Implement and Fine-Tune Transformer-Based Model

Dataset Description

The dataset utilized for the trading model is composed of high-frequency trading data, which inherently contains numerous instances where market prices exhibit minimal changes within short time intervals. To effectively analyze this data:

- **Indicators**: We employed a set of technical indicators from the PPO sample implementation. These include RSI, MACD, Stochastic Oscillators, and others, providing a comprehensive view of market dynamics.
- **Signal Split**: The signals were categorized based on price movement thresholds (e.g., 0.0001 or 0.01%). This thresholding is essential for handling the high-frequency nature of the data, where significant trading decisions hinge on minimal price changes. This led to a distribution of trading signals heavily skewed towards **'hold'** signals, which makes practical sense given the minor fluctuations typical of such datasets.
- **Data Temporality**: Given the sequential nature of the data, which already encapsulates temporal information, additional temporal processing was not necessary for the initial implementation.

Model Overview

The trading model is built using a **Transformer architecture**, known for its effectiveness in handling sequential data through self-attention mechanisms. The model specifics are as follows:

- **Architecture**: The Transformer model with parameters tuned for this particular application—input dimension of 5, feature size of 5, 1 attention head, 6 layers, and a feed-forward dimension of 256.
- **Learning Strategy**: Utilization of the Adam optimizer with a Cosine Annealing learning rate scheduler to adaptively adjust the learning rate, enhancing the model's ability to converge to optimal weights.

Training Process and Hyperparameters

The model was trained with a focus on optimizing various parameters to enhance its performance:

- **Number of Features:** The number of features from the indicators were tested starting from 14 and reaching a final of 5. With 5 features, there was a good fit model getting trained, a lot of features were causing large model overfits.
- **Batch Size and Epochs**: After experimenting with different batch sizes and epochs, a size of 128 was found to be optimal. The model was trained over 500 epochs, and the performance metrics were closely monitored through the loss and accuracy graphs.
- **Performance Metrics**: The final model achieved an accuracy of approximately 82.5% on the test set. The use of a validation set helped in fine-tuning and ensuring the model did not overfit to the training data.

Evaluation and Validation

The model's validation involved assessing its prediction capabilities on unseen test data:

- **Trading Environment with Blotter:** The environment used Simple Moving Average for executing trade suggestions, based on the model predictions from the trained model.
- **Accuracy Analysis**: Due to the type of data we trained the model on, there were mostly hold suggestions, which made sense given the very minute price fluctuations.
- **Trading Suggestions**: The model effectively suggested 'hold' actions during periods of minimal price change, and appropriately recommended 'buy' or 'sell' actions when the indicators identified potential opportunities.

Example Suggestions

1. `Hold at 192.47999572753906 per share`

   `Buy 259 shares at 192.47000122070312 per share`

   `Buy 258 shares at 192.47000122070312 per share`

   `Hold at 192.47000122070312 per share`

2. `Hold at 192.52000427246094 per share`

   `Hold at 192.52000427246094 per share`

   `Sell 259 shares at 192.52499389648438 per share`

This shows how effectively the model outputs hold or sell, when there's in prices, the action goes from Hold to Buy. During an increase in share price, the action is to sell shares.