Unsupervised Change Point Detection in Multivariate Time Series

Daoping Wu[1,2] Suhas Gundimeda[3] Shaoshuai Mou[4] Christopher J. Quinn[1]
[1]Dept. of Computer Science, Iowa State University
[2]Uber Technologies, Inc.
[3]School of Industrial Engineering, Purdue University
[4]School of Aero. and Astronautics, Purdue University

Abstract

We consider the challenging problem of unsupervised change point detection in multivariate time series when the number of change points is unknown. Our method eliminates the user's need for careful parameter tuning, enhancing its practicality and usability. Our approach identifies time series segments with similar empirically estimated distributions, coupled with a novel greedy algorithm guided by the minimum description length principle. We provide theoretical guarantees and, through experiments on synthetic and real-world data, provide empirical evidence for its improved performance in identifying meaningful change points in practical settings.

1 INTRODUCTION

There are many domains where researchers model non-stationary time series, such as neuroscience (Bassett and Sporns, 2017), climatology (Mudelsee, 2013), and economics (Granger and Newbold, 2014). Despite its prevalence, modeling non-stationary time series is generally challenging. One approach to model non-stationary time series is to split up the time series into multiple parts and fit a stationary model for each part. While the *local* problem of fitting each part with a stationary model may be simple, the *global* problem of determining where the change points should be in the first place makes the problem challenging.

In addition to being directly used for modeling non-stationary systems, change point detection is also used as a sub-routine in various applications, like fraud detection (Murad and Pinkas, 1999), quality control (Bissell, 1969), climate monitoring (Beaulieu et al., 2012), seizure detection (Schröder and Ombao, 2019) and motion transition detection (Liu and Chan, 2017).

While there have been many advances, there are a number of challenges that have limited universal effectiveness of many methods: 1. Existing methods usually require extensive parameter tuning (Aminikhanghahi and Cook, 2017; Truong et al., 2020). Two standard parameters are the threshold for the hypothesis test and the number of change points. However, tuning the parameters to an appropriate value is difficult. 2. Distributional changes could happen in the mean, variance, or a combination of several statistics. Some methods only focus on one specific change. A technique that can comprehensively detect various statistical changes is in need. 3. Change detection methods generally require additional statistical assumptions. For example, the Bayesian approach requires the knowledge of prior distribution (Adams and Mackay, 2007). The Hidden Markov Model method requires revisiting the same state multiple times. Another common assumption is that data in each stationary subsequence is independent and identically distributed (Aminikhanghahi and Cook, 2017). The above settings can be unrealistic for many real-world systems. 4. Few methods have been designed for multivariate time series. Existing works usually apply a univariate method to multivariate time series by aggregating data or statistics (Bardwell et al., 2019), which can miss changes in inter-dependencies that do not result in changes in the marginal distributions. 5. Some methods have high computational complexity.

Our Contribution: In this paper, we propose a novel multivariate change point detection method to address the above challenges.

Our method contains two stages. In the first stage, we cluster parametric distributions fit to data from sliding windows. For the second stage, we estimate

Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238. Copyright 2024 by the author(s).

distributions from the clusters and propose a greedy algorithm to keep a subset of the clusters and then find change points between the remaining ones. Our main contribution lies in the second stage. Our procedures use the minimum description length (MDL) principle to guide model selection. The procedure can detect changes in multivariate time series and, in terms of computation tractability, can scale to large time series.

Our proposed method addresses the challenges mentioned above. Our method has the following desirable properties: **(A) Less tuning:** Our method does not need to tune a threshold or require the number of change points to be known. While there are hyper-parameters, we demonstrate that they are much less sensitive to performance than the hyper-parameters other methods require **(B) Flexible:** Our method can be specialized to different settings. The data could be time-dependent, continuous or discrete-valued, not necessarily from a specific parametric family or have simplifying properties such as linearity. **(C) Multi-variate:** Our method is designed for the multi-variate setting. It can detect higher-order changes in the distribution that do not result in marginal changes (i.e. changes that univariate methods cannot detect). **(D) Scalable:** Our algorithm scales to long time series.

1.1 Related Work

We briefly review some key related works.

Among the few multivariate change point detection methods, SBS (Cho and Fryzlewicz, 2015), DCBS (Cho, 2016), E-Divisive, and E-Agglomerative (Matteson and James, 2014) are modifications of binary segmentation. BOCPDMS (Knoblauch and Damoulas, 2018) is the multivariate version of BOCPD. In addition to the traditional methods and their extensions, recent methods include (Davis et al., 2006), which uses a genetic algorithm, BNB (Hooi and Faloutsos, 2019), which applies random partitions, and (Wu et al., 2020), which utilizes clustering sliding windows and solves an integer linear program. However, it's important to note that the mentioned method is specifically designed for univariate time series.

1.2 Background

We now briefly review the minimum description length (MDL) principle. For more details, see (Grünwald, 2007). Conceptually, the MDL principle postulates that the best model for explaining a set of data is the model that can compress the data the most.

We will use a "two-stage" MDL encoding. Given a data set D and model class $\mathcal{M}$, we can measure the *coding length* $\mathrm{CL}(D|M)$ (e.g. number of bits) needed to encode the data D using the model $M \in \mathcal{M}$. The coding length is commonly measured using negative log likelihood. We can also measure the coding length $\mathrm{CL}(M)$ needed to encode the model M itself. See Ch. I.5 in (Grünwald, 2007) for details.

The overall coding length $\mathrm{CL}(D,M)$ is the sum of the two parts, $\mathrm{CL}(D,M) = \mathrm{CL}(D|M) + \mathrm{CL}(M)$. The best model $M^* \in \mathcal{M}$ is the one that minimizes

$$M^* = \underset{M \in \mathcal{M}}{\arg\min}\ \mathrm{CL}(D|M) + \mathrm{CL}(M). \tag{1}$$

2 PROBLEM FORMULATION

We next describe properties of piece-wise stationary time series and then state our optimization problem.

Let $Y_{1:T} := \{Y_1, Y_2, \cdots, Y_T\}$ denote a discrete-time, d-dimensional Markov order-p time series with time length T. Let $\mathcal{Y}$ denote the alphabet. Let $P_{Y_{1:T}|Y_{1-p:0}}(\cdot|\cdot)$ denote the joint distribution (density for continuous $\mathcal{Y}$) of the time series conditioned on length p history $Y_{1-p:0}$, with similar notation for marginal distributions. By the chain rule and the Markov property, for any realization $y_{1:T} \in \mathcal{Y}^T$ and any history $y_{1-p:0} \in \mathcal{Y}^p$, the joint distribution (conditioned on history $y_{1-p:0}$) can be factorized over time

$$P_{Y_{1:T}|Y_{1-p:0}}(y_{1:T}|y_{1-p:0}) = \textstyle\prod_{t=1}^{T} P_{Y_t|Y_{t-p:t-1}}(y_t|y_{t-p:t-1}). \tag{2}$$

Suppose that the time series' joint distribution is piece-wise stationary and parameterized by vectors in a parameter space Φ. Let τ_i for $i = 1, 2, \cdots, k$ denote the *change points* (i.e. the times when the conditional distribution changes). Let $\phi_i \in \Phi$ denote the parameter vector for $t \in (\tau_{i-1}, \ldots, \tau_i - 1)$. The change point τ_i signifies not only that $\phi_i \neq \phi_{i+1}$ (a necessary condition) but also that $D_{KL}(P_{\phi_i}||P_{\phi_{i+1}}) > 0$. (In exponential families, for instance, if the sufficient statistics are linearly dependent, different parameter vectors could induce the same distribution (see Section 1.5 of (Lehmann and Casella, 1998).)) For simplicity, we also denote the endpoints using the τ notation, with $\tau_0 = 1$ and $\tau_{k+1} = T + 1$. Since between the change points τ_i and τ_{i+1}, the conditional distribution, parameterized by ϕ_i, does not change, we can simplify (2),

$$P_{Y_{1:T}|Y_{-p:0}}(y_{1:T}|y_{-p:0}) = \prod_{i=1}^{k+1} \prod_{t=\tau_{i-1}}^{\tau_i - 1} P_{\phi_i}(y_t|y_{t-p:t-1}).$$

We refer to the period of time from one change point up to, but not including, the next change point as an *epoch*. We denote the lists of changepoints and parameter vectors as $\boldsymbol{\tau} = (\tau_1, \cdots, \tau_k)$ and $\boldsymbol{\phi} = (\phi_1, \cdots, \phi_{k+1})$, respectively. We allow repeated models in non-adjacent epochs, e.g. $\phi_2 = \phi_5$. We denote the number and set of *distinct* parameter vectors as $\widetilde{k}$ and $\widetilde{\boldsymbol{\phi}} = \{\widetilde{\phi}_1, \cdots, \widetilde{\phi}_{\widetilde{k}}\}$ respectively.

Let $\mathcal{M} \subseteq \Phi^T$ denote a class of piece-wise stationary models with Markov order p and parameter space Φ. Let $M(k, \boldsymbol{\tau}, \boldsymbol{\phi}) \in \mathcal{M}$ denote a piece-wise stationary model with the change point counts, the change point locations, and parameter vectors explicitly enumerated. Figure 1 (a) depicts a realization of a piece-wise stationary linear VAR model of length $T = 300$, with $\phi_i = (A_i, \mu_i, \Sigma_i)$ for $i = 1, 2, 3$ composed of the coefficient matrix, noise mean vector, and noise covariance matrix, respectively.

We can consider the following problem outlining our general goal of fitting a piece-wise stationary model to the data.

Problem 1. *Given a time series realization $y_{1-p:T} \in \mathcal{Y}^{T+p}$, a model class $\mathcal{M}$, and a loss function $\mathcal{L}$, solve*

$$M^*(k, \boldsymbol{\tau}, \boldsymbol{\phi}) = \underset{M(k,\boldsymbol{\tau},\boldsymbol{\phi})\in\mathcal{M}}{\arg\min} \sum_{i=1}^{k} \mathcal{L}(y_{\tau_{i-1}:\tau_i-1}|\phi_i). \quad (3)$$

Problem 1 can be challenging to solve. We first simplify the problem by estimating single change points, i.e., $k = 1$. We will next show how we extend it to solve the problem when there are multiple unknown numbers of change points.

3 ESTIMATING SINGLE CHANGE POINT

In this section, we discuss estimating single change point when the time series only contains two epochs. Using the log-likelihood as loss function, existing methods solve the following problem.

$$\underset{\tau:\ 1<\tau<T}{\arg\max} \sum_{t=1}^{\tau} \log \frac{P_{\hat{\phi}_1}(Y_t|Y_{t-p:t-1})}{P_{\hat{\phi}_2}(Y_t|Y_{t-p:t-1})} \quad (4)$$

Subject to:

$$\hat{\phi}_1 = \arg\max_{\phi} \sum_{t=1}^{\tau} \log P_\phi(Y_t|Y_{t-p:t-1})$$

$$\hat{\phi}_2 = \arg\max_{\phi} \sum_{t=\tau}^{T} \log P_\phi(Y_t|Y_{t-p:t-1})$$

The above problem requires solving inner optimization for each τ, which is computational prohibitive. We consider a different problem by presuming that we have good estimates of P_{ϕ_1} and P_{ϕ_2} before we estimate the change point. We then solve the problem using known $P_{\hat{\phi}_1}$ and $P_{\hat{\phi}_2}$.

$$\underset{\tau:\ 1<\tau<T}{\arg\max} \sum_{t=1}^{\tau} \log \frac{P_{\hat{\phi}_1}(Y_t|Y_{t-p:t-1})}{P_{\hat{\phi}_2}(Y_t|Y_{t-p:t-1})}. \quad (5)$$

The following theorem states that if the estimated distribution $P_{\hat{\phi}_i}$ is close to the true distribution P_{ϕ_i} and two true distribution are diverse to a degree, then we can have a good level of accuracy for the estimated change point.

Theorem 1. *((α, β)-accuracy) For true models P_{ϕ_1}, P_{ϕ_2}, suppose the data before and after the change point τ^* are drawn from P_{ϕ_1} and P_{ϕ_2} respectively. The MLE $\hat{\tau}$ using $P_{\hat{\phi}_i}$ satisfies $Pr[|\hat{\tau} - \tau^*| > \alpha] < \beta$ for any $\beta > 0$ and $\alpha = \frac{2A^2}{C^2} \log \frac{32}{3\beta}$, where $A = \max_t \frac{\log P_{\phi_1}(Y_t|Y_{t-p:t-1})}{\log P_{\phi_2}(Y_t|Y_{t-p:t-1})} - \min_{t'} \frac{\log P_{\phi_1}(Y_t|Y_{t-p:t-1})}{\log P_{\phi_2}(Y_t|Y_{t-p:t-1})}$ and*

$$C = \min\{D_{KL}(P_{\phi_1}||P_{\hat{\phi}_2}) - D_{KL}(P_{\phi_1}||P_{\hat{\phi}_1}),$$
$$D_{KL}(P_{\phi_2}||P_{\hat{\phi}_1}) - D_{KL}(P_{\phi_2}||P_{\hat{\phi}_2})\}.$$

Proof. Our proof closely follows Thm 5 in Cummings et al. (2018). The detailed proof is presented in the Appendix. □

This motivates us to consider that if we have access to estimated distributions, as well as the starting and ending time points of two adjacent epochs in a time series, we can utilize this information to estimate the change point between these epochs. Now, we are left with two even more challenging problems.

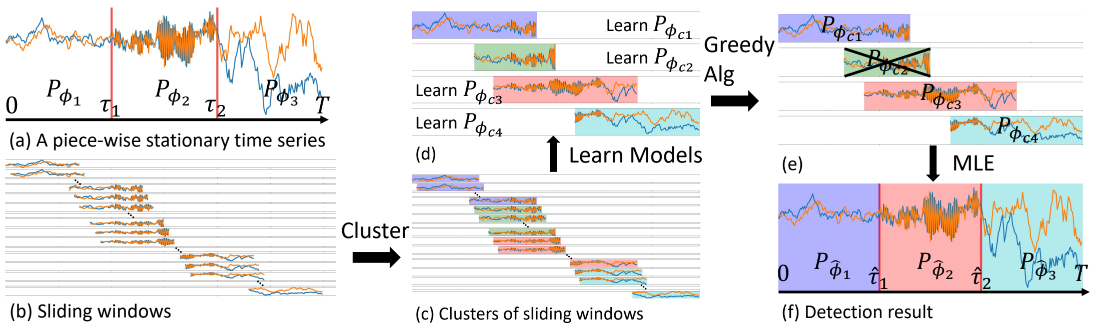
- **How to obtain a good estimation of true distribution from observational time series?**
- **How to solve problem 1 when there are multiple change points and we do not even know how many there are?**

We next propose a novel greedy search method to solve the above issues.

4 CHANGE POINTS GREEDY SEARCH

We next discuss our proposed method. Fig. 1 visually depicts the major steps of our proposed procedure, applied to a realization of a piece-wise stationary linear VAR model. High-level pseudo-code is shown in Algorithm 1.

Our proposed method begins by estimating conditional distributions for various subsequences. See Fig. 1 (b). We estimate the similarity of the estimated conditional distributions. We then cluster the conditional distributions (equivalently, the subsequences), resulting in a (soft) clustering. See Fig. 1 (c). As shown in Fig. 1 (d), we next estimate a single model for each cluster.



P_{ϕ_1} P_{ϕ_2} P_{ϕ_3}
0 τ_1 τ_2 T
(a) A piece-wise stationary time series
(b) Sliding windows
Cluster
(c) Clusters of sliding windows
Learn Models
Learn $P_{\phi_{c1}}$
Learn $P_{\phi_{c2}}$
Learn $P_{\phi_{c3}}$
Learn $P_{\phi_{c4}}$
(d)
Greedy Alg
$P_{\phi_{c1}}$ $P_{\phi_{c2}}$ $P_{\phi_{c3}}$ $P_{\phi_{c4}}$
(e)
MLE
0 $P_{\hat{\phi}_1}$ $\hat{\tau}_1$ $P_{\hat{\phi}_2}$ $\hat{\tau}_2$ $P_{\hat{\phi}_3}$ T
(f) Detection result

Figure 1: Diagram of the main steps of our method. (a) An observed multivariate, piece-wise stationary time series with unknown distribution and change points. (b) Sliding windows of the time series. (c) sliding windows clustered based on the similarity of their estimated distributions. Windows with the same color are in the same cluster. (d) Subsequences extracted from the clusters. Different colors represent different clusters. From each cluster, we infer a distribution. (e) The set of subsequences is greedily pruned. (f) The final model, the returned solution of Problem 1.

Our method is designed using the intuition that the empirical distribution of samples from the same population should be homogeneous to a certain degree. Although the number and locations of the change points are unknown, the conditional distribution $P_{\phi_1}(y_t|y_{t-p:t-1})$ is the same, parameterized by unknown ϕ_1, for the first epoch (from $t = 1$ up to the unknown $t = \tau_1$). Consequently, if we estimate parameter vectors for different subsequences of data for $t \in 1 : \tau_1 - 1$, in expectation, the estimated parameter vectors would be close (since they came from the same generative distribution). It is anticipated that by merging homogenous windows, we can get subsequences close to epochs (1st, 3rd and 4th subsequences in Fig. 1 (d)). More importantly, the above subsequences exhibit a one-to-one mapping to epochs. Hence, obtaining the subsequences helps to reveal partial information about epochs.

However, not all subsequences are desired. Windows might contain heterogeneous data. These similar windows might also be merged to be subsequences (2nd subsequence in Fig. 1 (d)). The existence of such subsequences causes ambiguity. Therefore, the key and challenging issue is how to efficiently select the correct subsequences. Due to that, we propose a greedy algorithm to select the desired subsequences. Our proposed procedure will then decide which clusters to ultimately keep as well as select estimated change points between those that are kept. That step is shown in Fig. 1 (e). After which, we have inferred change points and output a final model.

In the following subsections, we will discuss details of our proposed procedure and later demonstrate its efficacy on simulated and real-world data.

We informally call the subsequences which mainly contain data from the same epoch to be a *pure* subsequence (1st, 3rd and 4th subsequences in Fig. 1 (d)). We call the other subsequences *mixture* subsequences.

Algorithm 1: Main algorithm

```
Input  : Time series $Y_{1:T}$, Markov order $p$,
         Parameter space $\Phi$, oracle $\mathcal{A}$
Output: Piece-wise stationary model $M(k, \boldsymbol{\tau}, \boldsymbol{\phi})$
Estimate models from sliding windows
Cluster windows by model similarity
Estimate a model for each cluster
Apply Most-likely Mixture Alg to select models
  and subsequences
Estimate $\boldsymbol{\tau}$ and $k$ using the selected models and
  subsequences
Refit model to estimate $\boldsymbol{\phi}$ using estimated change
  points $\hat{\boldsymbol{\tau}}$
```

We now present how to select pure subsequences.

4.1 Identifying Pure Subsequences

Our selection procedure is guided by the MDL principle. Recall that after processing the clustering result, we have a list of ordered subsequences $S = (s_1, s_2, \cdots, s_R)$ and a corresponding list of models $\boldsymbol{\phi_s} = (\phi_{s_1}, \phi_{s_2}, \cdots, \phi_{s_R})$. See Fig. 2(b) for an example. For any two adjacent subsequences (which in general overlap), if we estimate that they both are pure and should be kept, then we will need to estimate a change point between them. We will discuss the details of our proposed local search for change points later in Section 3. We initially consider that all the subsequences are pure and find change points between each pair locally. The resulting model is

$$\begin{aligned} \boldsymbol{\tau} &= (\tau_1, \cdots, \tau_{R-1}) \\ \boldsymbol{\phi_s} &= (\phi_{s_1}, \cdots, \phi_{s_R}) \\ M &= M(R-1, \boldsymbol{\tau}, \boldsymbol{\phi_s}). \end{aligned} \tag{6}$$

Now we consider removing a subsequence s_i from S. We would get a modified list of subsequences $S' = S/\{s_i\} = (s_1, \cdots, s_{i-1}, s_{i+1}, \cdots, s_R)$ and a corresponding list of models $\boldsymbol{\phi_{s'}} = (\phi_{s_1}, \cdots, \phi_{s_{i-1}}, \phi_{s_{i+1}}, \cdots, \phi_{s_R})$. This model would have one less change point, $R-2$ total. If we re-estimate change points locally, then only the two changepoints bounding s_i with s_{i-1} and with s_{i+1}, τ_{i-1} and τ_i respectively, would be removed and replaced with a new estimated change point τ' between s_{i-1} and with s_{i+1}. The new model is

$$\begin{aligned} \boldsymbol{\tau}' &= (\tau_1, \cdots, \tau_{i-2}, \tau', \tau_{i+1}, \cdots, \tau_R) \\ \boldsymbol{\phi_{s'}} &= (\phi_{s_1}, \cdots, \phi_{s_{i-1}}, \phi_{s_{i+1}}, \cdots, \phi_{s_R}) \\ M' &= M(R-2, \boldsymbol{\tau}', \boldsymbol{\phi_{s'}}). \end{aligned} \tag{7}$$

The model corresponding to S' will have one less change point and, in some cases, one less parameter vector. We can compare the model (6) for S and the model (7) for S' by their overall coding lengths. If $\mathrm{CL}(Y_{1:T}, M') \leq \mathrm{CL}(Y_{1:T}, M)$, M' is a better model compared with M and then the subsequence s_i should be eliminated from the list of subsequences S. Otherwise, we keep s_i in the list of. Hence, we can define the *score* of a subsequence s_i as the difference of overall coding length,

$$\mathit{Score}(s_i) = \frac{1}{T}(\mathrm{CL}(Y_{1:T}, M) - \mathrm{CL}(Y_{1:T}, M')). \tag{8}$$

If $\mathrm{Score}(s_i) \geq 0$, the subsequence s_i is more likely a mixture subsequence. Otherwise, s_i is more likely a pure subsequence. Note that the score depends on the reference list of subsequences S. We first derive the coding length.

Coding Length: The coding length of time series $Y_{1:T}$ under model $M(k, \boldsymbol{\tau}, \boldsymbol{\phi}) \in \mathcal{M}$ can be approximated by the negative log likelihood (NLL). $\mathrm{CL}(Y_{1:T}|M) \approx -\sum_{i=1}^{k+1} \sum_{t=\tau_{i-1}}^{\tau_i - 1} \log P_{\phi_i}(Y_t|Y_{t-p:t-1})$. See Ch 3.1 in (Rissanen, 1998) for details. The coding length CL of model $M(k, \boldsymbol{\tau}, \boldsymbol{\phi})$ can be decomposed to several parts (see Appendix for more details),

$$\mathrm{CL}(M(k, \boldsymbol{\tau}, \boldsymbol{\phi})) = \mathrm{CL}(k) + \mathrm{CL}(\boldsymbol{\tau}) + \mathrm{CL}(\boldsymbol{\phi}). \tag{9}$$

First, $\mathrm{CL}(Y_{1:T}, M)$, the overall coding length of data and model M, is directly derived from (9). We now consider $\mathrm{CL}(Y_{1:T}, M')$, the overall coding length of data and model M'. The coding length of data under model M' is $\mathrm{CL}(Y_{1:T}|M') \approx \mathrm{NLL}(Y_{1:T}|M')$.

Next, we derive the coding length of model M'. According to the same decomposition as shown in (9), we have

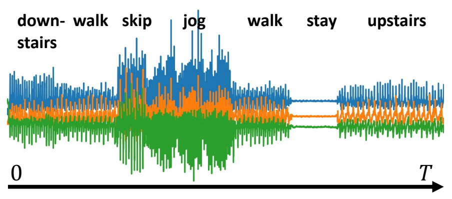
$$\mathrm{CL}(M') = \log(R-2) + (R-2)\log T + \mathrm{CL}(\boldsymbol{\phi}'_{\boldsymbol{s}}). \tag{10}$$

By rearranging the two parts of the overall coding length, we compute the score of the subsequence s_i as $\mathrm{Score}(s_i) = \frac{1}{T}(\mathrm{CL}(Y_{1:T}, M) - \mathrm{CL}(Y_{1:T}, M')) = \frac{1}{T}(\mathrm{CL}(Y_{1:T}|M) - \mathrm{CL}(Y_{1:T}|M') + \mathrm{CL}(M) - \mathrm{CL}(M'))$. See Appendix for formulas. Larger scores indicate a subsequence is more likely to be a mixture. Given a list of subsequences S, the *most-likely mixture subsequence* $s_* \in S$ is the subsequence with the largest score, $s_* = \arg\max_{s \in S} \mathrm{Score}(s)$. In Fig. 2(b), the scores are written next to the subsequences. Only S_2 and S_8 have positive scores and S_8's is larger. Thus, among the subsequences in Fig. 2(b), S_8 is the most-likely mixture subsequence.

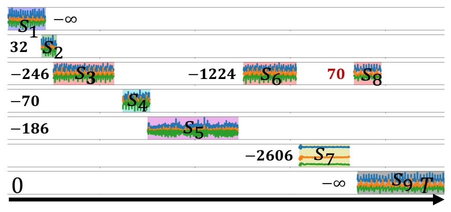
4.2 Most-likely Mixture Algorithm

Having a measure for the marginal gain in coding length by removing an arbitrary subsequence s from S, we propose our Most-likely Mixture algorithm (outlined in Algorithm 2). We first initialize a candidate list of pure subsequences S using all the subsequences $s \in S$ and compute the score for each. In the example shown in Fig. 2, there are 9 subsequences after clustering. Fig. 2(b) shows the initial scores computed for each subsequence. Then, as long the candidate list of subsequences contains a subsequence s likely to be a mixture, i.e., $\mathrm{Score}(s) \geq 0$, we will look for the most-likely mixture subsequence s_* and remove it from the list. In Fig. 2(b), s_2 and s_8 are both likely to be mixture subsequences. s_8 is the most-likely mixture. We then remove s_8 from the list. Every time we delete a subsequence $s_i \in S$, we update the score for the remaining subsequences. Note we only need to modify the score for the deleted subsequence's adjacent subsequences (s_{i-1} and s_{i+1}) and, if there was exactly one other sequence $s_j \in S$ with the same parameter vector as s_i, $\phi_{s_i} = \phi_{s_j}$, then the score of s_j as well. In Fig. 2(c), after we delete s_8, we first update the score for its adjacent subsequence s_7 (no need to update score of s_9 as the score of the last subsequence is always fixed to $-\infty$). We then find there are still two subsequences s_3 and s_6 sharing the same parameter vectors with s_8 in the list. We do not need to modify them. We repeat the above procedure until all scores are negative. In Fig. 2(d), after we delete s_2 and update the score for s_3, all scores are negative. We then end the loop.

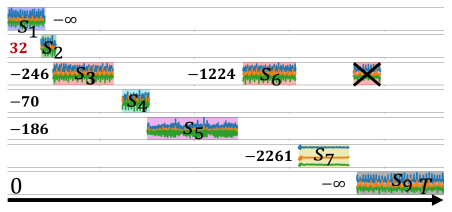
Our next theorem presents the theoretical guarantee for our proposed algorithm, specifically in terms of change point detection consistency analysis. We ap-



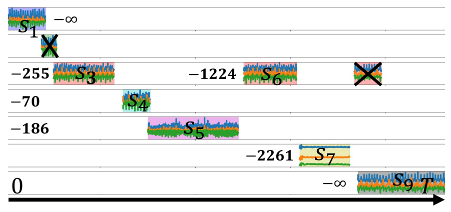
(a) A human motion time series



(b) Step 1: Compute the scores for each subsequence.



(c) Step 2: Delete s_8. Update the score for s_7



(d) Step 3: Delete s_2. Update the score for s_3. Ends the loop as no positive scores

Figure 2: Implementation of Algorithm 2: (b)(c)(d) are the iteration process of Algorithm 2 working on time series shown in (a). Each row in (b)(c)(d) represents a cluster and contains one or multiple subsequences with scores adjacent. The red color indicates the highest score.

Algorithm 2: Most-likely Mixture

```
Input   : The list of subsequences $S$
Output: The list of pure subsequences $S$
Initialize $S = S$ ;
Compute Score($s$) for $s \in S$ ;
while $\exists s \in S,\ s.t.$ Score($s$) $\geq 0$ do
    Find the most-likely mixture subsequence
      $s_* = \arg\max_{s \in S}$ Score($s$);
    Delete $s_*$ from $S$, i.e., $S = S/\{s_*\}$;
    Update Score($s$) for $s \in S$ ;
end
```

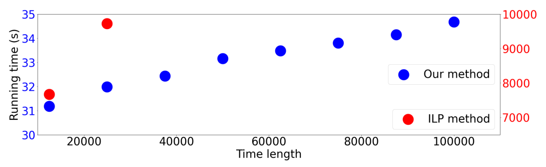


Figure 3: Average running times in seconds of our method (blue; left axis) versus the running time of the ILP method (red; right axis) for different lengths of data (horizontal axis), using a 20 core processor. Window size $W = 50$. The maximum number of windows $m = 2000$.

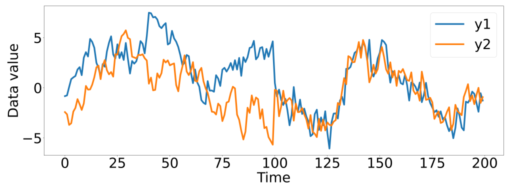
ply the same assumption that has been utilized in prior studies addressing the analysis of change point detection Bai and Perron (2003)Truong et al. (2020). We assume that the time horizon T as well as the epoch length $\frac{\tau_i}{T}$ goes to infinity. Similarly, we allow the window size to go to infinity with time horizon T, i.e., $W = \omega T$, $\omega \in [0, 1]$. Thus, the formed subsequence has infinite length. Under the "infinity" setting, by the consistency of MLE estimator, for each P_{ϕ_i}, there is a $P_{\phi_{s_i}}$ that for $\forall \epsilon > 0$, $D_{KL}(P_{\phi_i} || P_{\phi_{s_i}}) < \epsilon$. This gives us a trivial result that $lim_{T\to\infty} Score(s_i) < 0$. Our theorem discusses the Score for mixture subsequence.

Theorem 2. *(Consistency) Given two adjacent epochs with distributions P_{ϕ_1} and P_{ϕ_2} respectively, with change point to be τ. Consider a mixture subsequence s_2 with starting time point a and ending time point b satisfying $a < \tau < b$. $P_{\phi_{s_2}}$ is learnt from data which follows $\pi_2 P_{\phi_1} + (1 - \pi_2) P_{\phi_2}$, where $\pi_2 \in (0, 1)$. Having adjacent subsequences s_1 and s_3, with $P_{\phi_{s_i}}$ learnt from data which follows $\pi_i P_{\phi_1} + (1 - \pi_i) P_{\phi_2}$, $i = 1, 3$ and $0 \leq \pi_3 < \pi_2 < \pi_1 \leq 1$. Assume that $D_{KL}(P_{\phi_1} || P_{\phi_{s_1}}) < D_{KL}(P_{\phi_1} || P_{\phi_{s_2}})$ and $D_{KL}(P_{\phi_2} || P_{\phi_{s_3}}) < D_{KL}(P_{\phi_2} || P_{\phi_{s_2}})$, we can have*

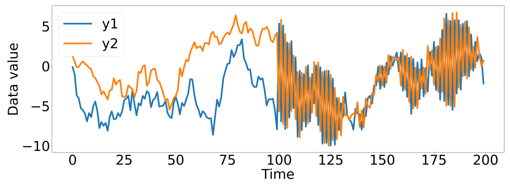
$$\lim_{T\to\infty} Score(s_2) > 0.$$

Proof. The detailed proof is in the Appendix. □

Computational Complexity: Figure 3 is a plot of empirical run-time for our procedure as a function of the time series length T. We see an approximately linear relation between running time and the time length. Details of the experiment are in the Appendix.

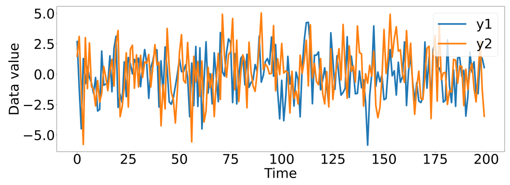


(a) Coefficient change. The statistic difference is 0.475.

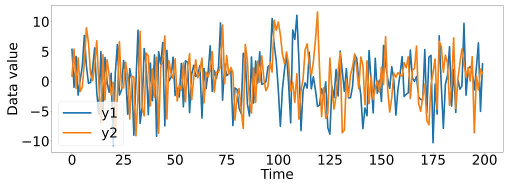


(b) Coefficient change: The statistic difference is 0.95.

Figure 4: Example time series with linear coefficient change.



(a) Covariance change. The statistic difference is 2.



(b) Covariance change: The statistic difference is 4.

Figure 5: Examples of time series with covariance change.

5 EXPERIMENTS

We next demonstrate the performance of our proposed method on artificial and real-world data sets.

State of the Art Baselines: We use the following baseline methods: Sliding windows (Truong et al., 2020), binary segmentation (Truong et al., 2020), dynamic programming (Truong et al., 2020), ILP method (Wu et al., 2020), BNB (Hooi and Faloutsos, 2019), BOCPDMS (Knoblauch and Damoulas, 2018), SBS (Cho and Fryzlewicz, 2015), DCBS (Cho, 2016), E-Divisive (Matteson and James, 2014), and E-Agglomartive (Matteson and James, 2014). Autoplait (Matsubara et al., 2014) and Fluss (Gharghabi et al., 2019).

Evaluation: We evaluate the performance using the precision, recall and F1 score. A true positive (TP) is identified if at least one estimated change point $\hat{\tau}$ is within δ time-point distance of a true change point, i.e, $|\tau - \hat{\tau}| \leq \delta$.

Datasets: We use both artificial and real-world data. See Fig. 4 and Fig. 5 for examples. We use artificial data to test different methods' sensitivity to changes in different parameters in a (linear) vector autoregressive model and in a (non-linear) vector autologistic regressive model. We use the following real-world data sets: Bee dance (Oh et al., 2008), HASC (Ichino et al., 2016), Occupancy (Candanedo and Feldheim, 2016) and MoCap Lab.

Clustering Algorithm and Window Size: In this work, we do not propose a specific clustering procedure. We use HDBSCAN McInnes et al. (2017) for clustering. One desirable property of HDBSCAN for our purposes is that it does not require a pre-specified number of clusters to pick.

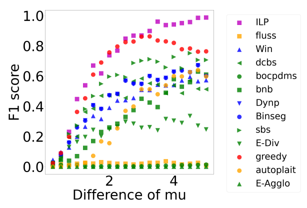
Window size is the hyperparameter of our method. We encourage the domain expert to pick the appropriate window size according to their domain knowledge. In our experiment, we searched for a range of window sizes and chose the one with the minimum MDL score. Details about our method's sensitivity to window size are in the Appendix.

5.1 Results

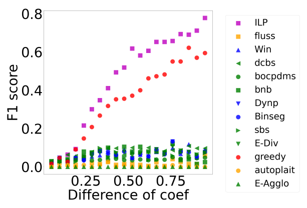
Artificial Data: Fig. 6 shows the F1 scores for the experiments with artificial data. Each sub-plot corresponds to a different type of change between the parameter vectors of the two epochs ϕ_1 and ϕ_2. The y-axis is the F1 score. Higher F1 scores indicate better performance. The x-axis is the difference of the changing statistic.

The performance of our method is depicted with red circles and denoted as 'greedy.' ILP, the univariate method in (Wu et al., 2020), is depicted in magenta squares. The three traditional change point baseline methods are depicted with blue markers. The state-of-the-art change point detection methods are depicted with green markers. Semantic segmentation methods are depicted with orange markers.

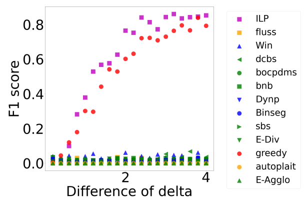
Our method (red circles) achieves performance nearly as good as ILP (magenta squares) in all the experiments. Our method also outperformed all other baselines in all four experiments (Figs. 6(a) to 6(d)). Our



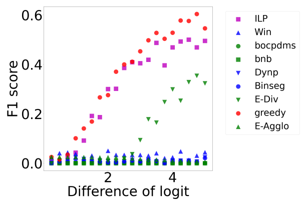
(a) Mean shift



(b) Coefficient change



(c) Covariance change



(d) Logit change

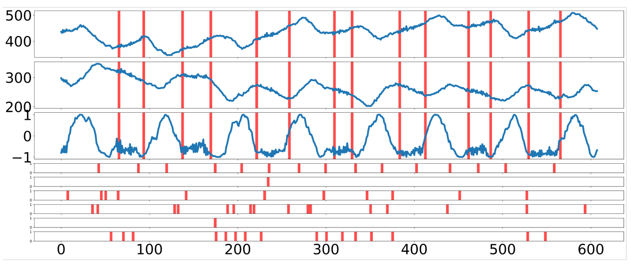
Figure 6: F1 scores for experiments with artificial data. ILP is in magenta. Our method is in red and performs comparably. For two experiments (linear coefficient change (b) and covariance change (c)), only our method and the ILP method were able to identify the change points.

method slightly outperformed ILP when the data is nonlinear (Fig. 6(d)).

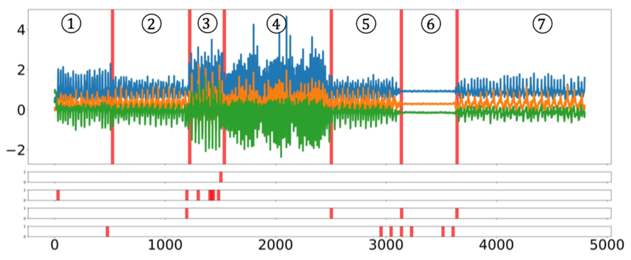
For changes in a linear VAR coefficient (Fig. 6(b)) and covariance (Fig. 6(c)), only our method and ILP were able to reliably detect changes. Fig. 4 shows example time series for linear VAR coefficient (Fig. 6(b)); the change in the time series is visually apparent, though no other baselines successfully detected the change. Fig. 5 shows example time series for linear VAR covariance (Fig. 6(c)); the change in the time series is less apparent visually, yet our method was nonetheless able to reliably detect the change. For changes in coefficients of a logistic regression model, only a single baseline, E-div (green triangles), was able to detect changes some of the trials, but had substantially worse performance than our method.

Real-world Data: In Table 1, we summarize the overall performance of our method and baselines on the real-world data sets. Performance is measured using F1 score for four different margin sizes. The precision and recall are shown in the Appendix in Table 2. We only include BNB, E-Divisive, BOCPDMS, SBS, Autoplait, and Fluss as baselines. Each of the remaining baselines (a) could not handle multiple $k > 1$ unknown change points, (b) had a high run-time even on the smallest real-world data set, or (c) both.

The highest F1 score and any lower scores within 5% of the highest score are in bold. We see our method outperforms all other methods on all of the real-world data sets, even with larger margins δ allowed for declaring a TP.



(a) Bee dance data 6



(b) HASC data

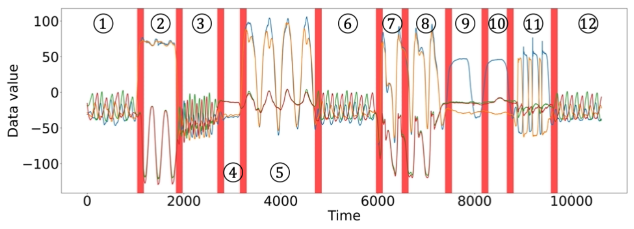
Figure 7: a. Bee dance data 6 (blue) and detected change points (vertical red line) by different methods. The time series in the first, the second, and the third panels correspond to x, y, $sin(\theta)$ in bee dance data. The change points detected by our method are presented in the first three panels. From the fourth panel to the last panel, red vertical lines show the change points detected by methods E-Divisive, sbs, bocpdms, bnb, autoplait, and fluss. b. The first panel shows the 3d time series and the change points detected by Alg. 1. Inferred epochs: 1. downstairs 2. walk 3. skip 4. jog 5. walk 6. stay 7. upstairs. From the second to the last panel, the vertical red lines show the detection result of methods of sbs, bnb, autoplait, and fluss.

To better illustrate the performance, we present examples of detected change points from datasets. The first three panels of Fig. 7(a) show an example of bee dance data and detected change points by Alg. 1. The periods with intensive fluctuations like $t = [137, 169]$ and $[223, 261]$ represent that the bee is waggling. The change points detected by Alg. 1 correctly correspond to the bee's movement transition time points. Fig. 7(b) shows one example of HASC data. We see there are several distinct phases in the first panel. The activities from right to left are going downstairs, walking, skipping, jogging, walking, staying, and going upstairs. Our method (detected change points shown in vertical lines from the first panel) successfully identifies all the transitions.

In Fig. 8, we show examples of MoCap data and detected change points by Alg. 1. Our method successfully detected all the significant motion transitions in

Table 1: Performance. F1 score.(- indicates the running time lasts more than 48 hours)

Datasets	δ	Our Method	BnB	E-Div	BOCPDMS	sbs	Auto-plait	fluss
Bee dance	5	**64.5**	29.2	32.5	**35.6**	6.4	10.6	34.2
	10	**76.8**	45.0	**52.1**	44.0	8.0	15.2	45.8
	15	**78.7**	55.0	**70.1**	49.2	9.6	16.7	45.8
	20	**79.6**	61.7	**79.5**	52.4	12.8	18.2	48.3
HASC	50	**54.5**	10.4	-	-	9.3	**60.5**,	25.9
	100	**70.3**	16.3	-	-	14.8	**71.1**	29.0
	150	**75.3**	23.1	-	-	18.5	**73.7**	31.8
	200	**76.7**	25.7	-	-	19.4	**74.0**	32.8
Occu-pancy	5	**42.9**	37.5	14.3	4.7	**40.0**	0.0	12.5
	10	**71.4**	**62.5**	14.3	4.7	60.0	12.5	12.5
	15	**71.4**	**62.5**	14.3	9.3	60.0	12.5	12.5
	20	**71.4**	**62.5**	14.3	9.3	60.0	12.5	12.5


(a) Subject 86 Trial 2 and detected change points by Alg. 1 (vertical red line).

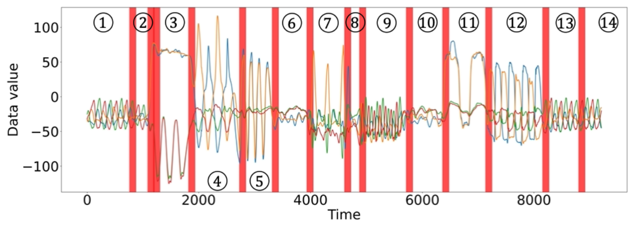

(b) Subject 86 Trial 8 and detected change points by Alg. 1 (vertical red line).

Figure 8: Identified change points (red) for MoCap data. (a) Inferred epochs: 1.Walk 2.Squat 3.Run 4.Stand 5.Stretch arms 6.Walk 7.Jump 8.Jump forward 9.Raise arm 10.Raise arm again 11.Punch 12.Walk. (b) Inferred epochs: 1-2.Walk 3.Squat 4.Twist 5.Roll arms 6.Stand 7.Kick 8.Lean back 9.Run 10.Stand 11.Raise arm 12.Punch 13-14.Walk

subject 86 trial 2 (shown in Fig. 8(a)). In Fig. 8(b), our method detects two close change points at the first motion phase change from walk to squat (phase 1 to 2 in 8(b)). Except for the major changes described in the caption of 8(b), our method also detects two extra change points at $t = 755$ and $t = 8796$. By watching the video, we see the person makes significant right turns at the two time points.

6 CONCLUSION

We propose a novel unsupervised change detection method based on clustering the sliding windows. The proposed method can be used with a wide range of parametric families. Our method is easy to use. It does not require extensive parameter tuning. Our method is scalable and is designed for multivariate time series. We provide theoretical analyses for our method. We conduct extensive experiments to show that it has good performance in detecting meaningful change points in real-world data.

References

R. P. Adams and D. J. C. Mackay. Bayesian Online Changepoint Detection. *Statistics*, 2007.

D. Agudelo-España, S. Gomez-Gonzalez, S. Bauer, B. Schölkopf, and J. Peters. Bayesian Online Prediction of Change Points. volume 124 of *Proceedings of Machine Learning Research*, pages 320–329, Virtual, 2020. PMLR.

R. Alami, O. Maillard, and R. Feraud. Restarted Bayesian online change-point detector achieves optimal detection delay. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 211–221. PMLR, 13–18 Jul 2020.

S. Aminikhanghahi and D. J. Cook. A survey of methods for time series change point detection. *Knowledge and Information Systems*, 2017. ISSN 02193116. doi: 10.1007/s10115-016-0987-z.

U. Appel and A. V. Brandt. Adaptive sequential segmentation of piecewise stationary time series. *Information Sciences*, 1983. ISSN 00200255. doi: 10.1016/0020-0255(83)90008-7.

I. E. Auger and C. E. Lawrence. Algorithms for the optimal identification of segment neighborhoods. *Bulletin of Mathematical Biology*, 1989. ISSN 15229602. doi: 10.1007/BF02458835.

J. Bai and P. Perron. Computation and analysis of multiple structural change models. *Journal of Ap-*

plied Econometrics, 2003. ISSN 08837252. doi: 10.1002/jae.659.

L. Bardwell, P. Fearnhead, I. A. Eckley, S. Smith, and M. Spott. Most Recent Changepoint Detection in Panel Data. *Technometrics*, 2019. ISSN 15372723. doi: 10.1080/00401706.2018.1438926.

D. S. Bassett and O. Sporns. Network neuroscience. *Nature neuroscience*, 20(3):353–364, 2017.

M. Basseville and A. Benveniste. Sequential Detection of Abrupt Changes in Spectral Characteristics of Digital Signals. *IEEE Transactions on Information Theory*, 1983. ISSN 15579654. doi: 10.1109/TIT.1983.1056737.

C. Beaulieu, J. Chen, and J. L. Sarmiento. Change-point analysis as a tool to detect abrupt climate variations. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 370(1962):1228–1249, 2012.

A. Bissell. Cusum techniques for quality control. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 18(1):1–25, 1969.

L. M. Candanedo and V. Feldheim. Accurate occupancy detection of an office room from light, temperature, humidity and co2 measurements using statistical learning models. *Energy and Buildings*, 112: 28–39, 2016.

W.-C. Chang, C.-L. Li, Y. Yang, and B. Póczos. Kernel change-point detection with auxiliary deep generative models. In *International Conference on Learning Representations*, 2019.

H. Cho. Change-point detection in panel data via double CUSUM statistic. *Electronic Journal of Statistics*, 2016. ISSN 19357524. doi: 10.1214/16-EJS1155.

H. Cho and P. Fryzlewicz. Multiple-change-point detection for high dimensional time series via sparsified binary segmentation. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, 2015. ISSN 14679868. doi: 10.1111/rssb.12079.

R. Cummings, S. Krehbiel, Y. Mei, R. Tuo, and W. Zhang. Differentially private change-point detection. *Advances in neural information processing systems*, 31, 2018.

R. Cummings, S. Krehbiel, Y. Lut, and W. Zhang. Privately detecting changes in unknown distributions. In *ICML*, 2020.

R. A. Davis, T. C. Lee, and G. A. Rodriguez-Yam. Structural break estimation for nonstationary time series models. *Journal of the American Statistical Association*, 101(473):223–239, 2006. ISSN 01621459. doi: 10.1198/016214505000000745.

P. Fryzlewicz. Wild binary segmentation for multiple change-point detection. *Annals of Statistics*, 2014. ISSN 00905364. doi: 10.1214/14-AOS1245.

S. Gharghabi, C. C. M. Yeh, Y. Ding, W. Ding, P. Hibbing, S. LaMunion, A. Kaplan, S. E. Crouter, and E. Keogh. Domain agnostic online semantic segmentation for multi-dimensional time series. *Data Mining and Knowledge Discovery*, 2019. ISSN 1573756X. doi: 10.1007/s10618-018-0589-3.

C. W. J. Granger and P. Newbold. *Forecasting Economic Time Series.* Academic Press, 2014.

P. D. Grünwald. *The Minimum Description Length Principle.* MIT press, 2007.

B. Hooi and C. Faloutsos. Branch and border: Partition-based change detection in multivariate time series. In *SIAM International Conference on Data Mining, SDM 2019*, 2019. ISBN 9781611975673. doi: 10.1137/1.9781611975673.57.

H. Ichino, K. Kaji, K. Sakurada, K. Hiroi, and N. Kawaguchi. HASC-PAC2016: Large scale human pedestrian activity corpus and its baseline recognition. In *UbiComp 2016 Adjunct - Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2016. ISBN 9781450344623. doi: 10.1145/2968219.2968277.

E. Keogh, S. Chu, D. Hart, and M. Pazzani. An online algorithm for segmenting time series. In *Proceedings 2001 IEEE international conference on data mining*, pages 289–296. IEEE, 2001.

E. J. Keogh, S. Chu, D. M. Hart, and M. J. Pazzani. Segmenting time series: A survey and novel approach. 2002.

J. Knoblauch and T. Damoulas. Spatio-temporal Bayesian on-line changepoint detection with model selection. In *35th International Conference on Machine Learning, ICML 2018*, 2018. ISBN 9781510867963.

C. G. Lab. Cmu graphics lab motion capture database. URL `http://mocap.cs.cmu.edu/`.

E. L. Lehmann and G. Casella. *Theory of point estimation.* Springer, New York, NY, 2 edition, 1998.

S. Li, Y. Xie, H. Dai, and L. Song. M-statistic for kernel change-point detection. In *Advances in Neural Information Processing Systems*, 2015.

K.-C. Liu and C.-T. Chan. Significant change spotting for periodic human motion segmentation of cleaning tasks using wearable sensors. *Sensors*, 17(1):187, 2017.

Y. Matsubara, Y. Sakurai, and C. Faloutsos. AutoPlait: Automatic mining of co-evolving time sequences. In *Proceedings of the ACM SIGMOD International Conference on Management* *of Data*, 2014. ISBN 9781450323765. doi: 10.1145/2588555.2588556.

D. S. Matteson and N. A. James. A nonparametric approach for multiple change point analysis of multivariate data. *Journal of the American Statistical Association*, 2014. ISSN 1537274X. doi: 10.1080/01621459.2013.849605.

L. McInnes, J. Healy, and S. Astels. hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, 2017. doi: 10.21105/joss.00205.

M. Mudelsee. *Climate time series analysis*, volume 30. Springer, 2013.

U. Murad and G. Pinkas. Unsupervised profiling for identifying superimposed fraud. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 251–261. Springer, 1999.

S. M. Oh, J. M. Rehg, T. Balch, and F. Dellaert. Learning and inferring motion patterns using parametric segmental switching linear dynamic systems. *International Journal of Computer Vision*, 77(1-3), 2008. ISSN 09205691. doi: 10.1007/s11263-007-0062-z.

J. Rissanen. *Stochastic Complexity in Statistical Inquiry*, volume 15. World Scientific, 1998.

A. L. Schröder and H. Ombao. Fresped: Frequency-specific change-point detection in epileptic seizure multi-channel eeg data. *Journal of the American Statistical Association*, 114(525):115–128, 2019.

C. Truong, L. Oudre, and N. Vayatis. Selective review of offline change point detection methods. *Signal Processing*, 167:107299, 2020. ISSN 0165-1684. doi: https://doi.org/10.1016/j.sigpro.2019.107299.

E. S. Venkatraman. *Consistency Results in Multiple Change-Point Problems*. Stanford University, 1992.

L. Vostrikova. Detecting "Disorder" in Multidimensional Random Processes. *Soviet Mathematics Doklady*, 24:55–59, 1981.

D. Wu, S. Gundimeda, S. Mou, and C. J. Quinn. Modeling Piece-Wise Stationary Time Series. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2020. ISBN 9781509066315. doi: 10.1109/ICASSP40776.2020.9053470.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [No] The real world data is publicly available. Details on the implementation and the artificial data are discussed in the supplemental material.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]

(b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]

(c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

7 EXAMPLE OF PIECE-WISE VAR MODEL

Example 1. *Consider the following piece-wise linear VAR time series model. Let I_2 denote the 2×2 identity matrix. Let $A_1 = A_3 = 0.95 I_2$ $A_2 = \begin{bmatrix} 0 & 0.95 \\ 0.95 & 0 \end{bmatrix}$, $\Sigma_1 = I_2$, $\Sigma_2 = 4I_2$, $\Sigma_3 = 9I_2$, $\mu_1 = \mu_2 = \mu_3 = \mathbf{0}$. When $t = 1, 2, \cdots, 100$, $Y_t = A_1 Y_{t-1} + \epsilon_t$, where $\epsilon_t \sim \mathcal{N}(\mu_1, \Sigma_1)$. When $t = 101, 102, \cdots, 200$, $Y_t = A_2 Y_{t-1} + \epsilon_t$, where $\epsilon_t \sim \mathcal{N}(\mu_2, \Sigma_2)$. When $t = 201, 202, \cdots, 300$, $Y_t = A_3 Y_{t-1} + \epsilon_t$, where $\epsilon_t \sim \mathcal{N}(\mu_3, \Sigma_3)$. Let $\phi_i = (A_i, \mu_i, \Sigma_i)$ for $i = 1, 2, 3$. Then, $M(2, (100, 200), (\phi_1, \phi_2, \phi_3))$ is a piecewise stationary model with two change points.*

8 CLUSTERING MODELS ESTIMATED FROM SLIDING WINDOWS

As we discussed in Section 4 (see also Fig. 1), we expect that models estimated windows contained within the same epoch will be similar. Instead of defining a hard threshold to use with the symmetric KL divergence to decide which distributions came from the same epoch, we propose to use a clustering algorithm for grouping estimated models, which can account for the *global* variations of similarity scores. In this work, we do not propose a specific clustering procedure. In our experiments, we use HDBSCAN (McInnes et al., 2017). One desirable property of HDBSCAN for our purposes is that it does not require a pre-specified number of clusters to pick. If a clustering algorithm is used that does, then our procedure could be forked, with parallel instances of our procedure running, one for each number of clusters. The final models from those parallel instances could then be compared using overall complexity to select an overall final model.

At first glance, we might hope each cluster corresponds to an epoch in the underlying model. For Example 1's realization in Fig. 1 (a), for instance, we might hope that there are exactly three clusters, one for each of the underlying epochs shown in Fig. 1 (a).

As discussed in Section 4, that does not generally happen. Fig. 1 (c) shows that an extra cluster was found for the time series in Fig. 1 (c). Specifically, the windows colored green were clustered together, but they contain data from epoch 1 and epoch 2. However, there is no extra cluster for windows overlapping epochs 2 and 3; so we cannot simply take every other cluster as matching an underlying epoch. Additionally, if the same conditional distribution is present in multiple non-adjacent epochs in the underlying model, models from windows that came from those non-adjacent epochs might get grouped together. An example of a cluster formed from windows in different epochs is shown in Fig. 2(b), which is from a real-world data set; the third cluster was formed of windows from three disjoint segments.

We next introduce some terminology and notation for clusters that will describe the later steps in our procedure. Let n_C denote the number of clusters. Let C_i denote the set of windows whose estimated models were grouped together in cluster i. Let $\widetilde{C}_i = \cup_{w_j \in C_i} w_j$ denote the set of timestamps covered by at least one window corresponding to cluster i. For example, in Figure 9, the first cluster C_1 contains the first 26 windows, $C_1 = \{\omega_1, \omega_2, \cdots, \omega_{26}\}$. With window size $W = 30$ and stride of 1, $\widetilde{C}_1 = \bigcup_{i=1}^{26} \{i, i+1, \cdots, i+W-1\} = \{1, 2, \cdots, 55\}$.

For each cluster C_i, we fit a model using data from all times $\widetilde{C}_i$, $\phi_{C_i} = \mathcal{A}(\{Y_t | t \in \widetilde{C}_i\})$.

We expect the model ϕ_{C_i} fit using all times associated with cluster C_i will be similar to, but more accurate than, the models $\{\phi_w\}_{w \in C_i}$ fit to the individual windows belonging to cluster C_i. Next, using these models $\{\phi_{C_i}\}_{i=1}^{n}$ fit to whole clusters, we will attempt to estimate which subsets of clusters to keep (as approximate epochs) and jointly identify corresponding change points between them.

9 ESTIMATING CHANGE POINTS FROM THE CLUSTERED MODELS

When the underlying generative model has non-adjacent epochs with the same parameter ϕ (and thus the same conditional distribution during those non-adjacent epochs), our procedure may have clusters formed from windows in those non-adjacent epochs. For example, Fig. 2(b) shows the clustering and detection result of human motion data. Some timestamps from the two walking phases are grouped into the same cluster. Specifically, the third cluster (third row in subfigures (b)-(d)) is formed by subsequences s_3, s_6, and s_8, where s_3 and s_6 corresponding to walking and s_8 was spurious. We use the term *subsequence* to refer to (maximal) subsets of windows within the same cluster whose union forms a sequence. For each cluster C_i, let r_i and $S_i = \{s_{i,1}, s_{i,2}, \cdots, s_{i,r_i}\}$ denote the number of and set of subsequences in the minimum covering of $\widetilde{C}_i$, respectively. Thus, if $\widetilde{C}_i$ is a

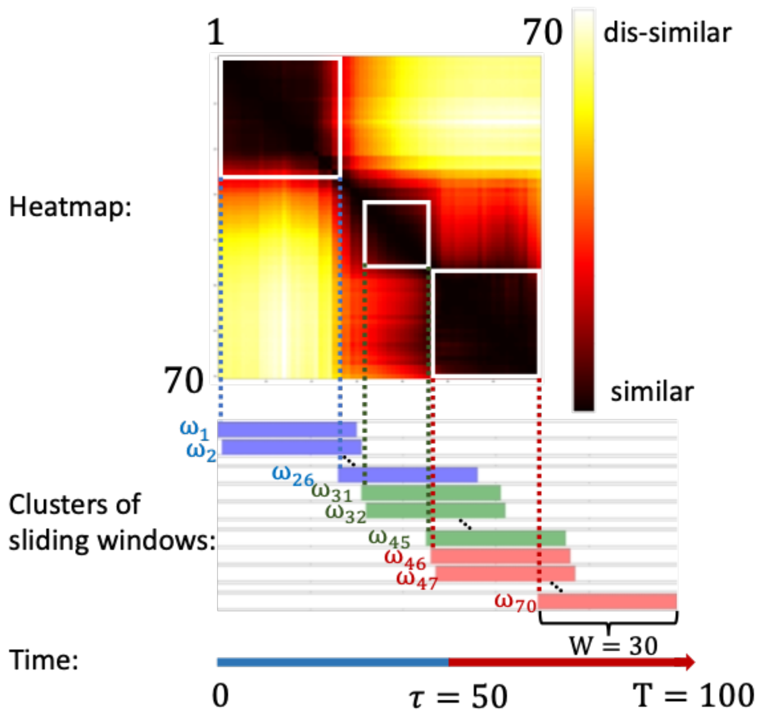


Figure 9: The heatmap of similarity measures between pairs of sliding windows, indexed by start times. Black indicates similarity. There is one true change point at $\tau_1 = 50$.

subsequence, then $r_i = 1$.

In Fig. 9, we show the heatmap of pairwise similarity of sliding windows and the corresponding clustering result. The time series has one change point at $t = 50$. We see that there are three clusters. In this specific example, each cluster only induces one subsequence. Given the ground truth that the change point is at $t = 50$, we can observe two different types of subsequences. Informally, we call a subsequence a *pure* subsequence if data whose timestamps in the subsequence is mainly come from the same epoch. Otherwise, we call the subsequence a *mixture* subsequence. The first subsequence $(1, 2, \cdots, 55)$ is a pure subsequence as the data whose timestamps from the subsequence is mainly from the first epoch $Y_{0:49}$. The third subsequence $(46, 47, \cdots, 100)$ is also a pure subsequence as the data is mainly from the second epoch $Y_{50:100}$. However, the second subsequence $(31, 32, \cdots, 75)$ is a mixture subsequence.

We next enumerate all subsequences by their starting time point. Let R be the total number of all subsequences induced by all clusters, then we denote $S = (s_1, s_2, \cdots, s_R)$ to be the list of all subsequences. The elements are increasingly ordered by their starting time point. A subsequence $s_i \in S$ if and only if there is one j, r such that $s_i = s_{j,r} \in S_j$. Fig. 2(b) shows one example of ordering subsequences. Correspondingly, we denote the list of parameters affiliated to S as $\boldsymbol{\phi_s} = (\phi_{s_1}, \cdots, \phi_{s_R})$. If $s_i \in S_j$, then $\phi_{s_i} = \phi_{c_j}$.

The pure subsequences reveal epochs and we will use them to estimate change points. The challenging part lies in identifying the pure subsequences among all subsequences. In practice, there could be one, multiple, or even no mixture subsequence between two pure subsequences.

10 CODING LENGTH

Encoding an integer k requires $\log_2 k$ bits. As $\tau_i \leq T$, $\text{CL}(\boldsymbol{\tau}) \leq k \log T$. Next, we only need to encode the $\widetilde{k}$ unique parameter vectors $\widetilde{\boldsymbol{\phi}}$, so $\text{CL}(\boldsymbol{\phi}) = \text{CL}(\widetilde{\boldsymbol{\phi}}) = \sum_{i=1}^{\widetilde{k}} \text{CL}(\widetilde{\phi}_i)$. Let n_i denote the number of observations to estimate $\widetilde{\phi}_i$ using maximum likelihood. Each parameter in $\widetilde{\phi}_i$ can be encoded using $\frac{1}{2}\log_2 n_i$ bits (Grünwald, 2007). Let $\|\phi\|$ denote the cardinality (L_0 norm) of a vector ϕ. Then, the coding length $\text{CL}(\boldsymbol{\phi})$ of the parameters is $\text{CL}(\boldsymbol{\phi}) = \sum_{i=1}^{\widetilde{k}} \frac{\|\widetilde{\phi}_i\|}{2} \log_2 n_i$.

11 SCORE COMPUTATION

The first term can be approximated as the following,

$$\begin{aligned}\mathrm{CL}(Y_{1:T}|M) - \mathrm{CL}(Y_{1:T}|M') \approx -\sum_{t=a_i}^{b_i} \log P_{\phi_{s_i}}(Y_t|Y_{t-p:t-1}) + \max_{\tau'} \sum_{t=\tau'}^{b_i} \log P_{\phi_{s_{i+1}}}(Y_t|Y_{t-p:t-1}) \\ + \sum_{t=a_i}^{\tau'} \log P_{\phi_{s_{i-1}}}(Y_t|Y_{t-p:t-1}),\end{aligned} \tag{11}$$

where a_i and b_i are the starting and ending time points of subsequence s_i. Eq. (11) is derived from a local estimation for τ'. In this greedy search, we always treat the first and the last subsequences are pure subsequences. Their scores are therefore fixed to $-\infty$.

Next, using (9) and (10) we get

$$\mathrm{CL}(M) - \mathrm{CL}(M') = \log \frac{R-1}{R-2} + \log T + \left(\frac{\|\phi_{s_i}\|}{2} \log n_i \right) \mathbb{1}_{\phi_{s_i} \notin \phi_{S'}}, \tag{12}$$

where n_i is the number of samples used to estimate ϕ_{s_i} and $\|\phi_{s_i}\|$ denotes the cardinality (L_0 norm) of ϕ_{s_i}. The last term in the sum (12) accounts for the situation where a parameter vector is not used by any other sequence in S' and thus results in further coding length savings.

12 COMPUTATIONAL COMPLEXITY

Our procedure's overall complexity depends on the model class $\mathcal{M}$, the model fitting oracle $\mathcal{A}$ and the clustering algorithm. We empirically measure the run-time incurred as a function of the time series length T. With $m = 2000$ windows and with $\mathcal{M}$ as a linear VAR model class, Fig. 3, we show the linear relation between running time and the time length. Using a 20 core processor, our method can handle a time series of length $T = 100,000$ in under 40 seconds.

13 RELATED WORK

In this section, we briefly review related works on change point detection and works on semantic segmentation.

Distributional-change detection: The study of distributional change point detection can be categorized into two main approaches: searching and hypothesis testing (Truong et al., 2020). Traditional searching methods include binary segmentation (BS) (Vostrikova, 1981; Venkatraman, 1992), sliding windows (SL) (Basseville and Benveniste, 1983; Appel and Brandt, 1983), and dynamic programming (DP) (Auger and Lawrence, 1989). Researchers have extended these methods over time. For example, Fryzlewicz proposed the WBS (Fryzlewicz, 2014), which integrates random sampling with binary segmentation to improve estimation consistency. The well-known Bayesian approach is the BOCPD, which utilizes survival analysis to model the period between two change points (Adams and Mackay, 2007). Recent methods based on BOCPD include (Agudelo-España et al., 2020; Alami et al., 2020). Although some of the aforementioned works are general searching methods, they do not explicitly support multivariate time series.

Semantic segmentation: Semantic segmentation of time series is a research topic related to change point detection. The goal of semantic segmentation is to find meaningful patterns in complex, real-world data (Keogh et al., 2001, 2002; Matsubara et al., 2014; Gharghabi et al., 2019). Change point detection, on the other hand, focuses on identifying changes in the (unknown) generative distribution of observed time series, such as a mean shift or a change in variance (Li et al., 2015; Chang et al., 2019; Cummings et al., 2020). Despite overlaps, some researchers suggest the goals of semantic segmentation and change point detection are distinct (Keogh et al., 2001; Gharghabi et al., 2019). Recently, some methods have proven effective for both problems (Hooi and Faloutsos, 2019).

14 STATE OF THE ART BASELINES

For the experiments on artificial data (short time series), we extended the univariate procedure proposed in (Wu et al., 2020) to the multivariate setting (namely by using the same first stage as our method) and referred to it as 'ILP.' We also include three traditional change point detection methods: sliding windows, binary segmentation, and dynamic programming as baselines. For each of those, we use the implementation in the Python package 'ruptures' (Truong et al., 2020). Unlike our method, the baselines require the number of change points to be given as input. We provide the correct number of change points, $k = 1$, as input. Also, those baselines are designed for univariate ($d = 1$) time series. We apply those univariate methods to the bivariate ($d = 2$) artificial data by detecting one change point in the two univariate time series. For each experiment, if the two detected change points are within two timestamps, we consider them as one detected change point (for the bivariate time series) and keep the one closer to the true change point.

We also use several state-of-the-art multivariate change point detection methods, BNB (Hooi and Faloutsos, 2019), BOCPDMS (Knoblauch and Damoulas, 2018), SBS (Cho and Fryzlewicz, 2015), DCBS (Cho, 2016), E-Divisive (Matteson and James, 2014), and E-Agglomartive (Matteson and James, 2014). For the above methods, we use the authors' implementations. BNB requires the number of change points as input. We set the number of change points from the ground truth for it. BOCPDMS requires a threshold and other parameters to be set. The authors of BOCPDMS applied it to the bee dance data set. We use their default parameters with the bee dance data for real-world data experiments. SBS and DCBS require users to specify the type of changes (mean or second-order structure change) to be detected. In the experiment with artificial data, we manually set the type of changes for them using the ground truth. For the experiments with real-world data, we simply set it to mean change. For other parameters, we use their default setting. For E-Agglomerative, we set segment members to be 100 uniformly spaced segments up to the time length. For E-Divisive, we use the default settings.

Additionally, we compare the performance of our method with two popular semantic segmentation methods, Autoplait (Matsubara et al., 2014) and Fluss (Gharghabi et al., 2019). Recall from our discussion of related work in Section 13 that the goal of semantic segmentation is different. Because they are designed for different purposes, we do not expect that those methods will necessarily perform well in experiments with artificial data when epochs' distributions vary subtly. However, they have been applied to some of the real-world data sets we use, so we include them along with change point detection methods as baselines.

We use the default setting when running Autoplait. We tune two hyper-parameters in Fluss, the number of regimes changes, and the subsequence length (similar to our window size but also acts as a constraint that enforces the minimum distance between adjacent change points). We set the number of regimes change to be 1, and the subsequence length to be 30 for artificial data. For the real-world data, we set the number of regimes to be the number of change points from the ground truth. For bee dance data, as the epoch size is small, we set the subsequence length to be 10. We observe the authors set the subsequence length to be 90 for MoCap data. We use the same value for the other data sets as well.

15 IMPLEMENTATION DETAILS OF OUR PROCEDURE

15.1 Model Class $\mathcal{M}$, Oracle $\mathcal{A}$, and Clustering Algorithm

In our experiments on artificial and real-world data, for simplicity, we mainly used linear vector auto-regressive (VAR) models with Markov order $p = 1$ for the model class $\mathcal{M}$. For one experiment with artificial data, we used a (non-linear) logistic auto-regressive model with Markov order $p = 1$. For those two model classes $\mathcal{M}$, we used model fitting software from the Python libraries statsmodels and SciPy as the oracle $\mathcal{A}$.

15.2 Sliding Windows–Count m and Size W

The number of windows m and window size W are two hyperparameters in our method. While the number of windows m has a default value of $m = T - W + 1$ (all possible sliding windows of size W), to reduce computation, the user can use a smaller value of m. For simplicity, we set $m = 500$.

For the window size W, we do not propose a specific preset value. Many state-of-the-art works require the user to specify the window size (Li et al., 2015) (Gharghabi et al., 2019). We instead propose to automatically

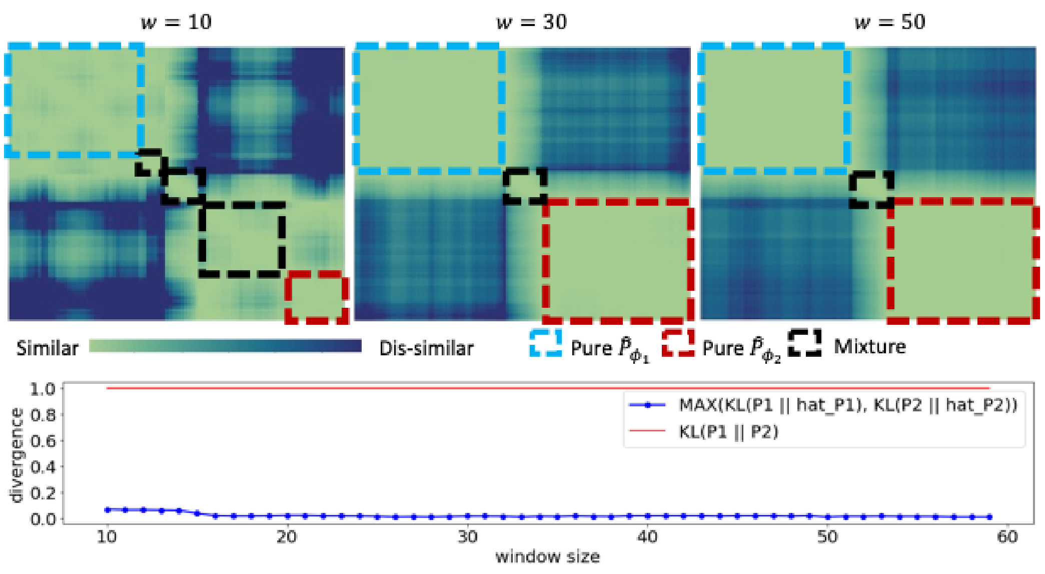


Figure 10: Clustering result for artificial time series with two epochs. The divergence between true distribution is fixed to be 1. The top figure shows the heatmap of window distance for different window size. The bottom figure shows the divergence between true and estimated distribution (blue line).

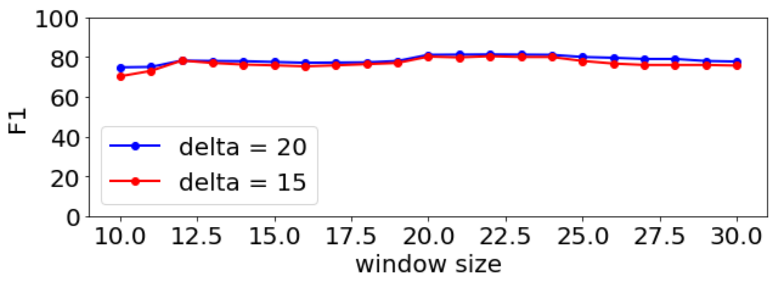


Figure 11: F1 score in different window size for Bee dance data.

select W by independently trying different values and using MDL to guide the selection (namely, whichever W results in the model with the lowest complexity). If W is too small, the parameters may overfit and/or result in distributions whose symmetric KL divergences are spuriously high, even for data from the same epoch. If W is too large, such as larger than some epochs, then there may not be any windows with data exclusively from shorter epochs, which could be missed.

Since what window sizes would work best will depend on the data set and its underlying generative process, we propose to search the window size from a bounded range $[W_{Min}, W_{Max}] = [15, 400]$. Our procedure can be run in parallel (independently) for different candidate W sizes. Among the models output from those parallel instances, the model with the smallest overall complexity is selected as the final model. For the sake of reducing computation, in our experiments on real-world data, when the time series has a time length T longer than 2000, we only searched 8 sizes for W uniformly spaced up to the W_{Max}. For the experiments on artificial data, due to the large number of trials we used (16,000 total), we simply fixed the window size to be $W = 30$.

Fig.10 shows that the estimated distribution is closer to the true distribution when window size grows (epoch length also grows). For real-world data, Fig. 11 shows that our method is robust to the variation in the window size.

15.3 Computer System

Our experiments are conducted on Dell compute nodes with two 12-core Intel Xeon Gold “Sky Lake” processors (24 cores per node) and 96 GB of memory.

16 EVALUATION

To avoid double counts, each estimated change point and each true change point is counted in at most one true positive. For artificial data, where we know the ground truth change points, we can control δ to satisfy that relation. The F1 score is defined as the harmonic mean of precision and recall, where *precision* $= \frac{\#\ \mathrm{TP}}{\|\hat{\boldsymbol{\tau}}\|}$ and *recall* $= \frac{\#\ \mathrm{TP}}{\|\boldsymbol{\tau}\|}$ and $\|\cdot\|$ denotes cardinality (L_0 norm) of a given vector.

17 ARTIFICIAL DATA – DETAILS

For each artificial data set, we randomly generated ϕ_1, the parameter vector for epoch 1. We then generated ϕ_2 so that P_{ϕ_2} was similar to P_{ϕ_1} except for one statistic. For the space Φ of stationary linear VAR models (Markov order $p = 1$), we varied (1) a noise mean value and (2) a coefficient. We also (3) varied coefficient and noise variances, so the joint distribution changed, but the marginal distributions did not. For the space Φ of stationary logistic VAR models (Markov order $p = 1$), we varied (4) a coefficient. For each of those changes, we generated times series that varied in the magnitude of change $\|\phi_1 - \phi_2\|_1$. The magnitude of the changes was among 20 uniformly spaced values. For each magnitude, we generated 200 time series. (Thus, 4000 time series for each of the four types of changes described above).

Fig. 6 shows that only our method and ILP were successful for anything other than a mean shift (with E-div better than others for logit change, though ours significantly better); Fig. 3 shows that our method is much faster than ILP.

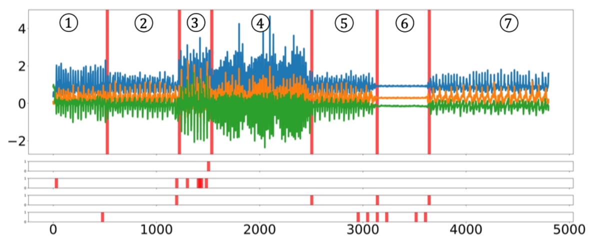
To visualize the distributional changes, we plot example time series for (2) linear coefficient change in Fig. 4 and (4) covariance change in Fig. 5. In Fig. 4, the difference between epoch 1 and epoch 2 is visually apparent. In Fig. 5, the difference between the two epochs is visually less apparent.

18 REAL-WORLD DATA SETS

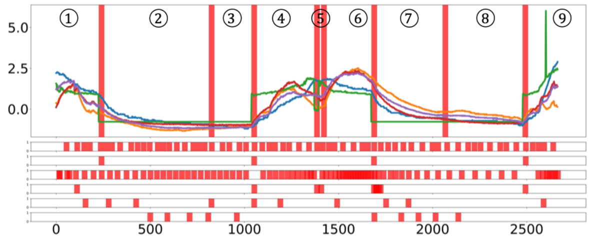
- Bee dance (Oh et al., 2008): The time series corresponds to bee movement trajectories captured by the camera. We used six time series, each with dimension $d = 3$ and with an average time length $T \approx 600$. Fig. 7(a) shows an example time series. Each timestamp is labeled with one of the states: turn right, turn left, and waggling. The change points are timestamps when the state transition occurs.
- HASC (Ichino et al., 2016): Human motion acceleration data. We used 50 time series (the first 50), each with dimension $d = 3$ and an average time length $T \approx 9000$. Fig. 7(b) shows an example time series. The timestamps of different phases of motion, like walking, running, etc., are recorded. The change points are when the motion changes.
- Occupancy (Candanedo and Feldheim, 2016): The data contains information about temperature, humidity, light, CO2, and humidity ratio in a room. The dataset contains one time series with dimension $d = 5$ and time length $T = 2665$. Fig. 12(b) shows the time series. Each timestamp is labeled either as being occupied or empty. We ignore the anomalous phases with less than ten timestamps and consider the true change points are at $t = 195$, 1044, 1371, 1400, 1674, and 2479.
- MoCap: Human motion sensor data[1]. We used two time series, each with dimension $d = 4$ and an average time length $T \approx 11,000$. Fig. 8 shows the two time series. The dataset does not have labeled ground truth. Instead, it provides recorded videos of the participant’s motions. We examine the video to check whether the change points identified by Alg. 1 is meaningful or not.

We do not preprocess the above data if there is no explicit statement from methods of the states of arts or the publisher of the data. For example, we do not subsample, rescale or normalize the data.

[1]http://mocap.cs.cmu.edu/



(a) The first panel shows the 3d time series and the change points detected by Alg. 1. Inferred epochs: 1. downstairs 2. walk 3. skip 4. jog 5. walk 6. stay 7. upstairs. From the second to the last panel, the vertical red lines show the detection result of methods of sbs, bnb, autoplait, and fluss.



(b) The first panel shows the 5d time series and the change points detected by Alg. 1. Inferred epochs: 1. occupied 2-3. empty 4. occupied 5. empty 6. occupied 7-8. empty 9. occupied. From the second to the last panel, the vertical red lines show the detection result of methods of E-divisive, sbs, bocpdms, bnb, autoplait, and fluss.

Figure 12: Detected change points (vertical red line) in HASC data (a) and Occupancy data (b)

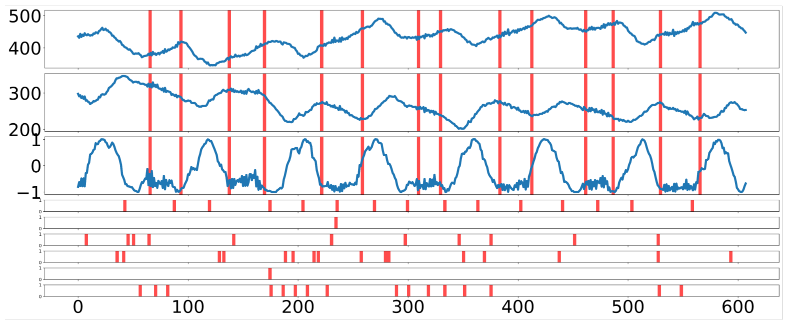


Figure 13: Bee dance data 6 (blue) and detected change points (vertical red line) by different methods. The time series in the first, the second, and the third panels correspond to x, y, $sin(\theta)$ in bee dance data. The change points detected by our method are presented in the first three panels (i.e. one set of change points, duplicated for each time series for visualization). From the fourth panel to the last panel, red vertical lines show the change points detected by methods E-Divisive, sbs, bocpdms, bnb, autoplait, and fluss.

19 PRECISION AND RECALL

In Table 2, we provide the precision and recall used to derive the F1 score in Table 1.

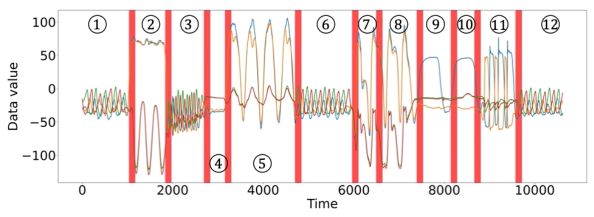
20 ABLATION STUDY

Without using the search algorithm, we can also estimate change points from every pair of clusters directly. Table 3 shows our greedy method is effective and necessary to estimate change points.

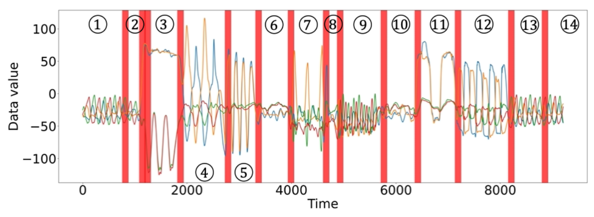
21 SENSITIVITY ANALYSIS

We now test the sensitivity of our method. We perform common preprocessing techniques to the data, including adding Gaussian noises, re-scaling, and Z-normalization. Table 4 shows our method is invariant to the above operations.

We also test the sensitivity of our method on hyperparameters and clustering algorithms. Fig. 10 and Fig. 11 show our method is robust to the variation of window size. Fig. 15 shows our method is robust to the variation of the clustering algorithm.



(a) Subject 86 Trial 2 and detected change points by Alg. 1 (vertical red line). Inferred epochs: 1. Walk 2. Squat 3. Run 4. Stand 5. Stretch arms 6. Walk 7. Jump in place 8. Jump forward 9. Raise arm 10. Raise arm again 11. Punch 12. Walk



(b) Subject 86 Trial 8 and detected change points by Alg. 1 (vertical red line). Inferred epochs: 1-2. walk 3. Squat 4. Twist 5. Roll arms 6. Stand 7. Kick 8. Lean back 9. Run 10. Stand 11. Raise arm 12. Punch 13-14. Walk

Figure 14: Identified change points (red) for MoCap time series.

Table 2: Performance. F1 score, (precision, recall).(- indicates the running time lasts more than 48 hours)

Datasets	δ	Our Method	BnB	E-Div	BOCPDMS	sbs	Auto-plait	fluss
Bee dance	5	**64.5**, (72.3, 58.1)	29.2, (28.5, 29.9)	32.5, (32.5, 32.5)	**35.6**, (45.9, 29.1)	6.4, (50.0, 3.4)	10.6, (46.7, 6.0)	34.2, (33.3, 35.0)
	10	**76.8**, (86.2, 69.2)	45.0, (43.9, 46.2)	**52.1**, (52.1, 52.1)	44.0, (66.8, 35.9)	8.0, (62.5, 4.3)	15.2, (66.7, 8.5)	45.8, (44.7, 47.0)
	15	**78.7**, (88.3, 70.9)	55.0, (53.7, 56.4)	**70.1**, (70.1, 70.1)	49.2, (63.5, 40.2)	9.6, (75.0, 5.1)	16.7, (73.3, 9.4)	45.8, (44.7, 47.0)
	20	**79.6**, (89.4, 71.8)	61.7, (60.2, 63.2)	**79.5**, (79.5, 79.5)	52.4, (67.6, 42.7)	12.8, (100, 6.8)	18.2, (80.0, 10.3)	48.3, (47.2, 49.6)
HASC	50	**54.5**, (48.7, 61.8)	10.4, (9.8, 11.1)	-	-	9.3, (31.7, 5.4)	**60.5**, (77.0, 49.9)	25.9, (24.4, 27.6)
	100	**70.3**, (62.8, 79.7)	16.3, (15.3, 17.3)	-	-	14.8, (50.8, 8.7)	**71.1**, (90.4, 58.5)	29.0, (27.3, 30.9)
	150	**75.3**, (67.3, 85.4)	23.1, (21.8, 24.7)	-	-	18.5, (63.5, 10.8)	**73.7**, (93.7, 60.7)	31.8, (30.0, 33.9)
	200	**76.7**, (68.6, 87.0)	25.7, (24.2, 27.4)	-	-	19.4, (66.7, 11.4)	**74.0**, (94.1, 61.0)	32.8, (30.9, 35.0)
Occu-pancy	5	**42.9**, (37.5, 50.0)	37.5, (30.0, 50.0)	14.3, (7.8, 83.3)	4.7, (2.5, 33.3)	**40.0**, (50.0, 33.3)	0.0, (0.0, 0.0)	12.5, (10.0, 16.7)
	10	**71.4**, (62.5,83.3)	**62.5**, (50.0, 83.3)	14.3, (7.8, 83.3)	4.7, (2.5, 33.3)	60.0, (75.0, 50.0)	12.5, (10, 16.7)	12.5, (10.0, 16.7)
	15	**71.4**, (62.5, 83.3)	**62.5**, (50.0, 83.3)	14.3, (7.8, 83.3)	9.3, (5.0, 66.7)	60.0, (75.0, 50.0)	12.5, (10, 16.7)	12.5, (10.0, 16.7)
	20	**71.4**, (62.5,83.3)	**62.5**, (50.0, 83.3)	14.3, (7.8, 83.3)	9.3, (5.0, 66.7)	60.0, (75.0, 50.0)	12.5, (10, 16.7)	12.5, (10.0, 16.7)

Table 3: Ablation study

Datasets	Bee Dance				HASC				Occupancy			
δ	5	10	15	20	50	100	150	200	5	10	15	20
W/O greedy	64.5/61.5	76.8/71.3	78.7/71.3	79.6/71.3	54.5/42.7	70.3/50.3	75.3/51.2	76.7/51.5	42.9/20.0	71.4/33.3	71.4/33.3	71.4/33.3

Table 4: Sensitivity analysis

Datasets	Bee Dance	HASC	Occupancy
δ	5	50	5
Noise-$\mathcal{N}(0,1)$	64.5	54.5	42.9
Noise-$\mathcal{N}(0,4)$	64.5	54.5	42.9
Rescaling	64.5	54.5	42.9
Z-normalization	64.5	54.5	42.9

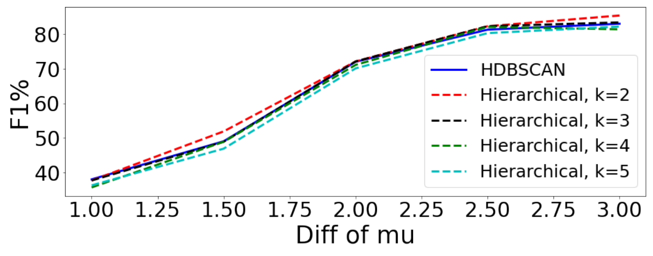


Figure 15: Sensitivity on clustering algorithm using artificial VAR data. The y-axis is the percentage of F1. The x-axis is the difference of mean μ.

22 MISSING PROOFS

22.1 Proof of Theorem 1

22.1.1 Technical Lemmas

We first state a few technical lemmas that will be used in our proof

(Ottaviani's inequality) For independent random variables $U_1, \cdots, U_m$ for $S_k = \sum_{i \in [k]} U_i$ for $k \in [m]$, and for λ_1, $\lambda_2 > 0$, we have

$$Pr[\max_{1 \le k \le m} |S_k| > \lambda_1 + \lambda_2] \le \frac{Pr[|S_m| > \lambda_1]}{1 - \max_{1 \le k \le m} Pr[|S_m - S_k| > \lambda_2]}.$$

Using Hoeffding's inequality, we can have the following Corollary. For independent random variables $U_1, \cdots, U_m$ with mean zero and strictly bounded by an interval length L and for $S_k = \sum_{i \in [k]} U_i$ for $k \in [m]$, and for $\lambda_1 > 0$ and $\lambda_2^2 > \frac{1}{2} m L^2 \log 2$, we have

$$Pr[\max_{1 \le k \le m} |S_k| > \lambda_1 + \lambda_2] \le \frac{2\exp(-2\lambda_1^2/(mL^2))}{1 - 2\exp(-2\lambda_2^2/(mL^2))}.$$

Proof. Since the random variables $\{U_i\}$ are independent and zero mean, the sum S_m is too, $\mathrm{E}[S_m] = 0$. Thus Hoeffding's inequality implies

$$\begin{aligned}\Pr(|S_m| > \lambda_1) &\le \Pr(|S_m| \ge \lambda_1) \\ &= \Pr(|S_m - \mathrm{E}[S_m]| \ge \lambda_1) \\ &\le 2\exp\left(-\frac{2\lambda_1^2}{mL^2}\right). && \text{(Hoeffding's ineq.)}\end{aligned}$$

Likewise, the sum $S_m - S_k = \sum_{i=k+1}^m U_i$ of $m-k$ independent zero mean random variables is also zero mean, $\mathrm{E}[S_m - S_k] = 0$. Thus, Hoeffding's inequality implies

$$\begin{aligned}\Pr(|S_m - S_k| > \lambda_2) &\le \Pr(|S_m - S_k| \ge \lambda_2) \\ &= \Pr(|S_m - S_k - \mathrm{E}[S_m - S_k]| \ge \lambda_2) \\ &\le 2\exp\left(-\frac{2\lambda_2^2}{(m-k)L^2}\right) && \text{(Hoeffding's ineq.)} \\ &< 2\exp\left(-\frac{2\lambda_2^2}{mL^2}\right). && (11)\end{aligned}$$

Since (11) holds for all $1 \le k < m$ (and trivially for $k = m$ in which case $S_m - S_k = 0$),

$$\max_{1 \le k \le m} \Pr(|S_m - S_k| > \lambda_2) < 2\exp\left(-\frac{2\lambda_2^2}{mL^2}\right). \tag{12}$$

Note that the left hand side is a probability, so this bound is only meaningful when

$$\begin{aligned} & 2\exp\left(-\frac{2\lambda_2^2}{mL^2}\right) < 1 \\ \iff \quad & \lambda_2^2 > \frac{1}{2} m L^2 \log 2 && (13)\end{aligned}$$

Combining these results and rearranging e.g. for $a, b \in (0,1)$, $a \le b \Leftrightarrow \frac{1}{1-a} \le \frac{1}{1-b}$).

□

22.1.2 Theorem 1

We first restate Theorem 1 using simplified notation.

Theorem 3. *For true models Q_0, Q_1, data X before change point k^* are drawn from Q_0, data X after change point k^* are drawn from Q_1. If we have estimated model P_0 approximate to Q_0 and estimated model P_1 approximate to Q_1, then the MLE $\hat{k}$ via P_0 and P_1 is (α, β)-accurate for any $\beta > 0$ and*

$$\alpha = \frac{2A^2}{C^2} \log \frac{32}{3\beta} \tag{14}$$

$A = \max_x \frac{\log P_0(x)}{\log P_1(x)} - \min_{x'} \frac{\log P_0(x')}{\log P_1(x')}$ *and*

$C = \min \{D_{KL}(Q_0||P_1) - D_{KL}(Q_0||P_0), D_{KL}(Q_1||P_0) - D_{KL}(Q_1||P_1)\}$.

Proof. We only have incorrect $\hat{k} \neq k^*$ if there exists some k such that the log-likelihood $l(k) > l(k^*)$.

Given some true change-point k^* and error tolerance $\alpha > 0$, we can partition the local interval $[t_1, t_2]$ (i.e. the feasible region for selecting $\hat{k}$) via a partitioning of $\mathbb{R}$ centered at k^* with sub-intervals with exponentially increasing size further out from k^*. We define the partition recursively, as

$$\begin{aligned} R_0 &= [k^* - \alpha,\ k^* + \alpha] \\ R_i &= [k^* - 2^i\alpha,\ k^* + 2^i\alpha] \setminus R_{i-1} \qquad \text{for } i = 1, 2, \dots \end{aligned} \tag{15}$$

Equivalently, for $i \geq 1$ we can identify R_i as the union of two sub-intervals, with R_i^- and R_i^+ denoting the parts left and right of k^* respectively, each of length 2^i,

$$\begin{aligned} R_i^- &= [k^* - 2^i\alpha, k^* - 2^{i-1}\alpha) \\ R_i^+ &= (k^* + 2^{i-1}\alpha, k^* + 2^i\alpha] \\ R_i &= R_i^- \cup R_i^+. \end{aligned}$$

Then we can bound the probability of the bad events as follows. The event $|\hat{k} - k^*| > \alpha$ is equivalent to the event $\hat{k} \in \cup_{i \geq 1} R_i$

$$\begin{aligned} \Pr\left(|\hat{k} - k^*| > \alpha\right) &= \Pr\left(\hat{k} \in \cup_{i\geq 1} R_i\right) \\ &\leq \sum_{i \geq 1} \Pr\left(\hat{k} \in R_i\right) \\ &\leq \sum_{i \geq 1} \Pr\left(\hat{k} \in R_i^+\right) + \Pr\left(\hat{k} \in R_i^-\right) \end{aligned} \tag{16}$$

We next bound $\Pr\left(\hat{k} \in R_i^+\right)$.

For simplicity, we let P_0 denote a fixed model with which we evaluate different candidate change-points, though for each candidate change-point k one could estimate models for each halve (split at k).

We observe that the event

$$\begin{aligned} \{\hat{k} \in R_i^+\} &= \{\max_{k \in R_i^+} \ell(k) = \max_{k \in \cup_{i \geq 0} R_i} \ell(k)\} \\ &\subseteq \{\max_{k \in R_i^+} \ell(k) - \ell(k^*) \geq 0\}, \end{aligned} \tag{17}$$

where (17) includes the larger event that the true change-point not as good as some candidate in R_i^+ (there could be an even better candidate elsewhere).

Now for a candidate $k \in R_i^+$, we expand the difference of log-likelihoods $\ell(k) - \ell(k^*)$ as

$$\begin{aligned}
\ell(k) - \ell(k^*) &= \left[\sum_{t=t_1}^{k-1} \log P_0(x_t) + \sum_{t=k}^{t_2} \log P_1(x_t)\right] \\
&\quad - \left[\sum_{t=t_1}^{k^*-1} \log P_0(x_t) + \sum_{t=k^*}^{t_2} \log P_1(x_t)\right] \\
&= \sum_{t=k^*}^{k-1} \log P_0(x_t) - \sum_{t=k^*}^{k-1} \log P_1(x_t) && (18) \\
&= \sum_{t=k^*}^{k-1} \log P_0(x_t) - \log P_1(x_t) \pm \log Q_1(x_t) && \text{(add and subtract a term)} \\
&= \sum_{t=k^*}^{k-1} \underbrace{-\left(\log \frac{Q_1(x_t)}{P_0(x_t)} - \log \frac{Q_1(x_t)}{P_1(x_t)}\right)}_{\tilde{U}_t} && (19)
\end{aligned}$$

where (18) follows since for this case the candidate $k \in R_i^+$ satisfy $k > k^*$, so data up to k^* are modeled the same and data after k are too, canceling out above. Thus, the only difference to the log-likehoods come from models used on data at times $t \in \{k^*, k^*+1, \dots, k-1\}$, we rearrange terms in (19) and we also introduce notation $\tilde{U}_t$ for the difference in log-likelihoods for any $t \in \{k^*, \dots, \max_{t \in R_i^+} t\}$.

Since data are distributed as $x_t \sim Q_1$ for $t \in \{k^*, \dots, k-1\}$, we have that the expected value of $\tilde{U}_t$ is

$$\begin{aligned}
\mathrm{E}[\tilde{U}_t] &= -\mathrm{E}_{Q_1}\left[\log \frac{Q_1(x_t)}{P_0(x_t)} - \log \frac{Q_1(x_t)}{P_1(x_t)}\right] \\
&= -(D_{KL}(Q_1||P_0) - D_{KL}(Q_1||P_1)). && (20)
\end{aligned}$$

Writing $U_t = \tilde{U}_t - \mathrm{E}[\tilde{U}_t]$, we can express the difference in log-likelihoods as

$$\begin{aligned}
\ell(k) - \ell(k^*) &= \sum_{t=k^*}^{k-1} \tilde{U}_t && \text{(from (19))} \\
&= \left(\sum_{t=k^*}^{k-1} \tilde{U}_t - \mathrm{E}[\tilde{U}_t]\right) - (k-k^*)(D_{KL}(Q_1||P_0) - D_{KL}(Q_1||P_1)) \\
&= \left(\sum_{t=k^*}^{k-1} U_t\right) - (k-k^*)(D_{KL}(Q_1||P_0) - D_{KL}(Q_1||P_1)) && (21)
\end{aligned}$$

We express the event $\ell(k) - \ell(k^*) \geq 0$ with $\{U_t\}$ as

$$\begin{aligned}
& \ell(k) - \ell(k^*) \geq 0 \\
\iff \quad & \sum_{t=k^*}^{k-1} U_t - (k-k^*)(D_{KL}(Q_1||P_0) - D_{KL}(Q_1||P_1)) \geq 0 && (22)
\end{aligned}$$

We can bound the probability of the event in (17),

$$\begin{aligned}
& \Pr\left(\max_{k \in R_i^+} \ell(k) - \ell(k^*) \geq 0\right) \\
&= \Pr\left(\max_{k \in R_i^+} \sum_{t=k^*}^{k-1} U_t - (k-k^*)(D_{KL}(Q_1||P_0) - D_{KL}(Q_1||P_1)) \geq 0\right) && (23)
\end{aligned}$$

Next, recall that $R_i^+ = (k^* + 2^{i-1}\alpha, k^* + 2^i\alpha]$, so for $k \in R_i^+$,

$$2^{i-1}\alpha < k - k^* \leq 2^i\alpha \qquad \Longleftrightarrow \qquad -2^{i-1}\alpha > -(k-k^*) \geq -2^i\alpha.$$

Thus,

$$\begin{aligned}
&\Pr\left(\max_{k\in R_i^+} \ell(k) - \ell(k^*) \geq 0\right) \\
&\qquad = \Pr\left(\max_{k\in R_i^+}\left\{\sum_{t=k^*}^{k-1} U_t - (k-k^*)(D_{KL}(Q_1||P_0) - D_{KL}(Q_1||P_1))\right\} \geq 0\right) \\
&\qquad \leq \Pr\left(\max_{k\in R_i^+}\left\{\sum_{t=k^*}^{k-1} U_t - 2^{i-1}\alpha(D_{KL}(Q_1||P_0) - D_{KL}(Q_1||P_1))\right\} \geq 0\right) \\
&\qquad = \Pr\left(\max_{k\in R_i^+}\left\{\sum_{t=k^*}^{k-1} U_t\right\} \geq 2^{i-1}\alpha(D_{KL}(Q_1||P_0) - D_{KL}(Q_1||P_1))\right)
\end{aligned} \tag{24}$$

With $(D_{KL}(Q_1||P_0) - D_{KL}(Q_1||P_1)) \geq C$, we have

$$\begin{aligned}
&\Pr\left(\max_{k\in R_i^+} \ell(k) - \ell(k^*) \geq 0\right) \\
&\qquad \leq \Pr\left(\max_{k\in R_i^+}\left\{\sum_{t=k^*}^{k-1} U_t\right\} \geq 2^{i-1}\alpha(D_{KL}(Q_1||P_0) - D_{KL}(Q_1||P_1))\right) \\
&\qquad \leq \Pr\left(\max_{k\in R_i^+}\left\{\sum_{t=k^*}^{k-1} U_t\right\} \geq 2^{i-1}\alpha C\right).
\end{aligned}$$

We recognize that for this case ($k \in R_i^+$) since $\tilde{U}_t = \log\frac{P_0(x_t)}{P_1(x_t)}$, $\tilde{U}_t$ is bounded in the interval $[\min_{x\in\mathcal{X}}\log\frac{P_0(x)}{P_1(x)},\ \max_{x\in\mathcal{X}}\log\frac{P_0(x)}{P_1(x)}]$, and U_t is just translated copy of $\tilde{U}_t$ to be mean zero, it is also bounded in an interval of length at most A. Also, R_i^+ is of length $2^i\alpha - 2^{i-1}\alpha = 2^{i-1}\alpha$. Invoking Section 22.1.1 with $\lambda_1 = \lambda_2 = \frac{1}{2}2^{i-1}\alpha C$, we have

$$\begin{aligned}
\Pr\left(\hat{k} \in R_i^+\right) &\leq \Pr\left(\max_{k\in R_i^+} \ell(k) - \ell(k^*) \geq 0\right) \\
&\leq \Pr\left(\max_{k\in R_i^+}\left\{\sum_{t=k^*}^{k-1} U_t\right\} \geq 2^{i-1}\alpha C\right) \\
&\leq \frac{2\exp(-2(\frac{1}{2}2^{i-1}\alpha C)^2/(2^{i-1}\alpha A^2))}{1 - 2\exp(-2(\frac{1}{2}2^{i-1}\alpha C)^2/(2^{i-1}\alpha A^2))} \\
&= \frac{2\exp(-2^{i-2}\alpha C^2/A^2)}{1 - 2\exp(-2^{i-2}\alpha C^2/A^2)}
\end{aligned}$$

provided $\alpha > \frac{A^2\log 2}{2^{i-2}C^2}$

The analysis of $\Pr\left(\hat{k} \in R_i^-\right)$ is similar,

$$\begin{aligned}\ell(k) - \ell(k^*) &= \left[\sum_{t=t_1}^{k-1} \log P_0(x_t) + \sum_{t=k}^{t_2} \log P_1(x_t)\right] \\ &\quad - \left[\sum_{t=t_1}^{k^*-1} \log P_0(x_t) + \sum_{t=k^*}^{t_2} \log P_1(x_t)\right] \\ &= \sum_{t=k}^{k^*-1} \log P_1(x_t) - \sum_{t=k}^{k^*-1} \log P_0(x_t)\end{aligned}$$

so the analogous $\tilde{U}_t$ is bounded in the interval $[\min_{x\in\mathcal{X}} \log \frac{P_1(x)}{P_0(x)},\ \max_{x\in\mathcal{X}} \log \frac{P_1(x)}{P_0(x)}]$ which is of length at most A. Thus, by symmetry we have the same bound on $\Pr\left(\hat{k} \in R_i^-\right)$, so

$$\Pr\left(\hat{k} \in R_i\right) \le \frac{4\exp(-2^{i-2}\alpha C^2/A^2)}{1 - 2\exp(-2^{i-2}\alpha C^2/A^2)}$$

Note that

$$\begin{aligned}\frac{4\exp\left(-2^{i-2}\alpha C^2/A^2\right)}{1 - 2\exp\left(-2^{i-2}\alpha C^2/A^2\right)} \le& 8\exp\left(-2^{i-2}\alpha C^2/A^2\right) &(25)\\ =& 8\left(\exp(\frac{-\alpha C^2}{2A^2})\right)^{2^{i-1}} &(26)\end{aligned}$$

Now we sum terms over all i,

$$\begin{aligned}\sum_{i\ge1} Pr[\max_{k\in R_i} \ell(k) - \ell(k^*) > 0] &\le 8\sum_{i\ge1}\left(\exp(\frac{-\alpha C^2}{2A^2})\right)^{2^{i-1}} &(27)\\ &\le \sum_{i\ge1}\left(\exp(\frac{-\alpha C^2}{2A^2})\right)^{i} &(28)\\ &\le \frac{8\exp\left(-\alpha C^2/A^2\right)}{1-\exp\left(-\alpha C^2/A^2\right)} &(29)\\ &\le \frac{32}{3}\left(\exp(\frac{-\alpha C^2}{2A^2})\right) &(30)\end{aligned}$$

We now complete the proof.

□

22.2 Proof of Theorem 2

Proof. Recall that $Score(s_i) = \frac{1}{T}(\mathrm{CL}(Y_{1:T}|M) - \mathrm{CL}(Y_{1:T}|M') + \mathrm{CL}(M) - \mathrm{CL}(M'))$. As $\mathrm{CL}(M) - \mathrm{CL}(M') = \mathcal{O}(\log(T))$,

$\lim_{T\to\infty} \frac{\mathrm{CL}(M)-\mathrm{CL}(M')}{T} = 0$. We then show $\frac{\mathrm{CL}(Y_{1:T}|M)-\mathrm{CL}(Y_{1:T}|M')}{T}>0$. We first decompose the log-likilhood into two parts

$$
\begin{aligned}
&-\frac{1}{T}\sum_{t=a}^{b}\log P_{\phi_{s_2}(Y_t|Y_{t-p:t-1})}\\
&=-\frac{1}{T}\left(\sum_{t=a}^{\tau}\log P_{\phi_{s_2}(Y_t|Y_{t-p:t-1})}+\sum_{t=\tau}^{b}\log P_{\phi_{s_2}(Y_t|Y_{t-p:t-1})}\right)
\end{aligned}
$$

We next derive the inequality for each part.

$$
\begin{aligned}
&\lim_{T\to\infty} -\frac{1}{T}\sum_{t=a}^{\tau}\log P_{\phi_{s_2}(Y_t|Y_{t-p:t-1})}\\
&\quad=\lim_{T\to\infty}\frac{1}{T}\sum_{t=a}^{\tau}\log\frac{P_{\phi_1(Y_t|Y_{t-p:t-1})}}{P_{\phi_{s_2}(Y_t|Y_{t-p:t-1})}}-\frac{1}{T}\sum_{t=a}^{\tau}\log P_{\phi_1(Y_t|Y_{t-p:t-1})}\\
&\quad=\lim_{T\to\infty}\mathbb{E}\left(\log\frac{P_{\phi_1}}{P_{\phi_{s_2}}}\right)-\frac{1}{T}\sum_{t=a}^{\tau}\log P_{\phi_1(Y_t|Y_{t-p:t-1})}\\
&\quad=\lim_{T\to\infty}D_{KL}(P_{\phi_1}||P_{\phi_{s_2}})-\frac{1}{T}\sum_{t=a}^{\tau}\log P_{\phi_1(Y_t|Y_{t-p:t-1})}\\
&\quad>\lim_{T\to\infty}D_{KL}(P_{\phi_1}||P_{\phi_{s_1}})-\frac{1}{T}\sum_{t=a}^{\tau}\log P_{\phi_1(Y_t|Y_{t-p:t-1})}\\
&\quad=\lim_{T\to\infty}\mathbb{E}\left(\log\frac{P_{\phi_1}}{P_{\phi_{s_1}}}\right)-\frac{1}{T}\sum_{t=a}^{\tau}\log P_{\phi_1(Y_t|Y_{t-p:t-1})}\\
&\quad=\lim_{T\to\infty}\frac{1}{T}\sum_{t=a}^{\tau}\log\frac{P_{\phi_1(Y_t|Y_{t-p:t-1})}}{P_{\phi_{s_1}(Y_t|Y_{t-p:t-1})}}-\frac{1}{T}\sum_{t=a}^{\tau}\log P_{\phi_1(Y_t|Y_{t-p:t-1})}\\
&\quad=\lim_{T\to\infty}-\frac{1}{T}\sum_{t=a}^{\tau}\log P_{\phi_{s_1}(Y_t|Y_{t-p:t-1})}
\end{aligned}
$$

Similarly, we have

$$
\lim_{T\to\infty}-\frac{1}{T}\sum_{t=\tau}^{b}\log P_{\phi_{s_2}(Y_t|Y_{t-p:t-1})}>\lim_{T\to\infty}-\frac{1}{T}\sum_{t=\tau}^{b}\log P_{\phi_{s_3}(Y_t|Y_{t-p:t-1})}
$$

We then have

$$
\lim_{T\to\infty}\frac{\mathrm{CL}(Y_{1:T}|M)-\mathrm{CL}(Y_{1:T}|M')}{T}>0
$$

which completes our proof. □