DRL para controle de um levitador magnético

Gabriel Evangelista

September 17, 2022

Ponto de Partida

Deep Reinforcement Learning for Process Control: A Primer for Beginners

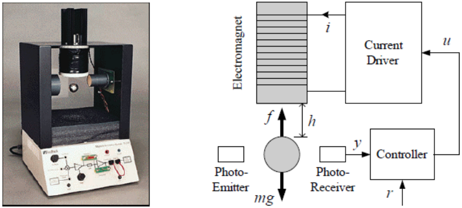
Steven Spielberg[a], Aditya Tulsyan[a], Nathan P. Lawrence[b], Philip D Loewen[b], R. Bhushan Gopaluni[a,*]

[a]*Department of Chemical and Biological Engineering, University of British Columbia, Vancouver, BC V6T 1Z3, Canada.*
[b]*Department of Mathematics, University of British Columbia, Vancouver, BC V6T 1Z2, Canada.*

Abstract

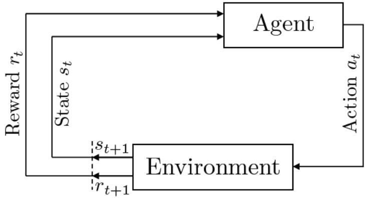
Advanced model-based controllers are well established in process industries. However, such controllers require regular maintenance to maintain acceptable performance. It is a common practice to monitor controller performance continuously and to initiate a remedial model re-identification procedure in the event of performance degradation. Such procedures are typically complicated and resource-intensive, and they often cause costly interruptions to normal operations. In this paper, we exploit recent developments in reinforcement learning and deep learning to develop a novel adaptive, model-free controller for general discrete-time processes. The DRL controller we propose is a data-based controller that learns the control policy in real time by merely interacting with the process. The effectiveness and benefits of the DRL controller are demonstrated through many simulations.

Keywords: process control; model-free learning; reinforcement learning; deep learning; actor-critic networks

(a)


(b)

Reinforcement Learning



$$R_t = \sum_{i=t}^{T} \gamma^{(i-t)} r(s_i, a_i)$$

$$J = \mathbb{E}_{s_i, a_i \sim \pi}[R_1]$$

Política: π (ou μ quando determinística).

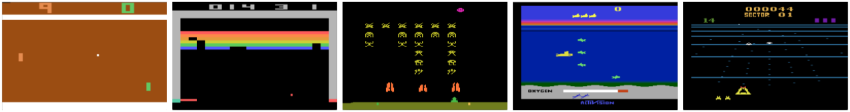
Equação de Bellman:

$$Q^{\mu}(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim E}[r(s_t, a_t) + \gamma Q^{\mu}(s_{t+1}, \mu(s_{t+1}))]$$

Onde, usualmente $\mu = argmax_a Q(s_t, a_t)$
Ideia chave: construir um estimador para Q, com parâmetros θ
A perda é dada por:

$$L(\theta^Q) = \mathbb{E}[(r + \gamma argmax_a Q(s_{t+1}, a_{t+1})|\theta - Q(s_t, a_t)|\theta)^2]$$

DQN



Inovações principais:

- Target network:

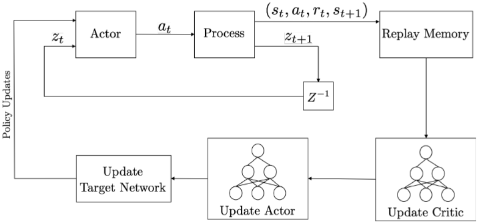
$$L(\theta^Q) = \mathbb{E}[(r + \gamma argmax_a Q_{tgt}(s_{t+1}, a_{t+1})|\theta - Q(s_t, a_t)|\theta)^2]$$

- Memory Buffer: Armazenamento e amostragem de transições. Amostra vai ser mais descorrelacionada e promove um aprendizado melhor do que o as amostras consecutivas.

DDPG

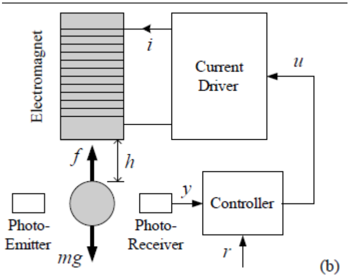
- Usa as técnicas do DQN para implementar a atualização da rede Q
- Atualização contínua com uma média: $\theta_{tgt} \leftarrow \tau\theta + (1 - \tau)\theta_{tgt}$ com $\tau << 1$
- Emprega a normalização dos valores
- Ator crítico: Uma rede para gerar as ações e outra para realizar a estimativa do Q
- Aprendizado pelo gradiente descendente de J a partir de valores de Q para atualizar π.
- **Ações contínuas!**

DRL Controller



- Baseado no DDPG
- Utiliza uma técnica de *gradient clip*
- Adiciona a variável de referência no algoritmo
- Relaciona as abordagens dos problemas:
  $a_t = f^{-1}(u_t)$
  $s_t =< y_t, y_{t-1}, ...y_{t-d}, a_t, a_{t-1}, ..., a_{t-d}, (y_t - y_{sp}) >$
  $r_t = -|y_t - y_{sp}|$, ou outra que se adapte ao problema, em função de y_t e y_{sp}.

A planta estudada



$$f = K\frac{i^2}{h^2}$$

$$m\frac{d^2h}{dt^2} = mg - K\frac{i^2}{h^2}$$

$$y = \gamma h + y_0, y \in (-2V, 2V)$$

$$i = \rho u + i_0, u \in (-3V, 5V)$$

$$\boxed{\frac{d^2y}{dt^2} = \gamma g - \frac{K(\rho u + i_0)^2\gamma^3}{m(y - y_0)^2}}$$

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = \gamma g - \frac{K(\rho u + i_0)^2\gamma^3}{m(y - y_0)^2}$$

$$s_t = < (y_t - ref), y_t, dy_t, u_t >$$

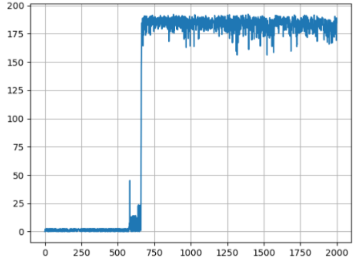
$$u_t = u_{t-1} + a_t$$

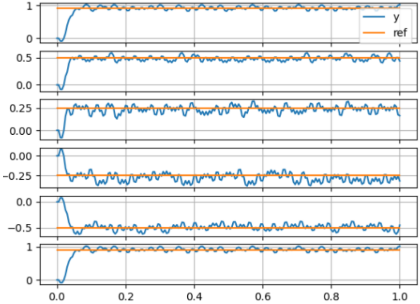
$$ref \rightarrow Const$$

$$ref \in (-0.5, 0.5), run-by-run$$

$$\boxed{rwd = -\exp(-e^2/0.05)}$$

Erros quadráticos médios (0.0133):
0.0256,
0.0090,
0.0037,
0.0053,
0.0103,

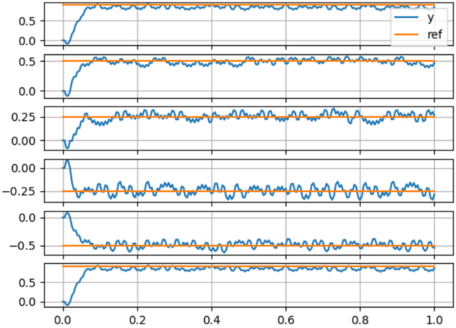




Resultado na Planta Proposta

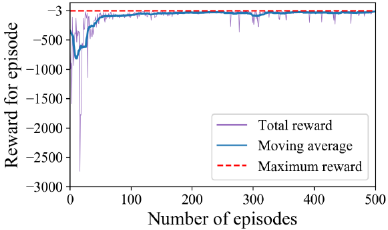
$$\boxed{rwd = -\exp(-e^2/0.05)\text{-}0.75\Delta u^2}$$

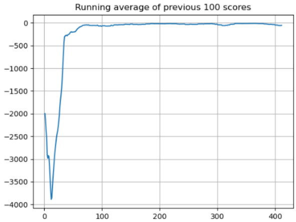
Erros quadráticos médios (0.0152):
0.0308,
0.0095,
0.0042,
0.0044,
0.0111,
0.0308

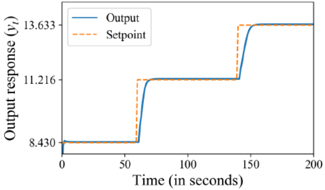


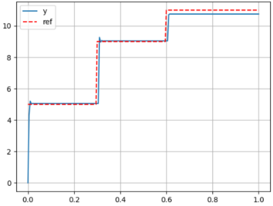
Reprodução do Resultado Original

$$G(z) = \frac{0.05z^{-1}}{1 - 0.6z^{-1}}$$









Conclusão

- As funções de recompensa apresentadas no artigo original não tratam bem problemas instáveis. A instabilidade gera o "morte prematura" do agente caso não hajam recompensas positivas para incrementos do número de passos dados.
- Apesar da solução funcionar bem nas plantas apresentadas no artigo original, houve dificuldade do emprego da técnica em uma planta instável.