controller, the controller checks for whether the cumulative reward is achieved or maximized. In the case of the cartpole problem, the angle of the pole in respect to the cart and distance from the center determines the value/reward achieved. If the cartpole is generally upright and near the center of the environment, a reward is given and maximized for that sequence. In the case that the reward is not maximized, the controller makes the necessary adjustments to the force and subsequent displacement. It is also important to note that there are two crucial conditions that may terminate or restart the action-environment-reward loop.

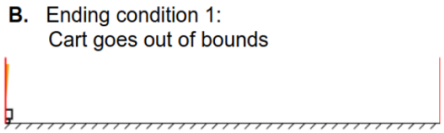


Figure IV. Cartpole system goes out of bounds

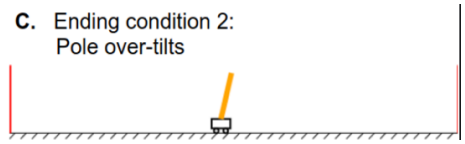


Figure V. Cartpole system overtilts beyond angle

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 10 May 2020 doi:10.20944/preprints202005.0181.v1

In both of the above cases, the agent's actions led to a case that did not maximize the reward in the specified environment. (cartpole was either not upright or not within the specified bounds) This could be seen as the "punishment" of the model. On the other hand, each time the reward is gained, the "score" of the cartpole increases by 1.

Now that we have discussed the physical properties of the cartpole system, the mathematical model used to solve it, and the application of RL towards the cartpole, we may move to analyze two algorithms that may allow for effective control and stability of the cartpole.

VI. Q-Learning

We know based off the possible states of the cartpole problem, that if we make the right decision, the cartpole will stay upright and balanced. Thus, we can identify the action-state pairs which lead to higher reward in the cartpole system. We can model each pair as a function of the probability of reward: $Reward = Q(s, a)$. In this case, the reward is known as a Q-value. The goal of Q-Learning, a RL algorithm, is to find this function $Q(s, a)$, while applying it iteratively to s' (future state). The initial Q-learning function can be represented as:

$$Q(s, a) = r + \gamma max_{a'} Q(s', a')$$

After obtaining some reward r by making an action a, we can reach the next state: s'. Upon reaching the next state (s'), the agent performs a new action (a') in regard to the reward. The weight we wish to focus on the next reward (r') is γ. Thus, we update the equation as follows:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma max_{a'} Q(s', a') - Q(s, a)]$$

VII. Application of Deep Q-Learning to the Cartpole System

Based on analysis of the elements of Q-learning, it is evident that the cartpole system can be effectively modeled using Q-learning. The cartpole problem has a state space of 4 dimensions of continuous values (θ, l, x, x') and an action space of 2 discrete values (move right or left). However, in typical Q-learning, we must shift our state for every slight change in the angle or position of the cartpole, and this would require extreme memory storage capabilities. Moreover, to apply Q-learning to balancing the cartpole system, we must approximate the model-free function $Q(s, a)$, where the input is a state-action pair (s, a) and the output is some expected reward. This technique of approximating the $Q(s, a)$ function to is known as a Deep Q-Network (DQN) and is more robust against parameter variations and frequent changes in the state. This technique follows the same process behind Q-learning, but rather utilizes a deep neural network to compute $Q(s, a)$ based on a trained network of nodes:

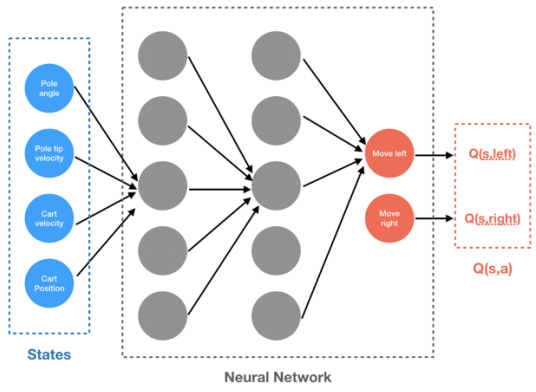


Figure VI. DQN for the cartpole system

As seen in the diagram above, the DQN uses the current states of the cartpole to calculate the expected reward and next action for the cartpole, returning a $Q(s, a)$ for both movement to the right and movement to the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 10 May 2020 doi:10.20944/preprints202005.0181.v1

left. The DQN would most likely need to be supplemented with a loss-function. We know that the updated Q-learning equation already calculates the value for $Q(s,a)$:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma max_{a'} Q(s',a') - Q(s,a)]$$

Thus, it is essential to have a loss function that minimizes the error between the approximation from the DQN and the true $Q(s,a)$ obtained from the equation. In summary, it is best to think of the overall process behind Q-learning and DQN as a "controlled trial and error" that looks to approximate the expected reward: $Q(s,a)$. Q-learning makes use of the updated Q-function which makes iterative adjustments between discrete state-action pairs. DQN seeks to avoid the memory overuse that may occur in Q-learning with near infinite state-action pairs, in favor of a Neural Network that approximates the expected reward from previous continuous state-action pairs.

VII. Training a DQN for the Cartpole System

DQNs are commonly used for the cartpole problem, and we may now understand the implementation of a DQN that maximizes the reward of the cartpole (the reward is the ability for the controller to both balance and control the cartpole). It is first important to note we can summarize the *state-action-reward-state* and environment of the cartpole system as a tuple: (S, A, R, P, ρ), where S is the state, A is the action, R is the reward function, P is the transition probabilities, and ρ is the initial state distribution. The reward function is:

$$r_{t+1} = R(s_t, a_t, s_{t+1})$$

This formulation of the cartpole system as a tuple is known as a Markov Decision Process (MDP). An MDP typically provides us with a method to accurately select an action a_t, given a state s_t. We then observe a_{t+1} and s_{t+1} based on the transition probabilities P. MDPs also provide techniques for helping agents find the optimized policy within the specified environment in the long run. Most implementations of DQNs use flat convolutional neural networks with batch normalization. This technique uses iterative adjustments towards $Q(s,a)$. After a DQN is implemented, it is necessary to train the model. The training of the DQN simply acts as a learning phase for the cartpole system in its environment. The training stage of any RL model is similar to the commonplace example of a child learning to walk. There are a few phases of learning that the cartpole must go through before it can effectively balance both the angle and position of the system: 1.) Learning to balance the pole alone 2.) Staying in bounds 3.) Staying in bounds but unable to balance pole 4.) Staying in bounds while effectively balancing the pole. The ultimate goal is to solve the environment as quickly as possible (solve in the least number of steps/episodes). As the model trains on state-action pairs, it will eventually improve the number of steps needed before solving. The figure below demonstrates a sample training for a DQN Cartpole. As seen in the graph, the cartpole eventually reduces the number of steps needed to solve the environment.

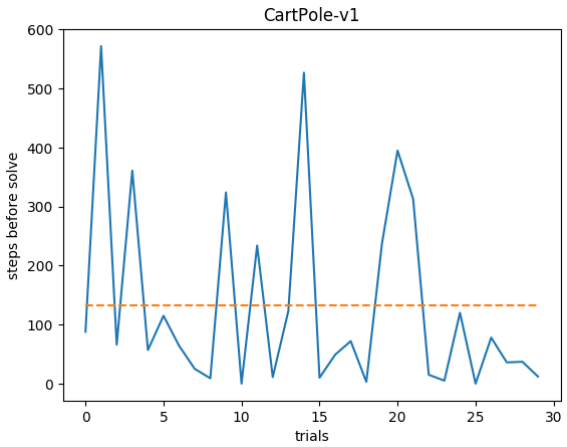


Figure VII. Training steps and trials for Cartpole DQ

VIII. Conclusions

Instead of using a PID controller or LQR controller for managing a cartpole problem, we can apply the earlier discussed RL algorithms to managing the forces and torques felt by the cartpole system. We had successfully derived the equation for a linear cartpole system that did not require swing up: $l\theta'' - gsin(\theta) = x''\cos(\theta)$. Q-learning and DQN can successfully model the trial-error process of trying to balance the cartpole (satisfy the linear equation above). These functions use state-action pairs to both calculate and approximate the $Q(s, a)$ (expected reward) of the specified environment. We can even implement a DQN that trains on MDP state-action pairs to find the optimal solution to the cartpole system. In the future, it may be necessary to investigate a more robust RL solution to the classic cartpole problem; other algorithms of interest include Monte-Carlo simulations, SARSA, and Actor Critic Policy-Gradient. Moreover, it may be of interest to identify a RL algorithm that can incorporate the swing-up of the cartpole system.

References

[1] *Comparison of Reinforcement Learning Algorithms applied to ... (n.d.). Retrieved from https://arxiv.org/pdf/1810.01940.*
[2] *Surma, G. (2019, January 18). Cartpole - Introduction to Reinforcement Learning (DQN - Deep Q-Learning). Retrieved from https://towardsdatascience.com/cartpole-introduction-to-reinforcement-learning-ed0eb5b58288.*
[3] *Kwon, S. (2007). Two Alternate Fuzzy Controllers for Cartpole System. 2007 International Conference on Machine Learning and Cybernetics. doi: 10.1109/icmlc.2007.4370220*
[4] *(n.d.). Retrieved from http://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum§ion.*
[5] *(1995, February 10). Retrieved from http://pages.cs.wisc.edu/~finton/qcontroller.html.*
[6] *(2016, November 15). Cart-Pole Balancing with Q-Learning. Retrieved from https://medium.com/@tuzzer/cart-pole-balancing-with-q-learning-b54c6068d947.*
[7] *IIT Bombay Graduate. (2019, May 6). Introduction to Deep Q-Learning for Reinforcement Learning (in Python). Retrieved from https://www.analyticsvidhya.com/blog/2019/04/introduction-deep-q-learning-python/.*
[8] *Phy, V. (2019, November 4). Reinforcement Learning Concept on Cart-Pole with DQN. Retrieved from https://towardsdatascience.com/reinforcement-learning-concept-on-cart-pole-with-dqn-799105ca670.*
[9] *Rodriguez, J. (2017, August 31). Reinforcement Learning Soup: MDPs, Policy vs. Value Learning, Q-Learning and Deep-Q-Networks. Retrieved from https://medium.com/@jrodthoughts/reinforcement-learning-soup-mdps-policy-vs-value-learning-q-learning-and-deep-q-networks-4ac137acd07.*
[10] *Deep Q-Learning with Keras and Gym. (2017, February 6). Retrieved from https://keon.io/deep-q-learning/.*
[11] *Brunskill, E. (n.d.). Slides on Q-learning and Monte Carlo Models by Stanford University. Stanford.*