DEEP REINFORCEMENT LEARNING FOR PID CONTROLLER TUNING

Kuang-Hung Liu[1*], Thomas A. Badgwell[1] and Michael H. Kovalski[2]
[1]ExxonMobil Research & Engineering
[2]ExxonMobil Information Technology
Clinton, NJ 08809

Abstract Overview

We present an application of deep reinforcement learning (DRL) technology for the problem of tuning Proportional-Integral-Derivative (PID) controllers. The DRL agent adjusts the PID tuning parameters in order to minimize the root-mean squared tracking error. Rather than postulating relevant features for the evaluation of the controller state we show that it is only necessary to provide raw values of the manipulated variable, controlled variable, and set-point as feedback to the agent.

Keywords

Deep reinforcement learning, PID controller, Actor-critic, Continuous control, Deep neural network, Convolutional neural network.

Introduction

Deep Reinforcement Learning (DRL) has gained widespread attention with the spectacular success of Google DeepMind's AlphaGo technology (Silver et al., 2016). While Reinforcement Learning theory and applications have progressed steadily since the 1990's (Sutton and Barto, 2018), the recent employment of Deep Neural Networks for function approximation makes it possible for DRL agents to extract relevant features from data without supervision (Levine, 2017), significantly widening the range of possible applications. While it is clear that DRL agents will not replace standard control algorithms, it is only natural to consider how this emerging technology may prove useful for optimization and control of chemical processes.

Here we present an application of DRL technology for the problem of tuning Proportional-Integral-Derivative (PID) controllers (Astrom and Hagglund, 1994). The DRL agent adjusts the PID tuning parameters in order to minimize the root-mean squared tracking error. Rather than postulating relevant features for the evaluation of the controller state we show that it is only necessary to provide raw values of the manipulated variable, controlled variable, and set-point as feedback to the agent.

Our work differs in two important ways from previous efforts to use Reinforcement Learning (RL) to tune PID controllers (el Hakim et al., 2013) (Sedighizadeh and Rezazadeh, 2008): (1) we use DRL so that we can avoid feature engineering, and (2) we compute incremental adjustments to the tuning parameters, rather than full values, leading to a more robust tuning policy.

PID Controller Tuning as a Reinforcement Learning Problem

Figure 1 shows a typical single-input, single-output control application (heater outlet temperature control) in which a PID controller adjusts a manipulated variable (MV) (fuel gas flow in Fig. 1) to keep a controlled variable (CV) (outlet temperature in Fig. 1) close to its setpoint (SP) (desired outlet temperature in Fig. 1). A PID typical controller measures the tracking error $e(t)$, defined as the

* To whom all correspondence should be addressed

difference between SP and CV, and applies a correction to the MV that depends on three tuning factors: a proportional gain (K_C), a reset time (τ_I), and a derivative time (τ_D).

$$mv_t = K_c \left[e_t + (\Delta t/\tau_I) \sum_{j=0}^{\infty} e_{t-j} + (\tau_D/\Delta t)(e_t - e_{t-1}) \right] \quad (1)$$

Here Δt is the sample interval. The values of the three tuning factors K_C, τ_I, and τ_D must be set correctly for any given process in order to achieve good PID control performance (Astrom and Hagglund, 1994), and they must be periodically updated as the process changes over time.

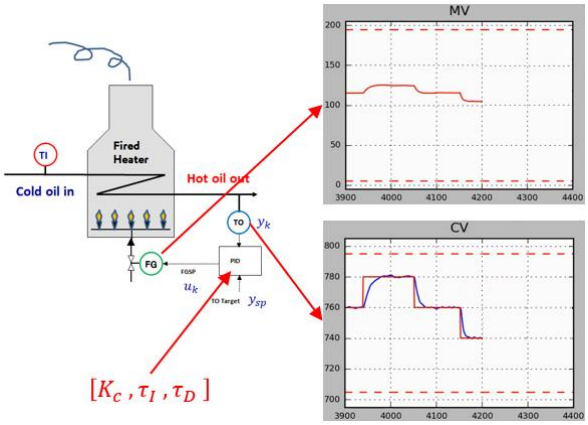


Figure 1. An example of PID controller controlling a fire heater

In this work, we viewed the PID controller tuning task as a reinforcement learning (RL) problem where an agent interacts with a PID control environment in discrete time steps. At each time step t the agent receives an observation s_t, takes an action a_t and receives a scalar reward $r(s_t, a_t)$. An agent's behavior is defined by a policy, π, which maps states to the actions, $\pi: S \rightarrow A$. The action-value function $Q_\pi(s_t, a_t)$ is the expected future accumulated reward after taking an action a_t in state s_t and then following the policy π afterwards (Sutton and Barto, 2018):

$$Q_\pi(s_t, a_t) = E\left[r(s_t, a_t) + \sum_{i=1}^{\infty} \gamma^i r(s_{t+i}, \pi(s_{t+i})) \right], \quad (2)$$

where γ is a discounting factor $\gamma \in [0, 1]$. The goal of reinforcement learning is to learn a policy which maximizes $Q_\pi(s_t, a_t)$ for any s_t and a_t. In the subsequent section we will discuss how we define state, action, and reward for the DRL adaptive PID tuning as well as briefly describe our DRL training process.

Training a Reinforcement Learning Agent for PID Tuning

In this application, after the DRL agent takes an action a_t to adjust the PID tuning parameters, we let the PID control execute for 200 samples before running the DRL agent again for the next adjustment a_{t+1}. When it is time to run, the state s_t that we pass to the agent is a collection of the 200 MV, CV, SP values since the previous adjustment, denoted as $\mathbf{mv_t}$, $\mathbf{cv_t}$, $\mathbf{sp_t}$, respectively. In addition, we also include the current PID tuning parameters $[K_{C_t}, \tau_{I_t}, \tau_{D_t}]$ in the state vector s_t:

$$s_t = [\mathbf{mv_t}, \mathbf{cv_t}, \mathbf{sp_t}, K_{C_t}, \tau_{I_t}, \tau_{D_t}] \quad (3)$$

Next, we define the action a_t taken by DRL agent as a set of multiplication factors to apply to the current PID tuning parameters. More specifically, $a_t = [a_t^1, a_t^2, a_t^3]$ with $a_t^1, a_t^2, a_t^3 \in [-1, 1]$ and

$$\begin{aligned} K_{C_{t+1}} &= K_{C_t}(1 + 0.5 * a_t^1) \\ \tau_{I_{t+1}} &= \tau_{I_t}(1 + 0.5 * a_t^2) \\ \tau_{D_{t+1}} &= \tau_{D_t}(1 + 0.5 * a_t^3) \end{aligned} \quad (4)$$

Let $\mathbf{e_t}$ denote the 200 tracking error measurements taken between time steps $t-1$ and t. We define the reward as the negative of the root-mean squared tracking error plus a penalty on the actions taken (promote efficient tuning):

$$r(s_t, a_t) = -(\|\mathbf{e_t}\|_2 + 0.01 * \|a_t\|_1) \quad (5)$$

Using our definition of state, action, and reward, we proceed to train a deep reinforcement learning agent on a simulated PID controller, with randomly generated setpoint changes, using the deep deterministic policy gradient (DDPG) algorithm (Lillicrap et at., 2015). DDPG is an actor-critic method that simultaneously learns both the action-value function and the policy function. Here both functions are approximated by a deep neural network. An illustration of our neural network architecture is shown in Figure 2. The first part of our architecture consists of multiple layers of convolutional neural networks (CNN) that are designed for efficient feature extraction from the raw sensor measurements. The second part of the architecture are two diverging sets of fully-connected layers that represent the policy function (with action output in Fig. 2) and action-value function (with value output in Fig. 2). Note that the first part of CNNs are shared between policy network and action-value network. This design choice is based on our observation that it is possible for a common set of features extracted from raw data using CNN to sufficiently determine both action-value function and policy function. We trained the entire network from scratch end-to-end without feature engineering.

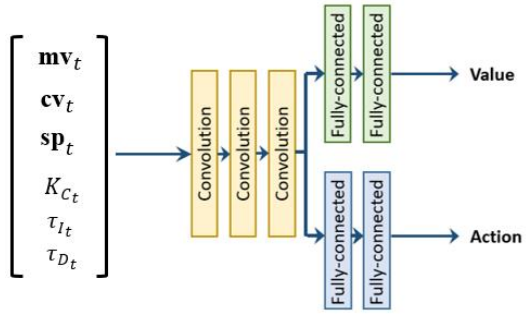


Figure 2. Deep neural network architecture for policy and action-value function.

We successfully trained DRL PID tuning agents to tune PID controllers for two different types of process dynamics: a second-order deadtime process (SODP), representative of many stable processes such as the heater outlet temperature control problem in Figure 1, and a pure integrating process (IP), which is representative of many tank level control problems. Training converged for the SODP example in roughly 500k iterations (encounters with the environment), and for the IP example in roughly 400k iterations.

A snapshot of a trained DRL agent tuning the SODP is shown in Figure 3. From Fig. 3(a)-(b), we can see that our trained DRL agent is able to tune the PID so that the CV tracks the SP closely. To quantitatively evaluate DRL agents' performance, we calculated the optimum constant tuning parameters for this process by direct dynamic optimation and obtained $Kc^* = 0.6129$, $\tau_I{}^* = 10.3614$, and $\tau_D{}^* = 7.0618$. As can be seen from Fig. 3(c)-(e), our DRL agents' policy found parameters near these, but chooses to continuously change them. By making continuous changes it functions effectively as a nonlinear controller, and so comparisons with the optimal constant tuning parameters are difficult to make. But it is clear from the good tracking performance that it is doing a respectable job of choosing tuning parameters. A snapshot of our trained agent in action on the IP process is shown in Figure 4. Again, seen from Fig. 4(a)-(b), our trained DRL agent is able to tune the PID so that the CV tracks the SP very closely. The optimal tuning parameters for this case are calculated as $K_C{}^* = 0.99$, $\tau_I{}^* = 1000$, and $\tau_D{}^* = 0$. From Fig. 4(c)-(e), we see that our DRL agent correctly set the K_C and τ_D terms while τ_I term is set to an unexpected value. Again, however, the small tracking error is evidence that these are acceptable values for the tuning parameters.

In summary, we have applied deep reinforcement learning to the problem of tuning PID controllers. We show that rather than postulating relevant features for the evaluation of the controller state as others have done, it is only necessary to provide raw values of the manipulated variable, controlled variable, and set-point as feedback to the reinforcement learning agent. Our future plans include evaluating the merits of continuously adjusting the PID tuning parameters, and development of a master PID tuning policy that can handle a wide variety of process dynamics and disturbances.

References

Astrom, K., Hagglund, T.,(1994), "PID Controllers: Theory, Design, and Tuning", ISA.

el Hakim, A., Hindersah H., Rijanto, E., (2013), "Application of Reinforcement Learning on Self-Tuning PID Controller for Soccer Robot Multi-Agent System," 2013 Joint International Conference on Rural Communication Technology and Electric-Vehicle Technology (rICT&IceV-T), November 2-28, Bandung-Bali, Indonesia.

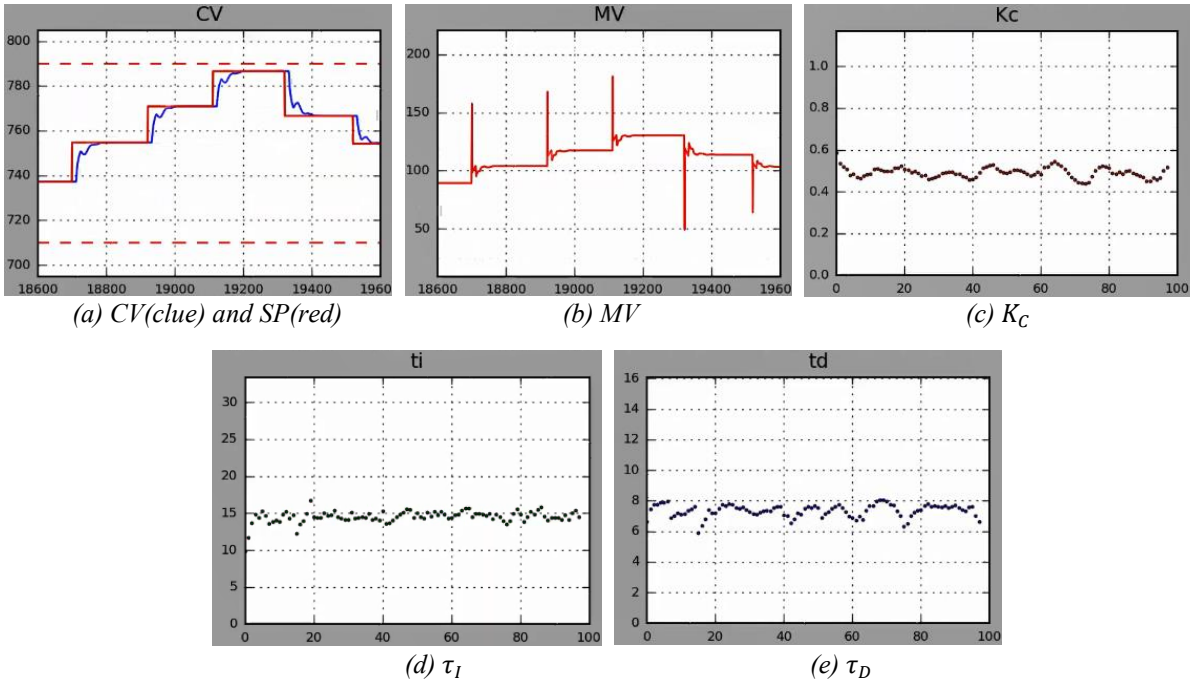
Levine, S., (2017). Deep Reinforcement Learning, Berkeley CS294-112.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2015). Continuous control with deep reinforcement learning. CoRR, abs/1509.02971.

Sedighizadeh, M., Rezazadeh, A., (2008), "Adaptive PID Controller based on Reinforcement Learning for Wind Turbine Control," World Academy of Science, Engineering and Technology, 37, 257-262.

Silver, D., Huang, A., Maddison, C., Guez, A., Sifre, L., Van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lancot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I. Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D., (2016). Mastering the game of Go with deep neural networks and tree search, Nature, 529, 484-489.

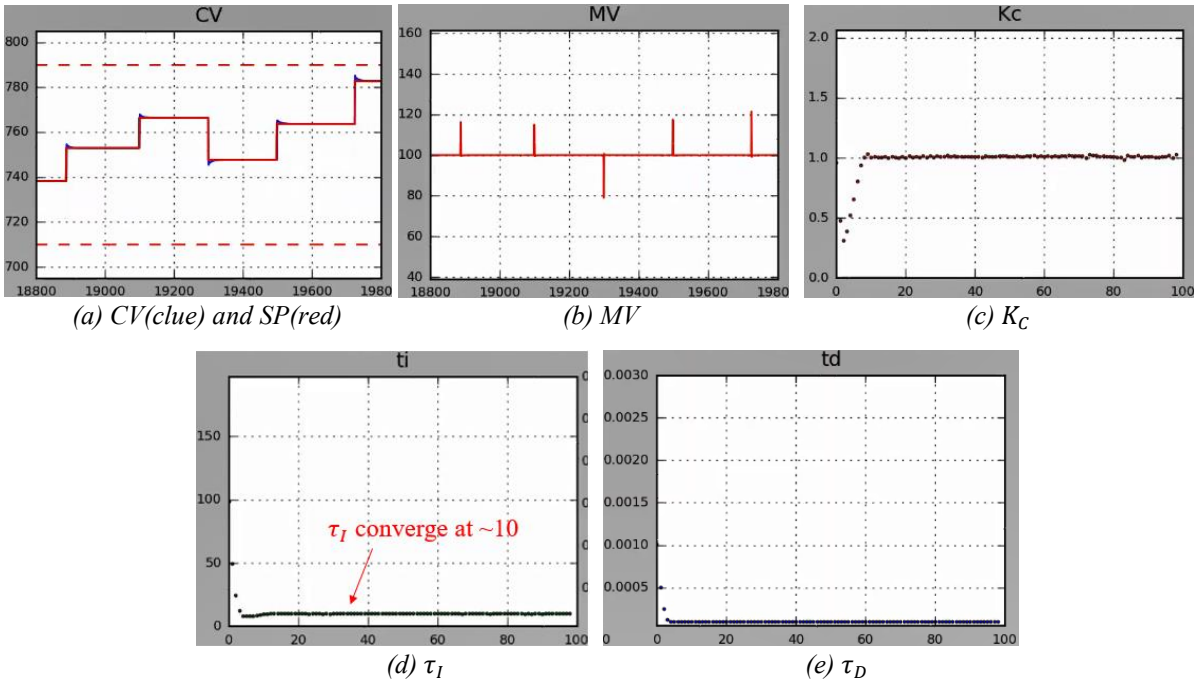
Sutton, R., Barto, A., (2018). Reinforcement Learning: An Introduction, Second Edition, MIT Press.



(a) CV(clue) and SP(red) *(b) MV* *(c) K_C*

(d) τ_I *(e) τ_D*

Figure 3. Trained DRL agent tuning a second-order dead time process (SODP) $\left[\frac{0.3e^{-10s}}{(1+5s)(1+5s)}\right]$; (a) CV (blue) and SP (red); (b) MV; (c)-(e)K_C, τ_I, τ_D value determined by a trained DRL agent at each time-step.



(a) CV(clue) and SP(red) *(b) MV* *(c) K_C*

(d) τ_I *(e) τ_D*

Figure 4. Trained DRL agent tuning an integrating process (IP) $\left[\frac{1}{s}\right]$; (a) CV (blue) and SP (red); (b) MV; (c)-(e)K_C, τ_I, τ_D value determined by a trained DRL agent at each time-step.