Deep Reinforcement Learning for Automated Stock Trading

113352030 金碩一 胡益鳴
113352032 金碩一 翁晟睿
113352019 金碩一 黃得晉
114352018 金碩一 楊成豐
114363013 企碩一 李姵琪

文章摘要及來源

- 摘要:
  - 用不同的演算法(A2C, PPO, DDPG)訓練三種不同的RL agent, 然後用這三個agent做股票交易並看績效

- 來源:
  - Yang, H., Liu, X.-Y., Zhong, S., & Walid, A. (2020). Deep Reinforcement Learning for Automated Stock Trading: An Ensemble Strategy. Proceedings of the First ACM International Conference on AI in Finance (ICAIF ’20), October 15–16, 2020, Manhattan, NY, USA. ACM, New York, NY, USA.

目錄

Actor-Critic架構

(1) Actor決定動作 S_t: state, P_t: stock price, H_t:shares, B_t: balance, c_t: cost rate, r_t:return

$$S_t = \begin{bmatrix} P_t \\ H_t \\ B_t \end{bmatrix} \longrightarrow \boxed{\begin{matrix}\textbf{Actor} \\ a_t \sim N(\mu, \sigma)\end{matrix}} \longrightarrow a_t \longrightarrow \begin{matrix}\Delta H_t = a_t \times 100 \\ \Delta B_t = -P_t \Delta H_t\end{matrix} \longrightarrow S_{t+1} = \begin{bmatrix} P_{t+1} \\ H_{t+1} \\ B_{t+1} \end{bmatrix}$$

(2) Critic評估Actor動作之成效

S_t、S_{t+1} → **Critic** →

$$\begin{aligned} r_t &= change\ of\ portfolio\ value - cost \\ &= (B_{t+1} + P_{t+1}H_{t+1}) - (B_t + P_t H_t) - P_t \Delta H_t c_t \\ V(S_t) &= E[r_t | S_t] \\ V(S_{t+1}) &= E[r_{t+1} | S_{t+1}] \end{aligned}$$

→ Advantage function(A)

$$A = r_t + \gamma V(S_{t+1}) - V(S_t)$$

γ: discount factor

(3) Actor更新

$$\min L_{actor}(\theta) \longrightarrow \nabla_\theta L_{actor}(\theta) \longrightarrow \theta \leftarrow \theta - \alpha \nabla_\theta L_{actor}(\theta)\ ,\ \alpha\text{: learning rate}$$

(4) Critic更新

$$\min L_{critic}(\theta) \longrightarrow \nabla_\theta L_{critic}(\theta) \longrightarrow \theta \leftarrow \theta - \alpha \nabla_\theta L_{critic}(\theta),\ \alpha\text{: learning rate}$$

A2C(Advantage Actor-Critic)

Actor目標式: $\max\ J(\theta) = E\left[\sum_{t=1}^{T} \mathrm{GAE}_t \times \ln f(a_t|S_t)\right]$

$$\text{其中}\mathrm{GAE}_t = \sum_{l=0}^{\infty} (\gamma\lambda)^l A_{t+l}\,,\ A_t = r_t + \gamma V(S_{t+1}) - V(S_t)$$

Critic目標式: $\min L_{critic}(\theta^V) = [r_t + \gamma V(S_{t+1}) - V(S_t)\ ]^2$

A2C範例

(1) Actor決定動作　S_t: state, P_t: stock price, H_t:shares, B_t: balance, c_t: cost rate, r_t:return

$$S_t = \begin{bmatrix} P_t = 10 \\ H_t = 0 \\ B_t = 1000 \end{bmatrix} \rightarrow \boxed{\begin{matrix}\textbf{Actor} \\ a_t \sim N(0,1)\end{matrix}} \rightarrow a_t = 0.5 \rightarrow \begin{matrix} \Delta H_t = a_t \times 100 \\ = 0.5 \times 100 \\ = 50 \\ \Delta B_t = -P_t \Delta H_t \\ = -10 \times 50 = -500 \end{matrix} \rightarrow S_{t+1} = \begin{bmatrix} P_{t+1} = 10.2 \\ H_{t+1} = 50 \\ B_{t+1} = 500 \end{bmatrix}$$

(2) Critic評估Actor動作之成效

$$S_t \text{、} S_{t+1} \rightarrow \boxed{\textbf{Critic}} \rightarrow \begin{matrix} r_t = change\ of\ portfolio\ value - cost \\ = (B_{t+1} + P_{t+1}H_{t+1}) - (B_t + P_t H_t) - P_t \Delta H_t c_t \\ = (500 + 10.2 \times 50) - (1000 + 0) - 500 \times 0.1\% \\ = 9.5 \\ V(S_t) = E[r_t | S_t] = 1000 \\ V(S_{t+1}) = E[r_{t+1} | S_{t+1}] = 1050 \end{matrix} \rightarrow \boxed{\begin{matrix}\text{Advantage function(A)} \\ A_t = r_t + \gamma V(S_{t+1}) - V(S_t) \\ = 9.5 + 0.99 \times 1050 - 1000 \\ = 49 > 0\end{matrix}}$$

(3) Actor更新

$$max\ J(\theta) = E\left[\sum_{t=1}^{T} A_t \times \ln f(a_t | S_t)\right] \rightarrow \begin{matrix} \nabla_\mu J(\theta) = A \times \frac{a_t - \mu}{\sigma^2} \\ = 49 \times \frac{0.5 - 0}{1^2} = 24.5 \\ \nabla_\sigma J(\theta) = A \times \frac{(a_t - \mu)^2 - \sigma^2}{\sigma^3} \\ = 49 \times \frac{(0.5 - 0)^2 - 1^2}{1^3} = -36.75 \end{matrix} \rightarrow \begin{matrix} \mu \leftarrow \mu + \alpha \nabla_\mu J(\theta) = 0 + 0.01 \times 24.5 \\ = 0.245 \\ \sigma \leftarrow \sigma + \alpha \nabla_\sigma J(\theta) = 1 + 0.01 \times (-36.5) \\ = 0.6325 \end{matrix}$$

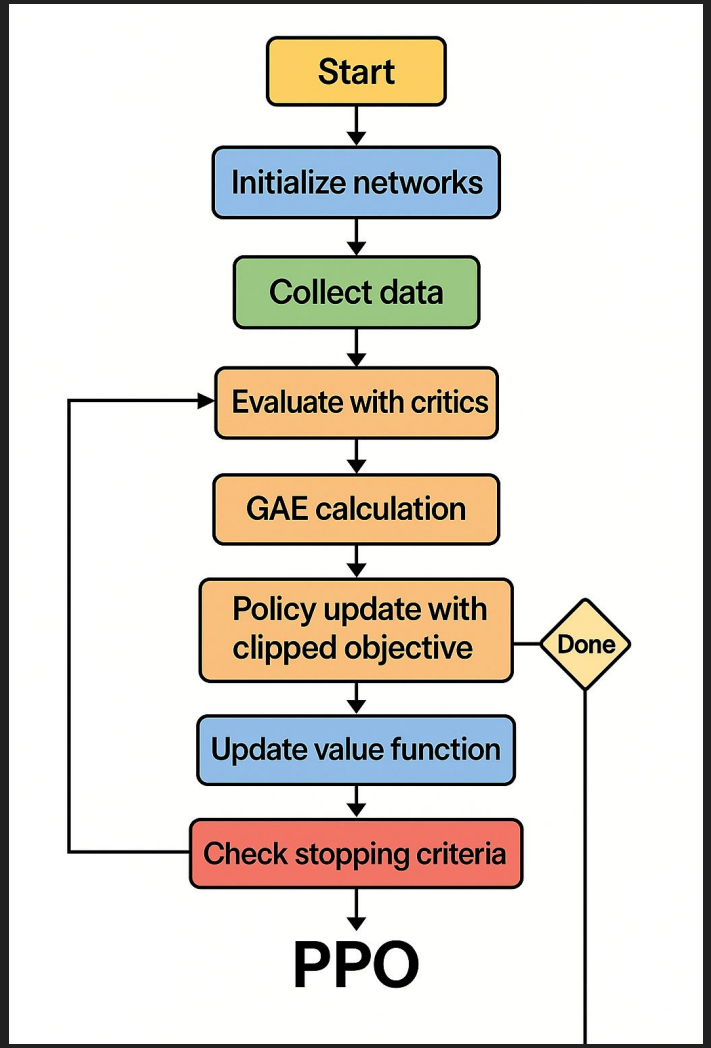
(4) Critic更新

$$\min L_{critic}(\theta^V) = [r_t + \gamma V(S_{t+1}) - V(S_t)]^2 = A_t^2 \rightarrow \nabla_\theta L_{critic}(\theta^V) = -2A_t \times \frac{\partial V(S_t)}{\partial \theta^V} \rightarrow \theta \leftarrow \theta - \alpha \nabla_\theta L_{critic}(\theta)$$

PPO(Proximal Policy Optimization)

PPO（Proximal Policy Optimization）是一種 on-policy 策略梯度演算法，為 A2C 的改良版。其核心設計是：

- PPO 與 A2C 一樣採用 Actor-Critic 架構，但強化了：
  - 使用 **Clipped Objective** 控制策略變動幅度
  - 支援 mini-batch 與多輪 epoch 訓練
  - 提升樣本效率與穩定性
- 在我們的實作中：
  - **Actor** 輸出 mean 和 std，建構 `Normal(mean, std)` 分布，從中 sample 動作。
  - **Critic** 輸出狀態價值 V(s)，協助計算 TD target 與 Advantage。

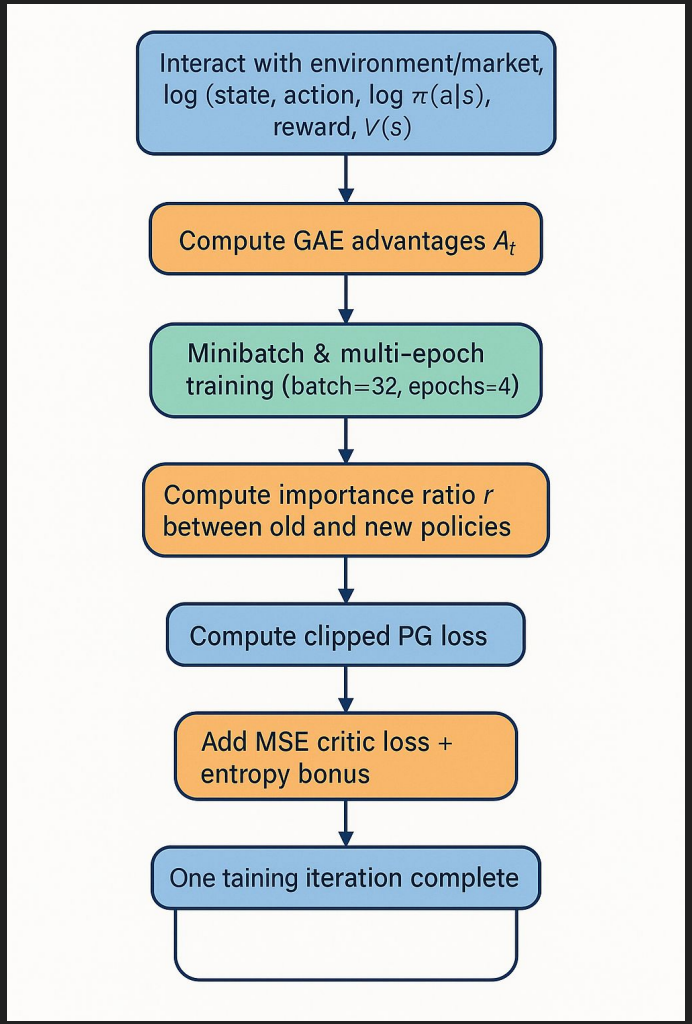

Start
Initialize networks
Collect data
Evaluate with critics
GAE calculation
Policy update with clipped objective
Done
Update value function
Check stopping criteria
PPO

訓練流程

1. 在環境中與市場互動，記錄狀態、動作、動作對數機率、獎勵與 Critic 估計值
2. 使用 GAE 計算每筆資料的 advantage 值。
3. 使用min-batch ＋ 多輪epoch訓練(如batch size = 32，epoch = 4)提升樣本效率。
4. 使用舊策略與新策略計算重要性比率 r：
$$r_t = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$$
5. 計算 clipped policy gradient loss這樣可避免策略大幅度偏離原本的 policy：

$$L = -\min\left(r_t A_t,\ \text{clip}(r_t, 1-\epsilon, 1+\epsilon) A_t\right)$$

6. 搭配 MSE Critic loss 與 entropy bonus，完成一次訓練迭代。


Interact with environment/market, log (state, action, log $\pi(a|s)$, reward, $V(s)$
Compute GAE advantages A_t
Minibatch & multi-epoch training (batch=32, epochs=4)
Compute importance ratio r between old and new policies
Compute clipped PG loss
Add MSE critic loss + entropy bonus
One taining iteration complete

損失函數組合

$$L = -\min(r_t A_t,\ \text{clip}(r_t, 1-\epsilon, 1+\epsilon)A_t) + c_v \cdot (V(s) - G)^2 - c_e \cdot H(\pi)$$

- 第一項為 Clipped Policy Loss：控制更新範圍
- 第二項為 Value Loss：TD-target 的 MSE
- 第三項為 Entropy Bonus：鼓勵多樣策略探索

特點與評價

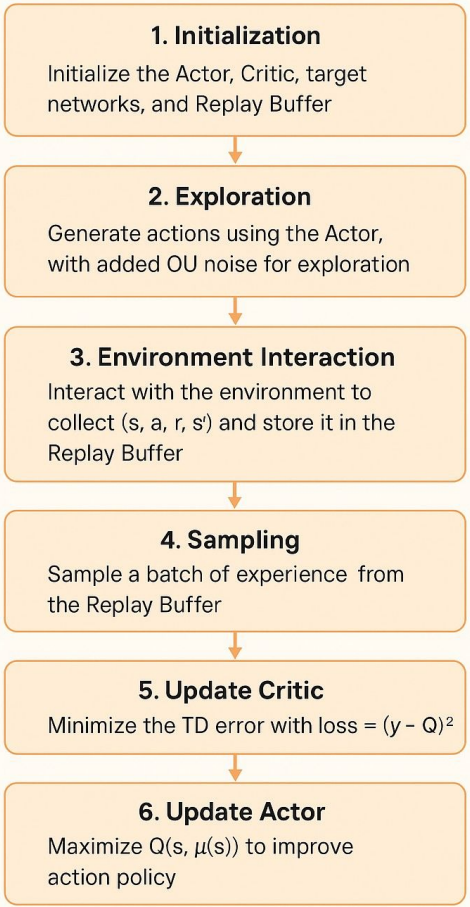
- 優點
  - 引入 Clipping 限制更新：提升策略學習穩定性
  - 支援 mini-batch 多輪訓練：提高 sample efficiency
  - 在連續動作問題上效果良好：適合金融市場資產配置應用
- 缺點
  - 計算成本較 A2C 高：每集需重複更新多次
  - 需要調整多個超參數（如 clip ratio, epoch）：增加實作與調參難度

DDPG（Deep Deterministic Policy Gradient）

DDPG 是一種專為**連續動作空間** 設計的強化學習演算法，結合了 Actor-Critic 架構與 DQN 技術（如 Replay Buffer 和 Target Network）。它屬於 **off-policy** 方法，能有效重複利用資料，是連續控制場景中常見的基礎架構。

- **Actor**：輸出每支資產的動作（即調整比例），不再是分布，而是**確定性動作** ：
- **Cri**$a = \mu(s|\theta^\mu)$ 個 (s,a) 配對，預測其對應的 Q 值
- 搭配以下三個核心模組強化穩定性：$Q(s, a|\theta^Q)$
  - **Replay Buffer**：儲存過去經驗打破樣本相關性
  - **Target Network**：訓練時使用延遲更新的 Actor/Critic Target，減少 Q 值估計不穩
  - **OU Noise（Ornstein-Uhlenbeck）**：時間相關噪音，模仿金融市場「慣性」波動

DDPG Flowchart



Critic更新邏輯

Critic 的目標是預測 Q(s,a)值，並對齊 TD 目標：

$$y = r + \gamma Q'(s', \mu'(s'))$$

其中：

- Q′、μ′：分別是目標 Critic 與目標 Actor 網路
- y：目標 Q 值（TD-target）

Critic loss 使用 MSE 誤差最小化預測與目標差距：

$$L_{\text{critic}} = (Q(s,a) - y)^2$$

Actor 更新邏輯

Actor 的目標是輸出讓 Critic 評價越高的動作，因此反向最大化 Critic 評價：

$$L_{\text{actor}} = -Q(s, \mu(s))$$

也就是說，Actor 在持續修正自己，讓它輸出的動作在 Critic 看來是「更有價值」的。

特點與評價

- 優點:
  - 支援精細的連續動作輸出:適合資產配置、能源控制等場景
  - 使用 Replay Buffer:提升樣本效率，支援長期訓練，也可以避免過度擬合
  - 使用 Target Network 與 OU Noise:穩定學習與有效探索兼具
- 缺點
  - 對初始化與參數設定敏感:若 OU noise 不佳，容易震盪或學不到策略
  - 訓練不如 PPO 穩定:不適用於劇烈 regime shift 的場景，DDPG 比較適合用在市場規律穩定的場景，如果市場突然劇烈變動，它調整得不夠快，策略容易失效。

特徵工程(隨機森林)

- 初始納入多種技術指標作為候選特徵，包括：
  - MACD
  - RSI（相對強弱指標）
  - CCI（順勢指標）
  - ADX（趨勢強度指標）
  - ATR（真實波動幅度）
  - VR（成交量比率）
  - Bollinger Bands（上下軌）
  - vol_5d（近五日成交量波動）

  總計超過十個特徵

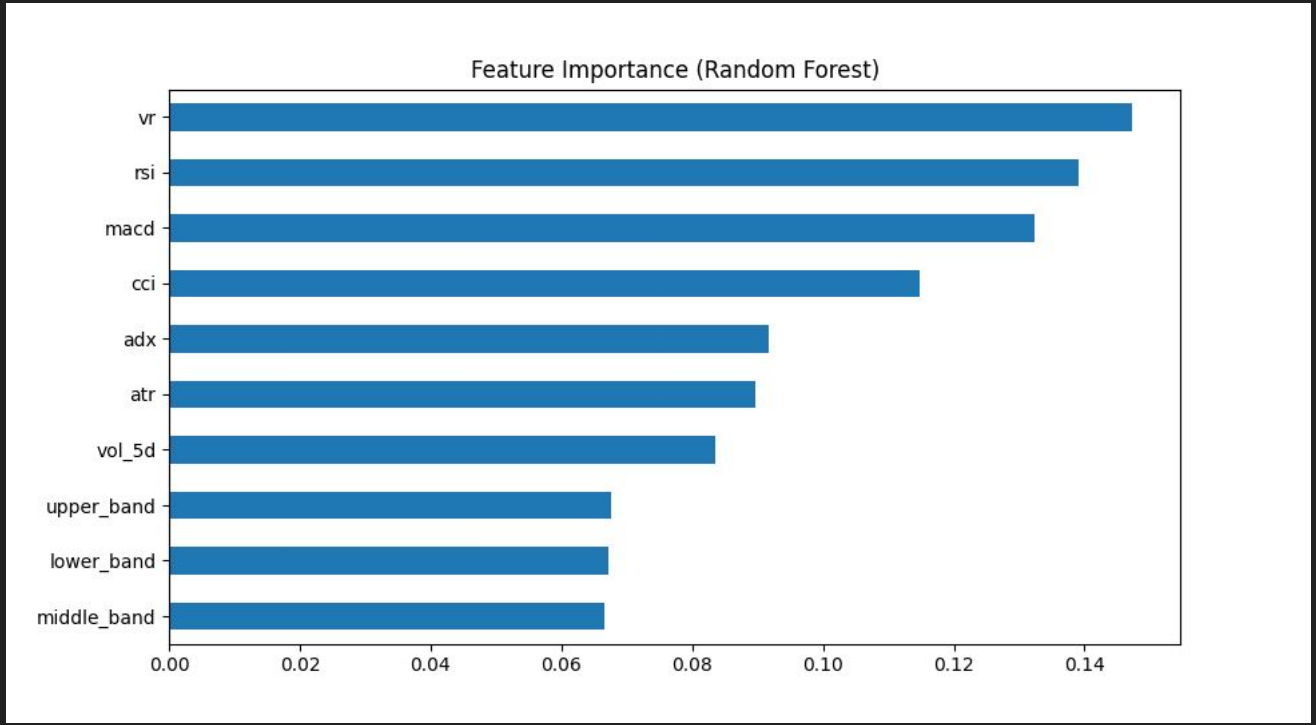
標籤設計（target）

- 以每支股票 **下一日（t+1）報酬方向** 為預測目標：

$$\text{target} = \begin{cases} 1 & \text{若 } \text{return}_{t+1} > 0 \\ 0 & \text{否則} \end{cases}$$

- 這樣設計可模擬「預測隔日是否上漲」的場景，使模型能透過技術指標識別短線動能。

選擇前五大特徵


Feature Importance (Random Forest)
vr
rsi
macd
cci
adx
atr
vol_5d
upper_band
lower_band
middle_band
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14

環境設計

為了讓強化學習模型能在類似真實市場的條件下學習投資決策，需要設計了一個符合 OpenAI Gym 標準的多資產交易環境，支援資產配置、持股限制、技術指標觀察與交易成本。

- 狀態空間設計（共 211 維）
  - 現金資產（1維）　：當前可動用資金
  - 股票價格（30維）：每檔股票的 adjusted close price
  - 持股資訊（30維）：每檔股票的持有張數
  - 技術指標（150維）：每檔股票的 5 種指標（MACD、RSI、CCI、ADX、VR）各 30 支

動作空間設計

- 每一個動作代表一檔股票的交易指令
- 連續值介於 -1 和 1 之間:
  - 正值:買入
  - 負值:賣出
- 真實執行時，會乘上最大單次交易數(例如 100 股)，並考慮現金餘額與手續費(每次交易0.1%)
- Reward 設計:
  - 每步 reward 以帳戶總資產變動計算:　$\text{reward}_t = (\text{資產}_t - \text{資產}_{t-1}) \times \text{scaling}$
  - scaling 設為 1e-4，防止數值過大導致學習不穩
  - reward 並非股票報酬率，而是整體帳戶回報，包含現金與持股

環境互動邏輯

1. 接收強化學習 Agent 的動作指令
2. 先賣出再買入（考慮現金與庫存限制）
3. 根據價格、成本計算實際成交與資產變動
4. 更新持股與現金，計算 reward
5. 輸出下一天狀態與 reward，直到資料結束

策略流程設計

強化學習交易策略採用 **滾動式視窗（rolling window）架構**，將整段歷史資料動態切分為訓練、驗證與交易三個階段，並透過 Ensemble 模型選擇機制提升決策品質與穩定性。

- 資料分段邏輯
  - train：從 2009/01/01 起至當期 Validation 開始前一天，用來訓練 A2C、PPO、DDPG 三種模型
  - Validation：接續 63 天，評估每一模型於當前市場狀況下的適應性
  - Trade：再接續 63 天，以表現最佳的模型進行實際交易模擬

每期結束後向前平移 63 天，持續滾動訓練與更新。

滾動式訓練與驗證邏輯

在每一次迭代中，依序執行以下流程：

1. 使用訓練資料，分別訓練 A2C、PPO 與 DDPG 模型。
2. 在 Validation 資料上執行策略，產出資產變化與 Sharpe ratio。
3. 根據 Sharpe ratio 選出當期表現最佳的演算法。
4. 使用該模型執行後續 Trade 資料段的交易模擬。
5. 儲存回測結果與模型選擇紀錄，進入下一期視窗。

策略架構優勢

- 特點:
  - 滾動更新訓練資料:可適應市場結構逐步變化
  - Ensemble 模型選擇:不同模型在不同階段可展現強項
  - 使用 Validation 資料選模:避免過度擬合過往市場

風險控制：Turbulence Index

Turbulence 指標是根據Mahalanobis Distance（馬氏距離）計算衡量當前市場價格向量與其歷史均值的偏離程度，數值越高，代表市場行為越異常、風險越高。

計算方式如下：

$$\text{Turbulence}_t = (p_t - \mu)^\top \Sigma^{-1} (p_t - \mu)$$

其中：

- pt：當前資產價格向量
- μ：歷史均值（252天）
- Σ：歷史價格共變異數矩陣

為避免初期樣本不穩，本研究前兩筆 turbulence 值設為 0，提升穩定性。

動態流程

風控門檻 並非固定值，而是結合**長期 baseline + 短期市場狀況** 共同決定，具備以下四步邏輯：

① 建立 insample turbulence baseline

- 初始使用 **2009/01 ~ 2015/10** 的資料，作為 insample baseline
- 對該區段 turbulence 指標取 **90% 分位數** 作為警戒線
- 每五年（約 1260 個交易日）**向前滾動一次 baseline**，模擬市場結構變化

② 動態監控市場當前風險

- 每一期前 63 天（約 1 季）為風險觀察期
- 計算該段 turbulence 的**歷史平均值**
- 若均值大於 insample baseline，則視為「風險過高」

③ 啟用 robust safety line（保守門檻）

- 在高風險情況下，不採用當期實際 turbulence，而回退使用保守固定門檻
- 本研究設定 robust threshold = **96**
- 確保在例如 2020 COVID-19 、2022激進升息此類極端時期，不會高曝險交易

④ 進入風控模式的行為反應（執行端）

- 當日 turbulence 超過 threshold 時，**禁止買進、強制清倉**
- 否則正常執行強化學習動作決策

實驗資料與設定說明

研究標的與資料來源：

- **標的資產**：道瓊工業指數 30 檔成分股（Dow 30）
- **期間範圍**：2009/01/01 ～ 2024/10/7（共約 4000 個交易日）
- **資料頻率**：日資料（Daily）
- **原始欄位**：tic、open、high、low、adjcp、volume
- **資料來源**：WRDS

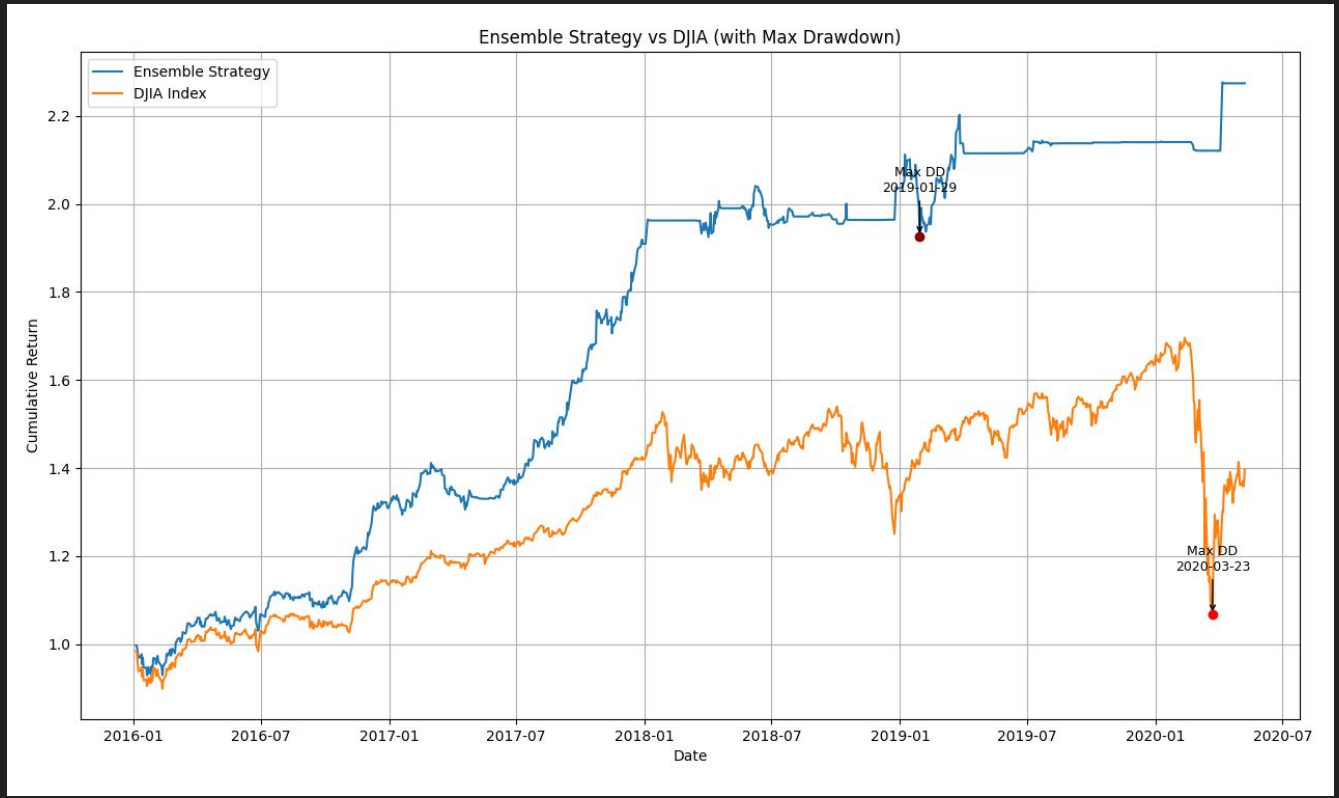
策略訓練與回測設定

項目	設定值
初始資產	$1,000,000
每檔最大交易量	100股
Rebalance 週期	每 63 天（約每季）
交易成本	每次交易0.1%
模型訓練結構	滾動式訓練－驗證－交易（Rolling Window）
Training 起點	2009/01/01
實際交易區間	2015/12/31-2024/10/04
訓練時長	5.84小時

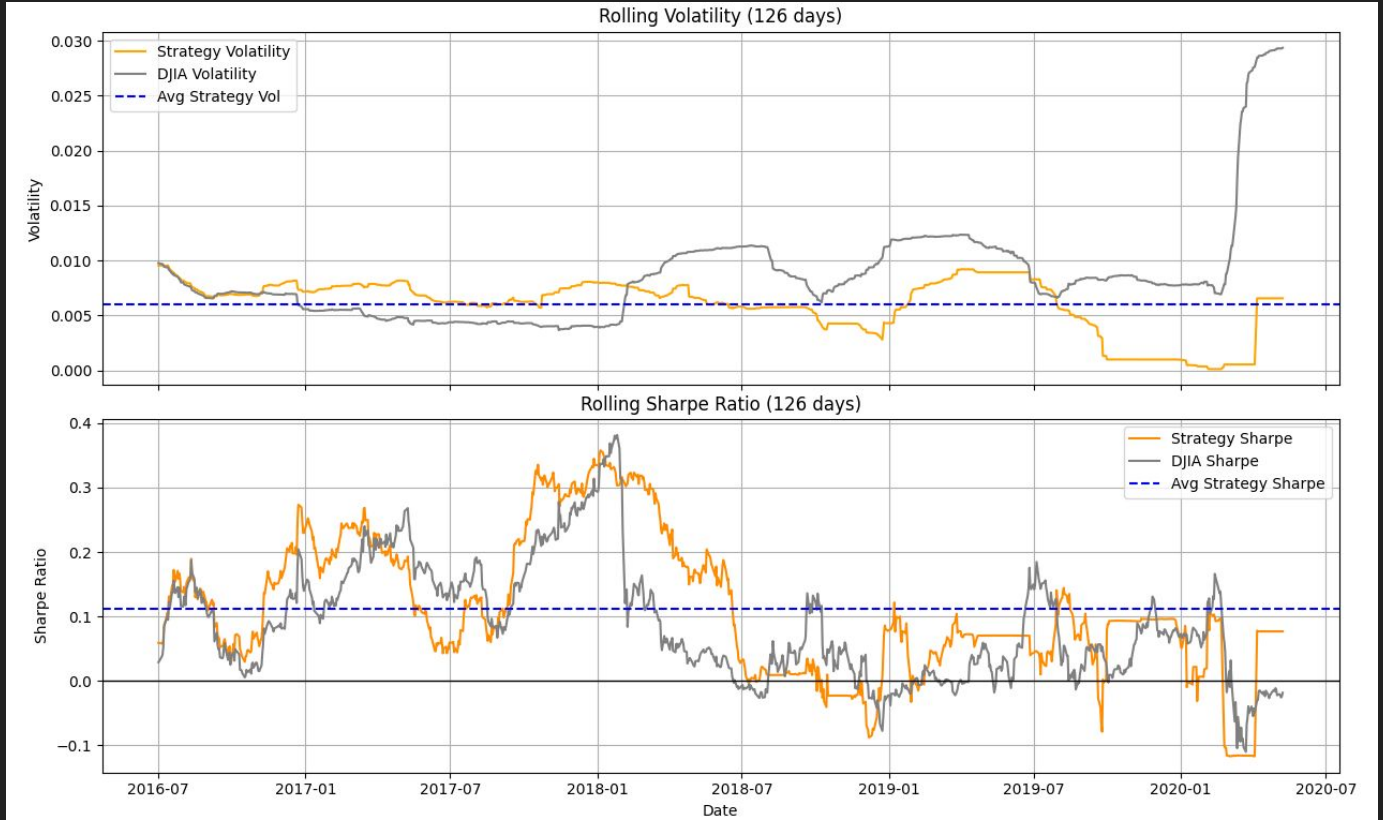
強化學習演算法設定

模型	Learning Rate	Episode 數
A2C	1e-4	50
PPO	1e-4	50
DDPG	Actor: 1e-3 Critic: 1e-3	50

績效呈現(2016-2020)



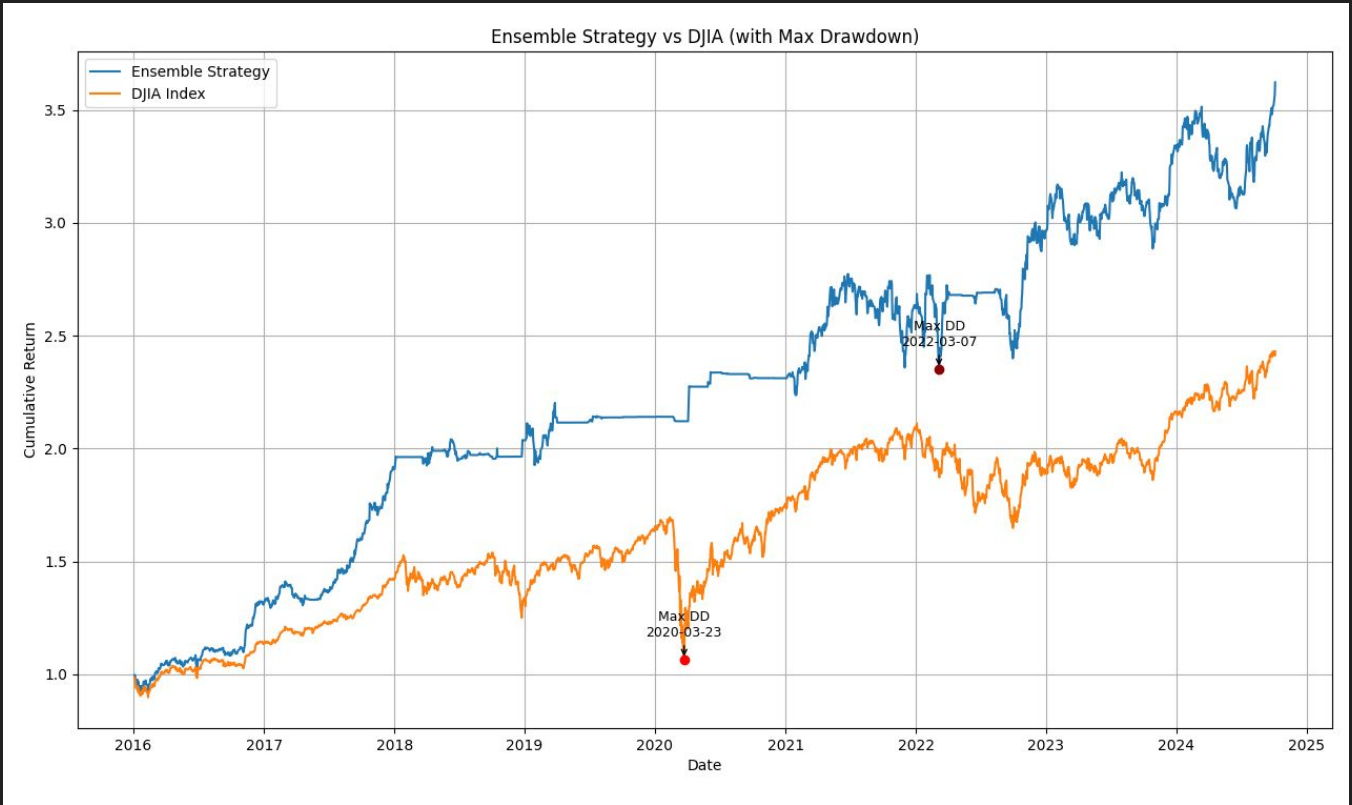
績效呈現(2016-2020)



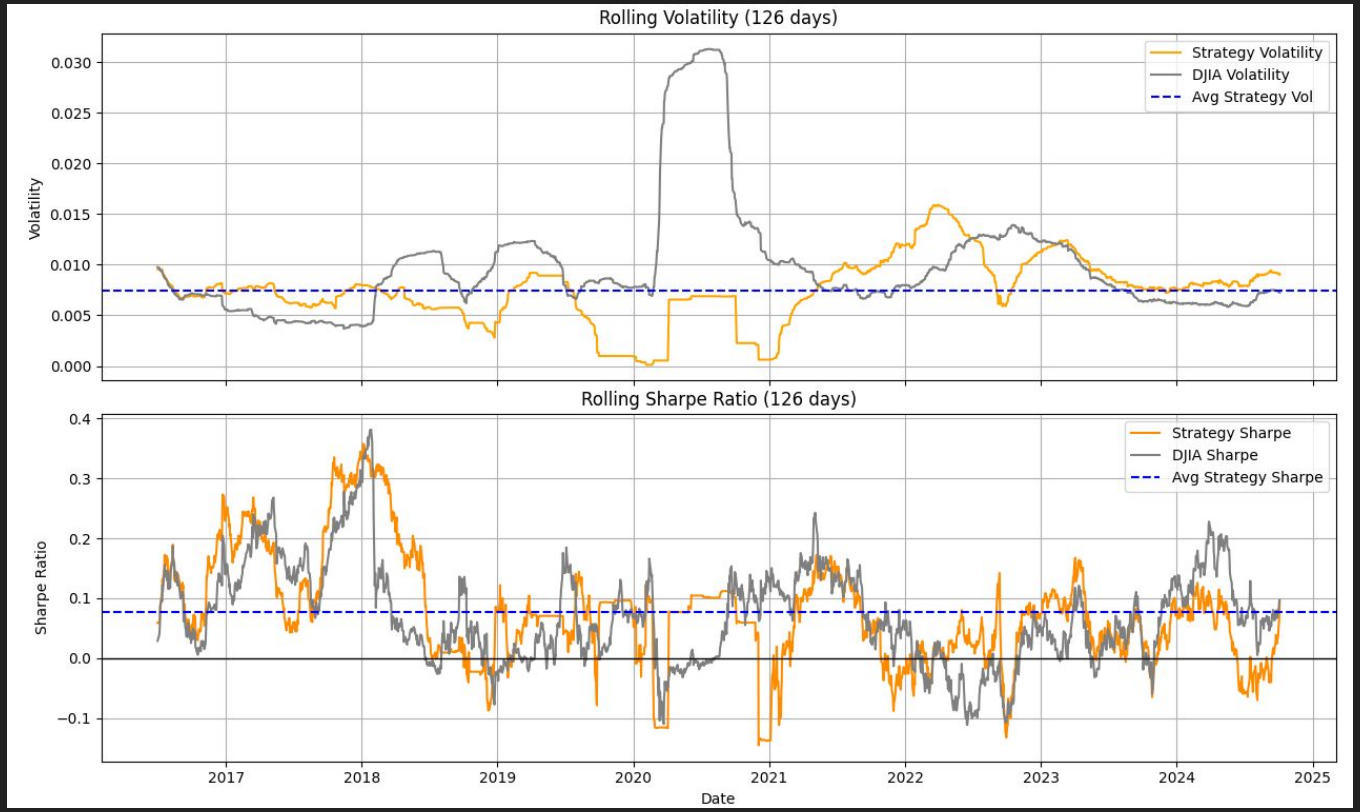
Ours Ensemble(2016-2020) vs. 論文ensemble

指標	Ours Ensemble (2016-2020)	論文 Ensemble (2016-2020)	道瓊指數(DJIA)
Cumulative Return	127.4141%	70.4%	38.6%
Annual Return	20.7930%	13%	7.8%
Sharpe Ratio	1.7837	1.3	0.47
Annual Volatility	10.9374%	9.7%	20.1%
Max Drawdown	-8.7839%	-9.7%	-37.1%

績效呈現(2016-2024)



績效呈現(2016-2024)



績效呈現

指標	Ensemble 策略	道瓊指數(DJIA)
Cumulative Return	262.1638%	143.0570%
Annual Return	15.8444%	10.6830%
Sharpe Ratio	1.1931	0.66
Annual Volatility	13.0469%	17.86%
Max Drawdown	-15.1474%	-37.0862%

論文比較

1. 使用隨機森林進行特徵選擇
    a. 原論文直接使用固定技術指標作為 state input
    b. 我們利用 Random Forest Classifier 分析各項技術指標對未來報酬的預測能力，篩選出最具解釋力的前 Top-K 特徵，以提升模型的績效
2. 風控機制採用動態滾動式 insample threshold 設定
    a. 原論文使用固定的 insample turbulence threshold（0.9 quantile of 2009–2015）
    b. 我們採用動態風控機制：
        i. 每 **5 年向前滾動** 更新 insample 樣本
        ii. 使用 insample turbulence 90 分位數作為 baseline
        iii. 依據當前市場的**近一季實際風險** 進行比對與切換，實現風控鬆緊動態調整

結論

- 使用隨機森林篩選出的技術指標，策略績效已超越原本的論文。
- 論文原有的硬性風控機制，隨著回測期間拉長、市場結構改變後逐漸失效。
- 改良後的動態風控機制，雖然仍會略增回撤，整體風險控制卻明顯優於原機制。
- 優化方向：
  - 做更多的特徵工程在環境上面優化
  - Reward Function的優化，不靠硬性風控靠Agent學習風控。

神經網路架構說明（Actor & Critic）

策略中所有強化學習演算法均採用具備兩層隱藏層的多層感知器（MLP）作為 Actor 與 Critic 的主體，並依據不同演算法需求設計出相應的輸出層與變化細節。

A2C & PPO：共用 Actor-Critic 結構

組件	結構設計
隱藏層	兩層 `Linear`，每層 `64 units + ReLU` 啟動函數
Actor 輸出	`nn.Linear(64, action_dim) + Tanh`，輸出動作的 mean 向量
Actor std	`actor_log_std` 為一組可訓練參數（torch.Parameter），用於建立 Normal 分布
Crtirc 輸出	`nn.Linear(64, 1)`，估計狀態價值 V(s)

DDPG：分離式 Actor & Critic 結構

組件	結構設計
Actor	MLP(64-64-Tanh)：輸出連續動作（在 [−1,1]區間）
Critic	MLP(64-64-1)：輸入為狀態 s 與動作 a 的串接向量 [s,a]，輸出為 Q 值
Target Network	Actor/Critic 各自維持一組 target model，以 soft update 更新參數穩定訓練
探索策略	使用 Ornstein-Uhlenbeck 過程產生 mean-reverting noise 加入動作探索

Actor-Critic架構

(1) Actor決定動作　S_t: state, P_t: stock price, H_t:shares, B_t: balance, c_t: cost rate, r_t:return

$$S_t = \begin{bmatrix} P_t \\ H_t \\ B_t \end{bmatrix} \rightarrow \boxed{\begin{matrix}\text{Actor} \\ a_t \sim N(\mu, \sigma)\end{matrix}} \rightarrow a_t \rightarrow \begin{matrix}\Delta H_t = a_t \times 100 \\ \Delta B_t = -P_t \Delta H_t\end{matrix} \rightarrow S_{t+1} = \begin{bmatrix} P_{t+1} \\ H_{t+1} \\ B_{t+1} \end{bmatrix}$$

(2) Critic評估Actor動作之成效

S_t、S_{t+1} → Critic →

$$r_t = change\ of\ portfolio\ value - cost$$
$$= (B_{t+1} + P_{t+1}H_{t+1}) - (B_t + P_t H_t) - P_t \Delta H_t c_t$$
$$V(S_t) = E[r_t | S_t]$$
$$V(S_{t+1}) = E[r_{t+1} | S_{t+1}]$$

→ Advantage function(A)

$$A = r_t + \gamma V(S_{t+1}) - V(S_t)$$

γ: discount factor

(3) Actor更新

$$\min L_{actor}(\theta) \rightarrow \nabla_\theta L_{actor}(\theta) \rightarrow \theta \leftarrow \theta - \alpha \nabla_\theta L_{actor}(\theta)$$, α: learning rate

(4) Critic更新

$$\min L_{critic}(\theta) \rightarrow \nabla_\theta L_{critic}(\theta) \rightarrow \theta \leftarrow \theta - \alpha \nabla_\theta L_{critic}(\theta)$$, α: learning rate

A2C範例

(1) Actor決定動作

S_t: state, P_t: stock price, H_t:shares, B_t: balance, c_t: cost rate, r_t:return

$$S_t = \begin{bmatrix} P_t = 10 \\ H_t = 0 \\ B_t = 1000 \end{bmatrix} \rightarrow \boxed{\begin{matrix}\text{Actor} \\ a_t \sim N(0,1)\end{matrix}} \rightarrow a_t = 0.5 \rightarrow \begin{aligned} \Delta H_t &= a_t \times 100 \\ &= 0.5 \times 100 \\ &= 50 \\ \Delta B_t &= -P_t \Delta H_t \\ &= -10 \times 50 = -500 \end{aligned} \rightarrow S_{t+1} = \begin{bmatrix} P_{t+1} = 10.2 \\ H_{t+1} = 50 \\ B_{t+1} = 500 \end{bmatrix}$$

(2) Critic評估Actor動作之成效

S_t、S_{t+1} → Critic →

$$\begin{aligned} r_t &= change\ of\ portfolio\ value - cost \\ &= (B_{t+1}+P_{t+1}H_{t+1}) - (B_t+P_tH_t) - P_t\Delta H_t c_t \\ &= (500 + 10.2 \times 50) - (1000 + 0) - 500 \times 0.1\% \\ &= 9.5 \end{aligned}$$

$$V(S_t) = E[r_t|S_t] = 1000$$

$$V(S_{t+1}) = E[r_{t+1}|S_{t+1}] = 1050$$

→ Advantage function(A)

$$\begin{aligned} A &= r_t + \gamma V(S_{t+1}) - V(S_t) \\ &= 9.5 + 0.99 \times 1050 - 1000 \\ &= 49 > 0 \end{aligned}$$

(3) Actor更新

$$max\ J(\theta) = E\left[\sum_{t=1}^{T} \mathrm{A} \times \ln f(a_t|S_t)\right] \rightarrow$$

$$\begin{aligned} \nabla_\mu J(\theta) &= A \times \frac{a_t - \mu}{\sigma^2} \\ &= 49 \times \frac{0.5 - 0}{1^2} = 24.5 \\ \nabla_\sigma J(\theta) &= A \times \frac{(a_t - \mu)^2 - \sigma^2}{\sigma^3} \\ &= 49 \times \frac{(0.5 - 0)^2 - 1^2}{1^3} = -36.75 \end{aligned}$$

$$\begin{aligned} \mu \leftarrow \mu + \alpha\nabla_\mu J(\theta) &= 0 + 0.01 \times 24.5 \\ &= 0.245 \\ \sigma \leftarrow \sigma + \alpha\nabla_\sigma J(\theta) &= 1 + 0.01 \times (-36.5) \\ &= 0.6325 \end{aligned}$$

(4) Critic更新

$$\min L_{critic}(\theta^V) = [r_t + \gamma V(S_{t+1}) - V(S_t)\ ]^2 = \delta^2 \rightarrow \nabla_\theta L_{critic}(\theta^V) = -2\delta \times \frac{\partial V(S_t)}{\partial \theta^V} \rightarrow \theta \leftarrow \theta - \alpha\nabla_\theta L_{critic}(\theta)$$

A2C(Advantage Actor-Critic)

A2C（Advantage Actor-Critic）是一種 on-policy 的強化學習演算法，結合了策略網路（Actor）與價值網路（Critic），可同時學習動作策略與狀態評估，並有效降低訓練時的方差。

- 模型架構
  - Actor 輸出每支資產的動作平均值（mean）與標準差（std），建立 `Normal(mean, std)` 分布。
  - Critic：估計狀態價值 V(S)，提供更穩定的 baseline，讓 Actor 更新時可以減少估計誤差與方差。

Critic更新邏輯

Critic 的目的是學會預測某個狀態下未來可能獲得的回報。為此，我們使用 TD-target 作為學習目標：

$$G_t = r_t + \gamma V(s_{t+1})$$

接著透過均方誤差（MSE）使預測值 V(st)趨近於 TD 目標：

$$L_{\text{critic}} = (V(s_t) - G_t)^2$$

這讓 Critic 漸漸具備穩定估值能力，成為 Actor 更新的重要基礎。

Actor更新邏輯

Actor 的目標是最大化累積回報，因此需根據 advantage At 進行 policy gradient 更新。

我們使用 Generalized Advantage Estimation（GAE）估計 At，以平衡 bias 與 variance：

$$A_t = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}, \quad \text{其中 } \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$$

更新策略時會考慮歷史 log-probability：

$$\nabla_\theta J = \mathbb{E}[\nabla_\theta \log \pi_\theta(a_t|s_t) \cdot A_t]$$

訓練流程

1. 在環境中與市場互動，記錄狀態、動作、動作對數機率、獎勵與 Critic 估計值
2. 每隔 `n_step` 或 episode 結束後進行一次更新
3. 使用 Generalized Advantage Estimation（GAE）估計 Advantage 值

   ：$A_t = \delta_t + \gamma\lambda A_{t+1}$
4. Actor loss 根據 advantage 計算 policy gradient
5. Critic 使用 TD target 進行 MSE 回歸
6. 加入 entropy bonus 鼓勵策略探索行為

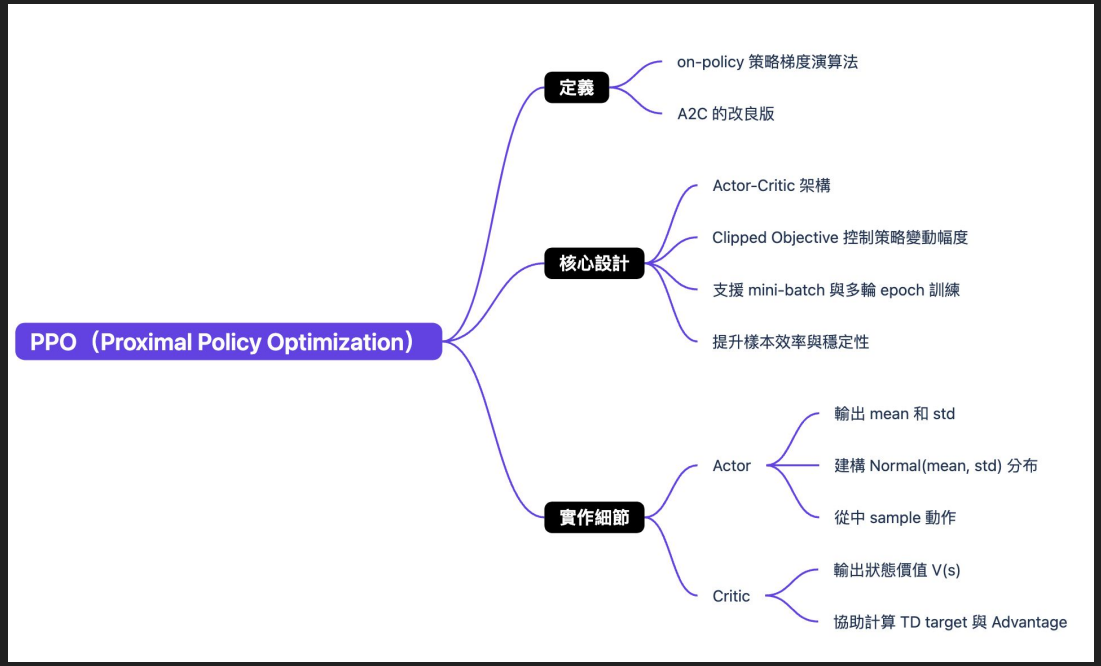
損失函數組合

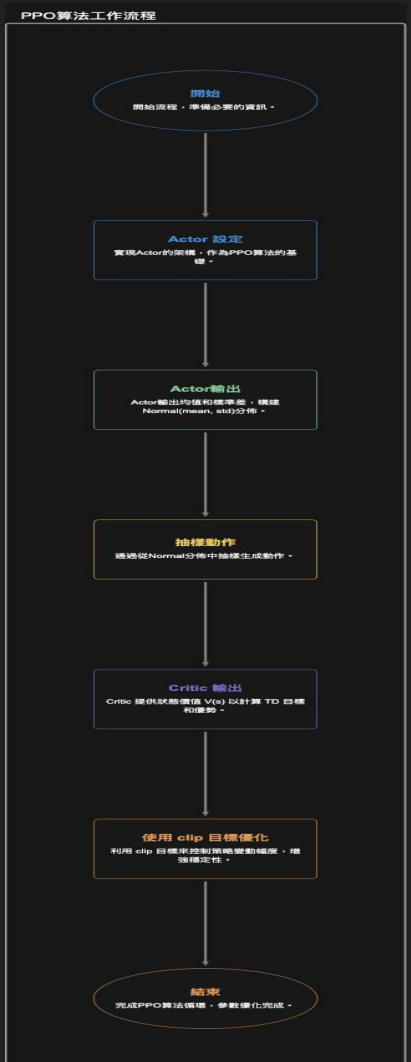
$$L = -\log \pi(a|s) \cdot A(s,a) + c_v \cdot (V(s) - G)^2 - c_e \cdot H(\pi)$$

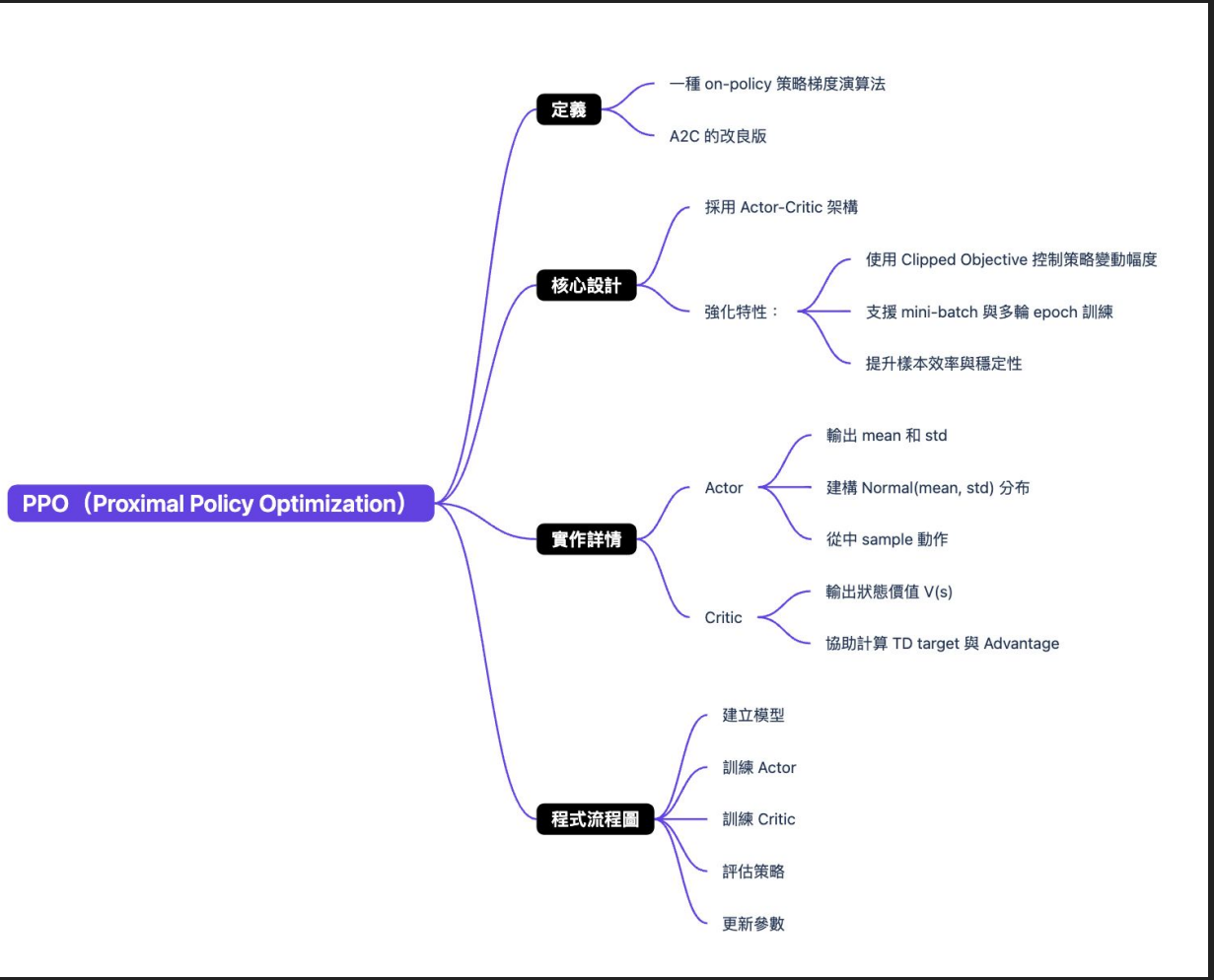
- 第一項為策略損失（policy gradient）
- 第二項為價值網路的 MSE 誤差
- 第三項為 entropy bonus，調整探索程度

特點與評價

- 優點:
  - 架構簡單, 實作快速:適合用於中小型環境與快速原型驗證
  - 使用 GAE 提升收斂穩定性:有效減少 variance
  - std 為可學參數:自動調整探索強度
- 缺點:
  - 無 replay buffer:樣本效率較低, 學習速度較慢(DDPG改善此問題)
  - 無 clipping 機制:遇高波動市場時, 更新可能不穩定(PPO改善此問題)


PPO（Proximal Policy Optimization）
定義
on-policy 策略梯度演算法
A2C 的改良版
核心設計
Actor-Critic 架構
Clipped Objective 控制策略變動幅度
支援 mini-batch 與多輪 epoch 訓練
提升樣本效率與穩定性
實作細節
Actor
輸出 mean 和 std
建構 Normal(mean, std) 分布
從中 sample 動作
Critic
輸出狀態價值 V(s)
協助計算 TD target 與 Advantage


PPO算法工作流程
開始
開始流程，準備必要的資訊。
Actor 設定
實現Actor的架構，作為PPO算法的基礎。
Actor輸出
Actor輸出均值和標準差，構建Normal(mean, std)分佈。
抽樣動作
通過從Normal分佈中抽樣生成動作。
Critic 輸出
Critic 提供狀態價值 V(s) 以計算 TD 目標和優勢。
使用 clip 目標優化
利用 clip 目標來控制策略變動幅度，增強穩定性。
結束
完成PPO算法循環，參數優化完成。


PPO（Proximal Policy Optimization）
定義
一種 on-policy 策略梯度演算法
A2C 的改良版
核心設計
採用 Actor-Critic 架構
強化特性：
使用 Clipped Objective 控制策略變動幅度
支援 mini-batch 與多輪 epoch 訓練
提升樣本效率與穩定性
實作詳情
Actor
輸出 mean 和 std
建構 Normal(mean, std) 分布
從中 sample 動作
Critic
輸出狀態價值 V(s)
協助計算 TD target 與 Advantage
程式流程圖
建立模型
訓練 Actor
訓練 Critic
評估策略
更新參數

Critic 更新方式

Critic 的任務與 A2C 相同，是估計每個狀態的價值 V(s)，其更新方式為：

$$G_t = r_t + \gamma V(s_{t+1})$$

$$L_{\text{critic}} = (V(s_t) - G_t)^2$$

我們使用 GAE 計算 At 與 TD target，並以 mini-batch 方式更新 Critic 多次（例如 4 個 epoch × batch size 32），在我們的模型中每128步更新一次策略，其中每32個state訓練一次，這樣訓練四輪，總共訓練16次。

Actor 更新方式

PPO 最大特色為 **限制策略更新幅度**，防止新策略與舊策略差異過大，造成學習崩壞。做法如下：

1. 儲存舊策略在每個動作下的 log 機率
2. 新策略經過多輪更新，重新計算新的
3. 比較兩者比值（重要性比率）：

$$r_t = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$$

4. 這個比值乘上 advantage At為 actor 的 policy gradient，但透過下方 clipping 機制控制更新幅度：

$$L_{\text{actor}} = -\min\left(r_t A_t, \text{clip}(r_t, 1-\epsilon, 1+\epsilon) A_t\right)$$

訓練流程

1. Actor 根據當前狀態輸出動作，加上 OU noise 作為探索。

2. 與環境互動，收集 (s, a, r, s', done) 並存入 replay buffer。

3. 當 buffer 大小超過 batch size 時，開始每步訓練：
   a. 從 buffer 隨機抽樣一批資料
   b. 計算 Target Q 值：$Q_{\text{target}} = r + \gamma Q'(s', \mu'(s'))$
   c. 更新 Critic 以最小化 TD 誤差
   d. 更新 Actor 以最大化：$L_{\text{actor}} = -Q(s, \mu(s))$
   e. 使用 soft update 更新 target networks：$\theta' \leftarrow \tau\theta + (1-\tau)\theta'$

損失函數組合

- **Critic Loss（TD Error）**：

$$L_{\text{critic}} = (Q(s,a) - Q_{\text{target}})^2$$

- **Actor Loss（最大化 Q 值）**：

$$L_{\text{actor}} = -Q(s, \mu(s))$$

強化學習演算法設定

模型	Learning Rate	Episode 數
A2C	1e-4	50
PPO	1e-4	50
DDPG	Actor: 1e-3 Critic: 1e-3	50