Pairs-Trading a Sparse Synthetic Control

Jesus Villota Miranda[†] [*]

⟨ [†]CEMFI, Calle Casado del Alisal, 5, 28014 Madrid, Spain ⟩
⟨ Email: jesus.villota@cemfi.edu.es ⟩

This version: 19th May 2025

Abstract

This paper defines a novel approach to pairs trading by borrowing the concept of synthetic control from the treatment literature. In this paper we select a stock (termed "*target*") and construct its replica as a sparse linear combination of other assets (termed "*synthetic*"). Then, we perform pairs trading on the target vs. synthetic assets; for this purpose, we (nonlinearly) model their joint dependence via a Student-t copula and then construct mispricing indices from the implied conditional densities. Finally, we feed the dynamics of our miscpricing indices to a reinforcement learning agent. Our findings show that our RL agent succesfully implement statistical arbitrage based on our mispricing signals with a high net profitability out of sample.

JEL Codes: C14, C32, C58, C61, G12, G14

Keywords: Pairs Trading, Synthetic Control, Mispricing, Copula, Basket trading, Sparse Mean Reverting Portfolios, Reinforcement Learning

[*] I am deeply grateful to Enrique Sentana, Rafael Repullo, Anatoli Segura, Maria Teresa Gonzalez-Perez and participants at the CEMFI Banking & Finance Seminar Series for their feedback. I gratefully acknowledge financial support from the Santander Research Chair at CEMFI.

1. Methodology

This paper describes the interaction of three agents: (1) a synthetic control builder, (2) a joint dependence modeller and (3) an intelligent trading agent. The first two agents are "*not*" intelligent, in the sense that they don't get better at their jobs through interaction with their environment (a future research direction would be to make these agents intelligent by endowing them with a learning ability). The third agent is intelligent because it is "*trained*". That is, the agent receives information from the current state of the environment and takes an action for the next period according to a policy function. In the next period, after the state is realized, the agent evaluates the optimzality of the action undertaken by means of a utility function. Finally, the agent reoptimizes its policy function according to the utility obtained.

First agent: *Synthetic Control Builder.*

This agent takes a time series of past prices from the target asset (trgt) and from a donor pool $\mathcal{D}$ (with $\mathcal{D} \cap \text{trgt} = \emptyset$) and derives the synthetic control weights to best replicate the log-price behavior of the target asset in sample. Formally, it solves:

$$\{\hat{w}^i\}_{i\in\mathcal{D}} = \arg\min_{\{\hat{w}^i\}_{i\in\mathcal{D}}} \left\{ \sum_{t\in\mathcal{T}^{tr}} \left(\log p_t^{\text{trgt}} - \sum_{i\in\mathcal{D}} w^i \log p_t^i \right)^2 + \lambda \sum_{i\in\mathcal{D}} |w^i| \right\} \quad \text{s.t.} \quad \sum_{i\in\mathcal{D}} w^i = 1.$$

which then defines the price for the synthetic asset (synth).

$$p_t^{\text{synth}} = \sum_{i\in\mathcal{D}} \hat{w}^i p_t^i$$

Algorithm 1. Synthetic Control Builder

```
Require: price time series of a target asset $\{p_t^{\text{trgt}}\}_{t\in\mathcal{T}}$; price time series of a donor pool $\{p_t^i\}_{i\in\mathcal{D},t\in\mathcal{T}}$
1: Compute synthetic control weights in-sample
       $\{\hat{w}^i\}_{i\in\mathcal{D}} \leftarrow \arg\min_{\{\hat{w}^i\}_{i\in\mathcal{D}}} \left\{ \sum_{t\in\mathcal{T}^{tr}} \left( \log p_t^{\text{trgt}} - \sum_{i\in\mathcal{D}} w^i \log p_t^i \right)^2 + \lambda \sum_{i\in\mathcal{D}} |w^i| \right\} \quad \text{s.t.} \quad \sum_{i\in\mathcal{D}} w^i = 1.$
2: Compute price time series of the synthetic asset
       for $t \in \mathcal{T}$ :   $p_t^{\text{synth}} \leftarrow \sum_{i\in\mathcal{D}} \hat{w}^i p_t^i$
Ensure: price time series of the synthetic asset $\{p_t^{\text{synth}}\}_{t\in\mathcal{T}}$
```

Second agent: *Mispricing detective.*

This agent computes the in-sample returns from the target and synthetic assets, maps them to the empirical quantiles and estimates their joint distribution by means of a Student-t copula.

$$F(\mathsf{r}^{\text{trgt}}, \mathsf{r}^{\text{synth}}) = C(u, v; \rho, \nu)$$

The agent then uses the in-sample joint distribution of returns to compute mispricing indices (MI) for the pair prices. Such indices are obtained from the implied conditional densities. In particular, it is computed as the probability that the target (synthetic) return is lower than its realized value conditional on the synthetic (target) return being equal to its current value. Formally:

$$\begin{aligned} MI_t^{\text{trgt|synth}} &= \mathbb{P}\left[\mathsf{r}^{\text{trgt}} \leq r_t^{\text{trgt}} \mid \mathsf{r}^{\text{synth}} = r_t^{\text{synth}}\right] \\ MI_t^{\text{synth|trgt}} &= \mathbb{P}\left[\mathsf{r}^{\text{synth}} \leq r_t^{\text{synth}} \mid \mathsf{r}^{\text{trgt}} = r_t^{\text{trgt}}\right] \end{aligned}$$

Subsequently, the agent computes a cumulative mispricing index (CMI) by accumulating these mispricing indices in an autoregressive way. The accumulation of the mispricing indices (conditional probabilities) is done in excess of 0.5 (i.e. the noninformative probability level, or simply, the probability of a fair coin toss).

$$\begin{aligned} CMI_0^{\text{trgt|synth}} &= 0; \qquad CMI_t^{\text{trgt|synth}} = CMI_{t-1}^{\text{trgt|synth}} + (MI_t^{\text{trgt|synth}} - 0.5) \quad \text{for } t > 1 \\ CMI_0^{\text{synth|trgt}} &= 0; \qquad CMI_t^{\text{synth|trgt}} = CMI_{t-1}^{\text{synth|trgt}} + (MI_t^{\text{synth|trgt}} - 0.5) \quad \text{for } t > 1 \end{aligned}$$

The CMIs are reset to 0 every time a position is closed. This means that the CMI will effectively reflect the mispricing probability accumulation at each trade.

CMI Resetting When a position is exited (either closing to neutral or flipping direction), both CMIs are reset to zero:

$$\text{If } (a_{t-1} \neq 0 \text{ and } a_t = 0) \text{ or } (a_{t-1} \neq 0 \text{ and } a_t \neq a_{t-1}) :$$

$$CMI_t^{\text{trgt|synth}} = 0, \quad CMI_t^{\text{synth|trgt}} = 0$$

This reset mechanism ensures that new trades are initiated based on fresh mispricing signals rather than historical accumulation.

1.0.1 Mispricing Detective

Algorithm 2. Pseudo Observations

```
Require: Price time series of the target and synthetic assets $\{(p_t^{\text{trgt}}, p_t^{\text{synth}})\}_{t\in\mathcal{T}}$
1: Compute returns:
       for $t \in \mathcal{T}$ :   $r_t^{\text{trgt}} \leftarrow \frac{p_t^{\text{trgt}} - p_{t-1}^{\text{trgt}}}{p_{t-1}^{\text{trgt}}}$,   $r_t^{\text{synth}} \leftarrow \frac{p_t^{\text{synth}} - p_{t-1}^{\text{synth}}}{p_{t-1}^{\text{synth}}}$
2: Fit (in-sample) empirical CDFs:
       $\widehat{F}^{\text{trgt}}(r) \leftarrow \text{LinearlyInterpolate}\left(\frac{1}{|\mathcal{T}^{tr}|}\sum_{\tau\in\mathcal{T}^{tr}} \mathbb{I}(r_\tau^{\text{trgt}} \leq r)\right)$
       $\widehat{F}^{\text{synth}}(r) \leftarrow \text{LinearlyInterpolate}\left(\frac{1}{|\mathcal{T}^{tr}|}\sum_{\tau\in\mathcal{T}^{tr}} \mathbb{I}(r_\tau^{\text{synth}} \leq r)\right)$
3: Compute the pseudo-observations of the returns
       for $t \in \mathcal{T}$ :   $u_t \leftarrow \widehat{F}^{\text{trgt}}(r_t^{\text{trgt}})$,   $v_t \leftarrow \widehat{F}^{\text{synth}}(r_t^{\text{synth}})$
Ensure: $\{u_t\}_{t\in\mathcal{T}}$, $\{v_t\}_{t\in\mathcal{T}}$
```

Algorithm 3. Calibrate the Student-t copula parameters (ν, ρ) in-sample

```
Require: In-sample pseudo-observations $\{u_t\}_{t\in\mathcal{T}^{tr}}$, $\{v_t\}_{t\in\mathcal{T}^{tr}}$
1: for $\nu \in \mathcal{V} := [1, 15]$ do
2:     Map pseudo-observations into Student-$t$ variates using the inverse CDF of a Student-$t$
       distribution with $\nu$ degrees of freedom, $t_\nu^{-1}(\cdot)$.
           $$\mathbf{x}_\nu \leftarrow \left\{t_\nu^{-1}(u_t)\right\}_{t\in\mathcal{T}^{tr}}, \quad \mathbf{y}_\nu \leftarrow \left\{t_\nu^{-1}(v_t)\right\}_{t\in\mathcal{T}^{tr}}$$
3:     Obtain the empirical covariance matrix
           $$\widehat{\mathbf{\Sigma}}(\nu) \leftarrow \begin{bmatrix} \hat{\sigma}_x^2(\nu) & \hat{\sigma}_{yx}(\nu) \\ \hat{\sigma}_{xy}(\nu) & \hat{\sigma}_y^2(\nu) \end{bmatrix} = \frac{1}{|\mathcal{T}_{tr}|} \begin{bmatrix} \mathbf{x}_\nu^\top \mathbf{x}_\nu & \mathbf{x}_\nu^\top \mathbf{y}_\nu \\ \mathbf{y}_\nu^\top \mathbf{x}_\nu & \mathbf{y}_\nu^\top \mathbf{y}_\nu \end{bmatrix}$$
4:     Evalutate the log-likelihood of the Student-$t$ copula
           $$\ell(\nu) \leftarrow \sum_{t\in\mathcal{T}_{tr}} \log c(u_t, v_t; \nu, \hat{\rho}(\nu)) \quad \text{where} \quad \hat{\rho}(\nu) \leftarrow \frac{\hat{\sigma}_{xy}(\nu)}{\hat{\sigma}_x(\nu)\hat{\sigma}_y(\nu)}$$
5: end for
6: Set $\nu_\star \leftarrow \arg\max_{\nu\in\mathcal{V}} \ell(\nu)$, and $\hat{\rho}_\star \leftarrow \hat{\rho}(\nu_\star)$
7: Compute Student-t copula density
       $$c(u, v; \nu_\star, \rho_\star) \leftarrow \frac{1}{\sqrt{1-\rho_\star^2}} \cdot \frac{\Gamma(\frac{\nu_\star+2}{2})\Gamma(\frac{\nu_\star}{2})}{\Gamma(\frac{\nu_\star+1}{2})^2} \cdot \frac{\left(1 + \frac{x^2+y^2-2\rho_\star xy}{\nu_\star(1-\rho^2)}\right)^{-\frac{\nu_\star+2}{2}}}{\left(1+\frac{x^2}{\nu_\star}\right)^{-\frac{\nu_\star+1}{2}} \cdot \left(1+\frac{y^2}{\nu_\star}\right)^{-\frac{\nu_\star+1}{2}}}$$ where $x \leftarrow t_\nu^{-1}(u)$,
   $y \leftarrow t_\nu^{-1}(v)$ and $t_\nu$ is the Student-$t$ CDF.
Ensure: Student-$t$ copula: $C(u, v; \rho_\star, \nu_\star) \leftarrow \int_u \int_v c(u, v; \nu_\star, \rho_\star) dv du$
```

Algorithm 4. Cumulative Mispricing Indices

```
Require: Out-of-sample pseudo observations $\{u_t\}_{t\in\mathcal{T}^{test}}$, $\{v_t\}_{t\in\mathcal{T}^{test}}$, calibrated Student-$t$ copula
    $C(u, v; \nu^\star, \rho^\star)$ and sequence of actions (trades): $\{a_t\}_{t\in\mathcal{T}^{test}}$
1: Initialize stuff:
       $CMI_0^{\text{trgt|synth}} \leftarrow 0, \quad CMI_0^{\text{synth|trgt}} \leftarrow 0$
2: for $t \in \mathcal{T}^{test} \setminus \{0\}$ do
3:     if $(a_{t-1} \neq 0) \wedge (a_t = 0)$ then
4:         Reset the CMIs
               $CMI_t^{\text{trgt}} \leftarrow 0, \quad CMI_t^{\text{synth}} \leftarrow 0$
5:     else
6:         Compute mispricing indices
               $MI_t^{\text{trgt|synth}} \leftarrow \frac{\partial C(u_t, v_t; \nu^\star, \rho^\star)}{\partial v_t} \quad MI_t^{\text{synth|trgt}} \leftarrow \frac{\partial C(u_t, v_t; \nu^\star, \rho^\star)}{\partial u_t}$
7:         Update cumulative mispricing indices
               $CMI_t^{\text{trgt}} \leftarrow CMI_{t-1}^{\text{trgt}} + (\text{MI}_t^{\text{trgt|synth}} - 0.5)$
               $CMI_t^{\text{synth}} \leftarrow CMI_{t-1}^{\text{synth}} + (\text{MI}_t^{\text{synth|trgt}} - 0.5)$
8:     end if
9: end for
Ensure: Sequence of cumulative mispricing indices: $\{CMI_t^{\text{trgt}}\}_{t\in\mathcal{T}^{test}}$, $\{CMI_t^{\text{synth}}\}_{t\in\mathcal{T}^{test}}$.
```

Third agent: *Trading Agent.*

The third agent is an intelligent trading agent that learns a policy π to map an observed state $\mathbf{s}_{t-1}$ to an optimal trading action a_t. The goal is to maximize a cumulative reward signal over time. This learning problem is formulated as a Partially Observed Markov Decision Process (POMDP), governed by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S} \subseteq \mathbb{R}^{2(2+w)}$ is the state space, $\mathcal{A} \in \{-1, 0, 1\}$ is the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition kernel, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\gamma \in [0, 1]$ is the discount factor. We solve this MDP using a Reinforcement Learning (RL) approach, specifically, a Deep Q-Network (DQN).

State space At the beginning of each time step t, before an action is taken, the agent observes a state vector $\mathbf{s}_{t-1} \in \mathcal{S}$. This vector is constructed from information available up to the end of period $t-1$:

$$\mathbf{s}_{t-1} = \left[CMI_{t-1}^{\text{trgt}|\text{synth}},\ CMI_{t-1}^{\text{synth}|\text{trgt}},\ a_{t-1},\ \tau_{t-1},\ \mathsf{r}_{t-1}^{\text{trgt}},\ \mathsf{r}_{t-1}^{\text{synth}},\right]^{\top}$$

where $a_{t-1} \in \mathcal{A}$ is the trading position taken by the agent in the previous period, $\tau_t \in \mathbb{N}$ is the normalized duration for which the position a_{t-1} has been held $\{\mathsf{r}_\ell^{\text{trgt}}\}_{\ell=t-w}^{t-1}$ and $\{\mathsf{r}_\ell^{\text{synth}}\}_{\ell=t-w}^{t-1}$ are the sequences of the w most recent returns for the target and synthetic assets, respectively.

Action After observing $\mathbf{s}_{t-1}$, the agent chooses an action $a_t \sim \pi(\mathbf{s}_{t-1}) \in \{-1, 0, 1\}$, according to some policy function $\pi : \mathcal{S} \to \mathcal{A}$, where $a_t = 1$ implies going long on target and short on synthetic, $a_t = -1$ implies going short on target and long on synthetic, and $a_t = 0$ implies no position.

Reward After taking action a_t, returns for that period are realized $(r_t^{\text{trgt}}, r_t^{\text{synth}})$ and the agent computes the period's reward as

$$\mathcal{R}_t = \mathcal{R}(\mathbf{s}_t, a_t, a_{t-1}) = a_t(r_t^{\text{trgt}} - r_t^{\text{synth}}) - \kappa|a_t - a_{t-1}|$$

where $\kappa|a_t - a_{t-1}|$ are the transaction costs and κ is set to $3n$ bps, with n being the total number of stocks traded (one stock for the target plus the number of stocks that build the synthetic control).

Objective The agent's objective is to learn the trading policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes the expected cumulative discounted reward:

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(\mathbf{s}_t, a_t, a_{t-1})\right]$$

Value function We solve the MDP using the Deep Q-Network (DQN) algorithm, which approximates the optimal action-value function $Q^*(\mathbf{s}, a)$, defined by the Bellman optimality equation:

$$Q^*(\mathbf{s}, a) = \mathbb{E}\left[\mathcal{R} + \gamma \max_{a' \in \mathcal{A}} Q^*(\mathbf{s}', a')\right],$$

where $\mathbf{s}' = \mathbf{s}_{t+1}$ is the next state. The DQN parameterizes $Q_{\boldsymbol{\theta}}(\mathbf{s}, a)$ with a multi-layer perceptron (MLP), where $\boldsymbol{\theta}$ are the network parameters.

Experience Replay To break the temporal correlation in the observation sequence and improve learning stability, we use experience replay. Transitions $(s_t, a_t, \mathcal{R}_t, s_{t+1})$ are stored in a replay buffer $\mathcal{D}$ of capacity N. During training, we sample mini-batches $\mathcal{B} \subset \mathcal{D}$ of transitions uniformly at random from the replay buffer:

$$\mathcal{B} = \{(s_j, a_j, r_j, s_{j+1})\}_{j=1}^{|\mathcal{B}|} \sim \mathcal{U}(\mathcal{D}) \tag{1}$$

Target Network To further stabilize learning, we use a target network with parameters $\boldsymbol{\theta}^-$ that are periodically updated to match the main network parameters $\boldsymbol{\theta}$. The target network is used to compute the target values for the Q-learning update:

$$y_t = \mathcal{R}_t + \gamma \max_{a' \in \mathcal{A}} Q_{\boldsymbol{\theta}^-}(\mathbf{s}_{t+1}, a') \tag{2}$$

Loss Function The Q-function is updated by minimizing the mean squared error between the predicted Q-values and the target values, known as temporal-difference (TD) loss.

$$\begin{aligned}\mathcal{L}(\theta) &= \mathbb{E}_{(\mathbf{s},a,r,\mathbf{s}')\sim\mathcal{B}}[(y_t - Q_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t))^2] \\ &\approx \frac{1}{|\mathcal{B}|}\sum_{j\in\mathcal{B}}(y_j - Q(\mathbf{s}_j, a_j; \boldsymbol{\theta}))^2\end{aligned}$$

The parameters $\boldsymbol{\theta}$ are updated using gradient descent:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \tag{3}$$

where α is the learning rate.

Exploration Strategy To ensure adequate exploration of the state-action space, an ϵ-greedy policy is employed during training. With probability ϵ, a random action is selected; otherwise, the action with the highest Q-value is chosen: $a_t = \arg\max_a Q_{\boldsymbol{\theta}}(\mathbf{s}_{t-1}, a)$. The value of ϵ is typically annealed from an initial value (e.g., 1.0) to a final value (e.g., 0.05) over a portion of the training steps (e.g., the first 20% of total timesteps).

The probability of selecting a random action decreases linearly from ϵ_0 to ϵ_f over a fraction of the total training steps:

$$\epsilon_t = \begin{cases} \epsilon_0 - (\epsilon_0 - \epsilon_f) \cdot \frac{t}{T_{\text{explore}}} & \text{if } t < T_{\text{explore}} \\ \epsilon_f & \text{otherwise} \end{cases} \tag{4}$$

where T_{explore} = exploration_fraction $\cdot$ T_{total}, T_{total} is the total number of training steps, and exploration_fraction is a hyperparameter.

The action selection policy during training is:

$$\pi : \begin{cases} \textit{random action} & \sim \mathcal{U}(\mathcal{A}) & \text{w/ prob } \epsilon_t \\ \textit{greedy action} & \arg\max_a Q_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t) & \text{w/ prob } 1 - \epsilon_t \end{cases}$$

Implementation Details The implementation details of the RL algorithm are as follows:

- **Neural Network Architecture:** The neural network used is a Multi-Layer Perceptron (MLP) with two hidden layers, each with 64 units.
- **Optimizer:** The Adam optimizer is used with a learning rate of 1×10^{-4}.
- **Experience Replay Buffer:** The experience replay buffer has a capacity of 50,000 experiences.
- **Exploration:** The exploration strategy used is ϵ-greedy, with ϵ decaying from 1.0 to 0.05 over the first 20% of the training steps.
- **Training Steps:** The agent is trained for 100,000 time steps.
- **Evaluation Frequency:** The agent is evaluated every 10,000 time steps.

The trading agent is trained using the DQN algorithm, which learns an optimal policy by approximating the action-value function $Q(\mathbf{s}, a)$ with a multi-layer perceptron (MLP). Key hyperparameters include a learning rate of 10^{-4}, a replay buffer size of 50,000, an exploration fraction of 0.2 with initial and final epsilon values of 1.0 and 0.05 respectively, a discount factor $\gamma = 0.99$, and a target network update interval of 1,000 steps. Training occurs over a total of 100,000 timesteps, with periodic checkpointing every 10,000 steps to save model progress. The environment is wrapped with a `Monitor` to track performance metrics during training, and random seeds are set to ensure reproducibility.

The agent employs an ϵ-greedy exploration strategy, with ϵ decaying linearly from 1.0 to 0.05 over the first 20% of training steps (i.e., 20,000 out of 100,000 total timesteps). Key hyperparameters include:

- Learning rate: 1×10^{-4},
- Replay buffer size: 50,000,
- Batch size: 64,
- Discount factor: $\gamma = 0.99$,
- Training frequency: Every 4 steps,
- Learning starts: After 1000 steps.

The implementation uses the Stable Baselines3 library, with the "MlpPolicy" specifying a default MLP architecture suited to the state dimensionality.

Training The agent is trained on historical in-sample returns data for 100,000 timesteps, interacting with a custom Gym environment that simulates the pairs trading dynamics. The environment resets the state at the beginning of each episode, initializing $CMI = 0$, $a_0 = 0$, $\tau_0 = 0$, and portfolio value $V_0 = 1$. Training leverages experience replay, storing transitions $(\mathbf{s}_t, a_t, r_t, \mathbf{s}_{t+1})$ in the buffer and sampling minibatches to update θ.

Post-training, the agent is evaluated deterministically (i.e., $a_t = \arg\max_a Q(\mathbf{s}_t, a; \theta)$) on out-of-sample test data.

1.0.2 Training algorithm

Algorithm 5. Deep Q–Learning for Pairs Trading

```
1:  Initialise replay buffer $\mathcal{D} \leftarrow \emptyset$
2:  Initialise networks $Q(\,\cdot\,;\theta)$ and $Q(\,\cdot\,;\theta^-)$ with $\theta^- \leftarrow \theta$
3:  Reset environment, obtain $\mathbf{s}_w$
4:  for $t = w, w+1, \ldots, T-1$ do
5:      Draw action $a_t \sim \pi_{\varepsilon_t}(\cdot \mid \mathbf{s}_t)$
6:      Execute $a_t$, observe $r_t$, $\mathbf{s}_{t+1}$, terminal flag $d_t$
7:      Store transition in buffer $\mathcal{D}$
8:      if $|\mathcal{D}| >$ learning_starts then
9:          Sample minibatch $\mathcal{B} \subset \mathcal{D}$
10:         Compute targets $\{y_i\}_{i\in\mathcal{B}}$ and update $\theta$ using $\nabla_\theta \mathcal{L}$
11:     end if
12:     if $t \mod \tau_{\text{target}} = 0$ then
13:         $\theta^- \leftarrow \theta$
14:     end if
15: end for
```

Result Through iterative interaction with the trading environment and updates to its Q-network, the agent learns a policy $\pi(\mathbf{s}) = \arg\max_a Q^*_{\boldsymbol{\theta}}(\mathbf{s}, a)$ that aims to maximize long-term cumulative rewards.

References

A. Appendix

A.1 Cointegration Meets Synthetic Controls: A Formal Equivalence

In this appendix section, we develop a formal argument showing how, under some stringent assumptions, our notion of *synthetic control* can be viewed as a special case of *cointegration.* This connection underlies the intuition that, when one normalizes the first variable of a cointegrated system to 1, the remaining cointegration relationships effectively produce the *synthetic* version of the first variable when the cointegration vector satisfies a specific restriction.

Let $\{y_{i,t}\}_{t=1}^{T}$ denote the time series sequence of log-prices for each asset $i \in \{1, \dots, N\}$. Throughout, we assume each $y_{i,t}$ is an $I(1)$ process (integrated of order 1). Formally, an $I(1)$ process is one that becomes *stationary* (and typically ergodic) upon differencing once: $\Delta y_{i,t} := y_{i,t} - y_{i,t-1} \sim I(0)$. The notion of cointegration, due to Engle and Granger, is central in analyzing potentially long-run equilibria among these variables.

Definition 1 (Engle and Granger (1987))**.** *The components of* $\mathbf{y}_t := [y_{1t}, ..., y_{Nt}]$ *are said to be cointegrated of order d, b, denoted* $\mathbf{y}_t \sim CI(d, b)$*, if (a) all components of* $\mathbf{y}_t$ *are* $I(d)$ *and (b) a vector* $\boldsymbol{\beta} \neq 0$ *exists so that* $\boldsymbol{\beta}'\mathbf{y}_t \sim I(d - b)$*,* $b > 0$*. The vector* $\boldsymbol{\beta}$ *is called the cointegrating vector.*

Definition 2 (Synthetic Control)**.** *Let* $\{y_1, y_2, \dots, y_n\}$ *be a collection of random variables, where* y_1 *is the "target" variable and* $\mathbf{y}_{2:n} = (y_2, \dots, y_n)$ *constitute the "donor pool". A* synthetic control *for* y_1 *is constructed by choosing weights* $\mathbf{w}$ *in the* $(n-1)$*-dimensional space* $\mathcal{W} := \{\mathbf{w} \in \mathbb{R}_+^{n-1} : \sum_{j=2}^{n} w_j = 1\}$ *that satisfy* $\mathbf{w} = \arg\min_{w \in \mathcal{W}} \sum_{t=1}^{T} (y_{1,t} - \mathbf{w}'\mathbf{y}_{2:n,t})^2$.

Given that cointegration relationships prevail up to scale and sign changes, then, under suitable conditions on the cointegration vector, there exists a nontrivial constant κ that allows us to reinterpret the cointegration relationship as one of a synthetic control. In particular,

Proposition 1. *For a cointegrated vector* $\mathbf{y}$ *with rank r, if (at least) one of the cointegrating vectors* $\boldsymbol{\beta}$ *satisfies the restriction* $\mathcal{R} = \{\mathbf{1}'\boldsymbol{\beta} = 0\}$*, then we can scale the cointegration vector by* $\kappa = 1/\beta_i$ *such that* $\kappa\boldsymbol{\beta}'\mathbf{y}$ *is stationary and describes a "synthetic control" relationship (as per Definition 2) between* y_i *and* $\mathbf{y}_{-i}$*.*

Proof. The proof is straightforward. For a cointegration vector $\boldsymbol{\beta}$ where $\mathcal{R}$ holds, we have that $\mathbf{1}'\boldsymbol{\beta} = \sum_{j=1}^{n} \beta_j = 0$, which trivially implies $\beta_i = -\sum_{j \neq i} \beta_j$. For the sake of the proof, set that β_i to the first component (β_1). Then $\beta_1 = -\sum_{j=2}^{n} \beta_j$ and $\kappa = (\beta_1)^{-1} = -(\sum_{j=2}^{n} \beta_j)^{-1}$

$$\kappa\boldsymbol{\beta}'\mathbf{y} = \frac{1}{\beta_1}[\beta_1 \quad \boldsymbol{\beta}_{2:n}]\mathbf{y}_t = \begin{bmatrix} 1 & \frac{-\boldsymbol{\beta}'_{2:n}}{\sum_{j=2}^{n} \beta_j} \end{bmatrix} \begin{bmatrix} y_1 \\ \mathbf{y}_{2:n} \end{bmatrix} = y_1 - \frac{\beta_2}{\sum_{j=2}^{n} \beta_j} y_2 - \dots - \frac{\beta_n}{\sum_{j=2}^{n} \beta_j} y_n \sim I(0)$$

describes a stationary cointegration relationship in $\mathbf{y}$, and since

$$\begin{aligned} y_1 &= \frac{\beta_2}{\sum_{j=2}^n \beta_j} y_2 + \cdots + \frac{\beta_n}{\sum_{j=2}^n \beta_j} y_n + \epsilon \\ &= \mathbf{w}' \mathbf{y}_{2:n} + \epsilon \end{aligned}$$

with $\epsilon \sim I(0)$ and $\mathbf{w} := \left(\frac{\beta_2}{\sum_{j=2}^n \beta_j}, ..., \frac{\beta_n}{\sum_{j=2}^n \beta_j} \right)' \in \mathcal{W}$, then this relationship is endowed with a synthetic control structure. A similar reasoning applies to any other β_i different from β_1. $\square$

A.2 Algorithms