Vaibhav Nandkumar Kadam
vkadam@wpi.edu

Pratik Kumbhare
pkumbhare@wpi.edu

Prasanna Natu
pvnatu@wpi.edu

Omkar Bharambe
ogbharambe@wpi.edu

Homework Assignment 5
CS 539 Machine Learning

2022-11-12

1 Introduction

Improve PID controller through reinforcement learning. IEEE 2018.

In this assignment we summarize the above research article. Authors discuss a new technique for tuning the non-trivial PID parametes. They utilize Deep Reinforcement Learning algorithm which a effective learning technique for making decisions to tune the parameters of PID controller. Authors report an improvement over the classical PID controller by training an adaptive PID controller called as DRPID. The controller uses a deep deterministic policy gradient, which is apt for solving continue action control problem.
The authors propose DRPID which basically trains a agent to output K_p, K_d, K_i which stablizes and tunes the controller very fast. The agent gets a reward when the controller performs well with set of parameters provided otherwise it punishes it with penalty thus its trained with deep deterministic policy gradient algorithm to train the agent.

2 Methodology

Algorithm
The authors discuss about two Deep reinforcement Learning algorithm such as "Deep Q Network(DQN)" and "Deep Deterministic policy gradient(DDPG)". However due to continuous nature of the control system the value base method of "DQN" is not capable of giving us the desired output, as a result the author switched to "DDPG" . In this paper the author states that While deciding the control parameter the Deep Reinforcement learning not only consider the current state of the system but also consider past K cases for much better result. The Decision making process is written in equation as:

$$(k_{kp}, k_{kd}, k_{ki}) = a_k = \pi(s_k, s_{k-1}, s_{k-2}, ...s_{k-L}) \tag{1}$$

Hence, the PID control Algorithm can be stated as:

$$u(k) = k_p * error(k) + k_i * \sum_{n=1}^{k} error(i) + k_d * (error(k) - error(k-1)) \tag{2}$$

where,

$$error(k) = input(k) - feedback(k) \tag{3}$$

As we will currently considering k-L state while calculating for the control parameters the authors have modified the integral part as below:

$$u(k) = k_p * error(k) + k_i * \sum_{n=1}^{k-L} error(i) + k_d * (error(k) - error(k-1)) \tag{4}$$

The algorithm for DDPG is For learning the control parameters using DDPG the author set s the critic and actor network along with the weights randomly. This initialisation is followed with Initialising the target network with its weights and relay buffer.
Then we start updating with episode 1 and random process N for exploration. For time t= 1 we calculate the (t-L) states considering it to be 1 except current state, all state can be considered as "0".
Then the author got the control parameter(k_p, k_d, k_i at time t. Then the author could get the control signal using the PID function. This control system are then tested over the simulation and environment for calculating reward and new state This new state create a new sequence and all the transition is stored in R.

Then a Mini batch of random N transition starting from the R to train both the critic and actor network DDPG. Then using this network the hyper parameter are updated. This entire process repeated again if the time t ≤ T and episode ≤ M.

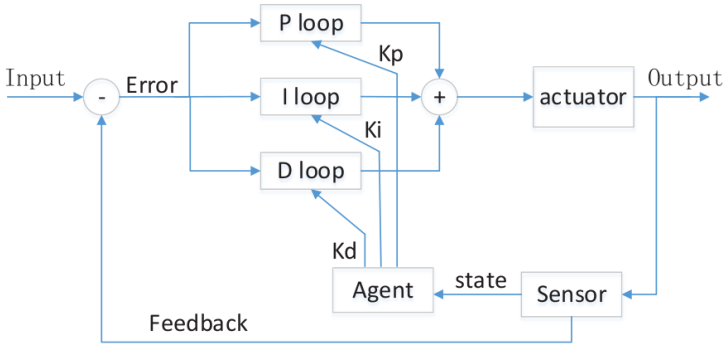

Figure 1: Block Diagram for PID parameter optimisation using DRL

3 Experiment

The algorithm is tested in the inverted pendulum simulation environment of the OpenAI gym module. The simulation consists of a sliding block on which an inverted pendulum is mounted. The objective is to control the sliding of the block left or right to prevent the pendulum from falling down. O is the base point of the horizontal axis, and the coordinate of the sliding block is x, and the angle between the pendulum and the vertical direction is theta. The state of the inverted pendulum contains four elements: x, derivation of x, theta, derivation of theta. The control signal input to the inverted pendulum is an integer: 1 meaning slide the block right and -1 meaning slide the block left.

For the experiment the simulation environment is modified to make it more suitable for our experiment:

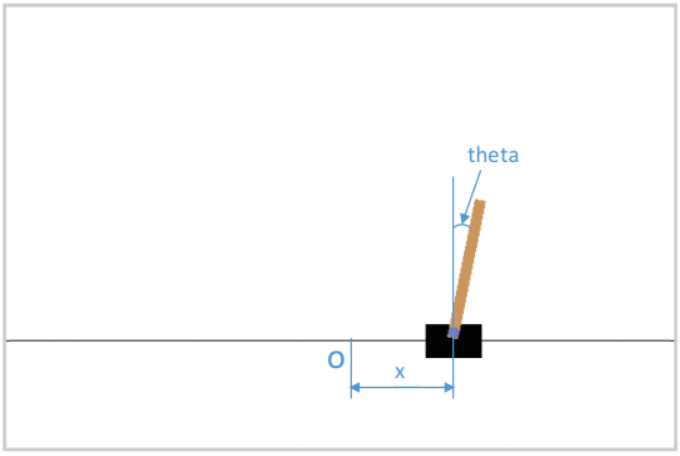

Figure 2: Simulation enviroment setup

1. The simulation environment is modified so that it can accept continuous signals in the range -1 to 1.
2. The reward function is changed to equation [5]:

$$reward = p * (1 - abs(x)) + q * (0.1 - abs(theta)) \tag{5}$$

Here abs(x) is the function that calculates absolute value of x, The parameters p and q are used to build the inverted pendulum's behaviour. If the parameter p is set to a large value, then the agent will focus more on controlling the inverted pendulum to the base point O. Or else if we set q to be a large value the agent will focus on the inverted pendulum to be vertical.

The training data comprises of the transition samples to train the agent. The transition sample memory is used to store the latest M=100000 transition samples. Each transition is formed as

$$s^t_{sequence} = [s_t, s_{t-1}, ..., s_{t-L}], (K_{pt}, K_{it}, K_{dt}) = p_t, r_t, s^{t+1}_{sequence} = [s_{t+1}, s_t, ..., s_{t-L+1}] \tag{6}$$

which means that the agent will output the parameters of

$$p_t \tag{7}$$

according to

$$s^t_{sequence} \tag{8}$$

Then the authors output the control signal by PID and the environment transitions to next state along with a reward return to the agent.

The authors then implement the following neural network (Figure 3):

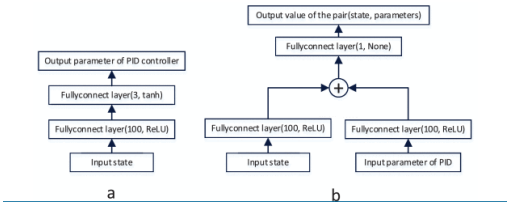

Figure 3: Structure of actor(left) and critic(right) neural network in DDPG

Here, the input to the actor neural network is the state sequence s_k,s_{k1},...,s_{kL}, and the output is the PID controller parameters k_p, k_d and k_i. The inputs to the critic neural network are the state sequence and the controller parameters, both. The output is the value of the pair $v((s_k, , s_{kL}), (k_p, k_i, k_d))$

The authors then train the neural network in DDPG. The optimization algorithm is Adam with a learning rate of 0.001 (actor) and 0.002 (critic) at the beginning. The learning rate decays exponentially. Soft parameter update method is used with $T = 0.01$ to update the parameters of both networks. Maximum steps in one T episode was set to 5000. The values of p and q from equation (5) were kept as 1 and 10.

The performances of different controllers were compared. The controller which held the inverted pendulum with more steps in one episode was determined to be the better controller. The performance comparison is shown in the figure below:

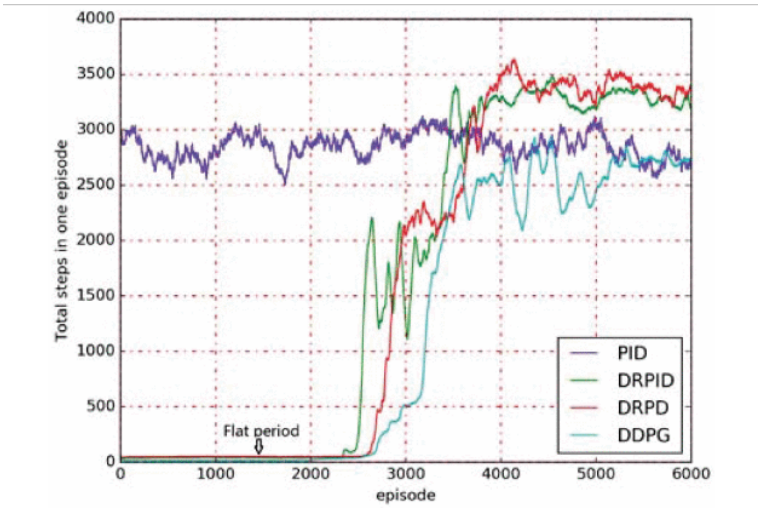

Figure 4: PID vs DRPID vs DRPD vs DDPG

4 Result

The performance of the four controllers was found to be as follows:

$$DRPD > DRPID > PID > DDPG. \tag{9}$$

The authors found out that the DRPD controller outperforms the DRPID controller. The authors speculate that the DRPD controller with adaptive parameters k_p and k_d is sufficient to control the inverted pendulum well, the integral loop in the controller may bring unnecessary control signals and cause overfitting.

5 Conclusion

The authors conclude that the classical PID controller can be improved immensely by integrating it with DRL algorithms. However the authors state that the performance of DRPID is not ready to be implemented in the real world, as the DRPID has to be trained first and the performance at the beginning to too poor. The authors propose to solve this problem by using a classical PID to train a weak DRPID controller first by imitation learning process. After imitation when the DRPID performs just well enough to control the actuator directly and then the weak DRPID will be trained with reinforcement learning algorithm simultaneously and eventually outperform PID, thus becoming a strong DRPID controller.
One of the strong points of this paper is the simplicity and clarity with which the DRPID and other algorithms are explained. However the DRPID algorithm standalone is found to be impractical to use in real life scenarios as the performance is not up to the mark. The authors should have tried to continue the research by proposing further improvements to the DRPID controller so as to make it practical to use in real life scenarios.

Literature

[1] Y. Qin, W. Zhang, J. Shi and J. Liu, ”Improve PID controller through reinforcement learning,” 2018 IEEE CSAA Guidance, Navigation and Control Conference (CGNCC), 2018, pp. 1-6, doi: 10.1109/GNCC42960.2018.9019095.