Classification-Based Barrier Change Point Detection Methods

Artemii Patov
Saint-Petersburg State University
Saint-Petersburg, Russia
patov.988@gmail.com

Viacheslav Gorikhovskii
Saint-Petersburg State University
Saint-Petersburg, Russia
v.gorikhovskii@spbu.ru

Vladimir Kutuev
Saint-Petersburg State University
Saint-Petersburg, Russia
v.kutuev@spbu.ru

***Abstract*—The change point detection problem in the time series arises in a wide variety of fields. In some situations, we lack the necessary resources to apply complex techniques, so lightweight approaches are needed.**

In this study, we consider lightweight approaches to the change point detection problem, namely, classification-based barrier methods. We want to investigate the use of various classifiers and classification evaluation metrics for change point detection, so we are creating a framework that makes it easy to build methods from different components. To study a large number of constructed methods, we create a flexible benchmarking system that allows one to evaluate methods using different metrics.

We conduct an empirical study of the methods, present the results and compare them with existing methods and with each other. Our implementation of the KNN based method shows high-quality results. However, we see potential in using at least one more tested classifier as well.

I. Introduction

Change point detection in the sequence of observations is an attempt to detect sudden changes in the incoming dataset, where not only fundamental changes are possible but also regular random fluctuations that should not be taken into account. This problem arises in completely different areas: finance [1], [2], where, for example, it is necessary to quickly register changes in the exchange rate; tracking the spread of epidemic diseases [3], where it is important to monitor the dynamics of the spread of the epidemic; security [4] requires an instant response to hacks.

A. Change point detection methods

There are many classifications of change point detection methods based on their properties. For example, methods are divided into two types by how completely they know the time series:

- *Offline* approach supposes that the data is fully available before processing it.
- *Online* approach does not assume knowledge of the entire data set. Only a part of the data is known at the beginning; new observations are loaded as the method proceeds.

Based on assumptions about the nature of the input data: parametric and nonparametric. *Parametric* methods require making distributional assumptions about the input data. Some of them can extract certain properties from the data at the training stage and change their work mode depending on them. The behavior of the *nonparametric* methods does not depend on the assumptions made about the input data, they are more universal and often less resource-consuming. However, parametric algorithms can demonstrate higher quality in special cases.

Based on the approaches used, for example:

- Bayesian [5] is a parametric online method based on analyzing distributions of run lengths (time elapsed since the last change point was detected) taking into account a priori parameters and input observations.
- Graph-based are offline nonparametric approaches that use graphs and algorithms to analyze them. For example, in [6], [7], a distance graph is constructed based on given observations.
- Classification-based approaches are based on ML algorithms. These approaches are the focus of the current study.

B. Study objectives

Our goal is to study one of the lightweight approaches for the change point detection problem. Namely, classification-based barrier change point detection methods. We want to be able to quickly create a large number of methods that work according to a single scenario. Therefore, we need to create a template method that can be parameterized with various classifiers and other parts used, implement these components, and construct various methods from them. We also want to automatically study the statistical properties of the constructed methods. To do this, we need to create a flexible framework capable of generating the necessary datasets and conducting various measurements on them. The research is being conducted within the algorithmic statistics project PySATL. It is worth noting that we deliberately chose the Python language, focusing not on the effectiveness of the implementations themselves, but on the possibility to quickly do a comprehensive research of the statistical properties of the methods. We also take into account performance of the methods, but we evaluate it not absolutely but relatively to each other. In the end, we want to provide recommendations for each of the methods, defining their regions of applicability.

C. Problem Statement

The *change point* is defined as follows. Consider a sequence of independent random variables $Y = \{Y_1, \ldots, Y_n\}$ that take their values in $\mathbb{R}^d$ $(d \geq 1)$. The sequence is identically distributed as F_0 until a time τ when the distribution changes to F_1:

$$Y_i \sim F_0, \quad i = 1, \ldots, \tau - 1,$$

$$Y_i \sim F_1, \quad i = \tau, \tau + 1, \ldots, n,$$

where $F_0 \neq F_1$. In the case of a detection problem, the task is to say if such τ exists in the input sample. In the case of localization, the task is to specify such a point.

D. Evaluation

Various studies consider different metrics for evaluating and comparing change point detection methods [8], [9]. The most of them are the derivatives of the confusion matrix calculation: power, f-measure, accuracy and others. It also makes sense to evaluate not only the fact of detection itself, but also the speed and error of the detection. This gives rise to another class of metrics, which includes metrics such as the Hausdorff metric, the mean absolute error, and others.

II. Related work

The field related to solving the change point detection problem is actively developing. Review articles significantly simplify the process of entering the field. The articles [9], [10] have a detailed description of the terminology used throughout the subject area and, among other things, consider possible ways to assess the quality of methods and suggest their classification of change point detection methods. The first focuses on the offline methods, the second one provides a general classification.

Also, the authors of the second study suggest their change point detection package `ruptures`[1]. However, it has a number of limitations: there is no functionality for processing time series in parts; it supports only the offline change point detection mode; there is no necessary infrastructure for doing experiments. In addition, there is a well-known `kats` package [11], which has exactly the same limitations. All these gaps are filled in our `pysatl-cpd`[2] package, within which the methods in the current study are implemented and evaluated.

Speaking of approaches that are closer to our subject, we will mention several studies. The [12] is a fundament for the current work: the method it proposes will be implemented and then generalized. The empirical and analytic results obtained in aforementioned research are very valuable, and a comparison will be made with them. The [13] provides nonparametric and parametric methods based on binary segmentation performed over the entire time series. The method thus has a number of limitations: firstly, the method is offline, secondly, it requires a priori knowledge of the number of change points. Due to the scrubbing technique, the method we propose will be able to get rid of the first limitation, and it does not use approaches that impose the second.

[1] https://github.com/deepcharles/ruptures

[2] https://github.com/PySATL/pysatl-cpd

III. Methodology

A. Scrubbing technique

In our work, the assumption is that change points do not occur too often. That is, there is only one change point in the input sample. However, we often deal with huge amount of data with multiple change points. In this case, we can feed the time series into the algorithm in small parts, assuming that there is no more than one change point in each part. The algorithm will quickly process each part, making a decision whether there is a change point or not.

As we have seen earlier (Section I-A), the problem of change point detection can be solved in different contexts: either the entire data set may be known a priori (offline) or only a part of it (online). Depending on the situation, the data can be processed in multiple ways. In the first case, it can be processed using a **sliding window**: a limited number of consecutive observations are considered at a time (Fig. 1). When processing of such interval ends, it shifts further along the time series, capturing new points and forgetting about the same number of old ones.

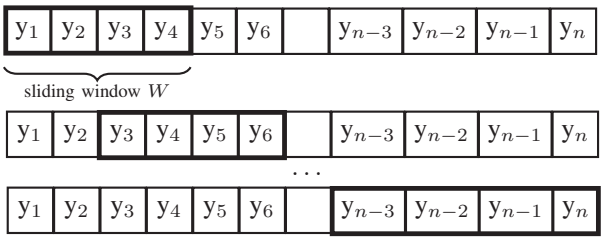

Fig. 1. Window sliding over a sample. A special case of Scrubbing technique

However, in many cases, the entire data set cannot be known from the beginning. In such situations, the time series is being processed using a *scrubbing* algorithm. Unlike a sliding window, the scrubber, if there are not enough observations to analyze, waits until a sufficient number of observations are received, so that they can be analyzed.

The *scrubber* can have several sample processing scenarios: for example, linear processing of available data and linear forgetting of old points is the same as sliding window that differs only in lazy loading of data. This scenario is used in the current study.

The scrubber can be configured to logarithmically forget data and to load new points linearly. In this case, the number of processing points will increase with each step. In addition, the observations considered by the scrubber may not be consecutive: it is possible to take completely different parts of the time series, and so on.

B. General classification-based barrier method

As mentioned earlier, the ideas from the [12] served as the basis for the concept of the generalized method. To be more specific, the two-step approach:

1) Splitting the input sample into two classes by selecting a barrier
2) Assessing the quality of the partitioning using a quality metric.

The concept of the generalized method is that the classifier, as well as the function for assessing classification quality, can be varied. Specifically, KNN, SVM, Decision Tree, Random Forest will be used in our work. As for the quality assessment function, we will try F1, MCC. The pseudocode of the algorithm can be seen in 1.

Algorithm 1 Classifier-Based Barrier CPD Method

1: **Input**: *sample* $y_1, \dots y_n$, *classifier*, *quality_metric*, *threshold*, *left* indent, *right* indent
2: **Output**: Detected *change_points*
3:
4: Split $sample$ into $train\_sample$ and $test\_sample$
5: $change\_points \leftarrow []$
6:
7: **for** $barrier \leftarrow left$ **to** $right$ **do**
8: $classifier.fit(barrier,\ train\_sample)$
9: $classes \leftarrow classifier.predict(test\_sample)$
10: $statistics \leftarrow quality\_metric(barrier,\ classes)$
11:
12: **if** $statistics > threshold$ **then**
13: $change\_points.append(i)$
14: **return** $change\_points$

The algorithm tests each point of the $sample$ for a change. It does not consider observations, that are too close to the boundaries of the sample, because in this case we have a strong imbalance between the number of points on the left and on the right, and therefore accurate analysis is not possible. So, it is reasonable to consider barriers located at some distance from the ends (Fig. 2). In pseudocode 1 these indents are denoted as $left$ and $right$.

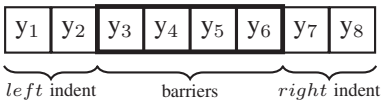

Fig. 2. An example of input sample of size 8. The length of indents is 2. The workspace consists of 4 central elements

The algorithm splits the input sample into $train$ and $test$ parts (Line 4). The algorithm iterates over the central part (workspace) of the input sample. Each element of the workspace is considered as a $barrier$. The $classifier$ is being fitted with the points that are not considered as the barriers (Line 8), and then it predicts the classes of the $test\ sample$ elements (Line 9). The classification is being evaluated using the $quality\ metric$ (Line 10). On line 12 the quality assessment is being compared to the $threshold$. If it is greater than the threshold, then the barrier is a change point. As we see, all parts are parameterizable.

C. Quality Metrics

As for the metrics for assessing the quality of classification, we can choose the usual metrics based on confusion matrix, only in this case negative values will correspond to class 0, positive values to class 1. Three of them were chosen for implementation. As in the case of classifiers, they have their own regions of applicability.

1)

$$Accuracy(W,\ \tau) = \frac{TP + TN}{P + N}$$

is the simplest metric symmetric with respect to classes. Its disadvantage is that in case of unequal number of elements in classes, the influence of smaller class decreases with decreasing number of elements in it. So, in case of a strong quantitative preponderance of one of the classes, the result is very likely to be faulty. It is used when its main property, symmetry, is required.

2)

$$F_1(W,\ \tau) = \frac{2TP}{2TP + FP + FN}$$

is a metric that is non-symmetric with respect to classes. It means it may change its value when labeling is changed. Also, F_1 does not take true negatives into account.

3)

$$MCC(W,\ \tau) = \tag{1}$$

$$\frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{2}$$

like F_1, is non-symmetric, but it takes into account true and false positives and negatives. And it does not depend on the varying number of elements in a class. Each class is assessed relative to the total number of elements in the class. Thus, MCC can be used even if the classes are of very different sizes. In many cases, it can give an informative assessment [14], [15] and can be more truthful than F_1.

D. Classifiers

Since the method is supposed to be used in conditions of having a small amount of resources, only lightweight classifiers were considered: SVM[3], Random Forest[4], Decision Tree[5], KNN [12]. They have a sufficient sensitivity not to miss the change point in a small window. All of them have their individual regions of applicability. One of our goals is to specify the regions.

IV. Implementation

A. Methods

Scikit[6] module supplies all the necessary classifiers (Section III-D). It is actively used in the implementation. As for classifier configurations, the default ones were mostly retained.

[3]https://scikit-learn.org/stable/modules/svm.html
[4]https://scikit-learn.org/1.5/modules/ensemble.html#random-forests
[5]https://scikit-learn.org/1.5/modules/tree.html
[6]https://github.com/scikit-learn/scikit-learn

However, for some of the methods, small experiments have also been made in order to select optimal hyperparameters. Currently, the default configuration of classifiers is the following:

- **KNN**: $k = 7$
- **Decision Tree**: nodes are expanded until all leaves are pure or until all leaves contain less than 2 samples
- **Random Forest**: 100 trees; for a single tree, the configuration is exactly the same as for Decision Tree classifier.
- **Kernel SVM**: linear kernel.

The methods implemented in our change point detection framework have been published on GitHub. It is worth noting that the implementations of SVM, Decision Tree and Random Forest are separate from the Accuracy, F_1, MCC metrics. Therefore, they can be freely combined, and in this case, the resulting methods are the Cartesian product of classifiers and metrics. As for the KNN-based method: we took the scikit KNN classifier and combined it with aforementioned metrics. We also implement an approach [12] that utilizes the classifier and the combinat orial metric monolithically. Their separation is fraught with performance loss.

B. Framework for benchmarking

A flexible framework[7] has been developed to make various measurements. The pipeline is built on the basis of workers in which algorithms are run and the necessary artifacts are saved. This design allows users to easily expand the framework by adding new workers with their own algorithm launch order and storing only the necessary benchmarking information.

An algorithm has been developed to automatically calculate the threshold for the given significance level. The change point detection algorithm runs once, saving statistics for each point of the sample. Next, the optimization problem is solved: the threshold is calculated using binary search, bringing the significance level closer to the required value.

V. Evaluation

The objectives of the study were to evaluate scrubbing classification methods using various statistical and performance measures and to compare them with each other and, where possible, with related studies. For each of the constructed methods, the following tasks were completed:

- Threshold calculation.
- Power analysis based on trivial datasets.
- Performance benchmarking based on synthetic datasets with many change points.

A. Experiment setup

The following methods were constructed from the implemented parts for benchmarking:

- KNN with combinatorial metric (CM).
- KNN with F_1 metric.
- KNN with MCC metric.
- Decision Tree with MCC metric.
- Random Forest with MCC metric.
- SVM with linear kernel and MCC metric.
- SVM with RBF kernel and MCC metric.

In all cases, the default configurations of the classifiers were taken (Section IV-A). The length of the scrubber window was 48 elements. The length of indents on both sides of the window was 12 elements, so the workspace consisted of 24 central elements. These scrubber hyperparameters proved to be optimal in general, but for particular cases and specific methods, they can be selected more individually.

B. Datasets for Monte-Carlo simulation

The purpose of the Monte Carlo simulation was to evaluate the quality of the methods on the same datasets. Each of the datasets consists of 10,000 samples with a size of 200, containing one change point at position 100. Within a single dataset, samples have the same pair of distributions: the first one is before the change point, the second one is after it. The descriptions of the univariate datasets taken can be found in Table I.

We test the hypothesis that there is a change point in the sample. A lot of different distributions were taken; mostly, three situations were considered: shift of deviation, mean shift, and distribution support change.

TABLE I. Univariate distributions

Notation	Type	Parameters
$\mathcal{N}^1$	normal	$\mu = 0.0$, $\sigma = 1.0$
$\mathcal{N}^2$	normal	$\mu = 0.0$, $\sigma = 3.0$
$\mathcal{N}^3$	normal	$\mu = 2.0$, $\sigma = 3.0$
$\mathcal{N}^4$	normal	$\mu = 2.0$, $\sigma = 1.0$
$\mathcal{U}^1$	uniform	$\min = 0.0$, $\max = 1.0$
$\mathcal{U}^2$	uniform	$\min = 0.0$, $\max = 2.0$
$\mathcal{U}^3$	uniform	$\min = -1.0$, $\max = 2.0$
$\mathcal{U}^4$	uniform	$\min = 0.25$, $\max = 1.0$
$\mathcal{U}^5$	uniform	$\min = 1.0$, $\max = 1.25$
$\mathcal{B}^1$	beta	$\alpha = 1.0$, $\beta = 1.0$
$\mathcal{B}^2$	beta	$\alpha = 0.5$, $\beta = 0.5$
$\mathcal{B}^3$	beta	$\alpha = 2.0$, $\beta = 2.0$
$\mathcal{W}^1$	weibull	$shape = 0.5$, $scale = 1.0$
$\mathcal{W}^2$	weibull	$shape = 1.0$, $scale = 1.0$
$\mathcal{W}^3$	weibull	$shape = 2.0$, $scale = 1.0$

Due to the fact that each method has its own region of applicability, a qualitative evaluation based only on univariate datasets is impossible: it is assumed that random forest and decision tree classifiers work well in the context of multivariate distributions, where it is possible to select more than one feature. The description of multivariate distributions can be found in Table II. The distribution before the change point is multivariate standard normal. After the change point, there is a mean shift. The shifts can be found in the Table II. 1000 samples were generated for each of the distributions.

[7]https://github.com/PySATL/pysatl-cpd/tree/scrubbing-methods-bench

TABLE II. MULTIVARIATE DISTRIBUTIONS WITH IDENTITY COVARIANCE MATRIX

Notation	Shift	Dimension
$MN - MN^1$	0.7	10
$MN - MN^2$	1.8	100
$MN - MN^3$	2.7	1000

C. Threshold calculation

Before starting benchmarking, it is necessary to choose the optimal thresholds. To do this, we fix the significance level, generate the datasets without change points, and calculate the threshold at which the significance level will be less than the set values. A standard set of significance levels of 0.5%, 1%, and 5% was taken. The goal was to calculate the optimal threshold values for I datasets; the distributions before the change point were taken; 10,000 samples without a change point were generated based on them. The threshold was calculated individually for each triple algorithm-distribution-significance level. The calculated thresholds can be seen in III.

Comparing the thresholds for the KNN based method calculated in the framework of the current work with the threshold from the original study [12], we can see that they are not very different, there may be a small error due to different significance levels, different distributions and generation randomness.

As for other methods, it is worth noting that for them the thresholds are in the range from -1 to 1; this is a property of the MCC metric. For F_1, the threshold would be in the range from 0 to 1.

D. Power analysis

Let us compare the implemented methods based on univariate datasets using the thresholds shown in Table III. We calculate the statistical power for each pair algorithm-distribution. The results are shown in Table IV. The values are sorted by column 7-NN CM.

Let us start the analysis from the lowest power values. The algorithms detect almost no change point in the data with $\mathcal{B}^1 - \mathcal{B}^2$ distribution. This can be explained by the similarity of the beta distributions with the parameters taken. Their similarity can be tested by calculating the likelihood functions for the sample points. In most cases, the values of likelihood functions for each of the distributions will be very close.

Within the distribution of $\mathcal{U}^1 - \mathcal{U}^4$, the support changes, but the change is not strong enough for the algorithms to detect it.

As for the Weibull distributions, three configurations and their symmetries were used. One trend is visible: algorithms perform worse if the distribution with a heavier tail goes before the change point. In the case of 7-NN, this is most clearly seen: the difference is approximately 7%.

The meaningful comparison of algorithms begins with the distribution $\mathcal{N}^1 - \mathcal{N}^2$. Two facts are worth noting. The first is that 7-NN with combinatorial is the only one who has done a good job of detecting. The second is that changing the core in SVM from Linear to RBF significantly increases the probability of detection. It is easy to explain why Decision Tree performs better than Linear SVM: in the one-dimensional case, DT is able to divide the support into several small intervals, classifying each into one of two classes. Linear SVM divides the whole support into two intervals, and therefore its accuracy is worse.

In the case of $\mathcal{U}^1 - \mathcal{U}^2$ and $\mathcal{U}^1 - \mathcal{U}^3$ a significant support change occurs. Using 7-NN, we see that the more the support changes, the more likely it is to detect a change point. But since expansion in the case of the second distribution occurs in both directions, it becomes more difficult for Linear SVM to find the central point, and the power decreases. The flexibility of RBF in this case greatly corrects the situation and increases the power by 50%. Some visual analysis of the samples plots was made, and it was noted that there may not be much difference in some local area around the change point, if only the deviation changes. In this case, it is reasonable to increase the window size.

As expected, the Random Forest classifier performed better than Decision Tree in all cases. But such a minor improvement is hardly worth the resources spent (Section V-F).

Next, let us move on to the analysis of multivariate distributions. The calculated statistical powers can be found in Table V. The motivation for choosing such distributions was the desire to compare with [12]. The results of statistical benchmarking of our implementation are much better than the results from the article. It turned out that to get closer to an absolutely accurate detection of the change point, it is enough to take $k = 7$. In the article, $k > 10$ was used to achieve a power of 0.8 in much more obvious situations.

If we look at Table V horizontally, we can see that the 7-NN with the combinatorial metric performed the best. Random Forest has significantly improved the results of Decision Tree. RBF SVM has improved the results of Linear SVM.

E. Change of quality metric

We demonstrate the use of various classification quality assessment metrics based on the 7-NN classifier. 7-NN was combined with the combinatorial metric described in [12]. We can experiment with different metrics by combining the implementation of the scikit KNN classifier and the metrics implemented in our project. The results can be seen in Tables IV and V.

Changing the classification quality assessment metric worsened the results. The combinatorial metric is the most appropriate of the metrics used. F_1 performed the worst, missing a change point, where other metrics would have been more likely to find it.

F. Performance analysis

To limit the region of applicability of the methods, an analysis of the spent resources is necessary. We measure the operating time of each of the methods on the same samples. We generate three samples of size 1000 with different number

TABLE III. CALCULATED THRESHOLDS. $\alpha = 5\%$

	CM	F_1	MCC				
Distr.	7-NN	7-NN	7-NN	Lin. SVM	RBF SVM	RF	DT
$\mathcal{N}^1$	3.25	0.859	0.5	0.414	0.453	0.516	0.516
$\mathcal{U}^1$	3.25	0.859	0.5	0.344	0.453	0.516	0.516
$\mathcal{B}^1$	3.25	0.859	0.484	0.344	0.453	0.516	0.516
$\mathcal{W}^1$	3.25	0.859	0.5	0.391	0.406	0.516	0.516
$\mathcal{W}^2$	3.25	0.859	0.5	0.406	0.438	0.516	0.516
$\mathcal{W}^3$	3.25	0.859	0.5	0.375	0.453	0.516	0.516
MN^1	2.813	0.878	0.5	0.531	0.484	0.563	0.57
MN^2	2.625	0.878	0.484	0.52	0.344	0.516	0.578
MN^3	2.625	0.878	0.5	0.486	0.34	0.488	0.578

TABLE IV. POWER. UNIVARIATE DISTRIBUTIONS

	CM	F_1	MCC				
Distribution	7-NN	7-NN	7-NN	RF	DT	Lin. SVM	RBF SVM
$\mathcal{B}^1 - \mathcal{B}^2$	0.1681	0.055	0.128	0.1022	0.1017	0.0587	0.1427
$\mathcal{U}^1 - \mathcal{U}^4$	0.2362	0.066	0.153	0.1361	0.1357	0.1479	0.206
$\mathcal{W}^2 - \mathcal{W}^3$	0.2674	0.075	0.22	0.1474	0.1467	0.1053	0.2479
$\mathcal{W}^1 - \mathcal{W}^2$	0.2871	0.051	0.182	0.1502	0.1499	0.1527	0.2036
$\mathcal{W}^3 - \mathcal{W}^2$	0.3426	0.058	0.147	0.1648	0.1644	0.1446	0.2494
$\mathcal{W}^2 - \mathcal{W}^1$	0.3440	0.073	0.209	0.1572	0.1561	0.1322	0.1527
$\mathcal{N}^1 - \mathcal{N}^2$	0.7858	0.081	0.417	0.3749	0.373	0.2945	0.6974
$\mathcal{W}^1 - \mathcal{W}^3$	0.8761	0.131	0.549	0.4281	0.427	0.2607	0.5427
$\mathcal{U}^1 - \mathcal{U}^2$	0.9003	0.11	0.657	0.4253	0.4227	0.922	0.8617
$\mathcal{W}^3 - \mathcal{W}^1$	0.9057	0.105	0.457	0.4986	0.4962	0.3393	0.5212
$\mathcal{N}^1 - \mathcal{N}^3$	0.9518	0.136	0.796	0.515	0.5134	0.8552	0.8675
$\mathcal{U}^1 - \mathcal{U}^3$	0.963	0.124	0.574	0.6572	0.6549	0.55	0.9503
$\mathcal{N}^1 - \mathcal{N}^4$	0.9976	0.452	0.94	0.7529	0.7516	0.9898	0.9667
$\mathcal{U}^1 - \mathcal{U}^5$	1.0	0.98	1.0	1.0	1.0	1.0	1.0

TABLE V. POWER. MULTIVARIATE DISTRIBUTIONS

	CM	F_1	MCC				
Distribution	7-NN	7-NN	7-NN	RF	DT	Lin. SVM	RBF SVM
$MN - MN^1$	0.994	0.225	0.841	0.799	0.349	0.757	0.976
$MN - MN^2$	1.0	1.0	1.0	1.0	0.986	1.0	1.0
$MN - MN^3$	1.0	1.0	1.0	1.0	0.999	1.0	1.0

of change points. The samples include random combinations of distributions from Table I. We ran each of the algorithms 100 times on each of the samples. We run our benchmarks on a machine running Ubuntu 22.04 with an Intel Core i7-4790 CPU at 3.60 GHz, using `pytest-benchmark` python package. The results can be found in Table VI.

We see that it is not possible to establish a special correlation between the number of change points and the time spent.

The absolute values are not significant because our implementation is not focused on computational efficiency; we consider only the ratios. Comparing Decision Tree and Random Forest-based algorithms, we see that there is almost a 140 times increase in spent time. Therefore, in the one-dimensional case, it is almost pointless to use the Random Forest classifier; we only get a slight improvement in detection quality. KNN-based method with combinatorial metric turned out to be 56

TABLE VI. TIME. MEAN ± STDDEV IN MS

Alg.\CP	5	10	20
DT	668 ± 4	672 ± 1	667 ± 1
L. SVM	754 ± 1	755 ± 3	755 ± 3
7-NN CM	42,175 ± 122	42,268 ± 133	42,336 ± 126
RF	93,895 ± 218	93,961 ± 178	93,845 ± 283

times slower than Linear SVM and 63 times slower than Decision Tree. However, the quality of detection has also increased significantly compared to the other two classifiers, as we have seen in Table IV. Linear SVM performed 1.1 times slower than Decision Tree.

VI. DISCUSSION

A fairly large pool of heterogeneous distributions was supposed to make it possible for all algorithms to show themselves in different conditions. Despite this, in absolutely all cases, 7-NN with combinatorial metric turned out to be statistically better than others. However, it is quite slow.

In terms of Random Forest, experiments have shown that the configuration that is used in the current study is not appropriate for the contexts taken. It takes a long time to detect with very low detection quality.

Choosing between Decision Tree and SVM at the moment, it is worth giving the preference to SVM: its processing time is almost similar to Decision Tree. However, the possibility of using different kernels makes it very flexible. If in some situations Decision Tree shows better results, changing the Linear kernel to RBF will significantly improve the results.

It should be noted that on all the datasets used (except the most obvious $\mathcal{U}^1 - \mathcal{U}^4$), the probability of detecting a change point using Decision Tree was less than 63%. This indicates that the method with its default configuration is insensitive. The SVM results are much better: on some of the distributions, it detects a change point with a probability of more than 75%. Using the RBF kernel significantly improves the results, in some cases, by more than 35%.

In the multidimensional case, the situation repeats itself: 7-NN shows incredibly good results. Kernel SVM performs better than Random Forest and Decision Tree. And changing the SVM kernel from Linear to RBF increases its power even more.

The analysis shows that if a fast method is required, then using the kernel SVM based method is worth considering. However, if the task is not to quickly analyze the data, but to achieve a high detection quality, one should take the sensitive 7-NN based method with combinatorial metric.

The analysis of applying various metrics for the 7-NN classifier implies that different metrics should be tried for other methods as well. Choosing the appropriate metric can greatly improve the results; for example, the SVM-based method in many cases gives better results than 7-NN with the MCC metric (Tables IV and V). A kernel SVM with a good quality metric can be a worthy competitor for 7-NN classifier with a combinatorial metric.

VII. FUTURE WORK

The choice of the optimal metric to assess the quality of the classification remains for the future.

In addition, we need a deeper study of classification-based barrier change point detection methods. It is worth analyzing other properties of the methods:

- Consumed memory
- How window size affects detection quality
- How workspace size inside the window affects detection quality

In the current study, the entire analysis is dedicated to the problem of change point detection. But the methods can also solve the problem of change point localization. In such a problem statement, we can study other properties of the methods, for example, the speed of detecting a change point, i.e. the minimum number of observations following the change point, necessary to detect the change point itself. However, in the case of scrubbing algorithms, this property needs to be reformulated. Scrubbing algorithms analyze one window completely and isolated from other parts of the time series. If there is a change point in the window, then all the elements of the window are used to detect it. Then the speed of detection of a change point is controlled by the size of the window. So, we need to look at how changing the window size affects the quality of detection.

VIII. CONCLUSION

We proposed a generalized classification-based barrier approach to the problem of change point detection. Within this approach, we implemented several methods. We also proposed our approach to the study of such methods. We have created a flexible benchmarking framework that allows us to generate datasets with different distributions, to automatically conduct empirical studies of methods, and to compare methods by their various properties.

More specifically, 7 methods were constructed and evaluated within the same set of different contexts. We have provided some of the results obtained using our benchmarking system in the current paper, but a large number of measurements remain for the future. The results were analyzed, and we made some judgments about the regions of applicability of each of the methods. We also compared our methods with existing ones. It turned out that our implementation statistically does a better job of detecting a change point than the implementation from the original study using nearest neighbors [12]. In addition, it was concluded that there are other classifiers, even more lightweight ones, which, if you choose the right metric, will be able to compete with KNN in the quality of change point detection.

The further development of the benchmarking system, which takes place within the pysatl-cpd project, will make it possible to compare different types of change point detection

methods, such as Bayesian [5], Graph-based [6], [7], and others, implemented in the framework.

ACKNOWLEDGMENT

This work was supported by St. Petersburg State University (Pure ID 116636233).

REFERENCES

[1] H. Takayasu, "Basic methods of change-point detection of financial fluctuations," 06 2015, pp. 1–3.

[2] A. Pepelyshev and A. S. Polunchenko, "Real-time financial surveillance via quickest change-point detection methods," 2015. [Online]. Available: https://arxiv.org/abs/1509.01570

[3] W. Yang, M. Lipsitch, and J. Shaman, "Inference of seasonal and pandemic influenza transmission dynamics," *Proceedings of the National Academy of Sciences*, vol. 112, no. 9, pp. 2723–2728, 2015. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.1415012112

[4] Y. Li, R. Qiu, and S. Jing, "Intrusion detection system using online sequence extreme learning machine (os-elm) in advanced metering infrastructure of smart grid," *PLOS ONE*, vol. 13, p. e0192216, 02 2018.

[5] R. P. Adams and D. J. C. MacKay, "Bayesian online changepoint detection," 2007. [Online]. Available: https://arxiv.org/abs/0710.3742

[6] H. Chen and N. Zhang, "Graph-based change-point detection," *The Annals of Statistics*, vol. 43, no. 1, pp. 139 – 176, 2015. [Online]. Available: https://doi.org/10.1214/14-AOS1269

[7] Y. Zhang and H. Chen, "Graph-based multiple change-point detection," 2021. [Online]. Available: https://arxiv.org/abs/2110.01170

[8] G. Burg and C. Williams, "An evaluation of change point detection algorithms," 03 2020.

[9] C. Truong, L. Oudre, and N. Vayatis, "Selective review of offline change point detection methods," *Signal Processing*, vol. 167, p. 107299, 09 2019.

[10] S. Aminikhanghahi and D. Cook, "A survey of methods for time series change point detection," *Knowledge and Information Systems*, vol. 51, 05 2017.

[11] X. Jiang, S. Srivastava, S. Chatterjee, Y. Yu, J. Handler, P. Zhang, R. Bopardikar, D. Li, Y. Lin, U. Thakore, M. Brundage, G. Holt, C. Komurlu, R. Nagalla, Z. Wang, H. Sun, P. Gao, W. Cheung, J. Gao, Q. Wang, M. Guerard, M. Kazemi, Y. Chen, C. Zhou, S. Lee, N. Laptev, T. Levendovszky, J. Taylor, H. Qian, J. Zhang, A. Shoydokova, T. Singh, C. Zhu, Z. Baz, C. Bergmeir, D. Yu, A. Koylan, K. Jiang, P. Temiyasathit, and E. Yurtbay, "Kats," 3 2022. [Online]. Available: https://github.com/facebookresearch/Kats

[12] H. Chen, "Sequential change-point detection based on nearest neighbors," *The Annals of Statistics*, vol. 47, no. 3, pp. 1381 – 1407, 2019. [Online]. Available: https://doi.org/10.1214/18-AOS1718

[13] M. Londschien, P. Bühlmann, and S. Kovács, "Random forests for change point detection," 05 2022.

[14] D. Chicco and G. Jurman, "The matthews correlation coefficient (mcc) should replace the roc auc as the standard metric for assessing binary classification," *BioData Mining*, vol. 16, 02 2023.

[15] D. Chicco, "Ten quick tips for machine learning in computational biology," *BioData Mining*, vol. 10, 12 2017.