You are tasked with designing a sophisticated algorithmic trading strategy to exploit volatility arbitrage opportunities across $N = 28$ currency pairs, denoted as $C_1, C_2, \ldots, C_{28}$. The market dynamics are nonlinear and exhibit temporal dependencies. The relationships among currency pairs are modeled using:

1. **Enhanced PCA** for feature extraction from the volatility matrices.
2. **Graph Neural Networks (GNNs)** to refine interdependencies and generate probabilistic outputs for trading actions.
3. **Cross-Entropy Similarity** to dynamically refine the relationships between GNN outputs for enhanced decision-making.

---

1. Inputs

Volatility Inputs

The volatility of a currency pair C_i at time t, denoted as $\sigma_i(t)$, is calculated as the rolling standard deviation over the past L time steps of its log returns:

$$\sigma_i(t) = \sqrt{\frac{1}{L} \sum_{k=1}^{L} (r_i(t-k) - \mu_i)^2} \quad ,$$

where:

- $r_i(t-k)$: Log returns for C_i at time $t-k$,
- μ_i: Mean of log returns over the window of L.

---

2. Kernel Transformation

Volatility Spread Matrix

Define the volatility spread matrix V where:

$$V_{ij}(t) = |\sigma_i(t) - \sigma_j(t)|, \quad \forall i,j \in [1,N].$$

Kernel Matrix

To capture nonlinear dependencies between volatilities, apply a Gaussian kernel to the volatility spread matrix:

$$K_{ij}(t) = \exp\left(-\frac{(V_{ij}(t))^2}{2 \cdot \sigma_K^2}\right),$$

where:

- σ_K: Kernel width parameter controlling sensitivity.

---

3. Enhanced PCA

Kernel PCA Decomposition

Decompose the kernel matrix $K(t)$ using eigenvalue decomposition:

$$K(t) = \sum_{k=1}^{N} \lambda_k \cdot v_k \cdot v_k^\top,$$

where:

- λ_k: Eigenvalues,
- v_k: Eigenvectors.

Principal Component Selection

Select the top m-principal components that explain at least 90% of the variance:

$$\frac{\sum_{k=1}^{m} \lambda_k}{\sum_{k=1}^{N} \lambda_k} \geq 0.90.$$

The reduced feature matrix for currency pair C_i is:

$$R_i(t) = \{\lambda_1 \cdot v_1, \lambda_2 \cdot v_2, \ldots, \lambda_m \cdot v_m\}.$$

---

4. Graph Neural Networks

Adjacency Matrix Construction

The adjacency matrix A_{ij} for each GNN is constructed from the volatility spread matrix:

$$A_{ij} = \exp\left(-\frac{\|R_i(t) - R_j(t)\|^2}{2 \cdot \sigma_A^2}\right), \quad \forall i, j.$$

Feature Propagation

Each GNN processes the PCA-reduced features through L-layers of propagation:

$$H_i^{(l+1)} = \mathrm{ReLU}\left(\sum_j A_{ij} \cdot H_j^{(l)} \cdot W^{(l)}\right),$$

where:

- $H_i^{(l)}$: Feature vector for C_i at layer l,
- $W^{(l)}$: Learnable weight matrix at layer l,
- $\mathrm{ReLU}(x) = \max(0, x)$: Activation function.

Probability Outputs

The final GNN output for C_i is a probability vector:

$$p_i = \{p_i^{\text{Buy}}, p_i^{\text{Sell}}, p_i^{\text{Hold}}\}, \quad \sum_k p_i^{(k)} = 1.$$

---

5. Cross-Entropy Similarity Between GNNs

Cross-Entropy Similarity

Define the similarity between GNN outputs for pairs C_i and C_j as:

$$\mathrm{Sim}_{ij} = -\sum_k p_i(k) \log p_j(k).$$

Similarity Matrix

Construct a similarity matrix S where:

$$S_{ij} = \mathrm{Sim}_{ij}, \quad \forall i, j.$$

---

6. Refinement Using Similarity

Refined Outputs

Use the similarity matrix S to refine GNN outputs:

$$p_i^{\text{refined}}(k) = \frac{\sum_j S_{ij} \cdot p_j(k)}{\sum_j S_{ij}}.$$

Signal Calculation

Compute the trading signal strength for C_i:

$$\text{Signal}_i = p_i^{\text{refined}}(\text{Buy}) - p_i^{\text{refined}}(\text{Sell}).$$

---

7. Trading Decisions

Dynamic Threshold

Define a dynamic threshold for trade execution:

$$\Theta(t) = \mu(\text{Signal}) + \delta \cdot \sigma(\text{Signal}),$$

where:

- $\mu(\text{Signal})$: Mean of signals across all pairs,
- $\sigma(\text{Signal})$: Standard deviation of signals,
- δ: Risk-adjustment coefficient.

Trade Execution

Execute trades based on normalized signals:

$$\text{Trade}_i = \begin{cases} \text{Buy}, & \text{if Signal}_i > \Theta(t), \\ \text{Sell}, & \text{if Signal}_i < -\Theta(t), \\ \text{Hold}, & \text{otherwise.} \end{cases}$$

---

8. Optimization

Sharpe Ratio

Maximize the Sharpe Ratio:

$$\text{Sharpe Ratio} = \frac{\text{Mean(Returns)} - \text{Risk-Free Rate}}{\text{StdDev(Returns)}}.$$

Variance Explained

Ensure the top m-components explain at least 90% of the variance:

$$\frac{\sum_{k=1}^{m} \lambda_k}{\sum_{k=1}^{N} \lambda_k} \geq 0.90.$$

Dynamic Similarity Updates

Adapt the similarity matrix S as market conditions evolve:

$$S_{ij}(t+1) = (1-\alpha) \cdot S_{ij}(t) + \alpha \cdot \text{Sim}_{ij}(t),$$

where α is a smoothing factor.

Problem Objectives

1. **Maximize Profitability**: Generate cumulative profits while maintaining a Sharpe Ratio above 2.0.
2. **Minimize Risk**: Ensure maximum drawdown remains below 5% of equity.
3. **Exploit Arbitrage Opportunities**: Identify nonlinear relationships in volatility spreads.
4. **Dynamic Adaptability**: Continuously update inter-GNN relationships using Cross-Entropy Similarity.