Semester project report

Department of Telecommunications and Media Informatics

Author:	**Márton Torner**
Neptun:	**OZFKFP**
Specialization:	**Infocommunication (HIT)**
E-mail address:	**torner.marton@gmail.com**
Supervisor:	**Bálint Gyires-Tóth, PhD**
E-mail addres:	**toth.b@tmit.bme.hu**

Title: Financial time series modeling and forecasting with deep neural networks

Task

The task is to collect limit order book data from a chosen crypto exchange, analyze and process the gathered records and to develop a neural network based solution to predict the direction of the price movement of the asset-pairs.

In this work level 100 data (limit order book, depth: 100) is used from Kraken crypto exchange. The live order flow is queried with the help of the provided API and saved on a local storage. The aim is to create a Deep Learning powered system which can predict the price movement direction from the given history. The first approaches are models based on autoregressive solutions (logistic regression and random forest regression) and then I try to use a deep neural network as the model.

In the future this work can be used as a base for creating a complex system for algorithmic trading.

2018/19 II. semester

1 Theory and previous works

1.1 Introduction

The computerization of financial markets and the availability of the electronic records of different stock exchanges like transactions, order flow and limit order books provide us a great opportunity to analyze and model the dynamics of these markets. Over the last years a huge number of studies have been done on this field which can serve as a good base for further studies.

With the computerization also came the rapid acceleration of the trading on these financial markets and big companies started to use algorithmic solutions to steal a march on their rivals and gain some higher profits.

The quick development that the field of Deep Learning had in the last decade and the impressive number of successful researches conducted on the topic of time series forecasting opened the door to change the traditional mathematical algorithms with a neural network based solutions. Price forecasting is just a step leading to more complex systems which use the predictions as inputs to deliver a trading strategy with the help of machine learning (reinforcement learning).

These solutions could change the role of financial specialists in the future.

1.2 Theoretical summary

1.2.1 Price formation mechanism

The market price per share of stock ("share price") is the amount of money an investor is willing to pay to own a company's single stock. The 'price formation mechanism' - given in Equation 1 - is a map which tries to represent the relation between market price and different variables affecting it, e.g. price history, order flow, news about the companies or the market etc.

$$Price(t + \Delta t) = F(Pricehistory(0...t), OrderFlow(0...t), OtherInfo) = F(X_t, \epsilon_t) \quad (1)$$

X_t: set of state variables (e.g. order flow, volatility, price, etc.)
ϵ_t: random noise (represents the arrival of new, unknown information)
Δt: time resoution

It is shown that this relation is universal (not asset-specific) and stationary (stable across time, even at out-of-sample predictions) [1]. These two features give us the opportunity to use all collected data points to design and train a universal model which can make predictions for all asset-pairs valid even in the future.

1.2.2 Limit Order Book

In this work limit order book data is used as the input so this section tries to provide a short overview what it is and how it works. On stock exchanges we want to buy shares or sell the ones we hold and to accomplish this we place orders of which the LOB consists. There are more special types of orders but in this work only limit orders are used. They remain in the LOB until being executed or cancelled.

The LOB has two sides, the ask and the bid side, shown on Figure 1. The ask side represents the sell orders (willing to sell the given amount at the given price) and the bid side the buy orders (willing to buy the given amount at the given price).

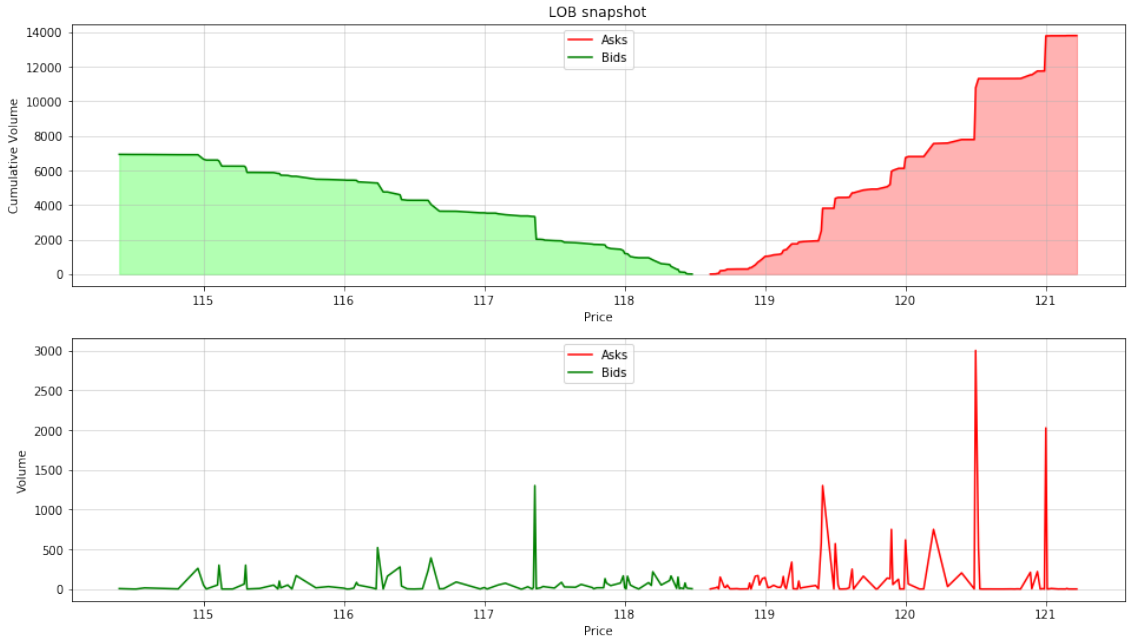

Figure 1: Limit Order Book snapshot (ETH_EUR). Bottom: Volume at each price level. Top: Cumulative volume at price levels.

A tick is the measure of the minimum upward or downward movement in a price of a security this means the price levels in the LOB must be the multiples of this value (so order prices can not differ less than this). The best ask price is the lowest sell order and the best bid price is the highest buy order. These are the prices at which the given number of shares can be bought or sold. The difference between best ask and bid price is called the spread and their average is the mid-price (2). The weighted average mid-price (WAMP) and volume weighted average price (VWAP) can be calculated as given in Equation 3 and Equation 4. The depth of the book is used in this paper as the number of best orders on each side (so depth=100 means the top 100 lowest priced ask- and the top 100 highest priced bid orders).

$$Midprice = \frac{p^{bid}_{best} + p^{ask}_{best}}{2} \tag{2}$$

$$WAMP = \frac{p^{bid}_{best} * v^{ask}_{best} + p^{ask}_{best} * v^{bid}_{best}}{v^{bid}_{best} + v^{ask}_{best}} \tag{3}$$

$$VWAP = \frac{(\sum_{i=1}^{depth}(p^b_i * v^b_i)) * v^a + (\sum_{i=1}^{depth}(p^a_i * v^a_i)) * v^b}{v^a + v^b} \tag{4}$$

p: price
v: volume (without lower index = sum over side)
b: bid
a: ask

Orders are submitted and cancelled continuously which means that the updates arrive in a very high frequency (millisecond) and it is also easy to see that consuming the order flow of stock exchanges leads to Terabytes of data. At crypto exchanges order manipulation is a big problem to be faced at filtering the collected data. So, to process a live order book we have to deal with various problems [TODO link ide a crawler szekcióhoz].

1.2.3 Deep Learning

Deep Learning models are deep neural networks (an example can be seen on Figure 2) trained on large datasets to uncover complex non-linear relations between the (high-dimensional) inputs and outputs. This can be simplified as it represents a functional relation like y = f(x) where y is the output and x is a (high-dimensional) input vector.

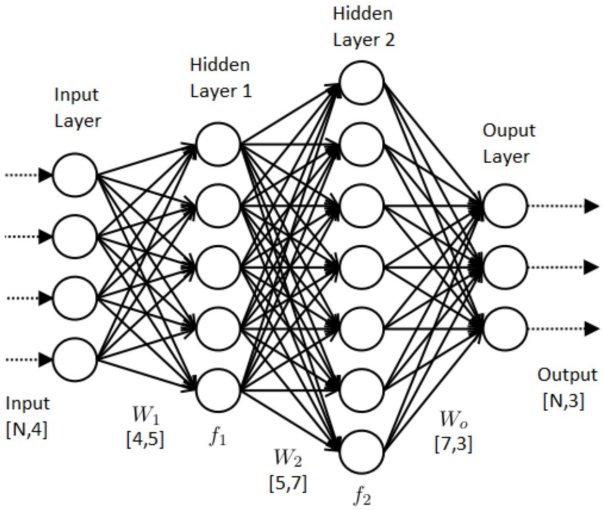

Figure 2: Simple multilayer perceptron.

The network iterates the input data through its hidden layer(s) (basically weighted sums are calculated followed by so called 'activation' functions). In supervised approaches the weights are modified in the backpropagation process in which a regularized cost function – reflecting the discrepancy between the inputs and the desired output – is tried to be optimized. Depending on the input and output dimensionality and the design of the network it can have millions of parameters which makes these calculations computationally expensive. This problem made the development of this field a standstill until the rapid spread and improvement of GPUs started providing a huge advancement in calculations.

1.2.4 Convolutional Networks

A class of deep neural networks are convolutional neural networks (CNN). The main design is the same as any other networks (input, hidden layers, output). General CNNs use a set of convolutional layers followed by fully-connected layers (all neurons in a layer are connected to every neuron in the preceding and following layers as seen on Figure 2) as hidden layers. Convolutional layers are used for representing small or high features in the data with applying a convolutional operation to their input. It uses filters which are parts of the data with the same or less dimension than the input. Using k filters k feature maps are generated.

Attention is a mechanism which can be added to reach better results with the cost of more computational expenses [2]. At a high-level attention can be described as giving the network the ability to learn which parts are more important to focus on. It is commonly and successfully used at natural language processing tasks.

1.2.5 Long Short-Term Memory

TODO

1.2.6 Generative Adversarial Networks

Generative adversarial networks (GAN) are deep neural architectures comprised of a generator and a discriminator network (Figure 3), introduced by Goodfellow et al. [3]. These two networks contest against each other in the following way. The aim of the generator is to produce data indistinguishable of the real data and the discriminator tries to classify whether its input was real data or fake, generated by the opposing network. The losses of the networks are propagated back combined making both networks' reward depending on both of their performance. This means if the generator lacks competence the discriminator's work will be easy thus the system will be incompatible of solving the problem. The design and training of GANs is a complex task.

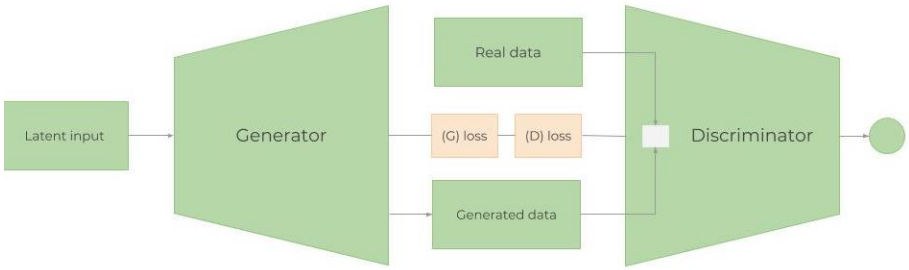

Figure 3: GAN architecture example.

1.3 Starting point, previous works on this project

At the start of the project I was no preceding work given and did not make any research on this topic before.

I started learning the theory of Deep Learning as part of the university course VITMAV45 and gained a well based knowledge on the subject to start with.

2 Own work on project

2.1 Data

The project's first task was to find or collect a database to work with. On financial field the big and high frequency raw datasets which are precise enough can cost a lot of money (3000 USD+). There was no funding available so I collected my own raw data from Kraken (www.kraken.com) which is a US based cryptocurrency exchange. Kraken gives the opportunity to access live order book (and of course other) data through a REST- and a Websocket based API.

In the training and evaluation process only ETH-EUR [1] snapshots are used but for further research on universality and stationarity [1]. I save 25 asset-pair's data in aggregate. For each currency 1 snapshot is querried every minute through REST API calls and the real-time order flow is recorded through the previosly mentioned Websocket API.

The saved data is stored in CSV file format, Table 1 and Table 2 show the structure. Every asset-pair is saved to a separate folder and at the end of each day all buffer files (updates and snapshots) are compressed (I chose gzip compression) and saved into a separate file this way saving space and ensuring that the unexpected corruption of a file causes minimal data loss. The saving method also makes paralell datapreparation on multiple CPUs possible. The dataset contains entries from 15th March 2019.

Field	Description
L1 Ask Price	The price of the ask order at 1st level (best ask)
L1 Ask Volume	The volume of the ask order at 1st level
L1 Ask Timestamp	The last modification time of the ask order at 1st level
...	
L100 Ask Price	The price of the ask order at 100th level
L100 Ask Volume	The volume of the ask order at 100th level
L100 Ask Timestamp	The last modification time of the ask order at 100th level
L1 Bid Price	The price of the bid order at 1st level (best bid)
L1 Bid Volume	The volume of the bid order at 1st level
L1 Bid Timestamp	The last modification time of the bid order at 1st level
...	
L100 Bid Price	The price of the bid order at 100th level
L100 Bid Volume	The volume of the bid order at 100th level
L100 Bid Timestamp	The last modification time of the bid order at 100th level
Timestamp	The timestamp of the snapshot

Table 1: One snapshot (one row in snapshot files).

Field	Description
AskOrBid	The update type: 0 = ask, 1 = bid
Price	The price of the order
Volume	The volume of the order (0 means that the level is deleted)
Timestamp	The last modification time

Table 2: One update (one row in update files).

[1] Ethereum - Euro: Ethereum is an open-source, public, blockchain-based distributed computing platform and operating system featuring smart contract (scripting) functionality. - *Wikipedia*

2.1.1 Data preparation

I wrote a parameterizable python script to do the whole preprocessing. The following section explains the steps executed by the generator.

To train my models I chose to use 30 days of ETH-EUR limit order book history. The first step of the preprocessing is to clean and reorder the update records to put them into an ascending flow. This is required because the Websocket API not always sends the packets in the right order or a short loss of the network connection can also cause some problems. At visualizing the data I also noticed that the API sometimes forwards updates with timestamps from few days ago. Although I could not explain this behaviour I had to remove these entries. The next task was to process the saved snapshots and create a full dataset which is saved to a single gzipped file. One parameter is which depth to use and it is important to emphasize that this is the first operation as we calculate every metrics for the following steps from the snapshots.

2.1.2 Labels

For each snapshot the WAMP (3) or the VWAP (4) is calculated since these are used in further steps. After this, we create the labels previously mentioned in 1.2.2. The research aims to successfully predict the direction of the price movement using 3 categories: UP (1), DOWN (-1), NO_MOVE (0). Snapshots are labelled using the following technique, based on the method introduced in the work by Zihao Zhang et al. ([5]). Although trades aren't executed exactly at the WAMP or the VWAP, they are a good estimation since they contain more information than the mid-price with also taking the volumes at the price levels in consideration. As financial time series, especially high frequency limit order book metrics are highly stochastic, the comparison between $WAMP_t$ and $WAMP_{t+1}$ would result noisy labels which are unable to compress bigger trends (here where the updates can happen on the scale of miliseconds this doesn't mean whole days, only minutes). I used a smoothing method which I recall briefly in Equation 5 and Equation 6. The first is the mean of the previous k - including the present one - (history), the second is the following k number of WAMPs (future). So k is the prediction horizon.

$$m\_(t) = \frac{1}{k}\sum_{i=1}^{k} \tag{5}$$

$$m_+(t) = \frac{1}{k}\sum_{i=1}^{k} \tag{6}$$

The labels are decided based on a threshold which is calculated as multplying the volatility - calculated on the full time period (history + future) - by α, a hyperparameter. An example can be seen on Figure 4. Red color indicates the down, green the up and white the stationary labels.

2.1.3 Scaling

After the labelling, the next step is to rescale the input (the snapshots as first approach). This is inevitable since there are multiple reasons for it. One is that if we use multiple asset-pairs the numbers can be on different scales e.g. the prices of Bitcoin and Ripple have a huge variance, this means we have to process them separately. The first approach was to use the z-score standardization method. As financial time series have regime shifts a static standardization would be inappropriate for a high frequency database over a considerable period of time, hence the previous 3 day's mean and standard deviation was used to standardize the actual day's prices. As for the volumes my first try was to simply normalize them, but a few tries showed that using the cumulative volumes at price levels boosts performance. This is reasonable, as humans can also

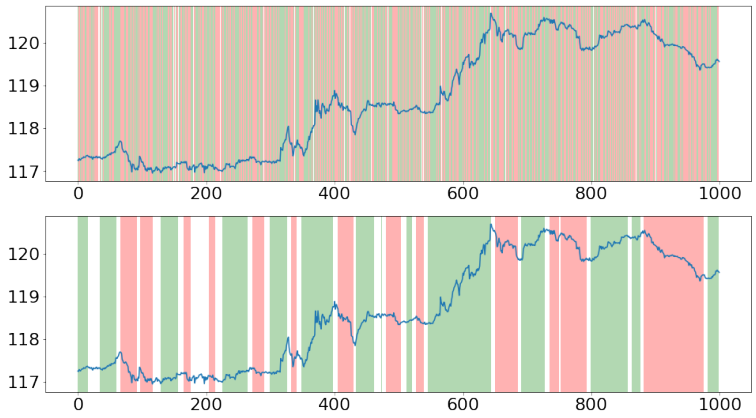

Figure 4: An example of labelling where the smoothing labels are used.

see more information from the cumulative volume chart than from the normal diagram (1). This approach also partly solved the problem of the outliers which means that some volumes were on an other scale than the othes in the same snapshot and they grew even bigger during the scaling because they were too rare to effect the standard deviation which stayed a small number this way and enlarged the huge numbers (as they were divided by them).

For the recurrent neural networks a last task was to resample the dataset with the number of parameters as timestep to create small series as inputs.

2.2 Baseline predictions

In my work I used three methods as a starting-point to see how a simple algorythm can solve this complex task and to have some results to which I can compare my future models.

2.3 Summary

A félévi munka során elért új eredmények ismételt, vázlatos, tömör Ebben a részben az adott félévre vonatkozó, az *Önálló laboratórium tárgy keretében elvégzett munka során* elért **új** eredmények ismételt, vázlatos, **tömör** összefoglalását várjuk, lehetőleg nem felsorolásként. Itt még egyszer ki lehet térni a leglényegesebb eredményekre, valamint a félév során felmerülő nehézségekre, de meg lehet említeni a továbbfejlesztési irányokat, lehetőségeket is.

Ezt a részt tagolható a következő pontok megválaszolásával:

- Mi volt az **aktuális kérdés**, probléma, amivel a félév során foglalkoztál?
- Mi a dolgozat **célja**, miért érdekes egyáltalán ezzel a problémával foglalkozni?
- Milyen **módszereket** használtál a probléma megoldása érdekében?
- Mik a legfontosabb **eredmények**?
- Milyen **következtetéseket** lehet levonni?

Ha valaki elolvassa ezt a részt, képet kell kapnia az egész dolgozatról. Ne legyen az absztrakt szó szerinti ismétlése.

Fontos, hogy az itt megadott sablontól el lehet térni, használata nem kötelező, csak segítséget jelenthet, viszont a fedőlap lehetőleg maradjon ugyanez és tartalmilag egyezzen meg a sablon irányelveivel. A beszámoló felépítésében nem érdemes eltérni a *Bevezető – Féléves munka és eredmények bemutatása – Összefoglaló* hármastól.

3 References

[1] Justin Sirignano and Rama Cont (2018). *Universal features of price formation in financial markets: perspectives from Deep Learning*

[2] *A Brief Overview of Attention Mechanism*

[3] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio (2014). *Generative Adversarial Nets*

[4] *Using the latest advancements in AI to predict stock market movements*

[5] Zihao Zhang, Stefan Zohren, and Stephen Roberts (2019). *DeepLOB: Deep Convolutional Neural Networks for Limit Order Books*

3.1 Source code

The whole project's source code (except the crawler's) can be found on GitHub:
`https://github.com/tornermarton/SemesterProject`
On some topics a deeper insight/longer explanation can be found in the Markdown files under the docs folder.
Most of the images used here can be found in the Jupyter Notebooks which are publicated also in this repository.

The following library served as a base for the crawler:
`https://github.com/krakenfx/kraken-wsclient-py`