Using Bitcoin Ledger Network Data to Predict the Price of Bitcoin

John Mern[1], **Spenser Anderson**[1], **and John Poothokaran**[1]

[1]Stanford University Department of Aeronautics and Astronautics, jmern91@stanford.edu
[2]Stanford University Department of Aeronautics and Astronautics, aspenser@stanford.edu
[3]Stanford University Department of Aeronautics and Astronautics, johnmp@stanford.edu

ABSTRACT

The Bitcoin digital crypto-currency has evolved from niche financial technology to major currency, with a market capitalization of $288 billion at the time of this writing. All Bitcoin activity is tracked in a blockchain ledger which maintains a complete history of every Bitcoin transaction, offering a unique opportunity to analyze the evolving micro-economy at full resolution. This work seeks to learn a model to predict the next-day trading price of Bitcoin by using data extracted from the transaction network and other economic indicators. We report the results of conducting an ablation study on the included data using multiple common learning models. From this study, we identify the optimal model to be a convolutional neural network. We continued to refine its training process resulting in a final model which generalizes moderately well to testing conditions, though additional improvements are still required to be viable for trading.

1 Introduction

In conventional quantitative finance, trends in speculative financial assets, such as Bitcoin, are predicted using hand-designed features or technical-indicators. While these can be moderately successful in identifying general trends in representative periods of behavior, they are typically not powerful enough to provide an accurate signal on their own and must instead be augmented heavily with labor-intensive analysis. This work presents an approach to apply modern machine learning to predict price directly from available data. Our dataset includes common economic indicators such as the Dow Jones Industrial Average (DJIA) and S&P 500 indexes (referred to as ”market data” throughout the rest of this paper), as well as analytics from Google reporting trends in searches for ”Bitcoin” and other related terms (referred to as ”trends data”). We additionally include features derived directly from the Bitcoin transaction ledger (”network data”). The graph features used in our dataset are extracted from a time-series history of the graph using Principle Component Analysis (PCA) in a process that is described fully in later sections.

The study is divided into two phases. In the first phase, we compare multiple model types to determine the best choice for further optimization. In this we considered basic approaches, such as logistic regression and support vector machines (SVMs) and deep-learning approaches with both dense and convolutional architectures. Each model produced either an estimate of the daily change in Bitcoin price, or a classification of the change direction. We also conducted an ablation study on the three effect of each data sub-set. In the second phase, the training process for the highest performing model was then optimized and the final performance of the model evaluated on an unseen test set.

2 Related Work

Several attempts have been made previously to analyze the Bitcoin transaction network for the purpose of macroeconomic analysis. Works by Kondor and Baumann[1,2] suggest that distinct subgraph features exist in the Bitcoin ledger network, and that many of these features are correlated with Bitcoin price. In a subsequent work, Kondor[3] shows that the most prominent features from the spectral space of the graph provide an efficient and compact representation that retains strong correlation with the trading price. In our work we adopt this same approach, though with some minor modification as described in later sections.

Researchers in computational finance have been using neural networks for automation of algorithmic trading and portfolio optimization since the early 1990's[4]. While many of these approaches were focused initially on high-frequency trading and arbitrage, recent work has focused on applying advancements in deep-learning to provide predictions on time-series data, typically on asset price. Some works have tested dense, feed-forward networks[5] as function model. These cannot, however, account for the time-series behavior without including hand-designed features in the input dataset. A natural choice to account for time-series dynamics would be a recurrent neural network (RNN) as in[6]. RNNs are often difficult to train for very long sequences. As a result, recent works have begun to focus on convolutional neural networks (CNNs)[7], with windows sliding over prior day states to account for time series. These networks seem to provide performance comparable to a similarly sized RNN. So while both the CNN and RNN approaches provide ways to uniquely account for the temporal dynamics, only the CNN will be explored in this work.

Data and Models

The dataset used in this study is comprised of three subsets - market, trends, and network. Each dataset has one-day resolution and covers the time period from 2011 through 2013 for training, and through 2014 for development and test. The market data subset included financial indicators we anticipated may be correlated with Bitcoin price. These include the following indexes gathered from Yahoo Finance®.

- The DJIA (Dow Jones Industrial Average)
- SPY (S&P 500)
- VIX (Volatility Index)
- BZF (Brazilian currency index)
- CEW (emerging markets currency index),
- Nikkei225 (major Japanese market index)
- BTC price moving average convergence-divergence (common technical indicator)

The trends dataset includes Google search frequency of the terms "Bitcoin", "Bitcoin Price", and "Bitcoin News." The network dataset is the same dataset as used in[1]. The final network in this dataset has 24.6M nodes and 129M edges. A contraction method was applied to group address IDs belonging to the same user, reducing the total number of unique transactions considered from 129M to 24M over the three year period. From this graph, we then developed two sub-graphs of long-term users, and the most active users. Each adjacency matrix was then reduced into an edge-list

vector, and the resulting vectors were stacked into a matrix (dimensions of days by transactions). We then conducted a PCA on this matrix (using singular value decomposition since the matrix was not square), and used top four left singular vectors as our feature set over time. These vectors are interpreted as the time-varying contribution of the top-k clusters of the sub-graph.

3 Methods

We are interested in predicting the inter-day change in Bitcoin price or binary change direction. That is, given data at day t, we want to predict either $P_{t+1} - P_t$ or $sign(P_{t+1} - P_t)$. For the regression task we tested linear regression (on high-order features), a feed-forward neural network, and a convolutional neural network (CNN). For classification, we tested logistic regression, Gaussian Discriminant Analysis (GDA), a support vector machine (SVM), a binary-output feed forward neural network, and a binary-output CNN. As will be seen, the models are more easily able to accomplish the classification task than the regression. Further, the SVM, logistic regression, and CNN are the top performing models, so these are the only ones we will discuss in depth here.

Logistic Regression

We applied the typical logistic regression process to develop our model. Feature sets were constructed from the base attribute set plus polynomial terms of up to degree five. The model was trained to provide a linear combination of these features for each output dimension. The value of this linear combination was then passed through the logistic (sigmoid) function to provide an output on $(0,1)$. The function was optimized using the Newton conjugate gradient solver in Scikit-learn. The logistic function and the basic form of the Newton solver are given below.

Sigmoid Function:

$$g(z) = \frac{1}{1+e^{-z}} = \frac{e^z}{e^z+1}$$

Newton update:

$$\theta := \theta - H^{-1}\nabla_\theta l(\theta)$$

where $l(\theta)$ is the log-likelihood of the logistic model, as given by

$$l(\theta) = \sum_{i=1}^{m} y^i log(h(x^i)) + (1-y^i) log(1-h(x^i))$$

Support Vector Machine

Since the dataset was likely to include several outliers, we tested the use of an SVM classifier in addition to the regression models. The SVM constrained optimization problem is given below.

$$min_{\gamma,w,b}\ \frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\zeta_i$$
$$s.t.\ \ y^i(w^T x^i + b) \geq 1 - \zeta^i$$
$$\zeta^i \geq 0$$

In this formulation, we seek to maximize the separating hyperplane margin (by minimizing the weight norm), though we allow for non-separable cases with the introduction of the ζ as a "hinge-loss" term. This problem is solved with libsvm which employs a variant of the SMO algorithm.

Convolutional Neural Network

The CNN model takes as input a stack of the prior k feature vectors arranged into a k by m matrix. Thus each input takes the form of an "image" of the evolution of the feature state of the past k days. In this work, we tested several values of k, though 10 was chosen as the ideal window size. An important difference between the design of this network and a network typically used for image processing is that there is no spatial relationship between feature order in the feature vector. Because of this, the first layer filter must span the width of the feature vector m. This ensures that combinations of all features are considered with equal probability. The final architecture is given in the table below.

Layer	Filter Dims.	Stride	Output Channels
Conv0	[3 X m]	1	64
Conv1	[3 X 1]	1	128
Conv0	[3 X 1]	1	256
Conv0	[8 X 1]	1	512
FC0	-	-	256
FC1	-	-	128
FC2	-	-	64
FC3	-	-	32
Output	-	-	2

Each layer uses rectified linear unit (ReLU) activations. The convolutional layers all use batch normalization. The first three fully connected layers include dropout with 50 percent drop rate for regularization. The unnormalized logits from the output layer are normalized via the Softmax function. The resulting probabilities are then used in a cross entropy loss with the actual labels. The network was trained with mini-batch gradient descent using the ADAM[8] optimizer in Tensorflow[9].

4 Experiments and Results

All the previously described models were trained and evaluated on each possible combination of the data sub-sets. Only the top performing model results are given here. For details on other models, interested readers may contact the authors directly.

Figure 1 below shows the performance of the polynomial logistic classifier, SVM, and the feed forward neural network (multi-layer perceptron), trained with different combinations of data. This served as an ablation study on the data included in the feature set. The training development validation dataset was a set of 90 days randomly sampled from the three year training period. These points were excluded from the training set. The models were trained for accuracy, but were evaluated based on return on investment (ROI) gained by following the purchasing signals output by the model.

As can be seen, inclusion of all considered datasets increases the ROI for all models. Interestingly, however, inclusion of trends without network data actually decreases performance of the models. This seems to suggest that there is some relationship between the two sets that is critical for a complete picture of the state.

Using this complete set, we then compared the performance of the three models above to the performance of the CNN. As shown in the table below, the CNN drastically outperforms all the other models in classification accuracy. It also provided the best regression performance, however, those results are not reported.

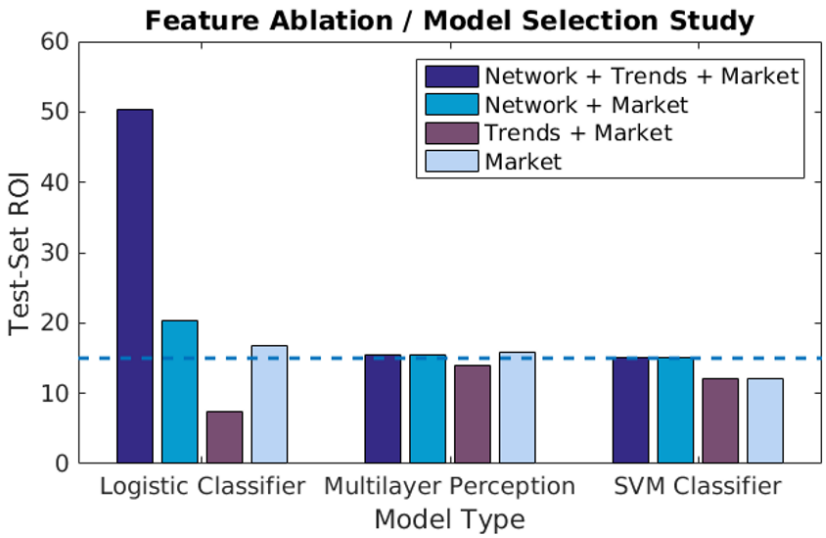

Figure 1. Dataset Ablation Study

Model	Validation Accuracy
Logistic Regression	0.653
SVM	0.673
MLP	0.684
CNN	0.967

We then focus exclusively on the CNN. We extend the training set through June 2014, and retain July through December 2014 as a test set. In this way, we test the ability of the model to generalize to potentially unseen macroeconomic conditions. We perform a hyper-parameter sweep over model training parameters, using June 2014 as the validation set. The final model retains accuracy of 59.7 % throughout the entire test set, and 66.7 % during the first month of the test set. This suggest that a sliding-window training approach would allow better long-term performance.

5 Conclusions and Next Steps

This work has shown that use of deep learning techniques, specifically a convolutional neural network, is able to generate a predictive model of Bitcoin price that generalizes reasonably well to unseen market conditions. This work also showed that use of microeconomic data from the transaction network can aid in performance. Future work should focus on incorporating RNN elements into the CNN architecture to induce dynamics directly into the network. Additional investigation is also needed into how to learn online to account for yet unseen market conditions. Work should also be done to improve the performance of regressive models, and to optimize trading strategy in-line with prediction.

6 Contributions

Each teammate participated in frequent discussions of the analysis approach, dataset, and overall direction of the project. Spenser Anderson and John Poothokaran handled collection and parsing of the market and trends data. John Mern parsed, contracted, and performed PCA on the network graph (as part of a CS224W project). Spenser Anderson was primarily responsible for the SVM

and the classification MLP. John Poothokaran conducted the logistic regression studies. John Mern developed the regression MLP and the CNN. All teammates feel the contributions have been equitable.

References

1. Kondor, D., Pósfai, M., Csabai, I. & Vattay, G. Do the rich get richer? an empirical analysis of the bitcoin transaction network. *PLOS ONE* **9**, 1–10 (2014). URL https://doi.org/10.1371/journal.pone.0086197. DOI 10.1371/journal.pone.0086197.

2. Baumann, A., Fabian, B. & Lischke, M. *Exploring the Bitcoin Network*, vol. 1 (2014).

3. Kondor, D., Csabai, I., Szüle, J., Pósfai, M. & Vattay, G. Inferring the interplay between network structure and market effects in Bitcoin. *New J. Phys.* **16**, 125003 (2014). DOI 10.1088/1367-2630/16/12/125003. 1412.4042.

4. Trippi, R. R. & Turban, E. (eds.) *Neural Networks in Finance and Investing: Using Artificial Intelligence to Improve Real World Performance* (McGraw-Hill, Inc., New York, NY, USA, 1992).

5. Dixon, M., Klabjan, D. & Bang, J. H. Classiffication-based financial markets prediction using deep neural networks. *CoRR* **abs/1603.08604** (2016). URL http://arxiv.org/abs/1603.08604. 1603.08604.

6. Fischer, T. & Krauss, C. Deep learning with long short-term memory networks for financial market predictions. FAU Discussion Papers in Economics 11/2017, Erlangen (2017). URL http://hdl.handle.net/10419/157808.

7. Borovykh, A., Bohte, S. & Oosterlee, C. W. Conditional Time Series Forecasting with Convolutional Neural Networks. *ArXiv e-prints* (2017). 1703.04691.

8. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. *CoRR* **abs/1412.6980** (2014). URL http://arxiv.org/abs/1412.6980. 1412.6980.

9. Abadi, M. *et al.* TensorFlow: Large-scale machine learning on heterogeneous systems (2015). URL https://www.tensorflow.org/. Software available from tensorflow.org.