Deep Learning in Non-Stationary Environments: Stock Price Forecasting

February 2, 2022

Andrea Trianni

1. Introduction

Stock Price forecasting is a popular task that consist in predicting the future value of a stock, based on some past evidence.
It is a well-known area of research since a desire of the human being has always been to predict the future, and therefore earn money. Furthermore, time series data have recently assumed a growing importance with the pandemic crisis, since forecasting systems has been an essential tool for driving public health decisions.
Despite the huge effort spent to solve the task, currently there isn't a general and robust solution for stock price forecasting. The main difficulty is that time series data follow a non-stationary distribution that machine and deep learning models can not handle. This because, the algorithms assume that both training and testing data are drawn from the same distribution.
The poor results are probably also related to the efficient-market hypothesis. This hypothesis affirms that the stock price follows a random walk, so the future price cannot be predicted based on past values. (Malkiel, 2020)
Although these drawbacks, data scientists try to overcome this theory, adapting deep models to deal with non-stationary data distributions. In this project, after a massive data pre-processing analysis, i have designed various deep-models based on different architectures. I have implemented both simple baseline models, and i also proposed my own approach aimed at improving the performances.
In order to do this, i have taken a trip through various solutions: from naive predictions, which simply forecast the last observed values as output, to the application of SMA[1] and the usage of statistical models. In fact, non-deep models can achieve quite good results for this task. In particular, the behaviors of ARIMA and Facebook Prophet (Sean J. Taylor, 2017) , was preliminary studied to figure out their strengths and understand how these ones can be exploited with a deep neural model.

Email: Andrea Trianni
<trianni.1806198@studenti.uniroma1.it>.

Deep Learning and Applied AI 2021, Sapienza University of Rome, 2nd semester a.y. 2020/2021.

[1]Simple Moving Average

2. Related Work

A lot of academic work has been done in this direction. For example in (Selvin et al., 2017) LSTM, GRU, and CNN based models have been tested and compared. The common approach is the sliding window representation, where the goal is to predict the price for the next step, looking few days back. Other methodologies relate to statistical methods and hybrid models.

3. Dataset

The dataset was generated from Yahoo. It contains the historical prices of Google (Alphabet Inc) of the last five years. The data points are sampled with a regular space interval, (e.g. each working day) and consist of this field: *Open, High, Low, Close, Adj, Volume*.
In order to avoid any information leakage and bias, i adopt a sequential strategy to split the dataset in train, validation and test set. The first four years (80%) were used to train the models, the next 6 months to validate, and the last 6 months to test them. This will ensure reliable test scores.

3.1. Dealing with Non-Stationarity

The main focus of the feature engineering part was to fight non-stationarity. Learning from non stationary distribution is difficult because the main statistics (mean, variance, ..) can change along the time. What the model has learn from the past can be no longer true for the future.
In order to help the learning process, i tried to convert a non-stationary environment to a stationary one. The first step is done rescaling the prices using the log transformation, since it squashes the data. It is frequently used in finance, and its importance can be understood with this example: *"Winning or loosing 1000 € is not the same important event if you already have 1 million €, or if you have nothing on your bank account."*
No other type of re-scaling technique was applied. For example min-max scaling is a wrong design choice, since we cannot assume there is a an absolute maximum in the training set. In the future, stock price can assume values never seen before, it is unbounded.
After applying the log, the second step consist in calculate

the return instead of directly use the price. The return is the difference between the price of today and the price of yesterday. The problem doesn't change since i can apply an inverse transformation to re-obtain the stock price.

$$price_k = price_0 + exp(\sum_{i=1}^{k} ret_i)$$

$$ret_i = log(price_i) - log(price_{i-1})$$

The stationarity was checked plotting the results of the partial autocorrelation function (PACF).

4. Models

For the purpose of the project four different deep models have been implemented. There are two baseline models: one that envolves the usage of a GRU cell, and another one based on an LSTM cell. Both the two architectures are able to learn from sequential data, using gates mechanism, and solving the vanishing gradient problem of the vanilla rnn (Hochreiter, 1998). A third network is based on bidirectional stacked lstm, that consist in a series of stacked lstm cells, that can learn the signal from left to right and vice-versa. The last model is a custom network that i have designed trying to raise the performances.
All the models are fed using a rolling window respect the dataset. They receive the data sequence that starts 14 day before, and they have to predict the next day values (1-*step forecast*), performing multidimensional regression.
Hyperparameters can be found on the notebook code.

4.1. My Approach

I tried to up-perform the baseline models adding some novelty idea to the network. I introduced the concept of time embedding, explained in the paper (Kazemi et al., 2019).

$$\mathbf{t2v}(\tau)[i] = \begin{cases} \omega_i\tau + \varphi_i & if \quad i = 0 \\ \mathcal{F}(\omega_i\tau + \varphi_i) & if \quad i \leq i \leq k \end{cases}$$

The time vector repr. is composed of two components:

- the first row represents a linear trend
- the second row represent a periodic trend ($\mathcal{F}$ *is just the sine function*)

It looks like the decomposition strategy of Prophet, which expect a trend line and more seasonal signals.
Another important upgrade was the introduction of a single-head self-attention mechanism that is able to connect each step of the time series the each other, understanding relations in a parallelized way (Vaswani et al., 2017).
The final network is built using this submodels. In other words, the input sequence feed an attention-layer, that consequently feed a GRU cell. The output of the cell is concatenated to the time vector embedding, and then there is a batch normalization layer. After that, the tensor pass trough a dense layer with dropout, that calculate the final output.

5. Experimental Results

The models have been tested and the results are collected in the table below. The metrics used to evaluate the performances are the usual metrics of any regression problem. In addition to them, i have also reported the binary accuracy (price can go up or down). Accuracy is important if we want use the model to build an automatic trading bot. If the price will increase, the bot has to buy now, viceversa it has to sell. Rephrasing, accuracy influence our earnings.

Table 1. 1-step forecasting Test Set Scores

Model	MSE Loss	R2 Score	Accuracy
Naive GRU	0.0237	-0.2442	47%
Naive LSTM	0.0239	-0.0510	56%
Stacked Bi-LSTM	0.0236	-0.0261	49%
Custom	0.0235	-0.0051	54%

The table show that the models obtained similar results. The custom model seems to be the more powerful, it has the best loss and the best R2 score. Accuracy is near to 50% for any of the model, confirming the random walk hypothesis.
Test set scores are not enough to fully evaluate the models. Looking at the table above, someone can think that all models have a small loss, so they perfectly fit the train set with one-step forecast, and everything is fine. This is only partially true, because we have also to compute incremental forecasting, that is what is used in a real world scenario to predict the future. At each step the new prediction is appended to the shifted input sequence. In the notebook i notice that the first three models can only predict a linear behavior, at least after few step. The custom solutions is the only one that is able to discover interesting non-linear patterns. This confirm the goodness of the design choices.

6. Conclusions

I tried both to overcome to the efficient market hypothesis and to deal with non-stationarity. The data pre-processing strategy has helped a lot in this sense, using the return of the log price. Furthermore, the custom model has raised the performance of the baselines, offering good result especially in incremental forecasting, that is what users are interested in. Anyway accuracy scores are all around 50%, this means that we can not make profit using this systems. In order to get better results it is reasonable to boost the historical price dataset with other kind of data, or for example with a real-time sentiment analysis system.
Concluding, stock price forecasting is a problem far from being completely solved.

References

Hochreiter, S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. 1998. Link to the source.

Kazemi, S. M. et al. Time2vec: Learning a vector representation of time. 2019. Link to the source.

Malkiel, B. G. *A Random Walk Down Wall Street: The Time-tested Strategy for Successful Investing*. 2020. Link to the source.

Sean J. Taylor, B. L. Forecasting at scale. 2017. Link to the source.

Selvin, S. et al. Stock price prediction using lstm, rnn and cnn-sliding window model. 2017. Link to the source.

Vaswani, A. et al. Attention is all you need. 2017. Link to the source.