Predicting Ethereum Next-Day Price Movement

Project Category: Finance and Commerce & Project Mentor: Honglin Yuan

Name: Andrej Elez	**Name: Koren Gilbai**	**Name: Eastan Giebler**	**Name: Andrei Mandelshtam**
SUNet ID: aelez	SUNet ID: kgilbai	SUNet ID: eastan	SUNet ID: andman

1 Abstract

This paper is meant to analyze the price dynamics of Ethereum (ETH), a cryptocurrency. This information is relevant for investors and participants in the ETH ecosystem. We used an LSTM model and a GRU to predict the next day price of ETH given past price and other feature data. The models achieved, in the best case, a root mean squared error of 0.042763 on the test set.

2 Introduction

In this project, we investigate the problem of next-day price prediction for Ethereum (ETH), a cryptocurrency. The price of ETH can affect the portfolio value of individual investors, the value of derivatives based on ETH such as options/futures/swaps, and can affect the timing of loan repayments or the valuation of collateral. Thus the future price of ETH has wide-ranging ramifications throughout the cryptocurrency ecosystem as a whole [1].

To create a baseline, we first tried standard logistic classification and a simple neural network to determine the next-day movement (up or down) of ETH based on a variety of proprietary features. We constructed these features based on personal knowledge of the variables which are determinative of future cryptocurrency prices in general.

Having a baseline classification model, we then transitioned to doing regression in order to predict the next-day price of Ethereum rather than simply predicting up or down movement. We added multiple features, including block difficulty (which measures how tough it is to actually mine a block in terms of hash power required).

We also added new models, including LSTM and GRU. Theoretically, we would expect these models to be more effective in predicting next-day price movements because features which affect price usually act with a lag. For example, there is no reason to expect a drop in gas usage for contracts to immediately be reflected in the price of ETH, but it does seem probable that over the course of days or weeks such a change in the constituents of the ETH ecosystem would reflect itself in the price. Thus models which have a component of timeliness are theoretically more effective for this particular problem than other models. We found that this held in practice, too.

3 Related Work

One of the first articles we found examined the effects of using Ethereum-specific blockchain data on the predictions of Ethereum prices [2]. This paper was interesting because in addition to Ethereum blockchain data, it included macroeconomic variables like stock market returns as features. This added context to the model which is theoretically relevant (i.e. if stocks go up because investors want risky assets, ETH will probably go up too). This paper used SVM's and ANN's to predict the price of Ethereum, and found that ANN's outperformed. This piece led us to another article[3] which used an LSTM model to predict the price of Ethereum. Their dataset was primarily composed of on-chain data, and their discussion of the properties of LSTM's that made them uniquely suited to this problem encouraged us to examine LSTM's for our own project. We found another article[4] which we thought was an interesting example of the use of neural networks. These authors used sentiment analysis, which included the results of web scrapers coded in python. The paper used neural networks, however the authors did not include details on their model construction, and so it is unclear how else their work differed from our own. Finally, we looked at another article [5] that uses three different types of RNN algorithms all working to predict the prices of Bitcoin, Ethereum, and Litecoin. The three types of algorithms were LSTM's, GRU's, and

bi-directional LSTM's. The only feature which this article used was price, but it was surprisingly predictive nonetheless. In the literature review which it contained it highlighted the fact that LSTM's had been found to be the most accurate models to predict the price of cryptocurrencies. This article found that the GRU was most effective at predicting future price, which conflicts with our results where the LSTM was more effective. However, that may be explainable by our richer feature set.

4 Dataset and Features

The open-source project Blockchain ETL provides an Ethereum dataset that is updated daily from the mainnet blockchain. It exposes an API for ETL (extract, transform and load) jobs for Ethereum blocks, transactions, ERC20 / ERC721 tokens, transfers, receipts, logs, contracts, and internal transactions [6]. To retrieve features, we queried the API for daily data. We also used the dataset "bigquery-public-data.crypto_ethereum" which is hosted on Google BigQuery, where we ran SQL queries to extract predictive features. The dataset includes 2463 examples, each corresponding to a single, discrete day between 08/08/2015 and 05/05/2022. We also used data from CoinMetrics, a free data provider which scrapes the blockchain for a variety of on-chain data sources. We extracted a total of 118 features, and used 17 features in the the models whose results are shown here. The features are all real numbers which represent values taken from a given day. Below we explain why we chose to retrieve certain features. We will explain the data processing and test/train distribution in the experiments section.

One feature we had was the number of active public keys, which is a proxy for the number of people engaged on the Ethereum ecosystem. Next we had gas, which refers to the fees paid to conduct transactions on the Ethereum blockchain. Next we had gas usage for contracts and gas usage for accounts. After submitting the interim report, we also added the features velocity and total value locked (TVL). Velocity denotes the ratio of the total value transferred divided by the active supply in the trailing 1 year. It captures the rate of turnover, as in the average number of times a unit of ether in supply has been transferred in the past year. TVL captures the the total volume held in the Ethereum decentralized finance ("defi") ecosystem.

5 Methods

The simple logistic regression models was run using stochastic gradient descent (SGD) was used as the optimizer with a learning rate of 0.01 on the simple models and 0.005 on the multilayer model. The loss function was Binary Cross Entropy loss (BCELoss).

$$\ell(x, y) = mean(L) = mean(\{l_1, \ldots, l_N\}^T), \quad l_n = -w_n \left[y_n \log x_n + (1 - y_n) \log(-x_n) \right]$$

After the interim report we decided to use recurrent neural networks - RNNs, specifically Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRUs). Both process data stepwise, and split each step into special processing gates. Whereas GRUs have two gates - the update gate and reset gate, LSTM has three - the input, forgetting, and output gate. In GRUs the process is slightly simplified, allowing for faster learning; the update gate chooses what to incorporate from the given data into the future, and the reset gate removes excessive data. As a result, this algorithm frees up some short-term memory while incorporating what is beneficial into the long-term. LSTM uses 3 gates with regularizing 'forgetful' sigmoid functions, which convert the data to be between 0 and 1 and therefore avoid blowing up data from the past. At the same time, they also incorporate information using tanh functions. The forget gate uses the sigmoid alone to forget 'bad' parts of the data (those close to 0 will be multiplied to a value close to 0, which is 'forgotten'). To visualize how one cell of this process works, consider the below figure.

Its inputs (the two arrows pointing right on the left of the figure) are the cell state and hidden state of the previous time iteration. The first vertical arrow of the diagram (with the sigmoid function) represents the forget gate. It multiplies the sigmoid function of the hidden state with the cell state, which effectively forgets all aspects of the cell state that are deemphasized in the hidden state. The next third is the input gate. It combines the current state - the data at the given time (which comes from the bottom-left vertical arrow) - and the hidden state and applies both the sigmoid and tanh functions to their combination. The tanh function effectively regularizes the data to stay within the interval $[-1, 1]$ of values of absolute value at most 1, which sets the cell up well for multiplication of the result of the sigmoid and the result of the tanh. This is then added to the result of the forget gate to obtain the new cell state. The final part of the process is the output gate, which multiplies the tanh of the new cell state and the sigmoid of a combination of the hidden state and current data to output the new hidden state. The reason for

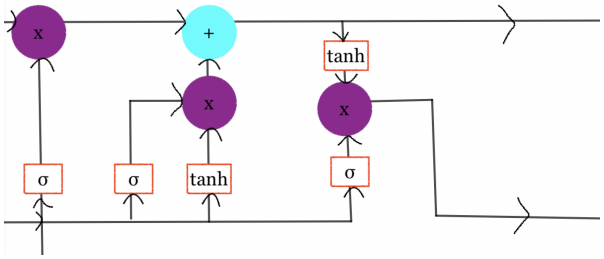

Figure 1
Long-Short Term Memory Cell State

this is because the hidden state is exactly the prediction of the model. The new hidden state is the new prediction, i.e. it multiplies the (processed together with the new data) previous hidden state by the new (regularized) cell state.

The diagram for a cell state of GRUs is quite similar, but as discussed before it incorporates one fewer gate. Specifically, it combines both the forget gate and the input gate together into one. This obviously retains less information than LSTM (which separates the 'forgotten' data and the 'input' data), but on the bright side it takes much less memory and time to train, and often works just as well or even better than LSTM. However, it incorporates the same sigmoid and tanh functions as before.

For all the RNN models, ADAM was used as the optimizer with the various learning rates described in table 2. The loss function was Mean Squared Error loss (squared L2 norm).

$$\ell(x, y) = mean(L) = mean(\{l_1, \ldots, l_N\}^T), \quad l_n = (x_n - y_n)^2$$

6 Experiments/Results/Discussion

Before discussing the performance of the RNN models on the regression task, we will examine the results of a simple logistic regression model on a classification task as a baseline of comparison. We implemented a logistic regression model with Pytorch [7] to make binary predictions about the next day price going up or down. The model was trained using the daily features described in section 3.

Using the raw, unmodified features, the model trained extremely poorly and could not determine any meaningful decision boundary from the input features (Fig. 2). This is likely due to the extreme variation of the range of input feature values across features. Without feature engineering, a simple logistic regression model was not expressive enough to determine a high dimensional decision boundary.

To improve model training, we performed feature engineering using the Scikit-learn[8] library. First, we added an intercept the the input features. Next, we performed min-max normalization upon the input features. Each input feature of the data was rescaled to the range of $[0, 1]$, with the general formula being:

$$x' = \frac{x - min(x)}{max(x) - min(x)}$$

The final preprocessing task for the data was the standardization of the input features. Specifically, each feature was standardized by removing the mean and scaling to unit variance via Z-score normalization:

$$x' = \frac{x - average(x)}{\sigma}$$

These two data preprocessing methods are widely used for machine learning algorithms, especially logistic regression. [9] Both models were trained for 100,000 iterations with a learning rate of 0.01. After reprocessing the data, the simple logistic regression achieved an accuracy of 53.5 percent on the test data.

In an attempt to improve the performance of a logistic regression model on the baseline classification task, we also implemented a simple neural network with 2 hidden layers. The final layer coupled a sigmoid activation function

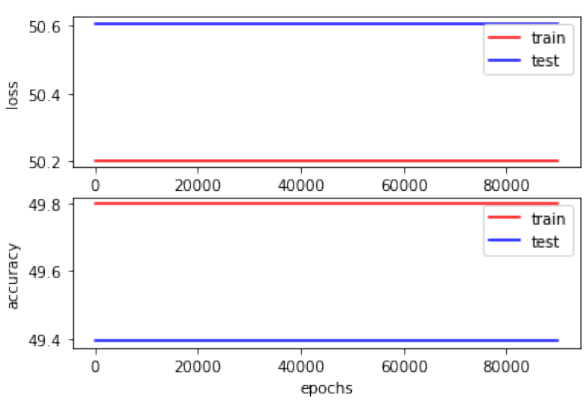

Figure 2
Without feature engineering

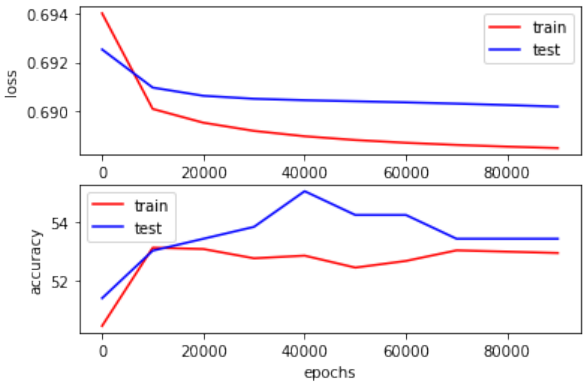

Figure 3
With feature engineering (normalization and standardization)

with Binary Cross Entropy loss to predict the next day outcome. The model was trained for 1 million iterations with a learning rate of 0.005. A relatively small learning rate was chosen as the model consistently converged to sub-optimal solutions with learning rates in the range of 0.01-0.3. This achieved an accuracy of around 55.5 percent on test data, which is slightly better than the single layer logistic regression. The low accuracy of the simple logistic regression and the multilayer model demonstrates that even with proper data reprocessing, a simple logistic regression lacks to the expressivity to classify next-day movement at a rate significantly better than a coin toss.

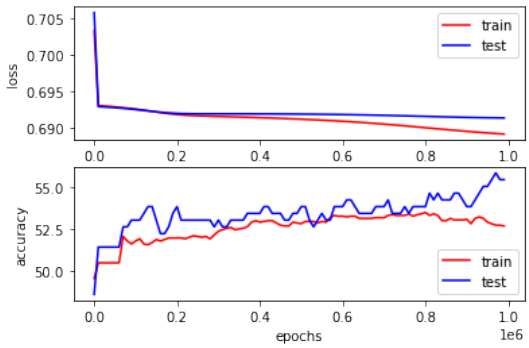

Figure 4
3 layer neural network, 1 million iterations

Following the mediocre results of the logistic regression baseline models, we decided to incorporate recurrent neural networks (RNNs) and time series into the analysis of Ethereum price. Ethereum prices inherently have many rapid changes from day to day, which make normal linear separations significantly weaker. A dynamic neural network allows for much more flexibility for the price to change, going up or down as a function of the time of day rather than large units of time. In RNNs, we can use time series data, and the output of the network will depend on the prior outputs while parameters are shared across network layers. This allowed us to construct more accurate regression from the features, while taking advantage of the timeliness inherent in price prediction. In contrast to the poor performance of the logistic regression in the classification task, the results from the RNN models on a regression task were indeed quite accurate.

We created 4 different RNN models to perform predictions on the regression task. Firstly, there were two types of RNN models we explored, LSTM and GRU. Second, we investigated the difference between RNN models using past price as the only feature to predict the next-day price and RNN models using the full 17 feature past dataset (including price) to predict next-day price. The data was preprocessed using the same methods utilized for the logistic regression models. The RMSE of each of the models predictions against the actual next-day price is as follows. The hyperparameters were tuned to each model type using a manual grid search.

Model	Training RMSE (10^{-3})	Testing RMSE (10^{-3})
Single Feature LSTM (Price)	4.229	67.936
Single Feature GRU (Price)	4.427	52.768
Multivariate LSTM	3.604	42.763
Multivariate GRU	3.983	52.473

Table 1: Comparison of the performance of the RNN models over the regression task.

Figure 5: Performance of the best RNN, the Multivariate LSTM, on the test dataset.

Table 2
Hyperparameters of the RNN models.

Parameter	Single Feature LSTM (Price)	Single Feature GRU (Price)	Multivariate LSTM	Multivariate GRU
Lookback (# of days)	50	50	10	10
Hidden Dimension	32	32	48	48
Number of RNN layers	2	2	1	1
Epochs	1000	1000	2000	2000
Learning Rate	0.005	0.002	0.002	0.001

6.1 Qualitative Findings

The first finding to note is that the lookback of the single (price) feature RNN models is much higher than that of the multivariate models. This suggests that the single variable RNNs needed more time series data than the multivariate models to improve prediction accuracy. As the number of expressive features used increased, the importance of each feature's time series memory decreased, lowering the optimal lookback. Furthermore, the single feature RNN models performed better when there were 2 RNN layers in the model compared to the multivariate models that only had 1 RNN layer. This suggests that the single feature models needed more expressivity with an additional RNN layer to extract more meaning patterns from the past price feature for the regression task. Since GRU's train faster and converge more rapidly than LSTM's, the learning rates for the GRU models was smaller than that of the LSTM models to prevent the model from diverging to suboptimal solutions.

7 Conclusion/Future Work

For future work, we would like to pursue a more systematic method for tuning the hyper-parameters of the RNN models. Specifically, we would perform an exhaustive grid search using Scikit-learn's[8] grid search package. Our LSTM algorithm was the highest performing, which we expected because of the timelines inherent in price prediction problems. If we had more resources available, we would like to optimize hyper-parameter discovery (as mentioned above) in order to further optimize our LSTM model.

8 Contributions

Eastan Giebler implemented the baseline & RNN models, and wrote up the discussion of the experiments and results. Koren Gilbai chose the features and wrote the SQL queries to pull the data from chain. Andrej Elez designed the problem structure, formatted the data and features for use in the model, and wrote up the interim report and much of the final report. Andrei Mandelshtam added mathematical analysis to the methods used, and analyzed the effects of running time as well as data structure on the fitting, and helped with the writeup.

References

[1] Ethereum price: How does eth price work & what influences it, Oct 2021.

[2] Han-Min Kim, Gee-Woo Bock, and Gunwoong Lee. Predicting ethereum prices with machine learning based on blockchain information. *Expert Systems with Applications*, 184:115480, 2021.

[3] Nishant Jagannath, Tudor Barbulescu, Karam M. Sallam, Ibrahim Elgendi, Braden Mcgrath, Abbas Jamalipour, Mohamed Abdel-Basset, and Kumudu Munasinghe. An on-chain analysis-based approach to predict ethereum prices. *IEEE Access*, 9:167972–167989, 2021.

[4] Mohammed khalid salman and Abdullahi Abdu Ibrahim. PRICE PREDICTION OF DIFFERENT CRYPTOCURRENCIES USING TECHNICAL TRADE INDICATORS AND MACHINE LEARNING. *IOP Conference Series: Materials Science and Engineering*, 928(3):032007, nov 2020.

[5] Mohammad J. Hamayel and Amani Yousef Owda. A novel cryptocurrency price prediction model using gru, lstm and bi-lstm machine learning algorithms. *AI*, 2(4):477–496, 2021.

[6] Blockchain etl: Facilitating data science on blockchain data.

[7] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[9] Joel Grus. *Data Science from scratch: First principles with python*. O'Reilly Media, 2019.