Cryptocurrency Pairs Trading with Co-integration and LSTM

Joshua Rosaler

Project Overview

In recent decades, pairs trading has been an extremely profitable trading strategy used by banks, hedge funds, and proprietary trading firms. The idea of the strategy is to find pair of assets such that the price of a certain linear combination of these assets is stationary. Stationarity entails that the price of a certain linear combination of assets oscillates around a fixed value, such that substantial deviations of the linear combination from a mean value can be exploited in cases where the price deviates significantly from this value. More precisely, one trades on the spread between two assets, buying or "going long" on the spread when it falls below the mean and selling or "shorting" the spread when it goes above the mean. A classic example used to illustrate pairs trading is the case of Coke and Pepsi: because the values of these two companies are driven by largely the same set of factors, they tend on the whole to vary together. When, say, Coke is usually around $3 more than Pepsi and this spread goes up to $6 (say, because Coke went up $2 and Pepsi down $1), we can short Coke and buy Pepsi in expectation that the spread will revert to its usual value of $3.

Problem Statement

There are many strategies used for pairs trading. The goal of this project is to determine whether recurrent neural networks (RNN's) and specifically Long Short Term Memory (LSTM) neural networks can be used to improve on the performance of one simple but popular method, known as the "distance method." We will investigate this question in the specific context of cryptocurrencies, in order to examine the profitability of LSTM-driven pairs trading strategies in this market. In particular we will examine whether LSTM based models of the spread exceed the performance of ARMA models used for the same purpose in terms of ordinary statistical measures like mean squared error. Employing comparable trading strategies based on these models, they will be compared in terms of financial metrics like Sharpe ratio and cumulative returns.

The will be approached as a **nonlinear regression** problem, in which an LSTM model will be used to predict the spread of a co-integrated pair of cryptocurrency assets on the basis of the past history of the spread. The co-integration will be performed by means of the usual Engle-Granger two step method in which ordinary least squares regression is performed of one price series against the other(s), and then the resulting linear combination of assets is tested for stationarity using the Augmented Dickey Fuller test. The input into the models used to predict the spread will be some number p lags of the spread time series (to be determined via trial and error) and the output will be a prediction of the price at the next time, or at some number of time steps into the future. It will be fit to data using using the mean squared error (mse) metric.

Data

Daily price data (Open, High, Low, Close) for several cryptocurrencies over several years, drawn from Kaggle. Following (Leung & Nguyen 2018), who investigate the construction of cryptocurrency portfolios for statistical arbitrage, we focus on Bitcoin, Ethereum, Litecoin, and Dash. All price histories considered end in February 2018. Bitcoin price data goes back to April 2013, giving 1,760 days of price data. Ethereum data goes back to Aug. 2015, giving 929 days of price data. Litecoin data goes back to April 2013, giving 1,760 days of price data. Dash data goes back to Feb. 2014, giving 1468 days of price data.

For the train/validation/test split we will find a span of about 929 days (the number of days of data for ethereum, the cryptocurrency for which there is the least data) using a roughly 60/20/20 split. Each training observation will be a window of, say, 100 days (this will be determined by more careful examination of the data). So, with a window of 100 days, the first 600 days will be used for training, giving roughly 500 training examples, each consisting of 100 days of closing price data as the features x and the following day's closing price as the label y. The next 200 days will be used for validation, and the following 129 days for testing. This will help to avoid look-ahead bias.

Benchmark Model

The base model against which the LSTM model of the spread series will be compared will be an **ARMA model**, which, like the LSTM model that we test, will take a certain number of lagged values as input and output a predicted price. Thus, both base model and LSTM model being tested are regression models.

Metrics

The LSTM and ARMA models will be fit using an **mean squared error** metric. Starting with the same initial portfolio, constructed via cointegration methods, the two strategies will be compared according to their cumulative returns and Sharpe ratios.

Strategy

The strategy that we will use to investigate this question was originally sketched in (Dunnis 2006, 2015) and further developed in (van der Have 2017). The detailed implementation, however, is original, although for various specific sub-tasks it synthesizes and modifies code drawn from various different sources, including Quantopian for the selection of cointegrated pairs and Jason Brownlee's Machine Learning blog for time series prediction of the spread with LSTM. The crytocurrency data is taken from Kaggle. Tips on construction of co-integrated cryptocurrency portfolios are taken from (Leung & Nguyen 2018).

Plan:

Part 1 - Data importing and transformation
Retrieve cryptocurrency data and put it into a Pandas data frame. Decide whether to use close price for each day, or some average of open, high, low, close. Transform price data to log price data.

Part 2 - Find co-integrated pairs
Use Engle-Granger method to find co-integrated pairs and associated cointegration vectors- adapt code from Quantopian, use statsmod library's coint function.

Part 3 - Portfolio construction.
Either choose one single co-integrated pair for which to model spread, or form portfolio of different pairs. I still haven't decided which; I may stick to first to keep things simple.

Part 4 - Benchmark Model: ARMA model
Use ARMA model to predict the spread. Find common trading strategy to use for both ARMA and LSTM. Use threshold filter from (Dunnis 2006): if the predicted return on the spread is greater than some chosen threshold X, go or stay long the spread; if the predicted return on the spread is below -X, go or stay short the spread. Try different X and see which maximize cumulative returns and Sharpe ratio on validation set.

Part 5 - Test model: LSTM
Train and test different LSTM models (see Brownlee for forecast details). Try different numbers of layers, neurons, lags for input series. Use same trading strategy as for ARMA model. Try different trading thresholds X, see which give best cumulative returns and Sharpe ratio.

Part 6 - Analysis: Model Comparison
Which model made better predictions according to the chosen mean squared error metric? Which model gave higher cumulative returns? Higher Sharpe ratio?

Sources

Overall approach adapted from:

- https://thesis.eur.nl/pub/41548/Have-van-der.pdf,
- https://www.researchgate.net/publication/327163916_Constructing_Cointegrated_Cryptocurrency_Portfolios_for_Statistical_Arbitrage
- https://link.springer.com/article/10.1057/palgrave.dutr.1840046
- https://link.springer.com/article/10.1057/palgrave.dutr.1840046

Cryptocurrency Data:

- https://www.kaggle.com/sudalairajkumar/cryptocurrencypricehistory/version/13

Cointegration Code:

- https://www.quantopian.com/lectures/integration-cointegration-and-stationarity
- https://www.quantopian.com/lectures/example-basic-pairs-trading-algorithm
- https://www.quantopian.com/lectures/example-pairs-trading-algorithm

LSTM time series prediction (for use on spreads of cointegrated pairs):

- https://github.com/jaungiers/LSTM-Neural-Network-for-Time-Series-Prediction
- https://machinelearningmastery.com/time-series-forecasting-long-short-term-memory-network-python/

Other:

- Vidyamurthy, Ganapathy. *Pairs Trading: Quantitative Methods and Analysis*, Wiley Finance, 2004.
- http://aaaquants.com/2017/08/27/machine-learning-lstm-networks/#comment-72