Cryptocurrency Price Prediction

Jiejun Lu, Shiyun Qiu

May 6, 2018

Source code for our final project can be found at: https://github.com/gwungwun/AM231_Project

1 Introduction

Cryptocurrencies, such as Bitcoin, Ethereum, and Ripple are the hottest topic in finance right now. They are known for high volatility, meaning that their prices can rise and fall quickly. In 2017, the price of Bitcoin rose almost 2000% from $900 to $20000. Despite the overall increasing trend, the price can slump 40% in less than three weeks as what we observed last September. Thus, accurately predicting the price of Bitcoin is hard but lucrative.

As a relatively new asset class, not many rigorous deep learning models have been applied to the analysis of cryptocurrencies, and the high variance of the currency is usually ignored. For this project, we aim to build a predictive model for Bitcoin prices with machine learning algorithms, such as Kalman Filter, Long-Short Term Memory (LSTM) and Deep Kalman Filter.

2 Data

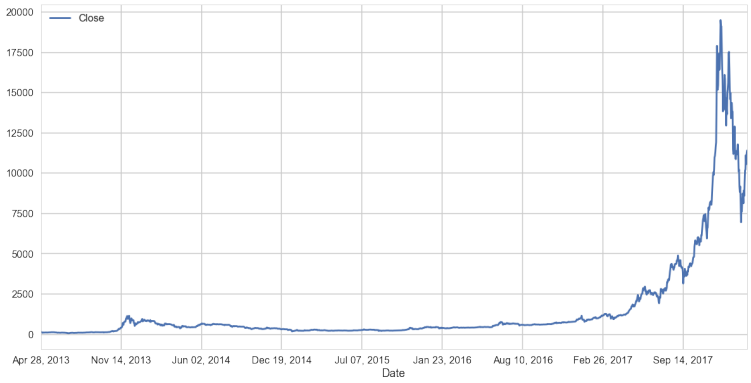

Figure 1: Price of Bitcoin from April 2013 to February 2018

The daily pricing data is collected from the Kaggle competition website, and the minute by minute data is downloaded from Coindesk.

As we can observe on Figure 1, Bitcoin price fluctuates greatly in 2017. The price can jump to two times its original value in a month, and drop back to its original value in two weeks. The overall increasing trend suddenly stopped in December, making the modeling and simulation process even harder.

3 Methodology

We adopted three models, Kalman Filter, LSTM and Deep Kalman Filter, for Bitcoin price prediction. We explained the details of our implementation and the reasons why we chose these models in the following three subsections.

3.1 Kalman Filter

Kalman filter is a conventional tool to deal with time seires forecasting. We assume the price follows a Simple Linear State Space Model(SLSSM).

$$Hidden : z_t = z_{t-1} + \epsilon_t, \epsilon_t \sim N(0, \sigma_\epsilon^2)$$
$$Observation : x_t = x_t + \delta_t, \delta_t \sim N(0, \sigma_\delta^2)$$

Here we assume there is a true hidden value for Bitcoin price at each time step, and the price we observe is generated by this true price plus a random Gaussian noise. Then we can use Kalman filter $x_{t+1|t}$ to predict the price at the next step $t+1$. However, we need to first specify the parameters. One of the solutions is to use the Expectation Maximization (EM) Algorithm to estimate the parameters σ_ϵ^2, σ_δ^2. The EM algorithm is composed of two parts: The E-step is used to derive the expected complete log-likelihood and the M-step is used to calculate the corresponding parameters that would maximize the log-likelihood. Here is the detailed derivation:

E-step:

$$E[log(p(\{z\}, \{x\}, ; \sigma_\delta^2, \sigma_\epsilon^2)|\{x\}; \sigma_\epsilon^{2(l)}, \sigma_\delta^{2(l)}]$$
$$\propto E[\sum_{t=1}^{T} -\frac{1}{2}\frac{(z_t - z_{t-1})^2}{\sigma_\epsilon^2} - \frac{N}{2}log\sigma_\epsilon^2 - \sum_{t=1}^{T} -\frac{1}{2}\frac{(x_t - z_t)^2}{\sigma_\delta^2} - \frac{N}{2}log\sigma_\delta^2|\{x\}; \sigma_\epsilon^{2(l)}, \sigma_\delta^{2(l)}]$$

We define the following terms for the M-Step.

$$z_{t|T} = E[z_t|\{x\}; \sigma_\epsilon^{2(l)}, \sigma_\delta^{2(l)}]$$
$$W_t = E[z_t^2|\{x\}; \sigma_\epsilon^{2(l)}, \sigma_\delta^{2(l)}]$$
$$W_{t,t+1} = E[z_t z_{t+1}|\{x\}; \sigma_\epsilon^{2(l)}, \sigma_\delta^{2(l)}]$$

By Kalman Smoothing, we have the following quantities [3].

$$z_{t|T} = z_{t|t} + (z_{t+1|T} - z_{t|t})$$
$$W_t = \sigma_{t|T}^2 + z_{t|T}^2, \ where \ \sigma_{t|T}^2 = (\frac{\sigma_{t|t}^2}{\sigma_{t|t}^2 + \sigma_\epsilon^{2(l)}})^2\sigma_{t+1|T}^2 + (\frac{\sigma_{t|t}^2}{\sigma_{t|t}^2 + \sigma_\epsilon^{2(l)}})\sigma_\epsilon^{2(l)}$$
$$W_{t,t+1} = \sigma_{t,t+1|T} + z_{t|T}z_{t+1|T}, \ where \ \sigma_{t,t+1|T} = (\frac{\sigma_{t|t}^2}{\sigma_{t|t}^2 + \sigma_\epsilon^{2(l)}})\sigma_{t+1|T}^2$$

Therefore, by taking the derivatives of the E-step and let it to be 0, we have the estimate of σ_ϵ^2 and σ_δ^2.
M-step:

$$\sigma_\epsilon^{2(l+1)} = T^{-1}(\sum_{t=2}^{T} W_t + \sum_{t=1}^{T-1} W_t - 2\sum_{t=1}^{T} W_{t,t-1})$$
$$\sigma_\delta^{2(l+1)} = T^{-1}(\sum_{t=1}^{T} x_t^2 + \sum_{t=1}^{T} W_t - 2\sum_{t=1}^{T} x_t z_{t|T})$$

We repeat the E-step and the M-step until they converge.

3.2 LSTM

There are several ways to improve the underlying assumptions of Kalman Filter. One problem of Kalman Filter is that it only assumes first-order dependencies. However, financial data are observed to have long-term dependencies. Therefore, we tried LSTM, which can automatically learn these long-term dependencies.[4]

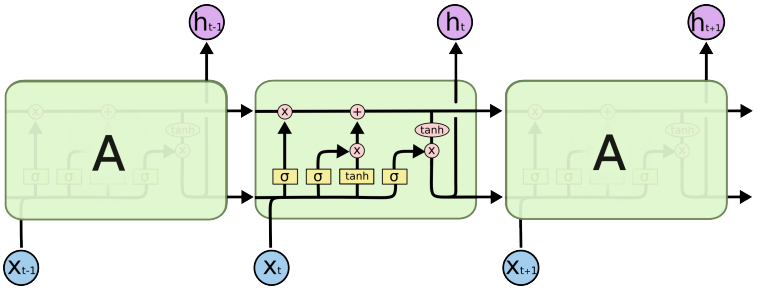

Figure 2: Core Architecture of LSTM, composed of a memory cell, an input gate, an output gate and a forget gate

However, one disadvantage of LSTM compared to Kalman Filter is its lack of interpretability. In practice, it is hard to explain exactly which features are stored or forgotten in the memory cell. In addition, it is easier to deal with inference problems using Kalman Filter as it is based on generative model.

3.3 Deep Kalman Filter

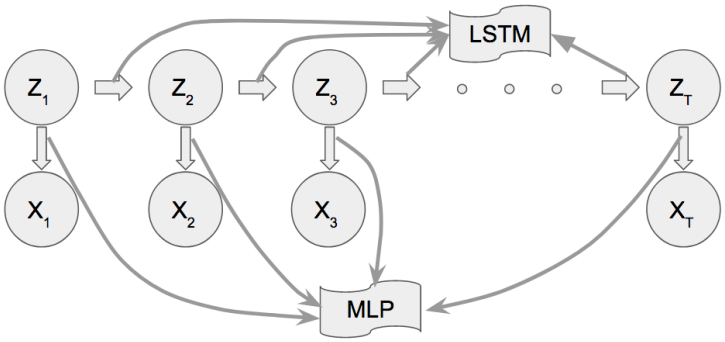

Figure 3: Structure of Deep Kalman Filter

We used a new model, Deep Kalman Filter, which combined the good properties of both Kalman filter and LSTM. We can learn the long term dependencies while still being able to deal with the inference problems.

Unlike the simple Kalman Filter mentioned in section 3.1, which uses linear functions perturbed by Gaussian noise for latent state evolution and emission distribution, Deep Kalman Filter parameterizes the latent state evolution by LSTM and the emission distribution by multilayer perceptron [1, 2].

Deep Kalman Filter will optimize the following lower bound, and update ϕ and θ using ADAM.

$$\log p_\theta(x) = \log \int_z \frac{q_\phi(z|x)}{q_\phi(z|x)} p_\theta(x|z)p_0(z)dz \geq \int_z q_\phi(z|x) \log \frac{p_\theta(x|z)p_0(z)dz}{q_\phi(z|x)}$$
$$= E_{q_\phi(z|x)}[\log p_\theta(x|z)] - KL(q_\phi(z|x)||p_0(z)) = L(x;(\theta,\phi))$$

There are two major difficulties in differentiating the lower bound:

- $E_{q_\phi(z|x)}[\log p_\theta(x|z)]$ depends on q_ϕ, where ϕ are parameters of neural nets
- The posterior distribution $q(z_1, ..., z_T|x_1, ..., x_T)$ is intractable to compute

In order to solve these problems, we assume that the latent state is normally distributed, i.e. $q_\phi(z|x) \sim N(\mu_\phi(x), \Sigma_\phi(x))$, and then we can perform stochastic backpropagation to obtain Monte Carlo estimates of the gradient of $E_{q_\phi(z|x)}[\log p_\theta(x|z)]$.

In summary, the Deep Kalman Filter uses a generative model and assumes that the latent state follows some Gaussian distribution just as simple Kalman Filter. The mean and covariance of the latent states are nonlinear functions of the previous state, previous action and time difference. G_α and S_β are parameterized by LSTM and F_k is parameterized by multilayer perceptron.

$$
\begin{aligned}
z_1 &\sim N(\mu_0, \Sigma_0) \\
z_t &\sim N(G_\alpha(z_{t-1}, \mu_{t-1}, \Delta_t), S_\beta(z_{t-1}, \mu_{t-1}, \Delta_t)) \\
x_t &\sim \Pi(F_k(z_t))
\end{aligned}
$$

4 Results

4.1 Convergence of EM Algorithm

We first modeled the daily price using the first 80% of data (before June 2017) and test on the rest 20% of data.

We run the E-step and the M-step iteratively until they converge. Figure 4 shows the results of the EM algorithm. Notice that σ_ϵ^2 is much greater than σ_δ^2, meaning the filters are dominated by the observations. Therefore, if we use Kalman Filter to predict the price at time t, it is similar to simply using the observed price at the last step $t-1$.

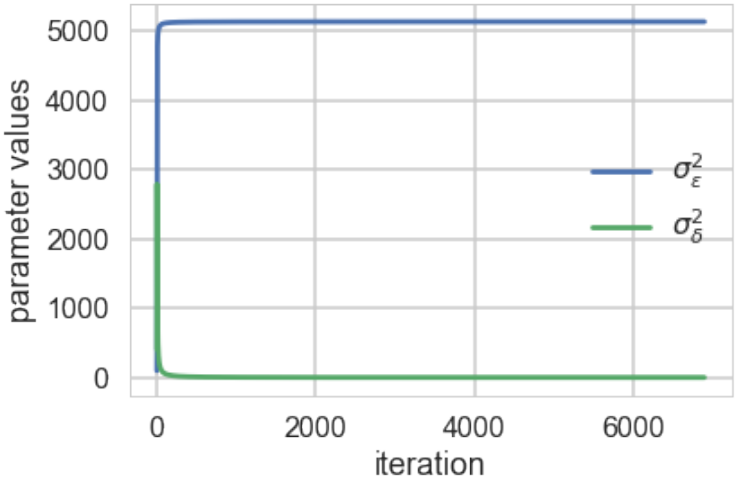

Figure 4: Convergence of EM algorithm

4.2 Comparison between LSTM and Kalman Filter

We compared the result of Kalman Filter with LSTM using daily data in Figure 5. The RMSE of Kalman Filter for training and test data is 28.065886 and 593.450079, respectively. While the RMSE of LSTM using 3-order dependencies for training and test data is 49.464597 and 599.043430, respectively, which are slightly

worse than that of Kalman Filter. That means, LSTM doesn't outperform Kalman Filter if we use fixed parameters for all periods. This makes sense since LSTM requires test data to have the same pattern as the training data, so that it can apply what it learned from the model. However, the last 20% of data is greatly different from the training data. This is another advantage of the Kalman Filter. In addition, training of deep neural network requires huge amount of data and fine tuning of parameters.

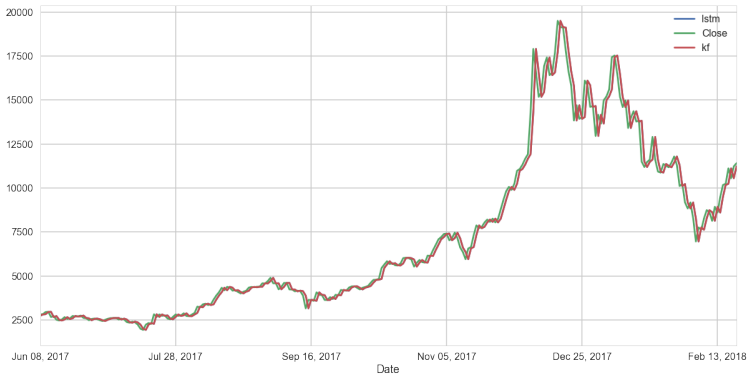

Figure 5: Comparison between LSTM and Kalman Filter

We also tried updating the parameters step by step using sliding window, but LSTM could only be slightly better than Kalman Filter in some periods.

4.3 Deep Kalman Filter

We experimented with daily data and minute by minute data using a sliding window of 10 observations. For daily data, we used 1555 samples for training, 4 samples each for validation and test. As for minute by minute data, we used 296000 samples for training, and 1989 samples each for validation and test. The performance is shown in the table below.

Deep Kalman Filter RMSE			
	Training	Validation	Test
Minute by Minute Data (MinMaxScaler)	0.022	0.036	0.037
Minute by Minute Data (Original Scale)	399.21	653.43	674.29
Daily Data (MinMaxScaler)	0.154	0.382	0.482
Daily Data (Original Scale)	2984.61	7375.02	9308.20

Deep Kalman Filter yields very bad performance on four years of daily data. With seven months of minute by minute data, we achieve a much lower RMSE. However, its performance is still worse than LSTM and Simple Kalman Filter, and it takes more than 12 hours to train a reasonably good model on minute by minute data.

Our Deep Kalman Filter cannot compete with LSTM and Simple Kalman Filter due to several reasons:

1. There are tens of parameters one can tune in the Deep Kalman Filter, such as the depth of MLP, dimension of the hidden and stochastic layers, learning rate, etc. It is very hard to find a good set of parameters. Our configuration is most likely suboptimal.

2. Since we are fitting two neural networks in the Deep Kalman filter, a huge amount of data is needed. As cryptocurrency is a relatively new asset, we do not have enough data to achieve good accuracy.

5 Conclusions and Future Work

In this project, we tested three different models, Kalman Filter, LSTM and Deep Kalman Filter, to model the dynamics of Bitcoin price. To our surprise, Kalman Filter performs the best. This may be due to the lack of data and suboptimal neural network structure and hyperparameters used in our model. Given enough data and time, we believe that LSTM and Deep Kalma Filter will outperform simple Kalman Filter.

Due to time limitation, we focused on price prediction and did not develop any trading strategy. However, people in the financial industry can easily design one based on our price prediction. Another interesting topic would be to analyze the price pattern using CNN, and predict trend of the price.

6 Reference

[1] Krishnan, Rahul G., Uri Shalit, and David Sontag. "Deep kalman filters." arXiv preprint arXiv:1511.05121 (2015).

[2] Krishnan, Rahul G., Uri Shalit, and David Sontag. "Structured Inference Networks for Nonlinear State Space Models." In AAAI, pp. 2101-2109. 2017.

[3] JONG, PIET DE, and Murray J. Mackinnon. "Covariances for smoothed estimates in state space models." Biometrika 75.3 (1988): 601-602.

[4] Hochreiter, Sepp, and Jrgen Schmidhuber. "LSTM can solve hard long time lag problems." Advances in neural information processing systems. 1997.