Branch and Border: Partition-Based Change Detection in Multivariate Time Series

Bryan Hooi[*][†] Christos Faloutsos[*]

Abstract

Given multivariate time series data, how do we detect changes in the behavior of the time series: for example, the onset of illnesses or complications in patients? Can we do this without making strong assumptions about the data? We propose BnB (Branch and Border), an online, nonparametric change detection method that detects multiple changes in multivariate data. Unlike existing methods, BnB approaches change detection by separating points before and after the change using an ensemble of random partitions. BnB is (a) *scalable:* it scales linearly in the number of time ticks and dimensions, and is online, thus using bounded memory and bounded time per iteration; (b) *effective:* providing theoretical guarantees on the false positive rate, and achieving 70% or more increased F-measure over baselines in experiments averaged over 11 datasets; (c) *general:* it is nonparametric, and works on mixed data, including numerical, categorical, and ordinal data.

1 Introduction

How do we detect change points in multivariate time series data? This problem has wide-ranging applications: in the medical domain, changes can indicate the onset of illnesses or complications, such as seizures, panic, and heart attacks [29]. Change detection on sensor data has been used to detect power plant failures and blackouts [3]. Other applications include intrusion detection in computer networks [35], disease outbreak detection [20], and location recognition for robots [24].

In many cases, we cannot assume that the data follows any standard distribution (e.g. Gaussian, Poisson). In realistic settings, data can be multi-modal, skewed, and nonlinear; moreover, different columns of the data may have different types: e.g. binary, numerical, ordinal, or count data. We also often want to detect change points in an **online** manner: many types of data (e.g. medical data) are received in real time. As each data point is received, the algorithm should update efficiently. Thus, our goal is an online, nonparametric change detection algorithm:

Informal Problem 1. (Online Change Detection)

- ***Given*** *a multivariate stream $X_1, X_2, \cdots$ (possibly containing numerical, ordinal or categorical data);*
- ***Find*** *a set of time ticks t where changes occurred, reported in a streaming manner.*

Change detection in time series [31, 7] has been studied extensively (expanded in Section 2). Our work differs in two key aspects. Firstly, most work focuses on purely numerical data, whereas we want to allow for e.g. count, ordinal and categorical data. Secondly, we use a nonparametric, random partition based approach. Algorithms which make distributional assumptions can be negatively affected when these do not hold: e.g. Gaussian or Poisson-based approaches will be overly influenced by outliers given heavy-tailed data, and report wrong change points. Moreover, many real world datasets may be hard to fit using **any** standard distribution: e.g. nonlinear or multimodal data.

Our contributions are as follows:

1. **Algorithm:** We propose an online nonparametric change detection algorithm based on random partitions, a novel approach for change detection.
2. **Scalability:** BnBO is linear (Figure 1a) and online, using bounded memory and time per iteration.
3. **Effectiveness:** Our algorithms outperform baselines in accuracy by 70% or more (Figure 1b), in experiments on real and synthetic data. Theorem 6.3 provides a theoretical guarantee on the false positive rate.

Reproducibility: our code and data are publicly available at `www.andrew.cmu.edu/user/bhooi/bnb`.

2 Related Work

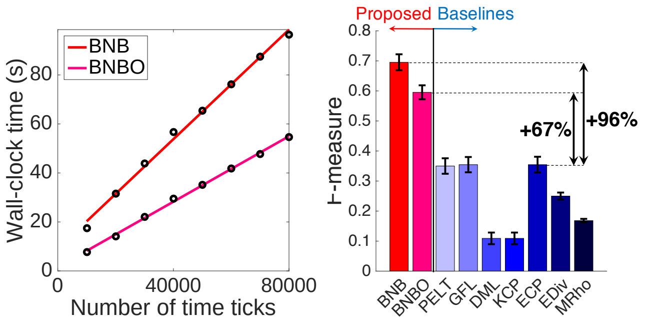
[2] reviews time series change detection methods.

Supervised Change Detection Supervised methods treat change detection as a binary classification task, or directly output class labels [30]. However, labeled data is often unavailable.

Parametric Change Detection Many approaches

[*]School of Computer Science, Carnegie Mellon University
[†]Department of Statistics, Carnegie Mellon University

Copyright © 2019 by SIAM
Unauthorized reproduction of this article is prohibited



(a) Scalability (b) Accuracy

Figure 1: **BnB is scalable and accurate:** (a) linear scalability of BnB and its online variant, BnBO (for a time series with 100 dimensions) (b) F-measure of detecting change points, averaged over 11 datasets.

rely on a parametric statistical model: e.g. Gaussians [17], and more general exponential family distributions [10]. Accompanying these are search algorithms: greedy binary segmentation [31], bottom-up segmentation [16], dynamic programming (DP) [15]. Wild binary segmentation [11] uses binary segmentation on random intervals. PELT [17] applies pruning to DP. Other approaches include Hidden Markov Models [18], Group Fused Lasso (GFL) [4], and Distance Metric Learning (DML) [34], which uses a Mahalanobis metric.

Online Change Detection FLOSS [12] is an online segmentation approach for multivariate data. GGS [13] is an online, multivariate, Gaussian approach. CUSUM [36] is an online approach for several parametric families; other control-chart approaches include e.g. MEWMA [28]. CD [27] uses a PCA-based, online multivariate approach.

Other Methods Nonparametric change detection methods include Multivariate Rho (MRHO) [19], and E-statistic based EDIV [26] and ECP [26], which use nonparametric goodness-of-fit measures. Other methods include Kernel Change Point Detection (KCP) [9], and neural networks [25].

Forest-Based Outlier Detection [23] proposed Isolation Forests, which detect outliers based on the intuition that outliers have low depth in randomly constructed trees. [32] proposes a similar but streaming approach. However, outlier detection and change detection require different intuitions: outlier detection is non-temporal, and finds deviations from the data, while change detection requires a time-series, and finds transitions from one regime to another.

Table 1 summarizes existing change point detection methods. BnB differs from existing methods in that it is online, nonparametric, and allows for multiple data types. Our approach is also very different, relying on an ensemble of random partitions.

Table 1: Comparison of relevant change detection approaches. 'Mixed Data Types' refers to allowing data types beyond just numerical data: including categorical, count, and ordinal data.

	PELT (Killick, 2012)	**GFL** (Bleakley, 2011)	**GGS** (Hallac, 2016)	**DML** (Xing, 2003)	**FLOSS** (Gharghabi, 2017)	**ZWZJ** (Zou, 2016)	**KCP** (Desobry, 2005)	**MRHO** (Kojadinovic, 2016)	**EDIV** (Matteson, 2014)	**ECP** (Matteson, 2014)	**CD** (Qahtan, 2015)	**BnBO**
Multivariate Data	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✔
Nonparametric					✓		✓	✓	✓	✓	✓	✔
Scales Linearly	✓	✓	✓		✓	✓		✓				✔
Online Algorithm			✓		✓	✓					✓	✔
Mixed Data Types												✔

3 Problem Definition

3.1 Problem Setting

Table 2 shows the symbols used in this paper.

Table 2: Symbols and definitions

Symbol	**Interpretation**
n	Number of time ticks in data
d	Number of dimensions in data
X_t	Data point at time t
X_{tj}	Data value at time t in dimension j
w	Window size
d_{lim}	Depth limit for trees
N	Number of trees
$s_{k,t}$	Separation depth for tree k at time t (Definition 2)
c_t	Change score for time t (Definition 3)
z_t	Indicator vector for ancestors of X_t (Section 5.3)
s_L, s_R	Left and right ancestor counts (Section 5.3)
s_I	Intersection set (Section 5.3)

We are given X, an $n \times d$ dataset, i.e. it has n time ticks with d dimensions. Denote by X_t the data point at time t, and denote by X_{tj} the data value in dimension j at time t. Our goal is to detect **change points**: i.e. time ticks where the behavior of the time series changes significantly. Formally:

Problem 1. (Multivariate Change Detection)

- ***Given*** *a multivariate dataset of t time ticks with d-dimensions (X_{tj} for $j = 1, \cdots, d$ and $t = 1, \cdots, n$) containing numeric, ordinal or categorical data;*
- ***Output*** *a set of time ticks where changes occurred.*

A major challenge is the **online** setting, in which we receive the data incrementally, one time tick at a time, i.e. $X_1, X_2, \cdots$. In some cases the stream may be infinite, in which case $n = \infty$. As we receive the data,

Copyright © 2019 by SIAM
Unauthorized reproduction of this article is prohibited

we should output which time ticks are change points.

However, it would be unrealistic to expect to determine whether time point t is a change point without any future information after time t: this is because the time ticks $t+1, t+2, \cdots$ are important in deciding whether an apparent change at time t was a true change point, or simply noise. Hence, we introduce a **window** parameter w: upon receiving time tick $t+w$, the algorithm must output whether time t is a change point.

Problem 2. (Online Change Detection)

- ***Given*** *a multivariate stream of d-dimensional data (X_{tj} for $j = 1, \cdots, d$ and $t = 1, 2, \cdots$), possibly containing numeric, ordinal or categorical data;*
- ***Output*** *at time $t+w$ whether a change point occurred at time tick t.*

4 Illustrative Example

The main idea of our approach is its use of random partitions to measure how good a change is. Consider a 1-dimensional toy dataset: $X = [0, 1, 0, 0, 10, 10, 9, 10]$, which clearly has a change point at $t = 5$. If we sample a random cut point uniformly between 0 and 10 (the min and max of the data), then with 80% probability, the cut point lies in $(1, 9)$. In this case, the cut point perfectly separates the points before and after $t = 5$. Intuitively then, $t = 5$ is a good change point because the points before and after (or equal to) it are easily separable by random partitions.

In real data, single cuts do not always suffice, and we may need multiple cuts, but the intuition is similar. Consider the drawing of a bird on a branch[1] in Figure 2a, containing an unknown change point, from drawing the branch to drawing the bird. The connecting line segments (in black) indicate that the drawing is a time series, not just a set of points. Finding the change point is challenging: the bird's feet and tail are hard to separate from the branch. Figure 2b shows that the Gaussian approach has difficulty: these approaches rely on a large difference in means before and after the change, but the bird and branch have fairly close means.

Figure 2c illustrates our random partition approach. We conduct random cuts (gray lines) that partition the space recursively. At the time of the correct change, the points before the change (in red) and after the change (in blue) can be cleanly separated by the random partition: i.e. all boxes contain either only red or only blue points. The stronger the change point, the easier the red and blue points are to cleanly separate, and the simpler the random partition we would need to cleanly separate them. Hence, in Figure 2d we compute a score for each possible change point based on how easily we can separate the points before and after the change using random partitions (we formalize this in Section 5). Averaging over many such partitions, Figure 2d shows that the highest change score indeed coincides with the true change point.

[1]Drawn based on http://getdrawings.com/bird-on-a-branch-drawing.

5 Proposed BnB Algorithm

5.1 Random Partition Tree We first explain how our random trees are built. Given a dataset X of size n with d dimensions, we first uniformly sample one of the d dimensions to split. Then, we sample the split point: we sample two independent uniform values between the minimum and maximum of the data values, and use their mean as the split point. This causes split points to tend toward the middle of the range, making the children more balanced in their node counts. We then recurse onto each child, stopping when a node contains only one data point, or reaches a depth limit d_{lim}.

Recall that we want to allow for ordinal and categorical data. These can be easily handled by splitting ordinal data at a randomly chosen split point, and splitting categorical data by randomly dividing the categories into two sets.

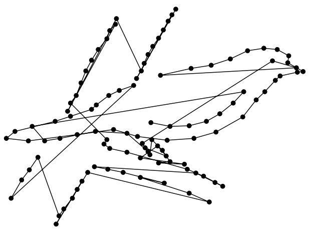
Definition 1. (Random Partition Tree) *A random partition tree is built by recursively sampling a feature, and then sampling a split point to partition the data space into two, continuing recursively until a depth of d_{lim} is reached.*

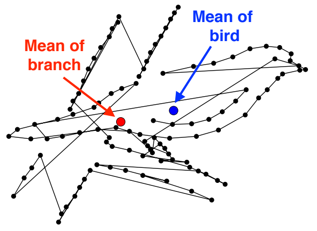
5.2 Change Score Our main intuition is that if time tick t is a good change point, then the data before and after the change point should be cleanly separable using relatively simple partitions, i.e. shallow trees. For a given candidate change point t, define the '*left set*' L as the w points before time t (i.e. $\{X_{t-w}, \cdots, X_{t-1}\}$). Similarly, define the '*right set*' R as the w points at or after time t (i.e. $\{X_t, \cdots, X_{t+w-1}\}$). Given a single random partition tree $\mathcal{T}$, define the '*separation depth*' as the minimum depth at which $\mathcal{T}$ fully separates the left set from the right set. Here 'fully separated' means that like in Figure 2c, each node of $\mathcal{T}$ at that depth either contains only points from L or R, but not both.

Definition 2. (Separation Depth) *The separation depth $s_{k,t}$ for the kth random partition tree at time tick t is the minimum depth at which $\mathcal{T}_k$ fully separates the left set from the right set.*

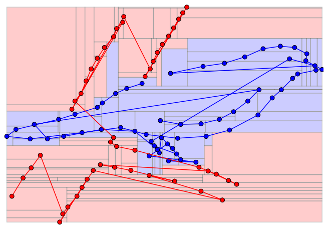
Intuitively, low separation depth means that L and R can be easily separated, which means t is a good change point. If L and R are not fully separated when reaching the depth limit d_{lim}, we set the separation

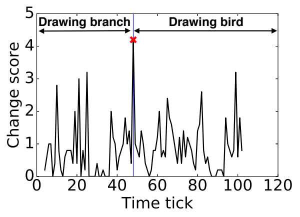
Copyright © 2019 by SIAM
Unauthorized reproduction of this article is prohibited


(a) Original drawing



(b) Gaussian approach fails


(c) Random partition approach finds change point



(d) Peak in change score coincides with ground truth

Figure 2: **BnB is adaptive and nonparametric:** (a) A drawing of a bird on a branch, with an unknown change point (when moving from the branch to the bird) (b) The bird and the branch have similar means, making them difficult to separate using standard approaches. (c) At the time of the true change, the red points (before the change) and blue points (after the change) can be fully separated using a relatively simple random partition, suggesting that this is a good change. The blue region (boxes only containing points after the change) conforms to the bird's outline well despite the bird's unique shape which is hard to separate from the branch. (d) Averaging over many such partitions, the change score is highest at the true change time.

depth to $d_{lim} + 1$, which functions as an upper limit to prevent any single tree from being too influential.

Finally, we average the separation depth values over N random partition trees. Since $d_{lim} + 1$ is an upper bound, we then use $d_{lim} + 1$ minus this average separation depth as our change score: thus, higher values indicate a better change, with 0 being the minimum value indicating a poor change.

Definition 3. (Change Score) *The change score c_t at time tick t is $d_{lim} + 1$ minus the average separation depth computed using N random partition trees:*

$$c_t = d_{lim} + 1 - \frac{1}{N} \sum_{j=1}^{N} s_{j,t} \tag{5.1}$$

5.3 Efficient Implementation The step of computing change scores can be sped up significantly over the naive approach. Implemented naively, at each of n time ticks, we would need to compute N separation depths, each taking $O(w)$ time, for a total of $O(n \cdot N \cdot w)$. However, we can save a factor of $O(w)$, by computing the change scores in a single pass over the dataset.

Consider a random partition tree $\mathcal{T}$. Every data point X_t for each t has been assigned to a leaf of $\mathcal{T}$ during the tree-building step. For convenience, let the nodes in $\mathcal{T}$ be assigned integer ids $i = 1, 2, \cdots$ arbitrarily. Define s_L ('*left ancestor count*'), a sparse vector: for each node i of $\mathcal{T}$, $s_L[i]$ is the number of points in the left set that node i contains. For example, in Figure 2c, each node corresponds to a box; then $s_L[i]$ is the number of points from the left set in that box. Define s_R ('*right ancestor count*') similarly using the right set. Define s_I ('*intersection set*') as the set of nodes which have nonzero counts in both s_L and s_R.

Lemma 5.1. *At time t, the separation depth is 1 more than the maximum depth among nodes in s_I.*

Proof. All nodes with nonzero counts in s_L are those with descendants in the left set, and similarly for s_R, so s_I contains nodes with descendants from both sets. If D is the maximum depth among nodes in s_I, the tree at depth D is not enough to perfectly separate the left and right sets, since a node at depth D has descendants from both sets (so the node contains points from the left and the right sets). However, the tree at depth $D + 1$ does perfectly separate the left and right sets, otherwise there would exist a node in s_I with depth $D + 1$. ■

Algorithm 1 shows our change scoring algorithm. We initialize s_L, s_R, s_I and separation depths $s_{j,t}$, then maintain them efficiently as we move forward in time.

Let z_t be a 0-1 vector with 1 in entry i iff node i is an ancestor of the leaf containing the data point X_t. Note then that s_L is just the sum of z_t over the left set, and similarly for s_R. Initially, the left set contains the first w time ticks, so s_L is initialized as $s_L = \sum_{t=1}^{w} z_t$ (Line 4). Similarly $s_R = \sum_{t=w+1}^{2w} z_t$ (Line 5). Next we initialize s_I as the set of nodes with nonzero counts in both s_L and s_R (Line 7), and compute the separation depth $s_{j,w+1}$ following Lemma 5.1 (Line 8).

To maintain these values, for s_L and s_R we add and subtract the relevant vectors z_t (Lines 12 and 13). For each such change, we incrementally update the intersection set s_I (Line 14) and thus the separation depth (Line 15). Finally, we compute the overall change score by averaging over all the trees (Line 16).

Algorithm 2 summarizes our full offline algorithm BnB. It estimates N random partition trees, then uses ChangeScore to compute scores.

Copyright © 2019 by SIAM
Unauthorized reproduction of this article is prohibited

Algorithm 1: ChangeScore

```
Input : Dataset $X$, $N$ random partition trees,
        window $w$
Output: Change scores $c_t$ for $w \le t \le n - w + 1$
1  ▷For each tree:
2  for $j$ in 1 to $N$ do
3      ▷Initialize left and right ancestor counts
4      $s_L = \sum_{t=1}^{w} z_t$
5      $s_R = \sum_{t=w+1}^{2w} z_t$
6      ▷Intersection set and separation depth
7      $s_I = \{k : s_L[i] > 0$ and $s_R[i] > 0\}$
8      $s_{j,w+1} = 1 + \max_{i \in s_I} \text{depth}(i)$
9  for $t$ in $w$ to $n - w + 1$ do
10     for $j$ in 1 to $N$ do
11         ▷Update due to change in left and right sets
12         $s_L \leftarrow s_L + z_t - z_{t-w}$
13         $s_R \leftarrow s_R - z_{t+w} - z_t$
14         $s_I = \{x : s_L[x] > 0$ and $s_R[x] > 0\}$
15         $s_{j,t} = 1 + \max_{x \in s_I} \text{depth}(x)$
16 return $c_t = d_{lim} + 1 - \frac{1}{N}\sum_{j=1}^{N} s_{j,t}, \ \forall\, t$
```

Algorithm 2: Offline Change Detection BnB

```
Input : Dataset $X$, window $w$, number of trees $N$,
        depth limit $d_{lim}$
Output: Change scores $c_t$ for $w \le t \le n - w + 1$
1  for $j$ in 1 to $N$ do
2      ▷Random Partition Tree creation (Section 5.1)
3      $\mathcal{T}_j = \text{MakeTree}(X, d_{lim})$
4  return ChangeScore$(X, w, \mathcal{T})$
```

5.4 Online Algorithm (BnBO) We now explain how BnB can be modified to be run online. The change score computation ChangeScore is already online: Lines 2 to 8 are based on only the first $2w$ time ticks, while Lines 12 to 15 are based only on the current window. Hence, it adapts to streams of arbitrary length. At time t, to compute z_t, for each tree, we 'classify' X_t by recursively following the split points, from the root node down to whichever leaf node X_t belongs to.

Only the tree generation algorithm MakeTree is not yet online, as it uses the full dataset to choose each split point. BnBO uses a small 'initialization' dataset of 250 points, which we use for MakeTree to decide the split points for all trees. The resulting trees are then used in the online version of ChangeScore for future data points in the stream.

5.5 Time Complexity For BnB's tree-building step, we have N trees, each requiring at most d_{lim} splits, where all splits combined in each level involve n data points. Hence, tree-building takes $O(N \cdot n \cdot d_{lim})$. For change scoring, we have N trees, each of which has to be updated $O(n)$ times, and each update is $O(d_{lim})$ since each z_t has at most d_{lim} nonzeroes, and Lines 12 to 15 are sparse vector operations with $O(d_{lim})$ nonzeroes. Hence overall runtime is $O(N \cdot n \cdot d_{lim})$.

For the same reasons, BnBO's tree building step requires $O(N \cdot d_{lim})$. For each online update, computing z_t is still proportional to tree height, since we travel the height of the tree, so each online update is $O(N \cdot d_{lim})$.

6 Theoretical Analysis

6.1 Interpretation of Separation Depth Can we better understand what BnB does, by re-interpreting it in a more theoretical way: e.g. is there a metric space (i.e. a space where we can measure distances), for which our approach is related to distances in this space? We show that the answer is 'yes'.

A **metric space** is a set $\mathcal{X}$ and a distance function $d : \mathcal{X} \times \mathcal{X} \to [0, \infty)$ such that:

1. $d(x, y) = 0$ iff $x = y$ (Identity)
2. $d(x, y) = d(y, x)$ (Symmetry)
3. $d(x, z) \le d(x, y) + d(y, z)$ (Triangle Inequality)

Given a tree $\mathcal{T}_k$, the tree partitions the data space, mapping each data point X_t into a leaf of $\mathcal{T}_k$: let leaf(X_t) map X_t to its leaf. Define $\mathcal{X}$ as the set of leaves, and $d(x, y)$ as the shortest path distance (in terms of number of graph hops) between leaves x and y.

Lemma 6.1. *$(\mathcal{X}, d)$ is a metric space.*

Proof. This is a special case of the fact that geodesic distance in a graph forms a metric space, if and only if the graph is connected [5]. In our case, the graph is a tree, hence the leaves indeed form a metric space. ∎

Theorem 6.1. *The separation depth $s_{k,t}$ can be rewritten in terms of distances in this metric space:*

$$s_{k,t} = d_{lim} - \frac{1}{2} \min_{x \in L, y \in R} d(\mathit{leaf}(x), \mathit{leaf}(y)) + 1 \tag{6.2}$$

Proof. We prove this in our supplement [1]. ∎

This implies that separation depth $s_{k,t}$, which measures how good a change is, is effectively a nearest-neighbor like statistic (though not in a Euclidean space).

6.2 Bounds on False Positive Rate Our main theorems bound our false positive rate. Assume that no change is present; as is standard in change detection literature, we use the null hypothesis H_0 : $X_{t-w}, \cdots, X_{t+w-1}$ are i.i.d., representing no change,

Copyright © 2019 by SIAM
Unauthorized reproduction of this article is prohibited

and bound the probability of high values of change score. First consider a single tree, which outputs the separation depth $s_{k,t}$ as change score. Then:

THEOREM 6.2. *For any $\lambda > 0$, letting $p_\lambda = 2^{\lambda-1}$, we have:*

$$P(s_{k,t} \le \lambda) \le (1/2)^{w-p_\lambda}. \tag{6.3}$$

Proof. We prove this in our supplement [1]. The main idea is to condition on the *set* of values taken by $X_{t-w}, \cdots, X_{t+w-1}$ (but not their order); then we can bound the probability that any partition fully separates the left and right sets. ∎

The next theorem concerns our final change score c_t. For any error threshold $\varepsilon > 0$, we have:

THEOREM 6.3.

$$P(c_t \ge d_{lim} + 1 - \log(1 + \log\frac{\varepsilon}{N} + w)) \le N(1/2)^{w-p_\lambda} \tag{6.4}$$

Proof. We prove this in our supplement [1]. ∎

This implies that any change scores above this value can be used to trigger an alarm with $1 - \varepsilon$ confidence, i.e. at most ε probability of a false positive.

7 Experiments

We design experiments to answer the questions:

- **Q1. Change Detection Accuracy:** how accurate are BnB and BnBO compared to baselines?
- **Q2. Scalability:** do they scale linearly?
- **Q3. Real-World Discoveries:** how do they perform on an indoor occupancy detection task?

Experiments were done on a 2.4 GHz Intel Core i5 Macbook Pro, 16 GB RAM running OS X 10.11.2. We implement our method in Python. For our methods we set $w = 15, d_{lim} = 15$ and $N = 50$, but show experiments on parameter sensitivity in this section.

We evaluate our algorithms on multivariate time series with various sizes and domains, described in Table 3. The `Occupancy` dataset contains ground truth change labels, which are the time points when the room changed from occupied to unoccupied, or vice versa.

7.1 Q1. Detection Accuracy

We now evaluate our algorithms' accuracy in detecting change points.

Single Change Points For each dataset, we sample a change point as a random time tick between $n/4$ to $3n/4$ (rounded). For data points on or after the change point, we add a Gaussian with mean 0 and standard deviation

Table 3: Datasets used.

Dataset name	Time Ticks	Dimensions	Content
`Chemical` [8]	9357	13	Chemical Sensor
`Beijing` [22]	43824	6	Air Quality
`Exchange` [21]	7588	8	Exchange Rate
`Measles` [33]	2501	50	Disease Counts
`Influenza` [33]	2501	50	Disease Counts
`Scarlet` [33]	2501	50	Disease Counts
`Whooping` [33]	2501	50	Disease Counts
`Vehicle1` [14]	264	2	Unmanned Vehicle
`Vehicle2` [14]	342	2	Unmanned Vehicle
`Vehicle3` [14]	578	2	Unmanned Vehicle
`Vehicle4` [14]	488	2	Unmanned Vehicle
`Occupancy` [6]	9752	5	In-room Sensors
`Synthetic`	10K-80K	100-800	Synthetic

for each dimension equal to the standard deviation of the data for that dimension. Then the goal of each algorithm is to detect the change point (exactly). For each algorithm we pass input parameters for detecting one change point, and evaluate the results according to F-measure against the true change point. As several of the baselines scale quadratically or slower, we subset all datasets to a size of 500. However, in Section 7.2 we show that our algorithms easily scale to $10K - 80K$ time ticks and $100 - 800$ dimensions. The results are averaged over 20 repetitions.

Baselines We compare BnB to the following recently proposed multivariate change detection methods:

- *Parametric:* PELT (Killick, 2012) is a dynamic programming (DP)-based approach. GFL (Bleakley, 2011) uses group-fused lasso. DML (Xing, 2003) is a distance metric learning approach, using a Mahalanobis metric.
- *Nonparametric:* KCP (Desobry, 2005) is a kernel-based approach. EDiv (Matteson, 2014) and ECP (Matteson, 2014) use the E-statistic, a nonparametric goodness-of-fit statistic, with hierarchical division and DP. MRho (Kojadinovic, 2016) uses Spearman's rho.

We show results for unnormalized data in Table 4, and normalized data in Table 5. In both, BnB and BnBO perform the best on almost all the datasets. Normalization does not significantly affect our methods, as all their steps are invariant to scaling. The baselines mostly perform slightly better with normalization than without. The results averaged over all 11 datasets is in Figures 3a and 3b. BnB and BnBO significantly outperform the baselines: e.g. by 82% or more for BnB. Between our two methods, BnB generally performs better, but BnBO still outperforms the baselines significantly despite all the baselines being offline.

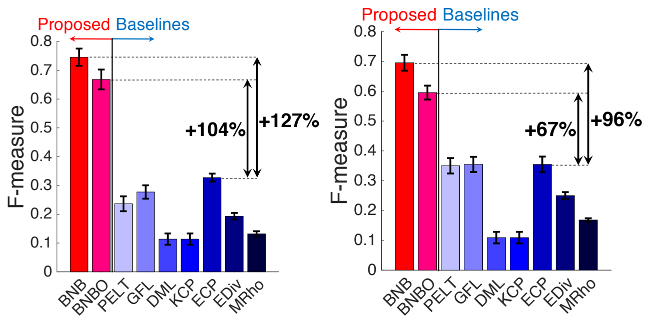
Copyright © 2019 by SIAM
Unauthorized reproduction of this article is prohibited

	BNB	BNBO	PELT	GFL	DML	KCP	ECP	EDiv	MRho
Chemical	**0.70**	**0.70**	0.50	0.50	0.15	0.15	0.45	0.28	0.00
Beijing	**0.80**	0.55	0.00	0.00	0.15	0.15	0.00	0.05	0.00
Exchange	**1.00**	0.90	0.00	0.10	0.00	0.00	0.15	0.00	0.00
Measles	**0.95**	**0.95**	0.40	0.45	0.20	0.20	0.15	0.33	0.33
Influenza	**0.95**	1.00	0.40	0.55	0.10	0.10	0.75	0.45	0.45
Scarlet	0.55	**0.60**	0.45	0.55	0.20	0.20	0.05	0.28	0.28
Whooping	**1.00**	0.70	0.00	0.00	0.20	0.20	1.00	0.40	0.40
Vehicle1	**0.65**	0.60	0.35	0.40	0.10	0.10	0.10	0.17	0.00
Vehicle2	0.40	**0.50**	0.10	0.10	0.05	0.05	0.15	0.05	0.00
Vehicle3	**0.25**	0.15	0.15	0.15	0.05	0.05	0.10	0.07	0.00
Vehicle4	**0.95**	0.70	0.25	0.25	0.05	0.05	0.70	0.05	0.00

Table 4: F-measure for single change point detection (without normalization)

	BNB	BNBO	PELT	GFL	DML	KCP	ECP	EDiv	MRho
Chemical	0.70	**0.75**	0.65	0.65	0.30	0.30	0.50	0.28	0.00
Beijing	**0.50**	0.40	0.40	0.40	0.20	0.20	0.10	0.20	0.00
Exchange	**0.95**	0.85	0.25	0.25	0.10	0.10	0.50	0.12	0.00
Measles	**0.95**	0.90	0.80	0.90	0.20	0.20	0.00	0.42	0.42
Influenza	**0.95**	**0.95**	0.20	0.20	0.20	0.20	0.40	0.47	0.47
Scarlet	0.60	0.35	**0.90**	**0.90**	0.40	0.40	0.40	0.40	0.40
Whooping	**1.00**	0.70	0.00	0.00	0.15	0.15	**1.00**	0.47	0.47
Vehicle1	**0.50**	**0.50**	0.30	0.30	0.10	0.10	0.25	0.20	0.00
Vehicle2	0.15	**0.30**	0.00	0.00	0.00	0.00	0.10	0.03	0.00
Vehicle3	**0.40**	0.30	0.00	0.00	0.00	0.00	0.00	0.03	0.00
Vehicle4	**0.95**	0.55	0.45	0.45	0.05	0.05	**0.95**	0.20	0.00

Table 5: F-measure for single change point detection (with normalization)



(a) Without normalization (b) With normalization

Figure 3: **Accuracy for single change detection:** (a) F-measure of detecting change points, averaged over 11 datasets. Error bars indicate one standard deviation. (b) Same results, with normalization for all methods.

Multiple Change Detection We use the same settings, except with 5 change points, sampled uniformly without replacement from $0.1n$ to $0.9n$ (rounded). We evaluate each algorithm based on its top 5 changes, again using F-measure, compared to the true changes. Results are averaged over 20 repetitions.

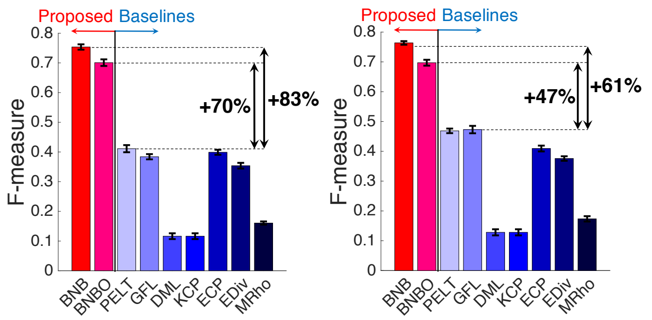
The results are shown in Table 6 for unnormalized data and Table 7 for normalized data. BnB and

	BNB	BNBO	PELT	GFL	DML	KCP	ECP	EDiv	MRho
Chemical	**0.68**	0.54	0.52	0.37	0.12	0.12	0.40	0.38	0.00
Beijing	**0.62**	0.61	0.21	0.23	0.03	0.03	0.19	0.23	0.00
Exchange	**0.91**	0.84	0.43	0.35	0.08	0.08	0.40	0.31	0.00
Measles	**0.90**	0.85	0.37	0.50	0.11	0.11	0.26	0.41	0.41
Influenza	**0.99**	**0.99**	0.29	0.59	0.14	0.14	0.61	0.50	0.50
Scarlet	**0.60**	0.56	0.31	0.40	0.10	0.10	0.10	0.29	0.29
Whooping	**0.98**	0.95	0.64	0.42	0.26	0.26	0.75	0.52	0.52
Vehicle1	0.68	**0.71**	0.28	0.29	0.09	0.09	0.21	0.20	0.00
Vehicle2	**0.50**	**0.50**	0.38	0.29	0.15	0.15	0.45	0.33	0.03
Vehicle3	**0.54**	0.44	0.41	0.36	0.06	0.06	0.34	0.30	0.02
Vehicle4	**0.89**	0.72	0.68	0.44	0.14	0.14	0.68	0.43	0.00

Table 6: F-measure for multiple change point detection (without normalization)

	BNB	BNBO	PELT	GFL	DML	KCP	ECP	EDiv	MRho
Chemical	**0.68**	0.55	0.58	0.63	0.14	0.14	0.45	0.41	0.02
Beijing	**0.74**	0.67	0.24	0.42	0.09	0.09	0.33	0.22	0.00
Exchange	**0.93**	0.87	0.57	0.51	0.11	0.11	0.47	0.37	0.02
Measles	**0.91**	0.87	0.36	0.69	0.14	0.14	0.26	0.45	0.45
Influenza	**0.98**	0.97	0.40	0.59	0.12	0.12	0.62	0.48	0.48
Scarlet	**0.71**	0.62	0.49	**0.71**	0.11	0.11	0.22	0.42	0.42
Whooping	**0.99**	0.95	0.64	0.48	0.17	0.17	0.72	0.55	0.55
Vehicle1	**0.61**	0.61	0.26	0.24	0.05	0.05	0.17	0.15	0.00
Vehicle2	0.49	**0.50**	0.41	0.28	0.08	0.08	0.41	0.32	0.00
Vehicle3	**0.56**	0.50	0.43	0.30	0.06	0.06	0.29	0.35	0.02
Vehicle4	**0.91**	0.63	0.62	0.47	0.21	0.21	0.62	0.47	0.00

Table 7: F-measure for multiple change point detection (with normalization)



(a) Without normalization (b) With normalization

Figure 4: **Accuracy for multiple change detection:** (a) F-measure of detecting multiple change points, averaged over 11 datasets. Error bars indicate one standard deviation. (b) Same results, with normalization.

BnBO again outperform the baselines on almost all the datasets. The average F-measure over all datasets are shown in Figures 4a and 4b. BnB outperforms the baselines by 70% or more F-measure on average.

Parameter Sensitivity Figure 5 compares performance for different parameter values, on multiple change point detection (without normalization). Re-

Copyright © 2019 by SIAM
Unauthorized reproduction of this article is prohibited

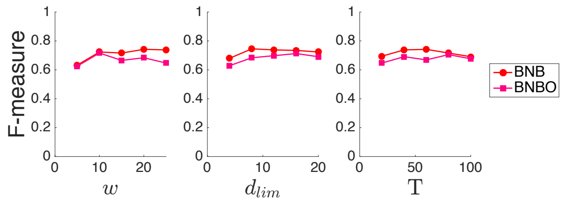


Figure 5: **Insensitive to parameters:** our methods perform consistently across parameter values.

sults are averaged over datasets and over 5 repetitions; the other settings follow the previous section. Our methods perform consistently well across values.

7.2 Q2. Scalability Next, we verify that our methods scale linearly. We repeatedly duplicate our `Occupancy` dataset in time ticks and dimensions, add Gaussian noise to each dimension with standard deviation equal to the standard deviation of that dimension, then subset the data to the required size. Figure 6a shows that BnB and BnBO scale linearly in time ticks; here dimensions is fixed at 100. Figure 6b shows that they scale linearly in dimensions; here time ticks is fixed at 10,000. Both methods are fast: for 100 dimensional data, BnB takes 1.2ms per time tick on average, while BnBO takes 0.7ms, on a laptop computer. The online nature of BnBO means that in settings where we need incremental results, it provides further efficiency gains compared to repeatedly running an offline algorithm.

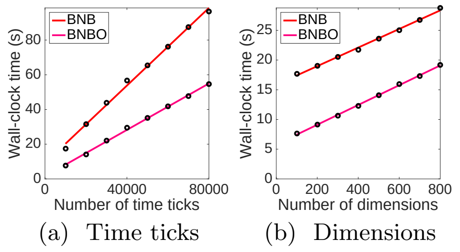


Figure 6: **Our methods scale linearly.**

7.3 Q3. Real-World Effectiveness We now evaluate our methods against the same baselines on a real-world occupancy detection task [6], with ground truth change points when the room changed from being unoccupied to occupied, or vice versa, resulting in 13 change points. The sensors measure the room's temperature, humidity, light and carbon dioxide levels. Ground truth occupancy was obtained from time-stamped pictures taken every minute. We evaluate using F-measure, where reported change points are considered as correctly matched to a given ground truth change if the two are at most '*tolerance*' minutes apart, where we plot different values of '*tolerance*' in Figure 7. We repeat each method 5 times and average the results.

BnB and BnBO outperform the baselines, with BnB having F-measure of 115% higher accuracy. We omit GFL, DML and KCP as they did not terminate after 16 hours. (BnB and BnBO took 57s and 28s.)

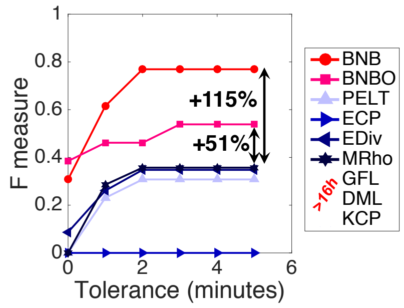


Figure 7: **Accuracy for occupancy detection:** our methods detect changes in occupancy in a room accurately, based on F-measure.

8 Conclusion

We propose BnB, which uses an ensemble of random partitions to measure change score. It generalizes to different data types , and BnBO provides an online approach. Our contributions are:

1. **Algorithm:** We propose a novel online nonparametric change detection approach.
2. **Scalability:** BnBO is linear (Figure 1a) and online, using bounded memory and time per iteration.
3. **Effectiveness:** Our algorithms outperform baselines in accuracy by 70% F-measure or more, on real and synthetic data. Theorem 6.3 provides a theoretical guarantee on the false positive rate.

9 Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant No. IIS-1247489, and by the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, or other funding parties. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

[1] Supplementary document. `http://www.andrew.cmu.edu/user/bhooi/bnb/supplement.pdf`, 2018.

[2] S. Aminikhanghahi and D. J. Cook. A survey of methods for time series change point detection. *KIS*, 51(2):339–367, 2017.

Copyright © 2019 by SIAM
Unauthorized reproduction of this article is prohibited

[3] T. Banerjee, Y. C. Chen, A. D. Dominguez-Garcia, and V. V. Veeravalli. Power system line outage detection and identificationa quickest change detection approach. In *ICASSP*, pages 3450–3454. IEEE, 2014.

[4] K. Bleakley and J.-P. Vert. The group fused lasso for multiple change-point detection. *arXiv preprint arXiv:1106.4199*, 2011.

[5] J. Bouttier, P. Di Francesco, and E. Guitter. Geodesic distance in planar graphs. *Nuclear Physics B*, 663(3):535–567, 2003.

[6] 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.

[7] H. Cho and P. Fryzlewicz. Multiple-change-point detection for high dimensional time series via sparsified binary segmentation. *JRSS(B)*, 77(2):475–507, 2015.

[8] S. De Vito, E. Massera, M. Piga, L. Martinotto, and G. Di Francia. On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario. *Sensors and Actuators B: Chemical*, 129(2):750–757, 2008.

[9] F. Desobry, M. Davy, and C. Doncarli. An online kernel change detection algorithm. *IEEE TSP*, 2005.

[10] K. Frick, A. Munk, and H. Sieling. Multiscale change point inference. *JRSS(B)*, 76(3):495–580, 2014.

[11] P. Fryzlewicz. Wild binary segmentation for multiple change-point detection. *The Annals of Statistics*, 42(6):2243–2281, 2014.

[12] S. Gharghabi, Y. Ding, C.-C. M. Yeh, K. Kamgar, L. Ulanova, and E. Keogh. Matrix profile viii: Domain agnostic online semantic segmentation at superhuman performance levels. In *ICDM*. IEEE, 2017.

[13] D. Hallac, P. Nystrup, and S. Boyd. Greedy gaussian segmentation of multivariate time series. *arXiv preprint arXiv:1610.07435*, 2016.

[14] N. Harth and C. Anagnostopoulos. Edge-centric efficient regression analytics. In *2018 IEEE EDGE*, pages 93–100. IEEE, 2018.

[15] B. Jackson, J. D. Scargle, D. Barnes, S. Arabhi, A. Alt, P. Gioumousis, E. Gwin, P. Sangtrakulcharoen, L. Tan, and T. T. Tsai. An algorithm for optimal partitioning of data on an interval. *IEEE Signal Processing Letters*, 12(2):105–108, 2005.

[16] E. Keogh, S. Chu, D. Hart, and M. Pazzani. An online algorithm for segmenting time series. In *ICDM*. IEEE, 2001.

[17] R. Killick, P. Fearnhead, and I. A. Eckley. Optimal detection of changepoints with a linear computational cost. *JASA*, 107(500):1590–1598, 2012.

[18] S. I. Ko, T. T. Chong, P. Ghosh, et al. Dirichlet process hidden markov multiple change-point model. *Bayesian Analysis*, 10(2):275–296, 2015.

[19] I. Kojadinovic, J.-F. Quessy, and T. Rohmer. Testing the constancy of spearmans rho in multivariate time series. *Annals of the Institute of Statistical Mathematics*, 68(5):929–954, 2016.

[20] M. Kulldorff, R. Heffernan, J. Hartman, R. Assunçao, and F. Mostashari. A space–time permutation scan statistic for disease outbreak detection. *PLoS medicine*, 2(3):e59, 2005.

[21] G. Lai, W.-C. Chang, Y. Yang, and H. Liu. Modeling long-and short-term temporal patterns with deep neural networks. *arXiv preprint arXiv:1703.07015*, 2017.

[22] X. Liang, T. Zou, B. Guo, S. Li, H. Zhang, S. Zhang, H. Huang, and S. X. Chen. Assessing beijing's pm2. 5 pollution: severity, weather impact, apec and winter heating. *Proc. R. Soc. A*, 471(2182):20150257, 2015.

[23] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In *ICDM*. IEEE, 2008.

[24] S. Liu, M. Yamada, N. Collier, and M. Sugiyama. Change-point detection in time-series data by relative density-ratio estimation. *Neural Networks*, 43:72–83, 2013.

[25] X. Liu and R. Lathrop Jr. Urban change detection based on an artificial neural network. *International Journal of Remote Sensing*, 23(12):2513–2518, 2002.

[26] D. S. Matteson and N. A. James. A nonparametric approach for multiple change point analysis of multivariate data. *JASA*, 109(505):334–345, 2014.

[27] A. A. Qahtan, B. Alharbi, S. Wang, and X. Zhang. A pca-based change detection framework for multidimensional data streams: Change detection in multidimensional data streams. In *KDD*. ACM, 2015.

[28] D. Qi, Z. Li, and Z. Wang. On-line monitoring data quality of high-dimensional data streams. *JSCS*, 86(11):2204–2216, 2016.

[29] M. Z. Rad, S. R. Ghuchani, K. Bahaadinbeigy, and M. M. Khalilzadeh. Real time recognition of heart attack in a smart phone. *Acta Informatica Medica*, 23(3):151, 2015.

[30] S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava. Using mobile phones to determine transportation modes. *ACM TOSN*, 6(2):13, 2010.

[31] A. J. Scott and M. Knott. A cluster analysis method for grouping means in the analysis of variance. *Biometrics*, pages 507–512, 1974.

[32] S. C. Tan, K. M. Ting, and T. F. Liu. Fast anomaly detection for streaming data. In *IJCAI*, 2011.

[33] W. G. Van Panhuis, J. Grefenstette, S. Y. Jung, N. S. Chok, A. Cross, H. Eng, B. Y. Lee, V. Zadorozhny, S. Brown, D. Cummings, et al. Contagious diseases in the united states from 1888 to the present. *NEJM*, 2013.

[34] E. P. Xing, M. I. Jordan, S. J. Russell, and A. Y. Ng. Distance metric learning with application to clustering with side-information. In *NIPS*, pages 521–528, 2003.

[35] K. Yamanishi, J.-I. Takeuchi, G. Williams, and P. Milne. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. *DMKD*, 8(3):275–300, 2004.

[36] C. Zou, Z. Wang, X. Zi, and W. Jiang. An efficient online monitoring method for high-dimensional data streams. *Technometrics*, 57(3):374–387, 2015.

Copyright © 2019 by SIAM
Unauthorized reproduction of this article is prohibited