台中，中華民國 111 年 11 月 05 日
Taichung, November 5th, 2022
論文編號：YY-XX

強化學習應用於無人機姿態控制

Apply reinforcement learning for UAV attitude control

吳柏勳 [a]、蕭富元 [b]
[a, b] 淡江大學航空太空工程學系

Wu, Po-Hsun[a], Hsiao, Fu-Yuen[b]
[a, b]Department of Aerospace Engineering, Tamkung University

摘要

本研究採用強化學習實現無人機的姿態控制，利用 OpenAI GYM package 建立強化學習的環境後，再使用強化學習演算法與 Python/Tensorflow 對環境進行學習。

關鍵字：無人飛行載具、強化學習、OpenAI GYM、Python/Tensorflow

一、緒論

1.1 研究動機

近年來機器學習技術日漸成熟和電腦運算速度的提升，機器學習開始大量的應用於影像辨識、自然語言處理、文本分析…等領域，透過大量的訓練資料和機器學習演算法來訓練決策使其達成我們所希望達到的目標。

而在控制領域通常都需要將非線性的模型線性化後，再運用 PID 或 LQR…等方法來設計出控制器，而設計出的控制器也會因為線性化模型的緣故，在遠離平衡點時，容易與真實狀況不符合導致系統無法有效達成目標。

若我們能夠利用機器學習演算法針對非線性的數學模型於電腦上進行大量模擬來訓練決策，即可得到一個以神經網路為基礎的控制器，也因為在訓練的過程中使用的模型是非線性的，所以在狀態遠離平衡點後就使控制器無法有效的達到目標。

1.2 文獻回顧

[1] [2] [3]

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

1.3 研究方法

本研究是利用強化學習演算法來訓練決策使其可以有效的控制無人機的姿態，首先利用 Python/Tensorflow 建立一個神經網路來作為產生動作的決策，再來利用 OpenAI GYM 和四階 Runge-Kutta 法來進行強化學習環境的建構與無人機的動態模擬，最後利用強化學習演算法蒐集決策與環境間互動的資料進行計算，更新神經網路的參數來最佳化獎勵函數來達到控制無人機姿態的目標。

二、強化學習

2.1 介紹

強化學習 (Reinforcement learning, RL) 屬於機器學習的一種，與其他機器學習方法不同的是，強化學習是基於與環境 (Environment) 進行互動獲得獎勵 (Reward) 來改善決策 (Policy) 最終得到一個能夠最大化獎勵函數的決策。

如圖 1，強化學習主要由決策、環境和演算法組成，決策會從環境中獲得狀態 (State) 後，根據不同的狀態反饋給環境不同的動作 (Action)，而環境得到決策所給予的動作後，也會計算出下一個狀態和獎勵值給予決策，而演算法會去蒐集每個時間下的狀態、動作和獎勵值，藉由獎勵值的大小來改變決策，獎勵值大的動作會使決策增加該動作的出現機率，反之，獎勵值小的動作則會減小決策執行該動作的機率，藉由大量的學習的過程使每次的動作都能產生最大的獎勵值。

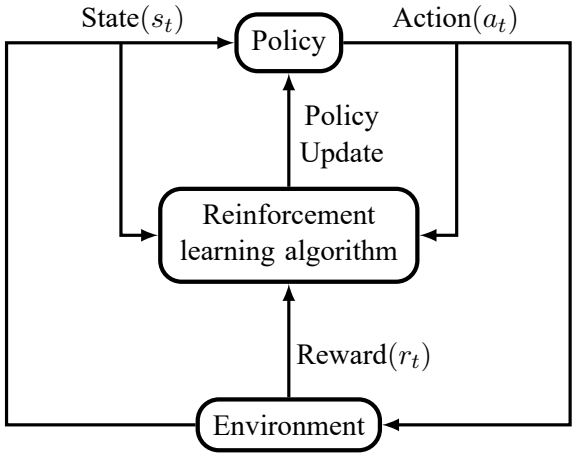

圖 1: 強化學習架構

2.2 PPO 演算法

PPO 演算法 (Proximal Policy Optimization alogrithm) 是由 OpenAI 於 2017 年提出的演算法，PPO 是基於 TRPO 演算法 (Trust region policy optimization) 改善而來，經文獻 [4] 證實 PPO 演算法具有 TRPO 的強健性，而在文獻 [4] 的結果裡 PPO 演算法比起其他種強化學習的演算法具有更好的整體性能。

在 TRPO 中，將最佳化的目標函數定義為式 (1)，而約束條件定義為式 (2)

$$\max_{\theta} \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t \right] \tag{1}$$

$$\text{subject to } \hat{\mathbb{E}}_t \left[\text{KL} \left[\pi_{\theta_{old}}(\cdot|s_t),\, \pi_\theta(\cdot|s_t) \right] \right] \leq \delta \tag{2}$$

在式 (1) 中，$\hat{\mathbb{E}}_t[\cdot]$ 代表某個路徑上的期望值平均函數，$\hat{A}_t$ 則是在某時間的優勢函數表示為式 (3)

$$\begin{aligned} &\hat{A}_t = \delta_t + \gamma\delta_{t+1} + \cdots + \gamma^{T-t+1}\delta_{T-1} \\ &\text{where } \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t) \end{aligned} \tag{3}$$

其中 r_t 為在某時間下的獎勵值；而在式 (2) 中，KL$[\cdot]$ 是 KL 散度 (Kullback-Leibler divergence) 代表當前的決策和舊的決策之間的差異必須小於 δ。

因為 TRPO 演算法中的約束條件為強約束條件，再加上 KL 散度難以估算，使 TRPO 演算法需要消耗更多的時間才能有較好的結果，所以在文獻 [4] 中提出將最佳化問題轉換為式 (4)

$$\begin{aligned} \min_{\theta} \hat{\mathbb{E}}_t [&\min(k_t(\theta)\hat{A}_t, \\ &\text{clip}(k_t(\theta),\, 1-\epsilon,\, 1+\epsilon)\hat{A}_t)] \end{aligned} \tag{4}$$

其中 $k_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$，而 clip 函數限制了機率比值的上下限在 $[1-\epsilon, 1+\epsilon]$ 之間，使其簡化式 (2) 避免計算 KL 散度又可以限制新舊決策間的差異，使 PPO 演算法可以具有 TRPO 的特性又可以強化收斂時的效率。

2.3 獎勵函數

在強化學習中獎勵函數的設計佔據了很重要的一部份，獎勵函數若設計的好可以使訓練出的決策有效的達成目標，而獎勵函數的目的是將決策與環境的互動狀況用數學函數來表達，好的動作則給予較高的獎勵，使演算法在更新決策時會使決策增加好的動作出現的機率。

在本研究中參考了 GYM 預設環境的獎勵函數的定義方式，該函數是定義了狀態的上下限，若決策可以將狀態維持在某個區間內則可以獲得獎勵，反之，若決策的動作會使狀態離開該區間，則在離開該區間後則無法獲得獎勵，表示為數學形式如式 (5)

$$R(t) = \begin{cases} 1, & \text{if } -x_{\text{bound}} \leq x(t) \leq x_{\text{bound}} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

三、環境與建模與訓練

3.1 無人機建模

在本研究中是使用文獻 [3] 中的橫向向動態方程式進行初步的驗證，如式 (6)

$$\begin{bmatrix} \Delta\dot{\beta} \\ \Delta\dot{p} \\ \Delta\dot{r} \\ \Delta\dot{\phi} \end{bmatrix} = \begin{bmatrix} -0.3981 & -0.0144 & -0.9626 & 0.9307 \\ -24.9043 & -10.2554 & 3.2605 & 0 \\ 7.5320 & -0.1847 & -0.7085 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta\beta \\ \Delta p \\ \Delta r \\ \Delta\phi \end{bmatrix} + \begin{bmatrix} 0 & 0.0080 \\ -2.3713 & 0.0699 \\ 0.1870 & -0.5106 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta\delta_a \\ \Delta\delta_r \end{bmatrix} \tag{6}$$

其中 β 和 ϕ 的單位為 rad，p 和 r 的單位為 rad/s，而 $\Delta\delta_a$ 和 $\Delta\delta_r$ 的單位為 deg，

3.2 OpenAI GYM 環境

OpenAI GYM 是一個廣泛應用於建構強化學習環境的 API，OpenAI GYM 提供一個標準化的環境架構給開發者進行強化學習環境的建構，使在學習的期間能夠更穩定的進行模擬與學習。

OpenAI GYM 主要由兩個函數組成，一個是 Step 函數，該函數是將動作 (Action) 作為輸入後，經過動態方程式和獎勵函數計算後，將下一個狀態 (State)、該動作的獎勵 (Reward)、是否終止 (Done) 和其他資訊 (Info) 四者作為輸出，另一個則是 Reset 函數，該函數是將所有先前模擬的變數進行初始化後，生成一個隨機的初始狀態後，將該狀態進行輸出作為該次模擬的起始。

3.3 訓練結果

四、結論

參考文獻

[1] W. Koch, “Flight controller synthesis via deep reinforcement learning,” *CoRR*, vol. abs/1909.06493, 2019.

[2] W. Koch, R. Mancuso, R. West, and A. Bestavros, “Reinforcement learning for UAV attitude control,” *CoRR*, vol. abs/1804.04154, 2018.

[3] 廖晉揚, “定翼無人機的最佳順滑模態控制,” M.S. thesis, 淡江大學航空太空工程學系碩士班, 2022.

[4] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *CoRR*, vol. abs/1707.06347, 2017.