Dexterity Capital Trading Algorithm Project

Zian (Shawn) Cheng
Management Science and Engineering
M.S. 2019, Columbia University

Table of Contents

Overview

- The goal is to develop a trading algorithm such that the ending balance of a trading day is maximized, given the starting balance of $25,000 (USD) and 0 BTC
- Attention needs to be paid to prevent overfitting
- Initial algorithm generated Decision Set 1 with an ending balance of $25701.98
- Improved algorithm generated Decision Set 2 with an ending balance of $25721.97

Data Exploration and Visualization - Arbitrage

- Plot of time series of prices for Exchange 1, 2 and 3; found arbitrage opportunities

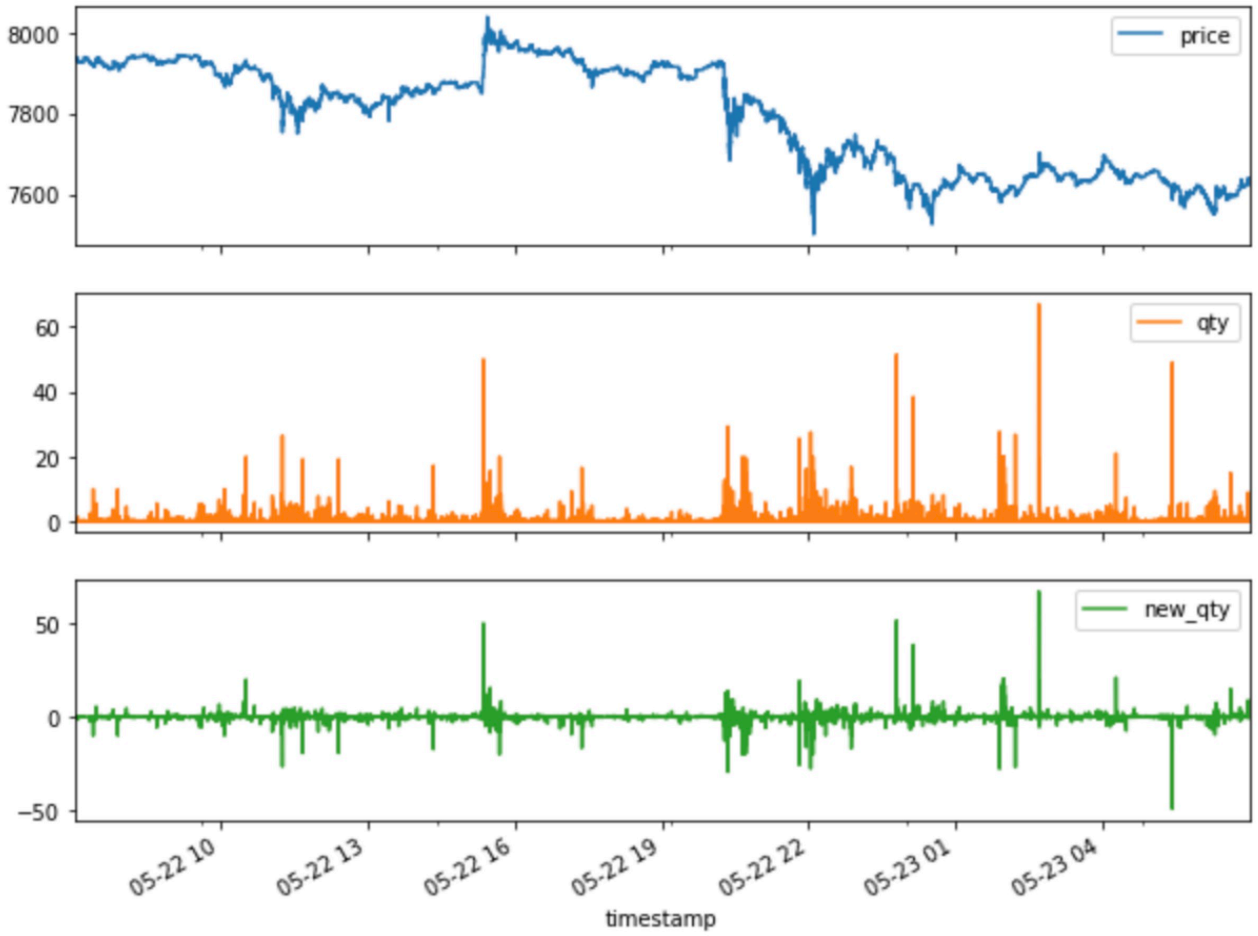
Data Exploration and Visualization

- Created “new_qty” = qty when side == “a”; “new_qty” = (-1) * qty when side == “b” (for easier Data Manipulation and Visualization)

- Tried using Relative Strength Index (RSI) as the signal for bid and ask; turned out to be a lagging indicator

Data Exploration and Visualization

- Time series plots of price, qty and new_qty for Exchange 1; discovered huge increase/drop in bid/ask volume before a huge price change



Data Exploration and Visualization

- Time series plot of MA5, MA15 and MA100 for Exchange 1

Initial Algorithm - Overview

- Loop through the combined data frame row by row
- Check if MA5 >= MA5_threshold; if yes, bid as much as the balance allows using the current lowest price, record this price as bid_price
- Check if price - bid_price >= price_difference_threshold or time is within 1 minute of end time; If yes, ask as much as the BTC allows using the current highest price

Initial Algorithm - Result

- MA5_threshold = 10; price_difference_threshold = 50
- Ending balance $25701.98; 24 bid attempts, 20 bid executions, 24 ask attempts and 20 ask executions
- Assumptions and simplifications
- Optimization tool (terminated due to overfitting)
- Could use Exponential Moving Average (EMA) and arbitrage opportunities to improve

```
2019-05-22T11:10:42.493374Z,2,7820.0,b
2019-05-22T11:10:42.493374Z,2,7820.0,b
2019-05-22T11:10:42.493374Z,2,7820.0,b
2019-05-22T11:10:42.493374Z,2,7820.0,b
2019-05-22T11:10:42.493374Z,2,7820.0,b
2019-05-22T11:10:42.493374Z,2,7820.0,b
2019-05-22T12:03:51.017620Z,1,7870.0,a
2019-05-22T12:03:51.017620Z,1,7870.0,a
2019-05-22T12:03:51.017620Z,1,7870.0,a
2019-05-22T12:03:51.017620Z,1,7870.0,a
2019-05-22T12:03:51.017620Z,1,7870.0,a
2019-05-22T12:03:51.017620Z,1,7870.0,a
2019-05-22T15:21:12.853661Z,1,7900.0,b
2019-05-22T15:21:12.853661Z,1,7900.0,b
2019-05-22T15:21:12.853661Z,1,7900.0,b
2019-05-22T15:21:12.853661Z,1,7900.0,b
2019-05-22T15:21:12.853661Z,1,7900.0,b
2019-05-22T15:21:12.853661Z,1,7900.0,b
2019-05-22T15:22:35.895514Z,1,7950.0,a
2019-05-22T15:22:35.895514Z,1,7950.0,a
2019-05-22T15:22:35.895514Z,1,7950.0,a
2019-05-22T15:22:35.895514Z,1,7950.0,a
2019-05-22T15:22:35.895514Z,1,7950.0,a
2019-05-22T15:22:35.895514Z,1,7950.0,a
```

Improved Algorithm - Overview

- Loop through the combined data frame row by row
- Check if EMA5 >= EMA5_threshold or current highest price - current lowest price >= price_difference_threshold; if yes, bid as much as the balance allows using the current lowest price, record this price as bid_price
- Check if price - bid_price >= price_difference_threshold or timestamp is within 1 minute of end time; If yes, ask as much as the BTC allows using the current highest price

Improved Algorithm - Result

- EMA5_threshold = 15; price_difference_threshold = 50
- Ending balance $25721.97; 102 bid attempts, 21 bid executions, 102 ask attempts and 21 ask executions
- # of bid/ask attempts increased; execution rate decreased

```
2019-05-22T09:53:41.122048Z,3,7864.9,b
2019-05-22T09:53:41.122048Z,3,7864.9,b
2019-05-22T09:53:41.122048Z,3,7864.9,b
2019-05-22T09:53:41.122048Z,3,7864.9,b
2019-05-22T09:53:41.122048Z,3,7864.9,b
2019-05-22T09:53:41.122048Z,3,7864.9,b
2019-05-22T10:16:33.423707Z,1,7914.95396004,a
2019-05-22T10:16:33.423707Z,1,7914.95396004,a
2019-05-22T10:16:33.423707Z,1,7914.95396004,a
2019-05-22T10:16:33.423707Z,1,7914.95396004,a
2019-05-22T10:16:33.423707Z,1,7914.95396004,a
2019-05-22T10:16:33.423707Z,1,7914.95396004,a
2019-05-22T11:05:02.022068Z,2,7828.96,b
2019-05-22T11:05:02.022068Z,2,7828.96,b
2019-05-22T11:05:02.022068Z,2,7828.96,b
2019-05-22T11:05:02.022068Z,2,7828.96,b
2019-05-22T11:05:02.022068Z,2,7828.96,b
2019-05-22T11:05:02.022068Z,2,7828.96,b
2019-05-22T14:32:10.906672Z,1,7879.0,a
2019-05-22T14:32:10.906672Z,1,7879.0,a
2019-05-22T14:32:10.906672Z,1,7879.0,a
2019-05-22T14:32:10.906672Z,1,7879.0,a
2019-05-22T14:32:10.906672Z,1,7879.0,a
2019-05-22T14:32:10.906672Z,1,7879.0,a
```

Further Improvements

- Simulation and sensitivity analysis
- Evaluate other methods of generating bid attempts (e.g. adjust how much to bid based on intensity of bid signals)
- Explore other methods of generating ask attempts (e.g. adjust the price_difference_threshold according to the intensity of bid signals)

Challenges and Milestones

- Study the rules and logic behind cryptocurrency trading
- Selecting parameters

Conclusion

- Challenging and fun

  Thank you for creating this project!

Appendix – Useful Resources

- https://medium.com/@coinloop/trading-with-rsi-2584deca18c8
- https://blog.hybridblock.io/8-factors-affecting-cryptocurrency-price/
- https://hackernoon.com/intricate-cryptocurrency-arbitrage-profit-like-a-ninja-fc60075cc22f).

Appendix – Initial Algorithm

```
# best result so far
# Simple MA5 threshold algorithm
from collections import defaultdict
from datetime import timedelta
import math
from datetime import datetime

balance = 25000
BTC = 0
# temp balance and temp BTC are the balance and BTC assuming all attempts are successful immediately
temp_balance = 25000
temp_BTC = 0

bid_price = 0
end_time = '2019-05-23 07:00:00'
fmt = '%Y-%m-%d %H:%M:%S'
tstamp_end_time = datetime.strptime('2019-05-23 07:00:00', fmt)
bids = []
asks = []
three_prices = {}
for index, row in df_all.iterrows():
    timestamp = str(row['timestamp'])
    add_one_millisec = str(row['timestamp'] + timedelta(seconds=0.001))
    MA5 = row['MA5']
    price = row['price']
    exchange_id = row['exchange_id']
    side = row['side']
    date = add_one_millisec.split(' ')[0]
    time = add_one_millisec.split(' ')[1]
    three_prices[exchange_id] = price
```

Appendix – Initial Algorithm

```
# bid attempt
maximum_can_buy = math.floor(temp_balance / (price * 0.5))
# exchange_id in [1, 2] because it seems only exchange 1 and 2 generate good signals
if(MA5 >= 10 and exchange_id in [1, 2] and maximum_can_buy > 0):

    bid_exchange_id = min(three_prices, key=three_prices.get)
    bid_price = price
    for i in range(0, maximum_can_buy):
        bids.append([row['timestamp'], bid_exchange_id,
                     bid_price, 'unsuccessful'])
        print(date + 'T' + time + 'Z,' +
              str(bid_exchange_id) + ',' + str(bid_price) + ',b')

    temp_balance -= bid_price * maximum_can_buy * 0.5
    temp_BTC += 0.5 * maximum_can_buy

# ask attempt
maximum_can_sell = int(temp_BTC * 2)
if(maximum_can_sell > 0 and
   (price - bid_price >= 50 or ((tstamp_end_time - row['timestamp']).total_seconds() <= 60 and temp_BTC > 0))):
    bid_exchange_id = max(three_prices, key=three_prices.get)
    for i in range(0, maximum_can_sell):
        asks.append(
            [row['timestamp'], bid_exchange_id, price, 'unsuccessful'])
        print(date + 'T' + time + 'Z,' +
              str(bid_exchange_id) + ',' + str(price) + ',a')

    temp_balance += bid_price * maximum_can_sell * 0.5
    temp_BTC -= maximum_can_sell * 0.5
```

Appendix – Improved Algorithm

```
# best result so far
# EMA5 threshold algorithm
from collections import defaultdict
from datetime import timedelta
import math
from datetime import datetime
balance = 25000
BTC = 0
# temp balance and temp BTC are the balance and BTC assuming all attempts are successful immediately
temp_balance = 25000
temp_BTC = 0

bid_price = 0
end_time = '2019-05-23 07:00:00'
fmt = '%Y-%m-%d %H:%M:%S'
tstamp_end_time = datetime.strptime('2019-05-23 07:00:00', fmt)
bids = []
asks = []
# initialize the three prices dictionary
three_prices = {}
three_prices[1] = 0
three_prices[2] = 0
three_prices[3] = 0
for index, row in df_all.iterrows():
    # prevent excessive bids/asks
    if(len(bids) > 100):
        break
    timestamp = str(row['timestamp'])
    add_one_millisec = str(row['timestamp'] + timedelta(seconds=0.001))
    EMA5 = row['EMA5']
    price = row['price']
    exchange_id = row['exchange_id']
    side = row['side']
    date = add_one_millisec.split(' ')[0]
    time = add_one_millisec.split(' ')[1]
    three_prices[exchange_id] = price
    current_highest = three_prices[max(three_prices, key=three_prices.get)]
    current_lowest = three_prices[min(three_prices, key=three_prices.get)]
```

Appendix – Improved Algorithm

```
# bid attempt
maximum_can_buy = math.floor(temp_balance / (price * 0.5))
if((three_prices[1] > 0 and three_prices[2] > 0 and three_prices[3] > 0) and  # check if the current price
   # exchange_id in [1, 3] because it seems only exchange 1 and 2 generate good signals
   ((EMA5 >= 15 and exchange_id in [1, 3]) or current_highest - 50 >= current_lowest) and
        maximum_can_buy > 0):
    bid_exchange_id = min(three_prices, key=three_prices.get)

    if(maximum_can_buy >= 1):
        bid_price = current_lowest

    for i in range(0, maximum_can_buy):
        bids.append([row['timestamp'], bid_exchange_id,
                     bid_price, 'unsuccessful'])
        print(date + 'T' + time + 'Z,' +
              str(bid_exchange_id) + ',' + str(bid_price+15) + ',b')

    temp_balance -= bid_price * maximum_can_buy * 0.5
    temp_BTC += 0.5 * maximum_can_buy

# ask attempt
maximum_can_sell = int(temp_BTC * 2)
if((current_highest - 50 >= bid_price or (tstamp_end_time - row['timestamp']).total_seconds() <= 60) and
   maximum_can_sell > 0):
    bid_exchange_id = max(three_prices, key=three_prices.get)

    for i in range(0, maximum_can_sell):
        asks.append([row['timestamp'], bid_exchange_id,
                     current_highest, 'unsuccessful'])
        print(date + 'T' + time + 'Z,' +
              str(bid_exchange_id) + ',' + str(current_highest-15) + ',a')

    temp_balance += bid_price * maximum_can_sell * 0.5
    temp_BTC -= maximum_can_sell * 0.5
```