AC 297 Final Report: Bitcoin Trading

Daniel Rajchwald, Wenwan Yang, Wenshuai Ye

May 12, 2015

1 Introduction

Bitcoin is a virtual currency developed by an anonymous hacker under the name Satoshi Nakamoto in 2009. It is a completely decentralized peer-to-peer system governed by open sourced algorithms. Bitcoin accounts are anonymous. They are just public key addresses without any personally identifiable information attached. Unlike other currencies, bitcoin has a maximum circulation of just under 21 million coins which cause the value of each coin to increase over time. Every transaction is verified by the entire Bitcoin network which makes it nearly impossible to counterfeit a Bitcoin. The motivation of this project is to implement some trading strategies and find the arbitrage opportunities of bitcoin. Some trading algorithms will be introduced in the following part. They are either based on price trend signals or arbitrage opportunities. Our research shows that arbitrage-based trading strategies outperform signal-based trading strategies. There are two ways to find the arbitrage opportunities. The first one is to buy a dual listed stock at a lower price in one market and simultaneously selling it at a higher price in another market. The second is the cross currency arbitrage which is an act of exploiting an arbitrage opportunity resulting from a pricing discrepancy among three different currencies in the foreign exchange currencies. We got the tick level data from Morgan Stanley and it is used to test performance of each algorithm.

2 Methodology

Given the unique nature of the Bitcoin asset, the natural methodology would be to try a variety of approaches over different market conditions. Given that techniques are favorable to different trading algorithms, we implement each technique with its own algorithm. For example, pattern recognition trading is best implemented when the predictions are made with a predictive confidence interval and momentum trading can be optimal when the long and short positions are closed during the trading cycle rather than just at the end of the trading cycle. Despite the laundry list of parameters that needs to be tuned for each technique, we compare all algorithms under a common simple trading strategy where the positions are only closed at the end and a fixed number of assets is bought or sold at each trade. We hope to get a better understanding of the Bitcoin market environment given the relative performance of the algorithms over different time periods.

3 Trading Algorithms

We tried various standard trading algorithms before building more sophisticated statistical models. These algorithms include momentum trading, pairs trading, backtesting, etc., which are simple and popular strategies that have been around for decades. By implementing all the strategies on one single dataset, we can compare the results and conclude which algorithms should be used in a given market. With these exploratory analyses done, our future work will focus on incorporating the bitcoin properties we have learned so far to build more sophisticated statistical models such as a Hidden Markov Model (HMM). HMM is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobserved (hidden) states. Our current thought

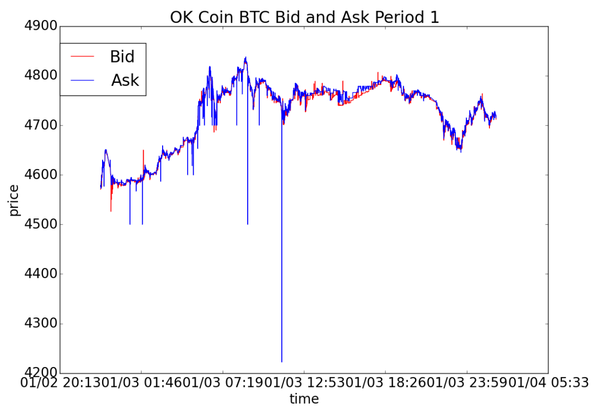

Figure 1: OK Coin Trading Period 1

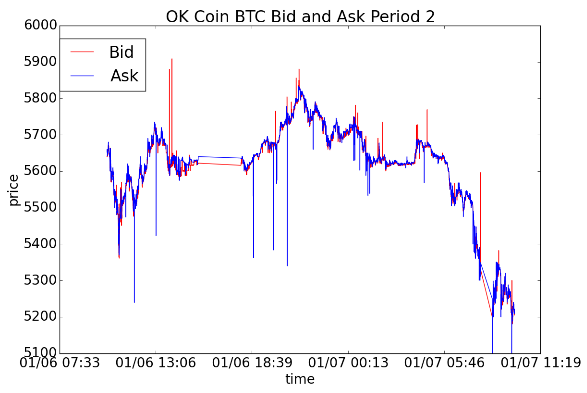

Figure 2: OK Coin Trading Period 2

is to model the intrinsic values of bitcoin as the unobserved state and the actual prices as the observed state. The trade signal relies on the current distribution of the intrinsic values. Depending on the performance, we might adjust our model when we implement it.

3.1 Momentum Trading

This strategy looks to capture gains by riding "hot" stocks and selling "cold" ones. To participate in momentum investing, a trader will take a long position in an asset, which has shown an upward trending price, or short sell a security that has been in a downtrend. The basic idea is that once a trend is established, it is more likely to continue in that direction than to move against the trend.

For example, one may consider taking a long position when the current price is greater than the moving average in a window, and a short position when the current price is less than the moving average in a window.

3.2 Pairs Trading

Given two correlated assets, pairs trading is a technique that uses mean reversion to make trades. For the Bitcoin trading, the correlated assets are actually the Bitcoin prices on two different Bitcoin exchanges. In this project, we use Huobi and Okcoin. At a time t, given the 2 exchanges, 1 and 2, one can calculate

$$D_t = F(ask_{1,t}) – bid_{2,t} \text{and} E_t = G(bid_{1,t}) – ask_{2,t} \tag{1}$$

where F(p) = a*p + b, a and b are determined through regression on training times (similarly for G):

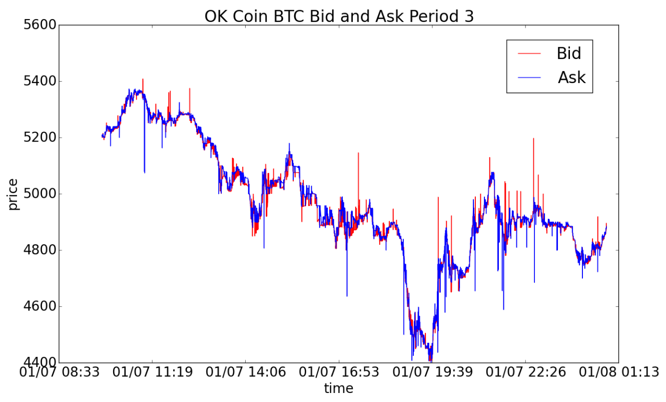

Figure 3: OK Coin Trading Period 3

$$F(ask_{1,train_1}) = bid_{2,train_1}, ..., F(ask_{1,train_m}) = bid_{2,train_m} \tag{2}$$

Then one calculates the current deviation of these quantities away from a moving average:

$$z_{D_t} = [D_t - mean(D_{t-n}, \ldots, D_{t-1})]/sd(D_{t-n}, \ldots, D_{t-1}) \text{ and}$$
$$z_{E_t} = [E_t - mean(E_{t-n}, \ldots, E_{t-1})]/sd(E_{t-n}, \ldots, E_{t-1})$$

Algorithm 1 Pairs Trading

```
Initialization: Trade = 0
for $t = 1$ to $N$ do
    if $z_{D_t} < -Threshold$ then
        Buy BTC from exchange 1
        Sell BTC from exchange 2
        Trade = 1
    end if
    if $z_{D_t} > Threshold$ then
        Sell BTC from exchange 1
        Buy BTC from exchange 2
        Trade = 1
    end if
    if $|z_{D_t}| < 0.5$ and $|z_{E_t}| < 0.5$ and Trade then
        Close positions
        Trade = 0
    end if
end for=0
```

If the z score is less than a threshold, then we expect the gap to increase at the next time step so we sell from the lower exchange and buy from the higher exchange. If the z score is more than a threshold, then we do the opposite, and close our positions in the third case.

3.3 Pattern Recognition (Pattern Matching)

Setup

This Strategy saves the current trend of bitcoin prices and looks for similar patterns in the historial trends by computing the similarity of two patterns. A pattern corresponding to a time point is defined as a list of length N that stores the percentage difference between the price of the time point and the price i time unit before that where i equals 1 to N. Similarity can be computed using normalized Euclidean distances or Pearson / Spearman Correlation Coefficient. The historical future prices of matched patterns are then collected to make predictions based on their distributions. We decided to use normalized Euclidean distance because it not only takes the linear relationship but also the scale into account. Given two patterns $A = [a_1, a_2, ..., a_N]$ and $B = [b_1, b_2, ..., b_N]$, we compute the similarity by calculating $\frac{100}{N}\sum_{i=1}^{N}(a_i - b_i)/b_i$. Because $(a_i - b_i)/b_i$ is less than 1, the similarity score here will not exceed 100. The next step is to set a similarity threshold for us to select qualified patterns. Figure 4 visualizes the matching technique by displaying matched patterns using lines. The dots represent the historical future outcomes and will be used for predictions, which would be discussed in detail.

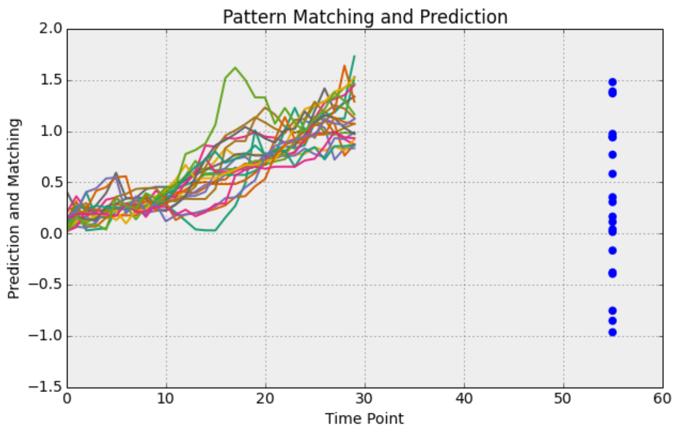

Figure 4: Pattern Matching

Prediction

With the historical future outcomes, we can construct a confidence interval to help us decide whether we should buy / sell bitcoins. In addition, we can use the confidence interval to adjust the volumes we desire to buy / sell.

3.4 HMM

Hidden Markov Model is a technique for inferring hidden states through observations when the structure of the model is known but the parameters are not. The parameters are estimated using a version of the Expectation Maximization (EM) algorithm known as the Bausch and Lomb algorithm. Formally, given

- Hidden States: $X = x_1, ..., x_n$
  - $x_i \in \{S_1, ..., S_k\}, S_j = N(\mu_j, \sigma_j^2)$
- Observations: $Y = y_1, ..., y_n$
- Parameters: $\Lambda = \{\pi = P(x_1), A_{ij} = P(S_i, S_j), \mu, \sigma\}$

$$P(X, Y) = \Pi P(x_1)\Pi P(x_i|x_{i-1})\Pi P(y_i|x_i) \tag{3}$$

After initializing parameters with k-means, one finds $\text{argmax}_\Lambda P(Y|\Lambda)$ through EM then calculates the most likely sequence of hidden states: $\text{argmax}_X P(X|Y, \Lambda)$ using the Viterbi algorithm. Then the next hidden observation is predicted: $\text{argmax}_y P(y|\text{argmax}_{x_{n+1}} P(x_n, x_{n+1}))$ from which a trading strategy can be easily generated. If a rise in price is predicted, one buys, if a fall is predicted, one sells, and if no change is predicted, one closes position. Lastly, one closes position at the end of the trading period. Note in the results that there is not a strict relationship between accuracy, RMSE, and profit among the stocks and number of hidden states, K. Given the nature of equity time series, arbitrage opportunities are very hard to track so there is a lot of randomness despite optimizing the predictions. We test for 2, 4, and 8 hidden states and pick the one that yields the most profit in the result tables.

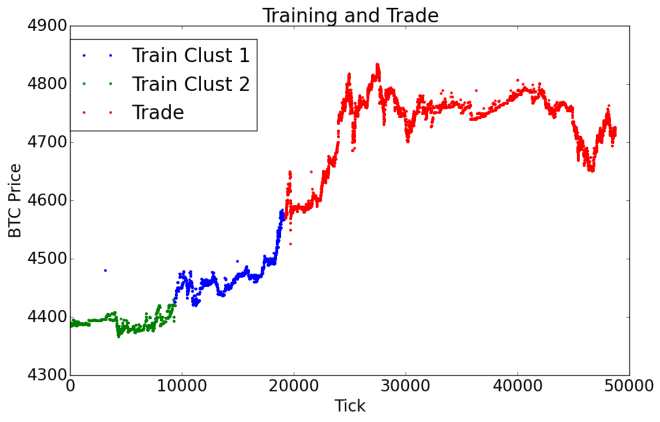

Figure 5: HMM Training

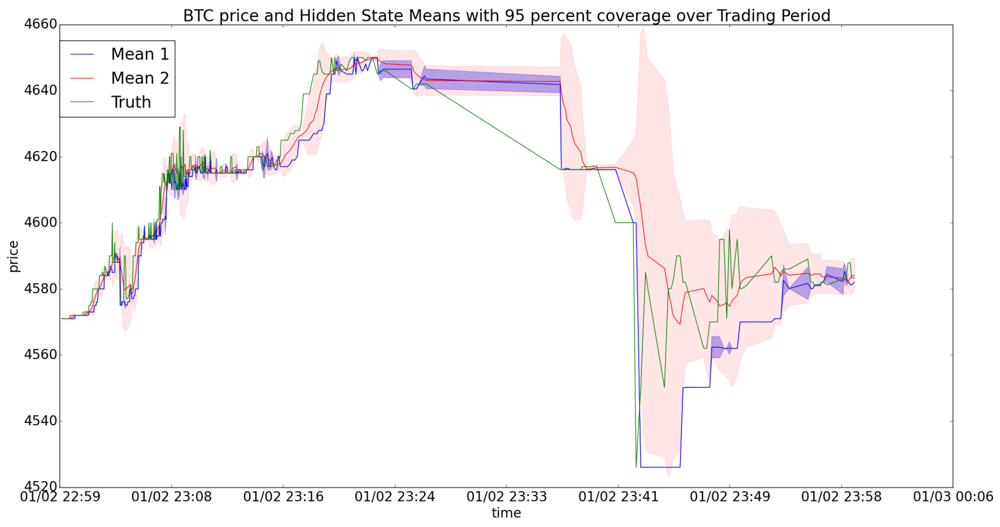

Figure 6: HMM Trading with 2 Hidden States

In Figure 5, the HMM parameters are initialized in the training region before the red curve. The number of hidden states must be specified and in this case it is 2. We use the K-means initialized with ”K-means Plus” to initialize the parameters. The HMM parameters over the first 500 trades of the bid price are illustrated in Figure 6. The relative distance and confidence intervals between the true bid price and the estimated means of the hidden states shows which hidden state the bid price is most likely to belong to at the moment.

3.5 MCMC

It has been proposed that stock pricing can be modeled by the geometric Brownian motion (GBM). One of the most important issues is to determine the distribution of stock prices, their return and other financial mean. Classical model of stock prices involves certain assumptions about the relevant financial data, which typically adopt the form of logarithmically normal distribution. The Markov chain Monte Carlo (MCMC) method is ideal to sample from empirical probability density of stock prices. This technique is flexible and calculates the probability at any given point. The purpose of this project is to model and predict stock prices using MCMC method.

Problem Setup

The stock price can be modeled by a logarithmic-normal dynamics, which can be written as:

$$d \ln S = \mu dt + \sigma S dW \tag{4}$$

where μ and σ are constants called the percentage drift and volatility, respectively. W is a Wiener process or Brownian motion. We consider the return over period Δt between two consecutive price observations to obtain:

$$Y_n = m + \sqrt{v} w_n \tag{5}$$

where $m = \mu \Delta t$ and $v = \sigma^2 \Delta t$ are the mean and variance of the return, respectively. Here we assume that the returns Y has size N, and express it as a set $Y = Y_n$, which represents the logarithmic return over a discrete price observations: $Y_n = \ln S_n - \ln S_{\mathrm{n-1}}$. w_n is a randomly distributed increment that follows normal Gaussian distribution. Our task is to estimate m and v.

Likelihood function

We view the whole historical series as an observation. Assume we start from an initial guess of parameter values m_0 and v_0. We will make a random draw with conditional probability $P(m, v|Y)$. By Clifford-Hammersley theorem, a multivariate joint distribution can be characterised completely by a set of conditional distributions. We can thus first sample for m assuming that v is fixed and then sample v assuming m fixed.

$$P(m|v, Y) = \frac{P(m, Y|v)}{P(Y)} = \frac{P(m|v)P(Y|m, v)}{P(Y)} \propto P(m)P(Y|m, v) \tag{6}$$

$$P(v|m, Y) = \frac{P(v, Y|m)}{P(Y)} = \frac{P(v|m)P(Y|v, m)}{P(Y)} \propto P(v)P(Y|v, m) \tag{7}$$

Y is given in our model by multivariate normal distribution

$$P(Y|m, v) \sim \varphi^{(n)}(Y; m, v) = \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi v}} exp(-\frac{(Y_n - m)^2}{2v}) \tag{8}$$

Conjugate Prior for the Mean

We assume m to be Gaussian distribution and parameters E_m^* and V_m^* for the centre and width of the distribution respectively

$$P(m) \sim \varphi(m; E_m^*, V_m^*) = \frac{1}{\sqrt{2\pi V_m^*}} exp(-\frac{(m - E_m^*)^2}{2V_m^*}) \tag{9}$$

Combined with the expression of Y we can get the posterior

$$P(m|v, Y) \sim \pi(m) = \varphi^{(n)}(Y; m, v)\varphi(m; E_m^*, V_m^*) \tag{10}$$

By the properties of normal distribution, it could be written as:

$$P(m|v, Y) \sim \pi(m) = \varphi(m; E_m, V_m) \tag{11}$$

with

$$V_m = (\frac{1}{V_m^*} + \frac{n}{v})^{-1} \tag{12}$$

$$E_m = V_m(\frac{E_m^*}{V_m^*} + \frac{\sum_{n=1}^{N} Y_n}{v}) \tag{13}$$

Conjugate Prior for the Variance

In our model, v is inverse gamma distributed with probability distribution function

$$f(v;\alpha,\beta) = \frac{\beta^\alpha v^{-\alpha-1} e^{\frac{-\beta}{v}}}{\Gamma(\alpha)} \tag{14}$$

$$p(v) \propto v^{-\alpha^*-1} v^{-\frac{N}{2}} exp(-\frac{\beta^*}{v}) exp(\sum_{n=1}^{N} -\frac{(Y_n - m)^2}{2v}) \tag{15}$$

$$p(v) \propto v^{-(\alpha^*+\frac{N}{2})-1} exp(\sum_{n=1}^{N} -\frac{\beta^* + \frac{1}{2}(Y_n - m)^2}{v}) \tag{16}$$

it is conjugate but with the posterior inverse gamma parameters given by

$$\alpha = \alpha^* + \frac{N}{2} \tag{17}$$

$$\beta = \beta^* + \frac{1}{2}\sum_{n=1}^{N}(Y_n - m)^2 \tag{18}$$

Algorithm with Gibbs Sampler

We estimate the parameters with MCMC simulation. We first use a Gibbs sampler and assuming conjugate priors for the parameters. The algorithm for the sampling process is briefly described.

Algorithm 2 Gibbs sampling

```
**Initialization:** $i = 0$, $m^{(0)} = m_0$, $v^{(0)} = v_0$
**for** $i = 1$ to $n$ **do**
  $m^{(i)} \sim N(E^{(i)}, V^{(i)})$
  $V^{(i)} = \left(\frac{1}{V^\star} + \frac{N}{v^{(i)}}\right)^{-1}$ and $E^{(i)} = V^{(i)}\left(\frac{E^\star}{V^\star} + \frac{\sum Y_n}{v^{(i)}}\right)$
  same for $v$
  $v^{(i)} \sim inverse\ Gamma(\alpha^{(i)}, \beta^{(i)})$
  $\alpha^{(i)} = \alpha^\star + \frac{N}{2}$ and $\beta^{(i)} = \beta^\star + \frac{1}{2}\sum(Y_n - m^{(i)})^2$
**end for**
=0
```

We treat the procedure as a random walk which is employed to produce a sample of possible model parameters. The estimation of the parameters with MCMC can be obtained by averaging over the samples:

$$\bar{m} = \frac{1}{n}\sum m^{(i)} \text{ and } \bar{v} = \frac{1}{n}\sum v^{(i)} \tag{19}$$

Then we can get the parameters from the logarithmic normal model:

$$\bar{\mu} = \frac{\bar{m}}{\Delta t} \text{ and } \bar{\alpha} = \sqrt{\frac{\bar{v}}{\blacksquare t}} \tag{20}$$

Burn-in Period

In case that we had a poor guess for the starting values. Assuming we reject the first M_B samples we start the summation from $i = M_B + 1$ and let the denominator be $M - M_B$ in the expressions for the sample averages above.

4 Results

4.1 Simple Trading

To compare the performance of all the trading strategies we implemented, we apply our predictions to a simple trading algorithm. That is, for each strategy, we predict whether the price is going up in the next time point. If it goes up, we buy 0.1 bitcoin. Otherwise, we sell 0.1 bitcoin. The position is closed at the end to calculate the remaining asset value.

Algorithm Performance (Test Period 1)

Algorithm	Profit	RMSE	Accuracy (Bid)	Accuracy (Ask)
Momentum	-12315	NA	0.11	0.08
Pairs	49709	NA	0.25	0.26
Patterns (CI)	-275	NA	0.29	0.28
Patterns (Median)	202	NA	0.29	0.29
HMM	3698	2.65	0.37	0.37
MCMC	-974	3.472	0.24	0.24

Algorithm Performance (Test Period 2)

Algorithm	Profit	RMSE	Accuracy (Bid)	Accuracy (Ask)
Momentum	-21951	NA	0.15	0.14
Pairs	56700	NA	0.28	0.29
Patterns (CI)	10148	NA	0.28	0.29
Patterns (Median)	-6010	NA	0.45	0.44
HMM	15415	141	0.22	0.21
MCMC	-12	2.892	0.23	0.24

Algorithm Performance (Test Period 3)

Algorithm	Profit	RMSE	Accuracy (Bid)	Accuracy (Ask)
Momentum	100651	NA	0.15	0.17
Pairs	74260	NA	0.30	0.28
Patterns (CI)	1308	NA	0.39	0.41
Patterns (Median)	-1693	NA	0.44	0.46
HMM	15965	130	0.22	0.27
MCMC	-722	1.673	0.23	0.25

4.2 More Complex Trading

Pattern Matching with Dynamic Volume Adjustment

Pattern maching utilizes confidence intervals to help the trader determine their next action. This can be extended to adjusting the volumes of bitcoins the trader wants to buy. For example, if the prediction outcomes are confident that the price will go up, the trader will buy more. Contrary, if the prediction outcomes do not signify any confidence in an increasing trend, the trader will buy less or do nothing. Results show that this trading strategy has stable performance and manages to earn profit in a downward trend period without short selling. For example, given the period displayed in figure 7, we found that the pattern matching strategy with volume adjustment can actually make profits in this downward trend.

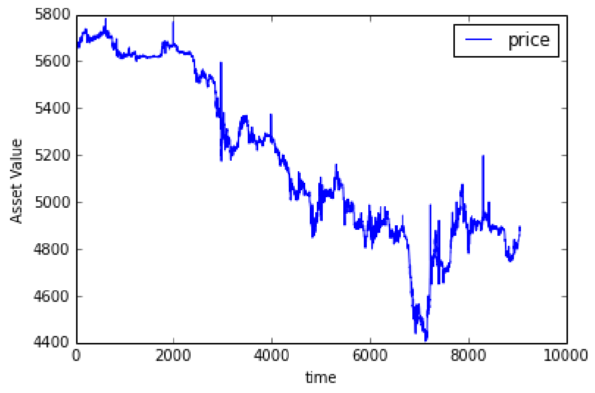

Figure 7: Bitcoin Prices (OK Coin)

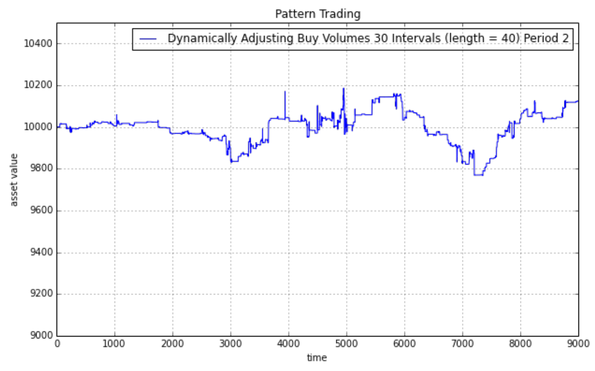

Figure 8: Asset Value Using Pattern Matching

Pairs Trading

Since pairs trading requires its own infrastructure, it did not use the simple trading algorithm described above. We varied the threshold parameter seen in Algorithm 1 to get the optimum return. As expected, prediction accuracy increased as the threshold increased since more extreme values are more likely to mean revert in the next time step.

5 Conclusion

In the results table, the performance of any algorithm is highly dependent on the time period. For example, the momentum trading algorithm yields negative returns in all periods except the third. There is also little correlation between accuracy and return. We believe the main reason for this to be that most of the time the Bitcoin prices do not change from tick to tick. For example, in the third trading period, 68 percent of Bitcoin tick prices did not change in the next tick. Thus the momentum trading algorithm predicted incorrectly for those 68 percent of ticks since the current prices is either above or below the current moving average. However, it predicts correctly 15.9 percent and incorrectly 15.7 percent when the price changes at the next tick. Therefore, algorithms may trade incorrectly without penalty during periods when the price does not change. Similar results hold for pairs trading. We also note that parameter tuning plays a huge role as pattern recognition trading succeeds over the 3 testing periods only if confidence intervals are used. Pairs trading does well in all three time periods due to the efficient market hypothesis that two different exchanges have to be highly correlated to prevent arbitrage opportunities. HMM does well in all three periods because we picked the best number of hidden states to use for each time period (either 2, 4, or 8). It was often the case that the other hidden states yielded negative returns. More work needs to be done to determine how to initialize the number of hidden states. MCMC, as shown in the table, does not have much prediction power. The reason lies in the fact that the samples generated from MCMC are only based on the expected return and variance. As a result, to predict the trend, we are actually drawing independent samples to predict next several ticks, which are extremely random. As a next step, we also plan to develop an ensemble trading strategies that takes advantage of different algorithms. For instance, combining the predictive power of momentum and HMM trading since the former is based on mean diverting predictions and the latter is based on mean reverting prediction. By taking a weighted average of the accuracy of the two methods over a historical window, once can dynamically determine the best way to trade. Overall, out efforts take a rigorous first step into Bitcoin trading and set the stage for future efforts.

References

[1] Idvall. Jonsson. Algorithmic Trading: Hidden Markov Models on Foreign Exchange Data, Linkopings University, January 2008

[2] P. Wilmott, Introduces Quantitative Finance, 2ed., John Willey, 2007

[3] B. Eraker, MCMC analysis of diffusion models with applications to finance, Journal of Business and Ecomomics Statistics, 2001, 19, 177

[4] Nakamura seminars, Quantitative Finance Review, 2014