Stock Trading Library

Assignment for Selected Topics in Programming

Marius Mikučionis <marius@cs.aau.dk>
May 5, 2022

Introduction

Stock exchange is a place where stock buyers and sellers meet and trade stocks for currency. A profitable trade involves buying at lower prices and later selling at higher price. Normally the price changes over time reflecting the expectations of buyers and sellers, however there are not many (human) participants and therefore it can be challenging for them to meet and agree on the price, especially when a company behind stock is not very popular. In order to facilitate a better trade availability, stock exchanges allow algorithmic trading. Algorithmic trades involve many instances of algorithms participating in public offerings and executing trades automatically on behalf of professional brokers. A typical instance of such trading algorithm involves monitoring price trends, computing specific signals (predicting future trends) and executing the corresponding transactions according to predicted trends and signals.

Figure 1 shows stock price changes over time aggregated into “candle-sticks” for some chosen period of time (hour, day, week, month, quarter etc):

- a candle stick is green if the *closing price* (last transaction in the period) was higher then the opening price (the first transaction in the period) -- an upward trend.
- a candle stick is red if the *closing price* is lower the opening price -- a downward trend.
- the opening and closing prices are connected with a solid rectangle denoting that the price fluctuates somewhere in between them.
- the whiskers (vertical lines) above and below the candle rectangles represent maximum and minimum price over that period.

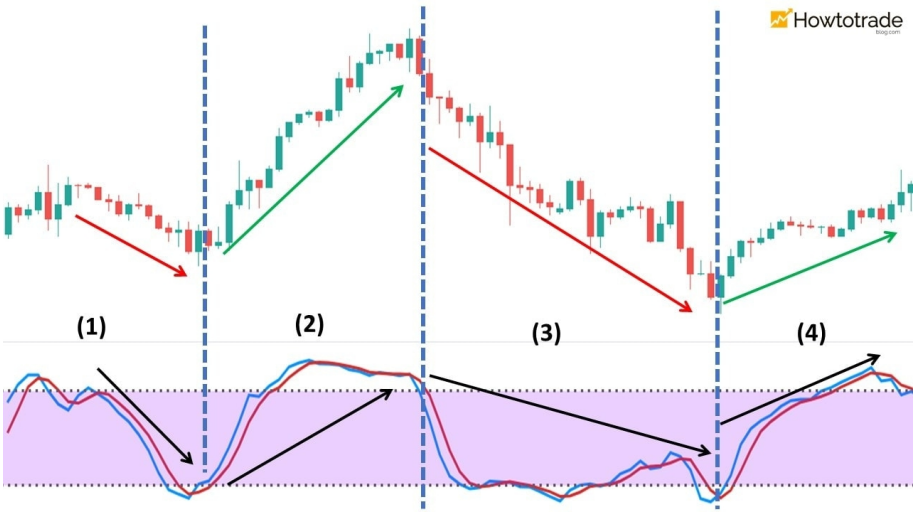

Figure 1 Candle-stick chart with stochastic oscillators [howtotrade].

Candle-sticks represent relatively short term and the price may change much more over longer periods of time, therefore the candle-stick data is only an intermediate step in recognizing longer trends providing larger changes in prices and thus potential for bigger profits/losses.

The bottom part of Figure 1 shows stochastic oscillators derived from the candle-stick data:

- The blue curve is a fast oscillator which reacts to the price changes quickly.
- The red curve is a slow oscillator which reacts slower
- Both curves oscillate between 0% and 100%, where the levels bellow 20% are called oversold (more selling than usually -- an opportunity to buy) and the levels above 80% are called overbought (more buying then usually -- an opportunity to sell).
- The two oscillators signal an upward trend when the blue curve crosses the red curve upwards and the signal is downwards when the blue curve crosses the red curve downwards.

The stochastic oscillator curves can be computed using the following formula [investopedia]:

$$K = \frac{C - L}{H - L} \times 100\%$$

Where C is a closing price of the last period, L is the lowest price over X previous trading periods, H is the highest price over X previous trading periods, K is the current value of the (fast) stochastic indicator. The slow stochastic indicator D is computed as ***a moving average*** of K over Y number of periods. *Usually the "buy" signal is when K crosses D curve upwards in the oversold region and "sell" signal is when K crosses D downwards in the overbought region*. One may also experiment just by monitoring the crossings and ignoring the overbought and oversold regions.

A typical strategy uses periods of 1 day and X = 14 days and Y = 3 days, but for some stocks it may not be profitable due to different buyer-and-seller dynamics (popular stocks receive more attention thus change faster, whereas niche stocks react slowly), hence these parameters (especially X) may need to be experimented with different values.

Requirements

The goal of the assignment is to develop a library for reading historical stock prices, simulating algorithmic trades and evaluating profitability of various strategies. The library implementation should support the following features:

1. Read the stock and trade information from JSON files.
2. The JSON parser should be generic (handle various data structures) and efficient (without copying an entire JSON file).
3. Compute candle-stick trajectories for a given selected period (e.g. one stick -- one day).
4. Compute stochastic indicators for given X and Y parameters from the candle-stick data.
5. Simulate a trade execution based on a given strategy based on a stochastic oscillator:
   a) An algorithm is given a fixed amount of money (e.g. 10000kr).
   b) Buy the stocks at current price ***after*** the oscillator signals "buy" and money is available.
   c) Sell the stocks at current price ***after*** the oscillator signals "sell" and get back the money.
   d) Repeat the last two steps until the trade data is over.
   e) At the end, sell all stocks (if any).
   f) Output the dates of trades and final profit/loss.
6. Use lambda expressions to either sort the trades, or create a stochastic indicator, or trade strategy.
7. Support and demonstrate parallel computation for either:
   a) Loading multiple stock data in parallel.
   b) Evaluating multiple trading strategies in parallel.
8. Use unit testing to demonstrate and test the parts of the library.
9. Benchmark: measure the time either for loading the stocks or simulating trade. Improve/vary the measured code/structures and measure again. Compare and make conclusion.
10. Visualize the stock price and oscillator signals by plotting them on screen.

The requirements are evaluated based on the demonstration basis, i.e. it is enough to demonstrate the requirement only on instance of one stock, trade strategy, parameter values etc., ***it is not necessary to compute the same results for multiple inputs*** (except for the requirement 7).

The final deliverable should be a compressed archive containing the following:

1. PDF listing of source code, unit tests and build script (e.g. CMakeLists.txt, Makefile or similar).
2. Simulated trades listings (when bought and sold and final profit/loss).
3. Benchmark measurements and conclusions.
4. Screenshots of visualized data.
5. Project sources and results (without build-system/binary-object files).