AI4T - Short Term Stock Price-Trend Prediction with LOB-Data

March 7, 2022

Mattia Capparella

Abstract

In the High Frequency Trading (HFT) scenario, being able to predict in fractions of a second the future trend for stock-price is crucial. Although some machine learning approaches have been already adopted, specific labelled data required for training these algorithms are not always at disposal. In this study, I considered short-term stock price prediction using a variant of the labelling method proposed in (Chang et al., 2021): a sliding time horizon to label stocks according to their predicted price trends, referred to as called slope-detection labelling, using predictions labels including "*rise plus*", "*stationary*" (my proposal) and "*fall plus*". The effectiveness of the proposed method was evaluated by application to the FI dataset (Ntakaris et al., 2019) seen during lectures.

1. Introduction

The idea of using mathematical models to analyze and predict trends of financial markets has manifested as the field of quantitative analysis. The premise of this field is the analysis of the time-series produced by the market: this allow the extraction of valuable predictions about various aspects of the market itself, *e.g.:* volatility, real value of an asset or the trend (the aspect we are interested in). However, the means giving these predictions often rely on handcrafted features and have their parameters tuned manually by observation of the irrational behaviour of the asset price movements, which can reduce the effectiveness of the predictions. It is for these reasons that Deep Learning has been perceived as the "perfect" solution for the aforementioned limitations. The main contribution of this work is the proposal of integrating deep learning technologies for predicting future mid-price movements from LOB-data, and labelling strategy originally proposed for labelling short-term stock price trend from OHLC data (*closing price*).

Email: Mattia Capparella <capparella.17146513@studenti.uniroma1.it>.

AI for Trading 2021, Sapienza University of Rome, 1st semester a.y. 2021/2022.

Code. `https://github.com/sh3rlock14/AI4Trading`. The code is divided into two *Colaboratory* notebooks inside *CNN Trading Agent Folder*. `"main"` contains the code for the data download and setup, the model architecture, the datamodel and the training pipeline. `"Slope-detection labelling method"` contains the implementation of the labelling described in (Chang et al., 2021), par. III.A.

2. Method

Slope detection labelling method This method has been adapted for LOB-data from (Chang et al., 2021) as follows: for a stock *n*, let p_d be the mid-price at time *d*, then define:

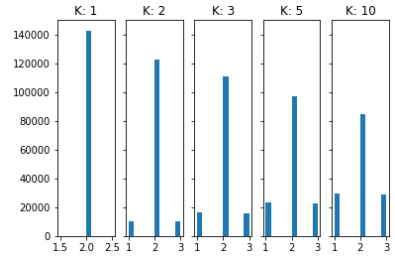
- $F_d = \sum_{i=1}^{K} p_{d+i}$: the average mid-price in the next **K** time slots
- $B_d = \sum_{i=0}^{K+1} p_{d-i}$: the average mid-price in the previous **K** time slots
- $\delta_d = F_d - B_d$: the *slope*
- μ_d: the mean value of mid prices
- σ_d: the standard deviation of the stock during the **2K** time slots, computed as:

$$\sigma_d = \sqrt{\frac{\sum_{i=-K+1}^{K} (p_d + i - \mu_d)^2}{2K}}$$

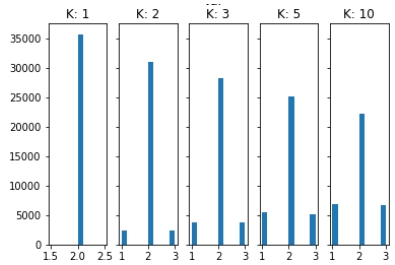
σ_d is then applied as a threshold against the mid-price as follow:

$$l(p_d) = \begin{cases} rise+, & \text{if } p_d > (\mu_d + \sigma_d) \text{ and } \delta_d > 0 \\ fall+, & \text{if } p_d < (\mu_d - \sigma_d) \text{ and } \delta_d < 0 \\ stat, & otherwise \end{cases}$$

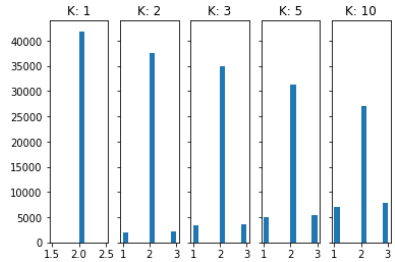
whereas *rise+* indicates the end of a rising period (peak), *fall+* indicates the end of a falling period (trough) and *stat*



(a) train labels



(b) val labels



(c) test labels

Figure 1. New label distributions

(stands for *stationary*) indicates that for the very next period the trend won't change its direction.

For this experiment I tried with $\mathbf{K} = [1, 2, 3, 5, 10]$ and fig.1 indicates how the new labels are distributed: as we can see, the distributions for all the sets are heavily unbalanced. Anyhow, I have used both **K=5** and **K=10**, resulting in the most balanced distributions.

Module Architecture For this experiment I reused the deep network seen during lecture (see fig.2 for the schematic) and described in (Zhang et al., 2019).

Training Framework For both Data&Architecture modelling and Training I used PyTorch Lightning Framework in combination with Weight & Biases platform for tracking experiments and visualizing results. I ran my experiments on *Google Colaboratory* using a GPU runtime for both training and testing. The listing 1 shows the sweep configuration ran by the *W&B* agent.

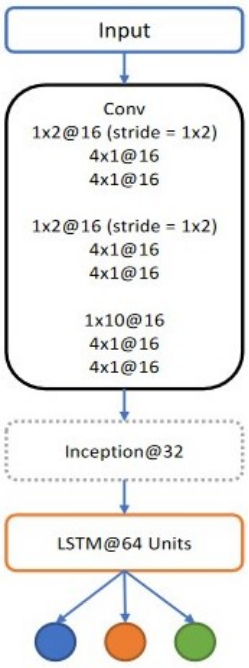


Figure 2. Model architecture schematic. Here XxY@Z represents a convolu- tional layer with Z filters of size (X × Y). 'X' convolves through time indices and 'Y' convolves different limit order book levels.

Listing 1. sweep configuration

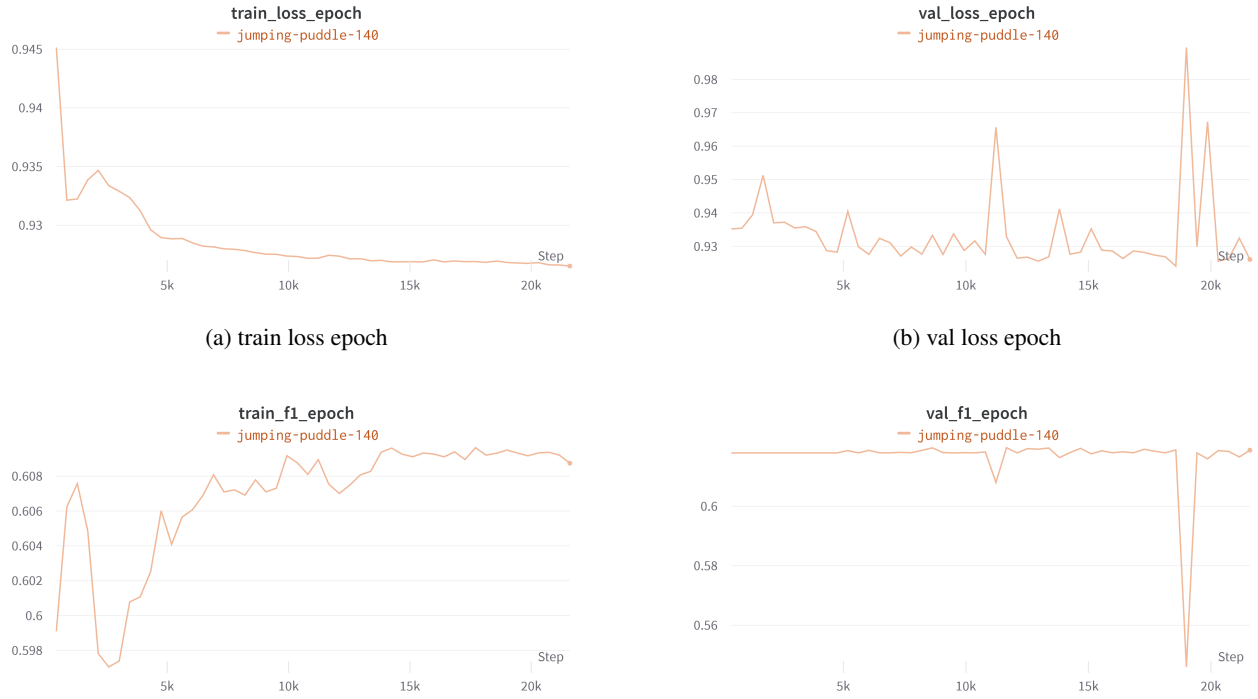
```
sweep_config = {
    "method" : "random", # Random Search
    "metric" : {
        "name" : "val_f1_epoch",
        "goal" : "maximize"
    },
    "parameters" : {
        "lr" : {
            # log uniform distribution between
            # exp(min) and exp(max)
            "distribution" : "log_uniform",
            "min" : -9.21, # exp(-9.21) = 1e-4
            "max" : -4.61, # exp(-4.61) = 1e-2
            },
        "batch_size" : {
            "values" : [128, 256]
            },
        "projection_horizon_index" : {
            "values" : [3, 4] # avoid 0,1,2
            },
        "weight_decay" : {
            "values" : [0.01, 0.03, 0.05]
            },
        "num_classes" : {
            "values" : [3]
            },}}
```

Best Results from Sweep Run In the table there are the best results and their relative parameters, obtained by the sweep run. Each run lasted at maximum 4 epochs.

batch size	lr	projection horizon	weight decay	epoch	test f1	test loss	train f1	train loss	val f1	val loss
128	0,000209	10	0,03	4	0,66	0,90	0,59	0,95	0,61	0,99

3. Results

These results fig.3 refer to the model trained for 50 epochs with the hyperparameters selected by the sweep run.



(a) train loss epoch

(b) val loss epoch

(c) train f1 epoch

(d) val f1 epoch

Figure 3. Quantitative results

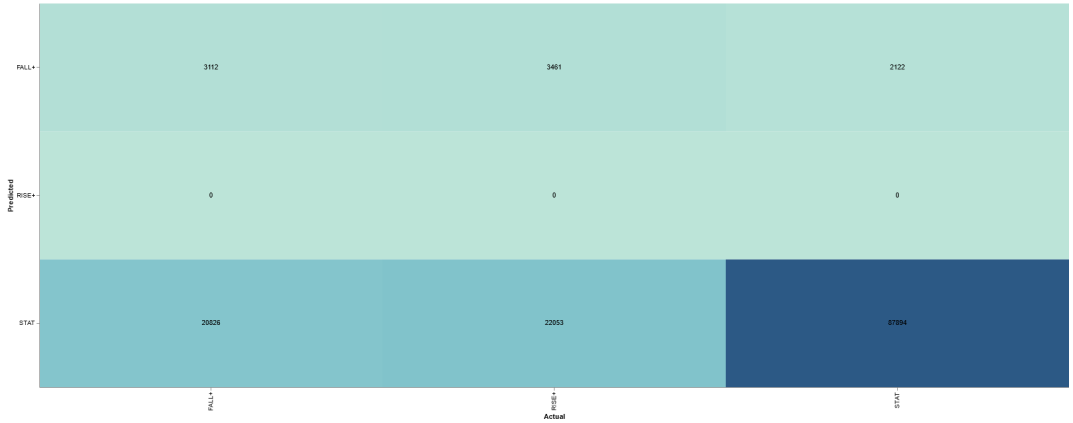


Figure 4. Test Set Confusion Matrix

4. Conclusion and Future Work

From the results above, we can conclude the while the model does not overfit, it achieves globally poor results, irrespective of the number of epochs it has been trained. The imbalanced nature of the dataset and the results shown by the confusion matrix should nudge us that probably we should either feed the model more data, or change the way the labels were created: if there is no *underlying pattern* to be discovered, then no matter how many examples we train on, eventually the model will start to simply memorizing the data (overfit) or won't improve the performance. To further investigate these results, a first attempt would be enlarging the *projection horizon* while creating the new labels: as seen in fig.1 the larger the horizon, the more the datasets labels are well distributed.

References

Chang, S.-H., Hsu, C.-W., Li, H.-Y., Zeng, W.-S., and Ho, J.-M. Short-term stock price-trend prediction using meta-learning. In *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 2900–2905. IEEE, 2021.

Ntakaris, A., Magris, M., Kanniainen, J., Gabbouj, M., and Iosifidis, A. Benchmark dataset for mid-price forecasting of limit order book data with machine learning methods, 2019.

Zhang, Z., Zohren, S., and Roberts, S. Deeplob: Deep convolutional neural networks for limit order books. *IEEE Transactions on Signal Processing*, 67(11):3001–3012, 2019.