Weekly Progress: 7 July - 13 July 2014

July 11, 2014

All of the source code for this work can be found on GitHub at:

http://github.com/cbuntain/ChangePointDetection

1 Achievements

- **Procured Bitcoin market value data across four currencies** – We acquired the MtGox Bitcoin exchange data from Quandl.com (link here: http://www.quandl.com/markets/bitcoin-data). This data set covers 3 October 2011 to 25 February 2014 (when MtGox was shut down).
- **Developed a test suite for the LRT and Cusum algorithms** – This suite supports testing across various dimensions of data ($k \in \{2, 4, 6, \ldots, 30\}$), a range of change points ($\{0, \ldots, 10\}$), randomly generated Φ matrices, and randomly generated data means. Each run of the test suite can perform an arbitrary number of runs across an arbitrary time series size, which are currently set to 500 and 10,000 respectively.
  - **Φ Generation** – Generating satisfactory Φ matrices is a non-trivial task as randomly generated matrices tend to violate constraints needed for VAR processing. To synthesize constraint-satisfying matrices, we performed the following:
    1. Generate a sparse $k \times k$ candidate matrix Φ_c with density=0.25. Ssparse matrices were used to avoid issues with generating a full $k \times k$ set of satisfactory values, which is unlikely for large k and results in long loops trying to generate good matrices.
    2. Construct a new matrix $\Phi_c = \Phi_c + \Phi_c^T + zI$ where z is some random integer in $\{2, \ldots, 9\}$. This formula was used to ensure Φ is symmetric positive-definite.
    3. Calculate the eigenvalues λ for the candidate Φ_c and use them to determine whether the zeros of the equation of the determinant $|\Phi_c(x)| = |I - \Phi_{c1}x|$ are on or outside the complex unit disk. To ensure this, we need to ensure all $\lambda^{-1} > 1$. If not, we try to scale down the Φ_c matrix by multiplying by $(\max(\lambda) + 1)^{-1}$. If this scaling still does not provide satisfactory Φ_c, we restart the process with a new candidate.
  - **Result logging** – Running logs of the suite's execution are saved to a log file, and results such as identified versus actual change points and computed W matrices are stored in a JSON file for review.
  - **Accuracy and scoring** – The number of true positives, false positives, and false negatives are also calculated for each run of each algorithm. A true positive is defined as a change point detected within ± 20 time steps of the actual change point. False positives are identified change points outside this range, and false negatives are change points missed by the algorithms. Once the test suite has finished its runs, it then calculates the accuracy and F1 score for each algorithm.
  - **Compared LRT and Cusum across dimensions** – With the test suite, we were able to compare the relative performance of LRT and Cusum as k increases (see Figure 1). Increases in k reduce accuracy in both algorithms, but Cusum is much more sensitive to this change.

  - **Identified a problem for high-dimensional data** – According to Galeano and Peña's paper, two potential changepoints must be at least d timesteps away to be considered distinct changepoints, where d is a function of the orders of the VARMA lag polynomials as well as the dimension k. This d is quadratic with respect to k, so when k is large, the minimum distance between potential changepoints is very large. This issue makes sentiment analysis for relatively high-dimensional data like Twitter problematic.

- **Wrote cleaner pseudocode for the Cusum algorithm** – The procedure as detailed in Galeano and Peña's paper is not very clear on several points. We drafted a cleaner description of the implementation for future reference.

- **Implemented Desobry's Kernel Change Detection algorithm** – Leveraged Python's Scikit package and its OneClassSVM implementation to construct dissimilarity indices for points in a given time series. This algorithm uses a sliding window of size $2m$ to calculate this index for a given t and compares it against a user-provided threshold η (generally, $\eta \in [0, 1]$).

  - **Preliminary experimentation** – This algorithm seems to work well for structural changes such as mean shifts but does not seem well-suited to discriminating two regimes between which only the covariance changes (Figure 2). We are currently exploring feature extractors to provide better information in this regard.

- **Attempted implementation of online algorithm based on Gaussian processes** – Studied a changepoint detection algorithm making use of Gaussian processes (by Saatçi, Turner, Rasmussen) based on the Bayesian Online Changepoint Detection algorithm (by Adams, MacKay) and deemed it too Bayesian to understand.

- **Started a changepoint detection algorithm based on direct density-ratio estimation** – Began implementation of a fourth, online changepoint detection algorithm using probability density ratios by Kawahara and Sugiyama.

- **Ran Bitcoin data through LRT, Cusum, and KCD** – We searched for change points in the Bitcoin data using all three algorithms currently implemented. Results are shown in Table 1.

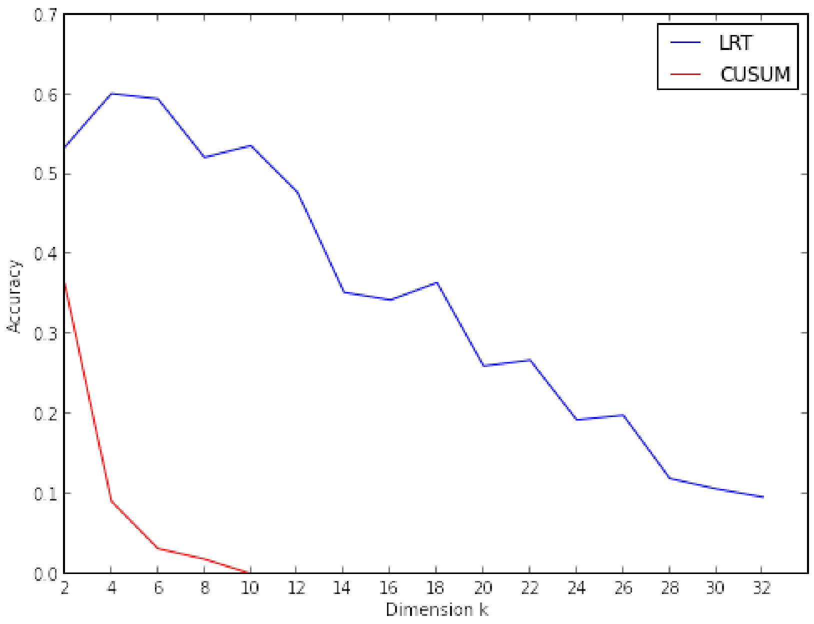
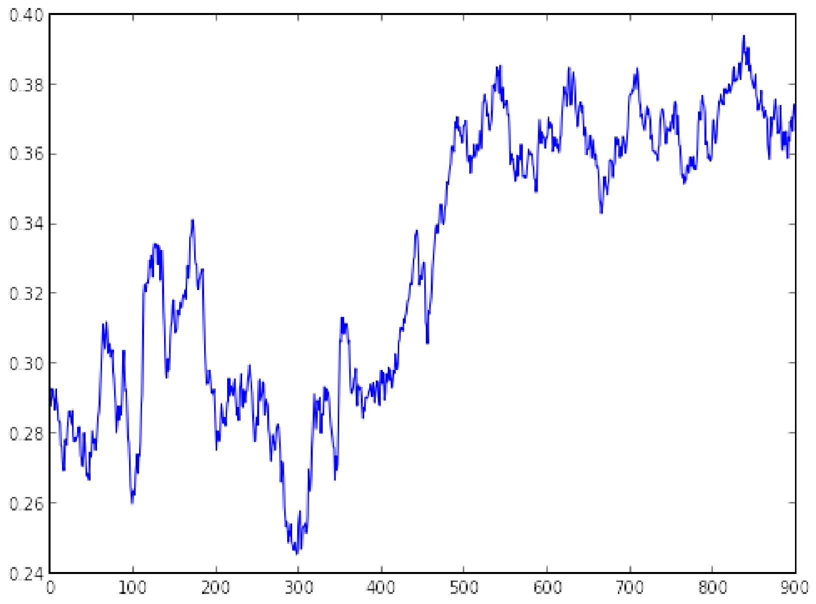
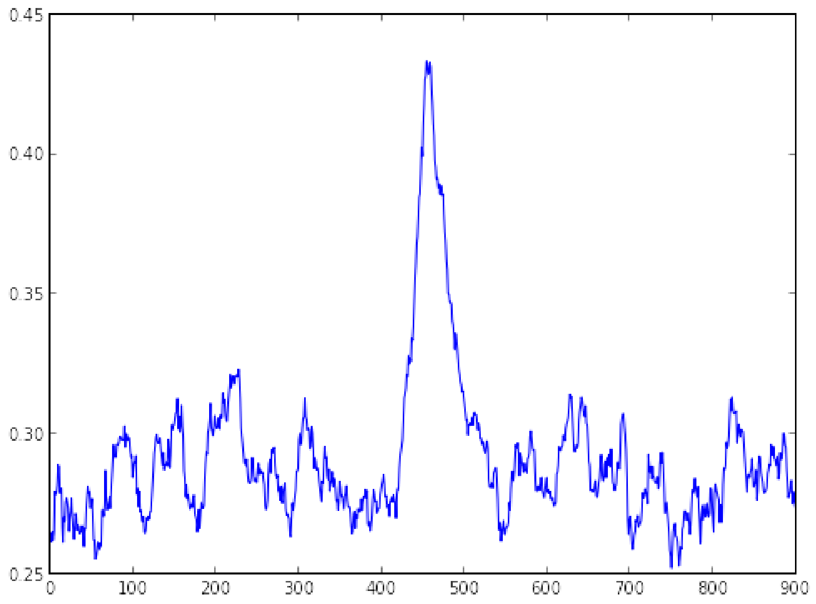


Figure 1: Accuracy of LRT versus Cusum



(a) Covariance Shift



(b) Mean Shift

Figure 2: KCD Indices

Table 1: Detected Change Points

LRT	**Cusum**	**KCD**
11/13/11		
12/29/11		
1/20/12		
3/27/12		
5/13/12		
6/7/12		
7/4/12	7/14/12	7/7/12
7/31/12		8/2/12
9/6/12	8/26/12	
12/11/12		12/17/12
1/19/13	1/19/13	
2/28/13		
3/26/13	3/9/13	3/26/12
5/3/13	5/3/13	
6/6/13		7/25/13
10/15/13		
11/5/13	11/9/13	

2 Plans for the Upcoming Week

- **Evaluate implementations on other model types** – Since our current implementations of LRT and CUSUM statistics rely on a library that can only perform MLEs for VAR-based series, we need to figure out how well (if at all) this works for VARMA or VARIMA and models of different (p, d, q) orders.
- **Continue implementing online algorithms** – We need to finish implementation of the fourth algorithm based on density-ratio estimation.
- **Dive deeper into the bridge and Bitcoin data sets** – While we have some data on detected change points, we need to figure out whether those change points actually correspond to real-world events.