

The Quantum Market Entanglement Problem

Context:

The global forex market, an entangled system of 28 interconnected currency pairs, exhibits nonlinear dynamics and temporal dependencies driven by volatility. Traders seek to exploit arbitrage opportunities using advanced computational techniques, yet the complex interrelationships between currencies remain an unsolved puzzle.

Imagine a scenario where the market behaves like a quantum system: each currency pair's volatility interacts probabilistically with others, creating a web of dependencies that evolve over time. Our goal is to decode this quantum-like entanglement, predict volatility-driven arbitrage opportunities, and act upon them.

Problem Statement:

Given 28 currency pairs $\{C_1, C_2, \dots, C_{28}\}$, each characterized by its time-varying volatility $\sigma_i(t)$, determine the **optimal arbitrage strategy** by solving the following challenges:

Step 1: Dynamic Volatility Spread Web

- Volatility Evolution:** For each currency pair C_i , the volatility at time t , $\sigma_i(t)$, is modeled as:

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

where $r_i(t)$ are log returns, μ_i is the mean of returns over a window L .

- Volatility Spread Matrix:** Define the pairwise volatility spread at time t as:

$$V_{ij}(t) = |\sigma_i(t) - \sigma_j(t)|, \quad \forall i, j \in \{1, 2, \dots, 28\}.$$

- Gaussian Kernel Transformation:** Capture nonlinear dependencies between pairs using:

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

where σ_K is a kernel width parameter. The result is a kernel matrix $K(t)$.

Twist:

The volatility spreads are **dynamic** and exhibit time-lagged correlations. The challenge is to model their interaction **over time**, akin to particles influencing each other in a quantum field.

Step 2: Enhanced PCA to Extract Latent Features

- Eigenvalue Decomposition:** Perform eigenvalue decomposition of the kernel matrix $K(t)$:

$$K(t) = \sum_{k=1}^{28} \lambda_k \cdot v_k v_k^T,$$

where λ_k are eigenvalues and v_k are eigenvectors.

- Latent Feature Extraction:** Select the top m eigenvalues such that:

$$\frac{\sum_{k=1}^m \lambda_k}{\sum_{k=1}^{28} \lambda_k} \geq 90\%.$$

Form a reduced feature matrix:

$$R_i(t) = \{\lambda_1 v_1, \lambda_2 v_2, \dots, \lambda_m v_m\}, \quad \forall i.$$

Twist:

The eigenvectors v_k represent **hidden market states** that encode inter-pair volatility dependencies. Extracting these features is analogous to identifying "hidden variables" in quantum systems.

Step 3: Graph Neural Networks for Market Entanglement

- Adjacency Matrix:** Construct the adjacency matrix $A(t)$ using the reduced features:

$$A_{ij}(t) = \exp\left(-\frac{\|R_i(t) - R_j(t)\|_2^2}{2\sigma_A^2}\right),$$

where σ_A controls sensitivity.

- Feature Propagation:** For L layers, propagate features through the network:

$$H_i^{(l+1)} = \text{ReLU}\left(\sum_{j=1}^{28} A_{ij} \cdot H_j^{(l)} \cdot W^{(l)}\right),$$

where $H_i^{(0)} = R_i(t)$, $W^{(l)}$ are learnable weights, and $\text{ReLU}(x) = \max(0, x)$.

3. **Output Probabilities:** The final layer outputs probabilities for each trading action:

$$p_i = \{p_i^{\text{Buy}}, p_i^{\text{Sell}}, p_i^{\text{Hold}}\}, \quad \sum p_i = 1.$$

Twist:

The GNN layers **amplify latent interactions** among currency pairs, analogous to how quantum states become entangled. The adjacency matrix $A(t)$ evolves over time, capturing market dynamics.

Step 4: Cross-Entropy Similarity for Dynamic Refinement

1. **Cross-Entropy Similarity:** Measure similarity between outputs of currency pairs:

$$\text{Sim}_{ij}(t) = - \sum_{k=1}^3 p_i(k) \log p_j(k).$$

2. **Similarity Matrix:** Construct the similarity matrix $S(t)$:

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

where α is a smoothing factor.

3. **Refined Outputs:** Refine GNN outputs using:

$$p_i^{\text{refined}}(k) = \frac{\sum_{j=1}^{28} S_{ij} \cdot p_j(k)}{\sum_{j=1}^{28} S_{ij}}.$$

Twist:

Cross-entropy similarity dynamically adjusts the interdependencies, simulating quantum decoherence as new information enters the system.

Step 5: Trading Decision and Optimization

1. **Signal Calculation:** Compute the trading signal for each pair:

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

2. **Threshold-Based Execution:** Execute trades based on dynamic thresholds:

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

where δ is a risk-adjustment coefficient.

3. **Profitability Maximization:** Maximize the Sharpe Ratio:

$$\text{Sharpe Ratio} = \frac{E[\text{Returns}] - r_f}{\sigma(\text{Returns})},$$

subject to constraints on drawdown and execution latency.

Twist:

The strategy's adaptability mirrors quantum systems' probabilistic nature, where predictions evolve with observations.

Challenge for the Reader:

- How does the similarity matrix $S(t)$ impact the refinement process, and can it lead to a situation where currency pairs cluster into "volatility regimes"?
- If $S(t)$ were treated as a quantum density matrix, could the problem be reframed using quantum computation principles?