Bitcoin price forecasting using GRU recurrent neural network

Flávio Silva
up201505646@edu.fe.up.pt

Gonçalo Duarte Nunes
up201504936@edu.fe.up.pt

***Abstract*—The present document is the report of the group project for the curricular unit of Machine Learning (EEC006). The approach followed to solve this forecasting problem consists on the design of a recurrent neural network model. Data was collected, analysed and processed, in order to ensure the quality of the training process. Features were selected based on correlation, excluding the ones with high correlation with price, since it is the parameter the model should estimate. Different model architectures with different combinations of hyperparameters were tested, in order to fine tune the model. The results were better than expected: the final model was able to forecast trends with great accuracy, accomplishing one of the main goals of this project, even tough it couldn't estimate precise values for the price.**

1. Introduction

Cryptocurrency is a buzz word nowadays, with more and more people getting interested in using it as an investment vehicle . Bitcoin is one of the many that's on the market today, but it was actually the first one to appear, in 2008, with the premises of creating a fully decentralized currency. In technical terms, bitcoin was created as a peer-to-peer electronic cash system, based on cryptographic algorithms such as hash functions that provide the necessary security, integrity and pseudo anonymous value exchange.
By the end of the year 2017, bitcoin had it's first big uprise, and became notorious. As popularity rose, investors started deviating from the idea of it being a currency, and more of a "speculative investment similar to the internet stocks of last century" [1].
Machine learning revolutionized the way to make predictions/decisions in many areas, from industrial, to medical, or even finance. It picks details from the past and respective outputs, and makes a model out of it to be able to predict what might happen in the future. According to several studies like [1] [2], even though there are plenty about cryptocurrency, not many of them are about predicting future prices, and the ones that are don't appear to have the best of results.
Bitcoin, and other cryptocurrencies, have a high price volatility that is not easily predicted. Some studies use internet trends, like google searches or twitter posts, to try and define this volatility, which will then be used as a feature. Picking the right features is essential in a machine learning problem, especially in this case, given the range of factors that might influence the price.
Another important aspect when dealing with this problem is to pick a good algorithm to train the model. The most commonly referred in the papers are RNN (Recurrent Neural Network) and LGTM (Long Short-Term Memory). The main difference to note is that LGTM gives more control over the outputs, preventing vanishing or uncontrolled growth of gradients (with the cost of complexity and resources).
The main objective of this project is to develop a simple predictive model for bitcoin prices, using neural networks. The work is divided in two big steps: feature selection and model construction.

2. Problem and Methodology

2.1. Feature Selection

In order to build the model, it is essential to have a good feature selection for training it. https://data.nasdaq.com/ has a set of features for several cryptocurrencies as well as other useful financial data from other markets. They also provide an API for data extraction that enables a simple integration to the code developed. After getting the features, they will need to be normalized, filtered and selected.
Normalization is essential in machine learning algorithms that use gradient descent, like the one that will be used. If we look at the formula for gradient descent:

$$\Theta_j = \Theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\Theta(x^{(i)}) - y^{(i)}) x_j^{(i)}, \qquad (1)$$

the multiplication by a factor of x (in the end) will produce different gradients steps if they are not normalized, leading to disparities. The normalization allows for a smoother rate of gradient change until they reach the equilibrium point.
The other aspect, comparing the features among each other to find out their correlation, is important because even though more features could lead to better results, if they have too high of a correlation they don't bring new information to the model.

2.2. Model Construction

As mentioned before, most of the papers talk about RNN and LSTM when predicting cryptocurrency prices. Most recent papers also talk about GRU algorithms (Gated Recurrent Unit), as an emergent and novel algorithm for time series forecasting. [3] paper compares LSTM with GRU, and concludes GRU is the best option overall, but there's no significant difference. It's also noted that GRU is really strong following tendencies, not so much the actual price.
Assuming those conclusions, a GRU network will be implemented. As for the architecture, different amounts of GRU blocks will be tested, as well as dropouts, that will help with over-fitting. The number of blocks (GRU plus dropout) will be defined after observing the results a given architecture produces. At the end of it will be a dense block, with linear activation function.

3. Implementation

As mentioned, we retrieve data about bitcoin using an API provided by Nasdaq. There are 33 different data sets available that will be downloaded and grouped in a dataframe, but not all have them are relevant since some of them are not being maintained anymore (won't be considered as they are not useful) and some other cases appear to have short periods without data, so an interpolation will be made in order to correct them.
With the features properly organized in a dataframe, an analysis on correlation between them is needed in order to drop the ones with the highest absolute correlation (0.95 and larger). The ones that have the highest correlation with the price were also excluded, because the model should be able to predict the price without directly or indirectly knowing it, in order to avoid overfitting. This selection process resulted in 13 relevant features, that will be used by the model:

- **USD Exchange Trade Volume** - The total USD value of trading volume on major bitcoin exchanges.
- **Difficulty** - A relative measure of how difficult it is to find a new block. The difficulty is adjusted periodically as a function of how much hashing power has been deployed by the network of miners.
- **Average Block Size** - The average block size in MB.
- **Cost % of Transaction Volume** - A chart showing miners revenue as percentage of the transaction volume.
- **Estimated Transaction Volume USD** - The Estimated Transaction Value in USD value.
- **Estimated Transaction Volume** - The total estimated value of transactions on the Bitcoin blockchain (does not include coins returned to sender as change).
- **Total Output Volume** - The total value of all transaction outputs per day (includes coins returned to the sender as change).
- **Total Transaction Fees USD** - The total value of all transaction fees paid to miners (not including the Coinbase value of block rewards).
- **Total Transaction Fees** - The total value of all transaction fees paid to miners (not including the Coinbase value of block rewards).
- **Total Bitcoins** - The total number of bitcoins that have already been mined; in other words, the current supply of bitcoins on the network.
- **My Wallet Number of Transaction Per Day** - Number of transactions made by My Wallet Users per day.
- **My Wallet Transaction Volume** - 24hr Transaction Volume of our web wallet service.
- **Mining Operating Margin** - Data showing miners revenue minus estimated electricity and bandwidth costs.

After analyzing the features, saving the ones that have the lowest correlation, the data has to be divided for training, validating and testing the model, and then normalized using packages like sklearn, that provide functions such as train_test_split and MinMaxScaler.
An important aspect when dealing with time series forecasting is that time is in constant move. The model will learn to estimate the instant n based on a defined look back. In other words, the forecasting of a certain instant is based on a defined number of past instants. The model should be trained in a way it ensures that behavior, so the training data should be sliced accordingly. For that goal, TimeSeriesGenerator from Keras library was used.
The starting architecture will consist of 5 blocks: 3 GRU blocks, 1 dropout (to avoid overfitting) and a dense block. Keras will be responsible for creating the model, represented in figure 1. The summary of this architecture shows that 40401 parameters will be trained.
The compiling of the model will use Adam optimizer and mean squared error to calculate the loss. The fitting will be done with 10 epochs, feeding it the generator that already contains the training data.

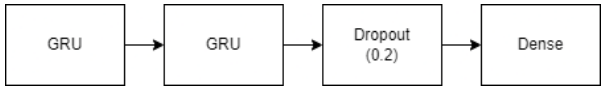

Figure 1. Model architecture

4. Results

In order to obtain the best result, some changes to the architecture were made, as well as changes to some hyperparameter. Figure 2 shows the validation for the starting architecture.

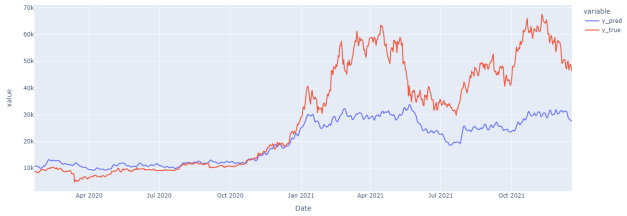

Figure 2. Validation for starting architecture

Then, it was observed that reducing the number of GRU blocks didn't affect too much the results, and the amount of time to fit the model reduced, as shown in figure 3.

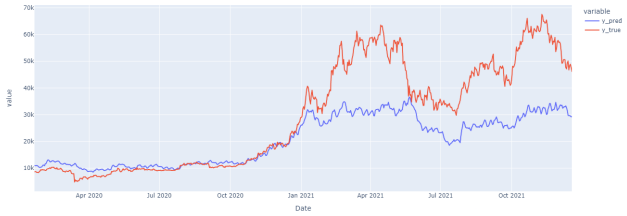

Figure 3. Validation for 2 GRU blocks

The best results were achieved after tweaking the dropout. The block was removed, and instead was used a parameter of the keras model - recurrent dropout. The results achieved are shown in figure 4.

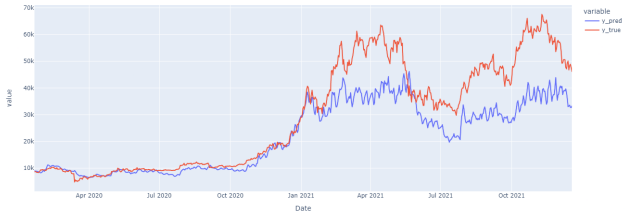

Figure 4. Validation for 2 GRU blocks and recurrent dropout

Increasing the number of epochs was also tested (figure 5), but the model didn't benefit from it at a later stage in the validation. During the first of the two years, the prediction was on point, but after that it was too sensitive to variations.

5. Conclusion

After observing the results, we can conclude that indeed the resulting trend predictions are very good, but the absolute value is not that great. Testing some other model architectures, as well as changing some hyperparameters was done in order to try and improve the results, but the observations made showed that there was some trade-off and no way of achieving a perfect result for all of the validation

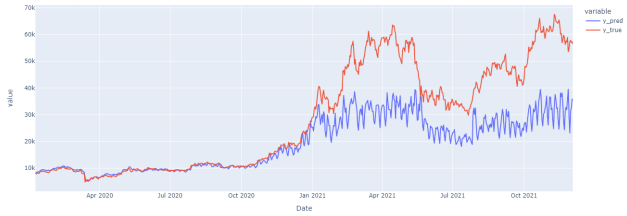

Figure 5. Validation with 50 epochs

spectrum.
The introduction of more dropout layers made the results 'flatter', even though it followed the price trend. Without them, the variations in price have a bigger slope.
Even though the results are better than expected, the introduction of some other features could benefit the model, such as 'fear and greed index' (a well known index in the market), that measures people sentiment towards the market. It would have helped to predict bigger rises/downfalls in the price, something that would be interesting to consider in a future project.

References

[1] Zheshi Chen, Chunhong Li, Wenjun Sun, Bitcoin price prediction using machine learning: An approach to sample dimension engineering, Journal of Computational and Applied Mathematics, Volume 365, 2020, 112395, ISSN 0377-0427, https://doi.org/10.1016/j.cam.2019.112395.

[2] H. Jang and J. Lee, "An Empirical Study on Modeling and Prediction of Bitcoin Prices With Bayesian Neural Networks Based on Blockchain Information," in IEEE Access, vol. 6, pp. 5427-5437, 2018, doi: 10.1109/ACCESS.2017.2779181.

[3] Hamayel, Mohammad J., and Amani Y. Owda. 2021. "A Novel Cryptocurrency Price Prediction Model Using GRU, LSTM and bi-LSTM Machine Learning Algorithms" AI 2, no. 4: 477-496. https://doi.org/10.3390/ai2040030