CITS4404 Assignment 2

Building Bots

1 Context and Aims

Automated algorithms, or *bots*, now perform a substantial percentage of the world's trading. It is difficult to know the exact numbers (and depends how broadly you define algorithmic trading). According to Wikipedia , for example, a study reported by Kissell found that "around 92% of trading in the Forex marked was performed by trading algorithms rather than humans". There is no doubt that trading bots are big business!

In this assignment we will use adaptive AI techniques to build *trading bots*. We'll do this for Bitcoin — the world's hardest currency — but the same bots could be used for other currencies, for crypto tokens, for stocks, commodities and the like. The bot is "agnostic" to what the underlying data represents.

The task of this assignment is to attempt to utilise techniques researched in Assignment 1 to create and optimise the performance of a trading bot.

While the aim is of course to generate the best bot possible, the real goal of the assignment is to provide an opportunity to investigate the use of adaptive algorithms on a real-world challenge.

2 Technical Analysis

Technical analysis (TA) (sometimes called *quantitative analysis*) refers to trading strategies that focus on price and volume history to try to predict future prices — or at least the direction of the market. This contrasts with *fundamentals* trading that might focus more on broader factors (such as geopolitics and macro factors) that are perceived to have an impact on price.

Many (human) traders of stocks, currencies and commodities trade primarily on technical analysis, and are known as *chartists* (or sometimes *quants*). We are certainly not advocating this approach! Nevertheless, it forms the basis for not only billions in trade, but also for most trading bots.

The key aspects of technical analysis needed for this assignment follow. You may find it useful to follow the examples at the (free) information website CoinGecko, where you can play around with the technical indicators discussed.

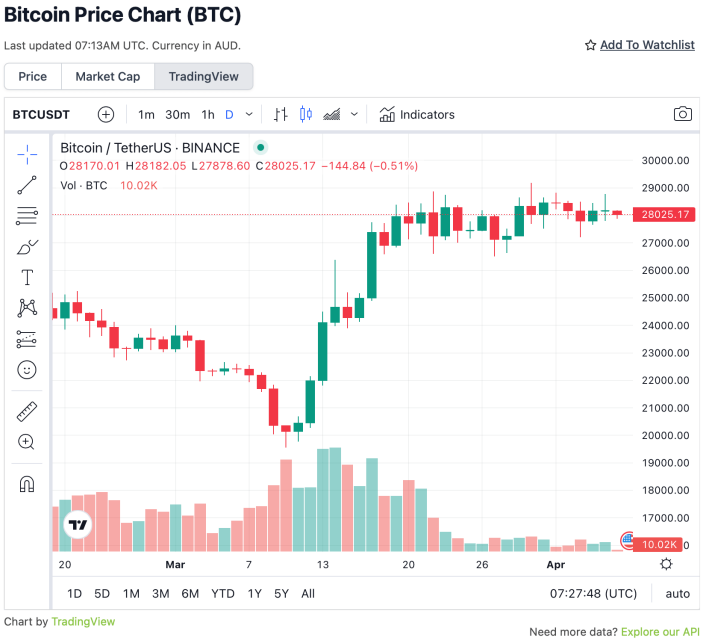

Figure 1: Candles for BTC/USDT

OHLCV data

Figure 1 shows a *candlestick chart*, in this case for Bitcoin in US dollars (also known as the BTC/USDT *pair*). This plot shows one candle for each day. The same can be done for different time periods, for example hours or weeks.

Each candle shows the open and close prices (thick line) and the intra-day high and low (thin line). The histogram at the bottom shows the volume of transactions in that time period. This standard representation of the historical data is also known as an *OHLCV (open-high-low-close-volume)* data.

Technical Analysis Indicators

TA seeks to identify trends, patterns or distinctive features in the historical data that may indicate future price movement. The data generated to represent these patterns are called *(techincal) indicators*. Many of these are commonly used and are given a name.

For example, one of the simplest indicators that many would be familiar with is a *simple moving average (SMA)*. This is a lagging indicator, because it averages data over a window of time that has passed.

An *exponential moving average (EMA)* also summarises data over a period of elapsed time, but gives greater weight to more recent values.

Figure 2 shows two moving average indicators: the green line has a greater delay and more smoothing (lower frequency), while the orange line reacts more quickly to changes in the data (higher frequency).

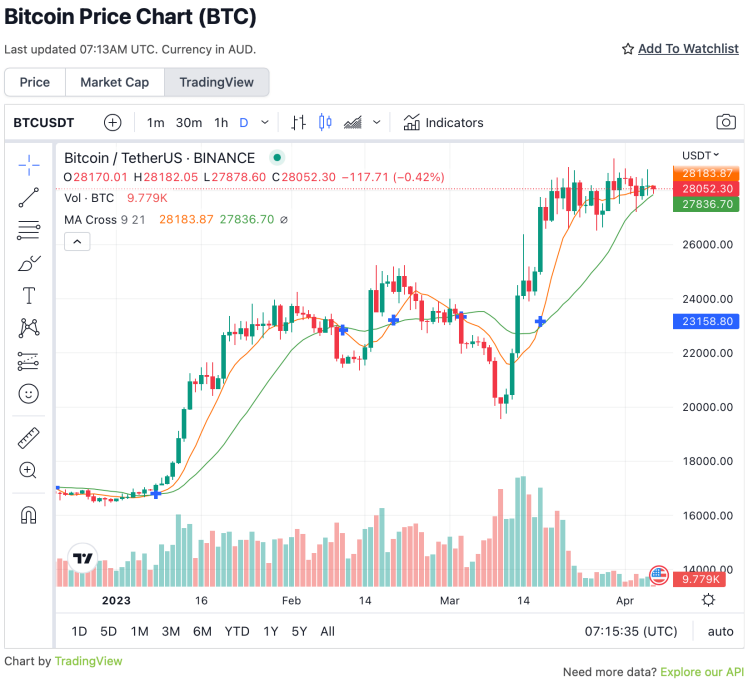

Figure 2: A chart showing two technical indicators with buy/sell triggers.

A simple trading *strategy* is to buy when the higher frequency indicator crosses above the lower frequency indicator and sell when the reverse happens. The blue crosses in Figure 2 indicate where the trades would take place using this strategy. We can say these *trigger* a buy or sell.

The idea is to catch the uptrends and skip the downtrends. It's a compelling story, but if it were that easy, everyone would be rich!

For a more involved example of TA, and the Bart Simpson indicator, see Kyle Doops.

Parameters

Most "indicators" really represent a class of indicators, parameterised by some values. A simple example is the SMA, which must specify the period over which the moving average is calculated, or the "window size". In fact, the lines in Figure 2 are just two SMAs, — the first (green), or lower frequency signal, is a 21 time period SMA, while the second, or higher frequency signal, is a 9 time period SMA. This is indicated by the two numbers after "MA Cross". Your can play with these online by double clicking on MA Cross and changing the parameter values, or select different indicators from the drop down menu.

An EMA will typically have two parameters, one for the window size, and a "smoothing factor" that relates to the exponent — or in practice, the degree to which the more recent values are weighted higher than older values.

Notice that all of these parameters are independent of the time scale – an SMA 21 will average over a window of 21 candles, whether they are minutes, days, or years.

Strategies and Bots

Notice that if you change the parameter values of "MA Cross" you will get different strategies — the buy and sell triggers will move to different points.

Of course, we can also get different strategies by choosing different indicators.

A trading *bot* is simply an implementation of a strategy.

Evaluation of Bots

We can think of bots as competing to implement the best strategy. Evaluating a bot over a period of data is easy — we simply start with a fixed amount, lets say $100, implement the trades, and see what value we have at the end. There will also normally be a costs, or a "spread", associated with each trade (taken by the exchange for providing the service), so we have to add those in to each transaction as well. This may be a constant amount or a percentage. These costs stack the odds slightly against the trader, and more so the more trades that are made.

This approach to evaluating a strategy or bot is known as *back-testing*. Of course, success on past sequences of data does not guarantee a strategy will be successful on future sequences!

Optimisation

It should be clear by now that (like most things in life!) designing a good trading bot is an optimisation problem.

In this case there are (at least) two things than can be adapted to try and achieve better results — the indicators chosen and the parameters of those indicators. (Which adaptations are likely to lead to better behaved search spaces?)

3 AI Optimisation

Technical Indicators and Data

For this assignment we will limit our bots to the technical indicators provided by the python `ta` library. You will only need to choose indicators and set their parameters.

Real-time data will be accessed from the Kraken exchange.

Use of these will be explained further in Section 4.

Problem Instance

A *problem instance* is defined by an $n \times 5$ array of OHLCV data – in this case we will use the daily candle data for the BTC/AUD pair between the specified dates, retrieved from Kraken. All bots will use the same set of data.

Since the data are fixed for a given problem instance, for brevity we will not explicitly show the data as an argument to the functions that follow.

Language

A *candidate solution* is defined as a pair of functions:

$$bot = (buy\_trigger(t, P), sell\_trigger(t, P))$$

where t is an index representing a (discretised) datetime and P is an ordered set of parameters.

buy_trigger and *sell_trigger* are simply discrete versions of an impulse function, defined by:

$$buy\_trigger(t, P) = buy(t, P) \land \neg buy(t-1, P) \land \neg(sell(t, P) \land \neg sell(t-1, P))$$
$$sell\_trigger(t, P) = sell(t, P) \land \neg sell(t-1, P) \land \neg(buy(t, P) \land \neg buy(t-1, P))$$

The bot will thus buy when the buy signal changes from false to true and there is no concurrent sell trigger. It will sell when the sell signal changes from false to true and there is no concurrent buy trigger.

Note: Since we are dealing with a finite set of data, we'll assume (in addition to the above) that the bot does not execute any sells prior to the first buy. For the purposes of evaluation we'll also assume a sale on the close of the last day.

The *buy* and *sell* functions will be expressed by logical expressions in disjunctive-normal-form. The following simple grammar describes our language buy and sell functions.

$$\begin{aligned}
buy(t, P) &\longrightarrow dnf \\
sell(t, P) &\longrightarrow dnf \\
dnf &\longrightarrow conj \lor dnf \\
dnf &\longrightarrow conj \\
conj &\longrightarrow lit \land conj \\
conj &\longrightarrow lit \\
lit &\longrightarrow \neg lit \\
lit &\longrightarrow value > c \times value \\
value &\longrightarrow indicator(t, P_i) \\
value &\longrightarrow candle_value(t) \\
value &\longrightarrow c
\end{aligned}$$

where

- P is an ordered set of constants (the "parameters")
- $P_i \subseteq P$
- $c_i \in P$
- *indicator* is an indicator signal (function) provided by the `ta` library
- *candle_value* is one of the candle functions $o(t)$, $h(t)$, $l(t)$, $c(t)$, $v(t)$

Optimisation

Optimisation may take place by:

- adapting parameter values within P
- adapting the expressions for *buy* and *sell* (within the restrictions of the language)

The choice is yours.

Strategy Evaluation

A bot is evaluated by applying its buy and sell triggers over the period of time, from the start to the finish.

You may assume:

- The bot starts with one hundred Australian dollars (AUD).
- The bot swaps all of its AUD for BTC when it receives the first buy trigger, then sells it all when it gets the subsequent sell trigger (any intermediate buy triggers after purchasing and before selling are ignored). It then buys again on the next buy trigger, and so on.
- Each buy or sell event costs 2% of current holdings.
- At the end of the test period the holding is sold (if currently in BTC) at the close price and evaluated in AUD.

4 Programming Setup

It is suggested that you use your own local python environment(s) for the coding. If you wish to use CoCalc you will need to install the `ta` and potentially `ccxt` libraries (you can use `pip install`). The `ccxt` library is strictly only needed once to download the data, which you could then upload to CoCalc if you wished.

Retrieving the Data

To retrieve the historical data we will use the `ccxt` package and the Kraken exchange. Kraken provides information on the BTC/AUD pair, and provides 720 data points without the need for credentialing — that is, you don't need an account.

A good tutorial on on using `ccxt` is provided by Part Time Larry. You only need to watch the relevant part. When viewing the video please note that you *do not need the API or secret keys* or the `config.py` file for our purposes.

Using the TA Indicators

A good follow on tutorial on using `ta` is also provided by Part Time Larry.

Note that an indicator class in `ta` may provide more than one indicator signal. For example, `ta.volatility.BollingerBands` provides a number of signals, notablly `bollinger_hband()`, `bollinger_lband()` and `bollinger_mavg()`.

Binary comparison indicators such as `bollinger_hband_indicator()` should not normally be needed, since our language already provides for comparisons.

The contents of the two tutorials is sufficient to start writing your own bot.

5 Deliverables

The project has three deliverables: a video presentation, a report, and a code repository.

Presentation

Each team will be required to provide a 15 minute video presentation on their project. The presentation should include the following:

- A high level explanation of the optimisation algorithm chosen, and any distinguishing features. The explanation should make use of diagrams and/ or pseudocode rather than actual code. Issues in the implementation may be discussed, but a full methodology is not required.
- A discussion of the considerations that went into choosing your (subset of the) hypothesis space — for example, what indicators were chosen, what parameter ranges were explored.
- Your evaluation/testing regime.
- Presentation of results. Visual representations are preferred (images, tables, animations/simulations).
- You may wish to interleave the above two points (design changes and results) to demonstrate how changes in the design affected the results.
- Conclusions from your work.

Report

The report should be in the style of a short conference paper, and complement your video. It should include an abstract (not more than half a page) that provides a high level overview of your work — what you selected to build on, the choices you made, and the results.

The body of the paper should provide more detail in support of the video. Unlike a conference paper which would be fully self-contained, you *do not need to repeat information from this project specification*. You can take this document "as read". You also do not need to provide extensive context or background around the chosen algorithm — you may reference material from Assignment 1 if you wish.

Code

The code, and any results files, should be provided as a link to a repository such as GitHub (preferred) or in CoCalc. The code should allow reproduction of the results. Brief instructions for running the code can be included in a README file or notebook.

The report should provide a pointer to where the code, and instructions, can be found.

The coding will not be marked *per se*, but the results should be repeatable if required.

6 Practicalities

- This is a group project with allocated teams of 3–5 members. You may wish to choose a team name to help distinguish the team in queries, marking and so on. The team name or number, the team members and student numbers should appear on the front page of the report.
- The deliverables are due by 11:59pm, May 13th. Late assignments will penalised according to the University's standard rules for late submissions described in the Unit Outline.
- The report is limited to a maximum of 3000 words excluding diagrams and references. Text beyond the maximum word count will not be marked. A word count must be included on the title page, along with the names and student numbers of the team members.
- The report must be submitted as a pdf file. If submitted as a file, the video should be in mp4 format unless otherwise arranged. Alternatively the video may be submitted as a link to a streaming service such as YouTube.
- Referencing should follow the IEEE style. As usual, all materials obtained from other sources should be referenced.
- This assignment is group work. Submitting work is considered a declaration that the work submitted is entirely the work of the team members except as referenced.
- A Teams channel has been provided for queries relating to the assignment.