A Brief Comparison of Algorithms for Detecting Change Points in Data

Cody Buntain
Department of Computer Science
University of Maryland
College Park, MD 20742
cbuntain@cs.umd.edu

Christopher Natoli
Department of Statistics
University of Chicago
5801 S Ellis Ave, Chicago, IL 60637
chrisnatoli@gmail.com

Miroslav Živković
Institute of Informatics
University of Amsterdam
1012 WX Amsterdam, Netherlands
m.zivkovic@uva.nl

***Abstract*—This is an IEEE based template that can be used for presenting your work on the Open Science Data Cloud. Use it for the PIRE Workshop challenge and other submissions such as the Supercomputing 2014 conference. Write your own abstract here.**

I. Introduction

In many applications where monitoring plays a role (Internet, smart energy grids, reservoir engineering), it is crucial to be able to detect points in time in which anomalies occur (indicating potential failures or attacks). This task often involves multidimensionality (more than one sensor) with dependence between sensors and time. Many of the well-known anomaly detection methods, however, assume one-dimensional and independent input. These assumptions are of course too unrealistic in many cases, and straightforward application to more complex problems frequently leads to erroneous results. Dependence is thus a common characteristic of multidimensional data (measurement). There exist several generic models known to have good capabilities to model data originating from multidimensional and dependent measurements, especially multidimensional autoregressive, moving average (ARMA) time series models, which are popular in econometrics research. Existing literature on machine learning techniques show that violated model assumptions do not necessarily obviate the efficacy of a given algorithm. As such, the research documented herein compares three algorithms for detecting change points, two of which are parametric and make assumptions of the underlying data, and the third is non-parametric and does not try to model underlying data distributions. Additionally, our investigation focuses on multidimensional data that exhibit "abrupt" change points and can include multiple change points over time.

We start by giving a short overview of generic models for multidimensional data in Section 2, describing in particular the multidimensional ARMA processes. Then we will give a literature overview of change point detection methods for time series data, focusing in particular on multidimensional data, in Section 3. From this overview we will describe the methods we have chosen to implement in more detail in Section 4. Section 5 then covers the comparative performance among the algorithms we implemented before we close in Section 6.

II. Generic Models For Multidimensional Time Series Data

Pull from Miroslav's content

III. Related Work

Pull from Miroslav's content

IV. Implemented Algorithms

A. Likelihood Ratio Test

In Galeano and Peña's work on detecting covariances changes in multivariate data, they proposed two methods for calculating test statistics from which change points could be identified [1]. These methods model the given data as a vector autoregressive integrated moving average (vARIMA) popular in economics and financial market analysis, extracting the errors (or innovations) from this data, and applying these methods on this error data. The first such statistic, on which we focus here, uses a likelihood-ratio test (LRT) to compare two hypotheses: the null hypothesis H_n that the covariance of this error data is best characterized by a single covariance matrix Σ versus the alternative hypothesis H_a that, at some time point h, the data is best characterized by two separate covariances matrices Σ_1 before h and Σ_2 after h. The logarithm of a modified form of the ratio H_n/H_a then generates a test statistic LR_h that existing literature shows is governed by a chi-squared distribution with degrees of freedom proportional to the dimensionality k of the data. From simulations of this distribution, we can generate a critical value given some α against which to compare this test statistic to determine whether a change point actually exists at some time h.

1) Algorithm: Given some time-series data $\tilde{y}_t$ and confidence α, we use the following algorithm to identify points of change in covariance:

```
Function LRT($\tilde{y}_t, \alpha$) Algorithm by Galeano and Peña [1]
fit VARIMA($p, d', q$) model to $\tilde{y}_t$ ;
compute residuals $\hat{e}_t$ ;
$k \leftarrow$ dimension($\tilde{y}_t$) ;
$d \leftarrow k(p+q+1) + \frac{k(k+1)}{2} + 1$ ;   /* minimum points needed */
$n \leftarrow$ len($\tilde{y}_t$) ;
$df \leftarrow \frac{k(k+1)}{2}$ ; /* degrees of freedom for $\chi^2$ */
$C \leftarrow$ simulateChiSquareMax($df, \alpha$) ;  /* obtain the critical value */
$LR \leftarrow$ zeros($n$) ;
$S \leftarrow \frac{1}{n}\Sigma_{i=1}^{n} e_i \cdot e_i'$ ;
for $h \in [d, n-d-1]$ do
    $v \leftarrow h/n$ ;
    $S_1 \leftarrow \frac{1}{h}\Sigma_{i=1}^{h} e_i \cdot e_i'$ ;
    $S_2 \leftarrow \frac{1}{n-h}\Sigma_{i=h+1}^{n} e_i \cdot e_i'$ ;
    $LR[h] \leftarrow n \ln \frac{|S|}{|S_1|^v |S_2|^{1-v}}$ ;
end
$h_{max} \leftarrow argmax_h(LR)$ ;
$\Lambda_{max} \leftarrow LR[h_{max}]$ ;
changePoints $\leftarrow$ [ ] ;
if $\Lambda_{max} > C$ then
    changePoints += $h_{max}$ ;
    $W \leftarrow$ transformation governing new data regime (see [1]);
    changePoints += apply LRT to $\hat{e}_t[0 : h_{max}]$ ;
    changePoints += apply LRT to $W \cdot \hat{e}_t[h_{max}+1 : n]$ ;
end
return changePoints
```

2) Implementation Details: To implement LRT, we used Python and Scikit's statsmodels package for fitting data to VAR() models. One should note this restriction to VAR() models is a result of an existing constraint in the statsmodels package.

We also implemented a version of the LRT algorithm that does not rely on calculating the W transformation matrix. Rather than evaluating W, we leveraged statsmodels and its maximum likelihood estimation to fit the data to two new VAR() models for each regime. The above algorithm performs better than this secondary implementation because it obviates the need for separate rounds of maximum likelihood estimation for each level of recursion.

B. CUSUM

The following changepoint detection algorithm, which focuses on changes in the covariance matrix, was designed by Galeano and Peña [1].

Suppose the k-dimensional time series $\boldsymbol{y}_t$ can be represented as a VARIMA process

$$\Phi(B)\boldsymbol{y}_t = \boldsymbol{c} + \Theta(B)\boldsymbol{a}_t,$$

where B is the backshift operator and the error or innovation $\boldsymbol{a}_t$ is an iid Gaussian random variable with mean $\mathbf{0}$ and covariance matrix Σ_i, where i indicates the particular regime.

The cusum test statistic is derived from accumulating the sum of squared errors. Since we do not know the true errors $\boldsymbol{a}_t$, we first fit a VARIMA process $\hat{\boldsymbol{y}}_t$ to the time series $\boldsymbol{y}_t$ and use the residuals $\boldsymbol{e}_t = \boldsymbol{y}_t - \hat{\boldsymbol{y}}_t$ as estimates for the errors. Consider the interval $\{\ell, \dots, r\}$. We can estimate the covariance matrix of the time series in this interval by

$$\hat{\Sigma}_\ell^r = \frac{1}{r-\ell} \sum_{t=\ell}^{r} \boldsymbol{e}_t \boldsymbol{e}_t^\top.$$

Denote the sum of squares accumulated up to time h by

$$A_\ell^r(h) = \sum_{t=\ell}^{h} \boldsymbol{e}_t^\top \left(\hat{\Sigma}_\ell^r\right)^{-1} \boldsymbol{e}_t,$$

where the squared errors are in some sense normalized by the empirical covariance $\hat{\Sigma}_\ell^r$ of the entire interval.

The cusum test statistic for a changepoint at time $h+1$ is then

$$C_\ell^r(h) = \frac{h}{\sqrt{2k(r-\ell+1)}} \left(\frac{A_\ell^r(h)}{h} - \frac{A_\ell^r(r-\ell+1)}{r-\ell+1} \right).$$

The term outside the parentheses is another normalization factor. The left-hand term inside the parentheses is the cumulative sum of squares up to time h. The right-hand term inside the parentheses is the cumulative sum of squares for the entire interval $\{\ell, \dots, r\}$. The test statistic $C_\ell^r(h)$ compares these latter two terms. For example, if there is no changepoint in $\{\ell, \dots, r\}$, then $\frac{A_\ell^r(h)}{h}$ and $\frac{A_\ell^r(r-\ell+1)}{r-\ell+1}$ will be approximately equal. However, if there is a single changepoint at time $t = h+1$, then the left-hand term is the sum of squares for all of the first regime (and only the first regime), while the right-hand term is the sum of squares over the entirety of both regimes. This suggests that the cumulative sums of squares are most different, and thus $C_\ell^r(h)$ is greatest in magnitude, at the changepoint $t = h+1$.

This observation is confirmed by plotting $|C_\ell^r(h)|$ vs. h for simulated data. Therefore, let

$$\Gamma_\ell^r = \max_{h\in\{\ell,\dots,r\}} |C_\ell^r(h)|$$
$$\bar{h}_\ell^r = \operatorname*{argmax}_{h\in\{\ell,\dots,r\}} |C_\ell^r(h)|.$$

Galeano and Peña prove that Γ_ℓ^r asymptotically has the distribution of the supremum over $[0,1]$ of a Brownian bridge. This is a known distribution, allowing us to calculate the critical value C_α for any given significance level α.

1) Algorithm for finding multiple changepoints: The approach to finding multiple changepoints is similar to binary segmentation. Let $d = k(p+q+1) + \frac{k(k+1)}{2} + 1$ be the resolution of the changepoint detection algorithm. First fit a VARIMA model to the data and computing the residuals. Then let $h_{\text{first}} = 1 + d$ and $h_{\text{last}} = T - d$, the final time step. Search for a changepoint in $\{h_{\text{first}}, \dots, h_{\text{last}}\}$ by checking if $\Gamma_{h_{\text{first}}}^{h_{\text{last}}}$ exceeds the critical value. Let $\bar{h}_{\text{old}} = \bar{h}_{h_{\text{first}}}^{h_{\text{last}}}$.

If a candidate is found, split the entire time series at the time $t_2 = \bar{h}_{\text{old}}$ of the candidate changepoint. Check $\Gamma_{h_{\text{first}}}^{t_2}$ if greater than the critical value; if so, redefine t_2 as $\bar{h}_{h_{\text{first}}}^{t_2}$. Repeat until $\Gamma_{h_{\text{first}}}^{t_2}$ is no longer significant. This procedure thus finds the earliest point in the time series at which a changepoint could occur. Redefine h_{first} as the last significant value of $\bar{h}_{h_{\text{first}}}^{t_2}$. Perform the same procedure over the interval $\{\bar{h}_{\text{old}}, \dots, h_{\text{last}}\}$, acquiring a new h_{last} as the latest point in the time series at which a changepoint could occur. If $|h_{\text{first}} - h_{\text{last}}| < d$, i.e., the resolution of the algorithm is not high enough to distinguish between h_{first} and h_{last}, then record $\bar{h}_{\text{old}}$ as a candidate changepoint. Otherwise, record both h_{first} and h_{last} as candidate changepoints. Then repeat this procedure in the narrower interval $\{h_{\text{first}}, \dots, h_{\text{last}}\}$. Thus, rather than continually cutting the interval in two and repeating the procedure in each part, the algorithm instead narrows down the interval in which changepoints can occur, using previous changepoints as the new endpoints of the narrower interval.

The algorithm ends up find excess candidates. To solve this retrospectively, let $x_1 = 1$, $x_s = T$, and $\{x_2, \dots, x_{s-1}\}$ be the sorted list of candidate changepoints. In each interval $\{x_i + 1, \dots, x_{i+2} - 1\}$, drop x_i from the list of candidates if $\Gamma_{x_i+1}^{x_{i+2}-1}$ is insignificant. Repeat this procedure until convergence. Then remove x_1 and x_s from the list of candidate changepoints. Denote the winnowed list of candidates by X. Then the final changepoints detected by the algorithm are $\{x + 1 : x \in X\}$.

The entire procedure is detailed more explicitly in Algorithm **??**.

Algorithm 1: Cusum algorithm by Galeano and Peña. The four steps correspond to Galeano and Peña's enumeration.

```
Step 1   fit VARIMA model to $\boldsymbol{y}_t$;
         compute residuals $\boldsymbol{e}_t$;
         $candidates \leftarrow \{1, T\}$;
         $h_{\text{first}} \leftarrow 1 + d$;   $h_{\text{last}} \leftarrow T - d$;
         while True do
Step 2       if $\Gamma_{h_{\text{first}}}^{h_{\text{last}}} < C_\alpha$ then
                 break;
             else
                 $\Gamma_{\text{old}} \leftarrow \Gamma_{h_{\text{first}}}^{h_{\text{last}}}$;   $\bar{h}_{\text{old}} \leftarrow \bar{h}_{h_{\text{first}}}^{h_{\text{last}}}$;
                 $\Gamma \leftarrow \Gamma_{\text{old}}$;   $\bar{h} \leftarrow \bar{h}_{\text{old}}$;
Step 3a          while $\Gamma > C_\alpha$ do
                     $t_2 \leftarrow \bar{h} - 1$;
                     $\Gamma = \Gamma_{h_{\text{first}}}^{t_2}$;
                 end
                 $h_{\text{first}} \leftarrow t_2$;
                 $\Gamma \leftarrow \Gamma_{\text{old}}$;   $\bar{h} \leftarrow \bar{h}_{\text{old}}$;
Step 3b          while $\Gamma > C_\alpha$ do
                     $t_1 \leftarrow \bar{h} + 1$;
                     $\Gamma = \Gamma_{t_1}^{h_{\text{last}}}$;
                 end
                 $h_{\text{last}} \leftarrow t_1$;
Step 3c          if $|h_{\text{last}} - h_{\text{first}}| > d$ then
                     append $h_{\text{first}}, h_{\text{last}}$ to candidates;
                     $h_{\text{first}} = h_{\text{first}} + d$;   $h_{\text{last}} = h_{\text{last}} - d$;
                 else
                     append $\bar{h}_{\text{old}}$ to candidates;
                     break;
                 end
             end
         end
         sort candidates;
         $\{x_1, \dots, x_s\} \leftarrow candidates$;
Step 4   repeat
             for $i \in \{1, \dots, s-2\}$ do
                 if $\Gamma_{x_i+1}^{x_{i+2}-1} < C_\alpha$ then
                     remove $x_{i+1}$ from candidates;
                 end
             end
         until convergence;
         remove $1, T$ from candidates;
         $changepoints \leftarrow \{x + 1 : x \in candidates\}$;
```

C. *Kernel Change Detection*

Offline, parametric methods for change point detection often outperform their online, non-parametric competitors, but the flexibility gained by looser model restrictions and the ability to run in a streaming context are at times more valuable. The kernel-based change detection (KCD) algorithm proposed by Desbory et al. is one such online algorithm [2]. Rather than rely on a priori knowledge of the generating distribution for a given series of data, KCD instead leverages support vector machines (SVMs) to construct descriptions of the data in a higher dimensional space defined by some given kernel (for our cases, the radial basis function, or RBF, kernel). One can then use these high-dimensional descriptions to develop a dissimilarity statistic that characterizes the differences in center and spread of two possible regimes in the data. Additionally, SVM's popular kernel trick allows one to generate this dissimilarity statistic in input space rather than feature space.

1) Geometric Interpretation: Desobry et al. provide a compelling geometric interpretation of the KCD based on angles between vectors as a measure of dissimilarity and spread (see Figure 1 for a two-dimensional view). This geometric interpretation exists in feature space and assumes that two one-class SVMs were used to describe data from a possible past regime and a possible future regime respectively. The angle between the vectors normal to the decision planes $\widehat{c_pc_f}$ can then be used as a dissimilarity metric between the two sets of data. To account for within-class spread, the authors then normalize this angle by the sum of the angles between the respective normal vectors a support vector of each regime $\widehat{c_ps_p}$ and $\widehat{c_fs_f}$. The result is the KCD dissimilarity statistic shown in Eq. 1.

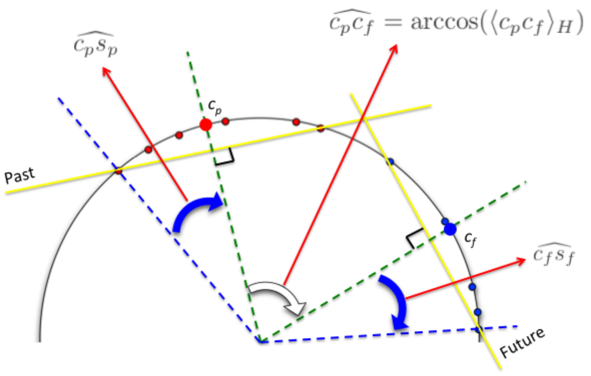

Fig. 1: KCD Geometry in Feature Space

$$KCD_{stat} = \frac{\widehat{c_pc_f}}{\widehat{c_fs_f} + \widehat{c_ps_p}} \tag{1}$$

These angles can be calculated using the inverse cosine of the dot product between the weight vectors, as shown in Eq. 2. One should note the denominator in Eq. 2 is for normalizing the vectors to unit length. This normalized dot product can be calculated in input space using SVM's learned weights $\boldsymbol{\alpha_p}$ and $\boldsymbol{\alpha_f}$ and the kernel matrices K as shown in Eq. 3. To obtain the within-class spread for each class, we use a similar form but rely on the SVM's intercept ρ rather than a dot product, as shown in Eq. 4.

$$\widehat{c_pc_f} = \arccos\left(\frac{\langle w_p, w_f\rangle_F}{\|w_p\|\|w_f\|}\right) \tag{2}$$

$$\frac{\langle w_p, w_f\rangle_F}{\|w_p\|\|w_f\|} = \frac{\boldsymbol{\alpha}_p^T K_{p,f}\boldsymbol{\alpha}_f}{\sqrt{\boldsymbol{\alpha}_p^T K_{p,p}\boldsymbol{\alpha}_p}\sqrt{\boldsymbol{\alpha}_f^T K_{f,f}\boldsymbol{\alpha}_f}} \tag{3}$$

$$\widehat{c_is_i} = \arccos\left(\frac{\rho_i}{\sqrt{\boldsymbol{\alpha}_i^T K_{i,i}\boldsymbol{\alpha}_i}}\right), i \in \{p, f\} \tag{4}$$

2) Algorithm: KCD has four parameters:

- m – Window size, or the number of points on either side of a candidate change point.
- γ – SVM parameter governing bandwidth for the RBF kernel (other kernels are possible, but we did not experiment with them).
- ν – One-class SVM parameter governing proportion of points that should be counted as outliers when training.
- η – Threshold parameter such that, if $KCD_{stat} \geq \eta$ for some time point h, h is considered a change point.

We assume X is the input data of size $n \times k$.

V. Performance Comparison

Graphs

VI. Conclusions

Conclusions

Acknowledgment

This work made use of the Open Science Data Cloud (OSDC) which is an Open Cloud Consortium (OCC)-sponsored project. The OSDC is supported in part by grants from Gordon and Betty Moore Foundation and the National Science Foundation and major contributions from OCC members like the University of Chicago.
Insert PIRE fellow name
was supported by the National Science Foundation Partnerships for Research and Education (PIRE) Award Number 1129076. Any opinions, findings, and conclusions or recommendations expressed are those of the

Function KCD(X, m, γ, ν, η) Algorithm by Desobry et al.[2]

```
$n \leftarrow$ len($X$) ;
changePoints $\leftarrow$ [ ] ;
for $h \in [m, n-m-1]$ do
    $X_p \leftarrow X[h-m, h]$ ;
    $X_f \leftarrow X[h, h+m]$ ;
    svm$_p \leftarrow$ SVM.fit($X_p, \gamma, \nu$) ;
    svm$_f \leftarrow$ SVM.fit($X_f, \gamma, \nu$) ;
    $\boldsymbol{\alpha}_p \leftarrow$ svm$_p.\alpha$ ;
    $\boldsymbol{\alpha}_f \leftarrow$ svm$_f.\alpha$ ;
    $\boldsymbol{\rho}_p \leftarrow$ svm$_p.\rho$ ;
    $\boldsymbol{\rho}_f \leftarrow$ svm$_f.\rho$ ;
    $K_{p,p} \leftarrow$ rbf($X_p, X_p$) ;
    $K_{f,f} \leftarrow$ rbf($X_f, X_f$) ;
    $K_{p,f} \leftarrow$ rbf($X_p, X_f$) ;
    $\widehat{c_p c_f} \leftarrow \arccos\left(\frac{\boldsymbol{\alpha}_p^T K_{p,f} \boldsymbol{\alpha}_f}{\sqrt{\boldsymbol{\alpha}_p^T K_{p,p} \boldsymbol{\alpha}_p}\sqrt{\boldsymbol{\alpha}_f^T K_{f,f} \boldsymbol{\alpha}_f}}\right)$ ;
    $\widehat{c_p s_p} \leftarrow \arccos\left(\frac{|\boldsymbol{\rho}_p|}{\sqrt{\boldsymbol{\alpha}_p^T K_{p,p} \boldsymbol{\alpha}_p}}\right)$ ;
    $\widehat{c_f s_f} \leftarrow \arccos\left(\frac{|\boldsymbol{\rho}_f|}{\sqrt{\boldsymbol{\alpha}_f^T K_{f,f} \boldsymbol{\alpha}_f}}\right)$ ;
    $KCD_{stat} \leftarrow \frac{c_p c_f}{c_p s_p + c_f s_f}$ ;
    if $KCD_{stat} > \eta$ then
        changePoints += $h$
    end
end
return changePoints
```

author(s) and do not necessarily reflect the views of the National Science Foundation.

Make sure at some point in the paper to cite the main Open Science Data Cloud paper [**?**].

References

[1] P. Galeano and D. Peña, "Covariance changes detection in multivariate time series," *Journal of Statistical Planning and Inference*, vol. 137, no. 1, pp. 194–211, Jan. 2007. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0378375805002673

[2] F. Desobry, M. Davy, and C. Doncarli, "An online kernel change detection algorithm," *Signal Processing, IEEE Transactions on*, vol. 53, no. 8, pp. 2961–2974, Aug. 2005.