Unsupervised Offline Changepoint Detection Ensembles

Théo VINCENT theo.vincent@eleves.fr

March 30, 2022

1 Introduction and contributions

This report is focusing on the work of I. Katser et al. [1]. Classical changepoint detection algorithms are using a single cost function to predict the changepoints. In this paper, the idea is to build an ensemble method on top of several cost functions. A code allowing the user to test the developped methods is coming along the paper. It is available at: `https://github.com/YKatser/CPDE`.

In the report, a short and concise overview of the method introduced in the article of I. Katser et al. is proposed. A tutorial explaining the method will also be added to the library **ruptures** [5]. Finally, a bug-free/fault-free version of the original code is released at `https://github.com/theovincent/CPDE` in which additionnal experiments have been added. Two notebooks summarising the experiments, are also in the code. They can be launched on Google Colab with a simple click. The experiments are done on two benchmarks (Tennessee Eastman Process (TEP) [4], Skoltech Anomaly Benchmark (SKAB) [2]) composed of data coming from the industry.

2 Method

A changepoint detection algorithm is composed of a *cost function* and a *search method*. It takes as input a time series and outputs the timesteps on which it thinks the statistical properties of the input have changed. A cost function takes as input a portion of the time series and outputs the cost of considering that all element of that portion have the same statistical properties. The search method queries the cost function to decide on which timesteps the input signal should be split to reduce the overall cost. The choice of the search method has to be decided according to the computational power that is available. Indeed, the exact mathematical problem can be solved by dynamic programming at the cost of a long time run. Other methods exist to reduce the time complexity. For example, binary segmentation produces sub-optimal results for a lower complexity. At the other end, a window search produces results that are even more sub-optimal but has a really competitive complexity. The choice of the cost function is a lot more complex since the statistical properties of the signal can be defined in several ways. Indeed, some cost are monitoring the change in mean (l2 cost) and others are trying to see if an element can be predicted from the previous ones (autoregressive cost). This is why the choice of the cost function has to be adapted to the signal to be segmented. The best cost function is sometimes hard to guess and this is problematic because the performances of the algorithm will suffer from a bad choice.

I. Katser et al. propose an approach where this choice does not have to be done by the user. Indeed, they build an ensemble method that is taking care of that choice. They provide an algorithm for three different search methods: window search, binary segmentation and dynamic programming. The main idea is to use the search methods as they normally work but this time the cost function is

an aggregation of several cost functions. The aggregation step is performed with a *scaling function* and an *aggregation function*. The question is then, when to perform the aggregation step? In each search method, there is a step where the changepoints are decided. For example, in the window search method, the last step is to select the peaks of the score. It is before this step that the different scores coming from the different cost functions are aggregated. The fact that the cost functions do not have to be chosen is a real incentive. Nevertheless, this algorithm forces the user to choose a scaling function and an aggregation function.

The three search methods that are studied in the paper are giving scores that are of different nature. Indeed, for the window search method, the higher the score for a timestep is, the higher the probability of this timestep of being a changepoint is. The same works for the binary segmentation. On the contrary, for the dynamique programming search method, it is the opposite: if the score is high, it means that the timestep has low probability of being a changepoint. To account for that difference, the opposite of the score is taken into account for the window search method and the binary segmentation search method. In the following, we will consider that it is the case so that we can only speak about search methods and avoid distinguishing the cases.

Since the search methods are always looking at minima (thanks to the "trick" presented in the last paragraph), it is as if every cost function is voting for changepoints. The scaling and aggregation function will decide on the rules of this voting process. In the paper, several possibilities are presented. The scaling function *MinMax* is interesting because it scales the scores to be between 0 and -1 with the maximum of the score being 0 and the minimum being -1. Indeed, the function can be written for a time series (s_t) as the following: $\text{MinMax}(s)_t \mapsto \frac{s_t - \max_i s_i}{\max_i s_i - \min_i s_i}$. By taking the aggregation function Min over the scaled costs, i.e for N time series $(s_t^i)_{i\in\{1,N\}}$, $\text{Min}(s)_t = \min_{i\in\{1,N\}} s_t^i$, all the cost functions vote equally. The result is close to the union of all the predicted changepoints. This combinaison of scaling and aggregation function is interesting when all the cost are designed to detect different types of changepoints. On the contrary, with the aggregation function Max over the scaled costs, i.e for N time series $(s_t^i)_{i\in\{1,N\}}$, $\text{Max}(s)_t = \max_{i\in\{1,N\}} s_t^i$, the intersection of the predicted changepoints is taken as the result of the ensemble model. Finally, the aggregation function WeightedSum is a happy medium, it is defined for N time series $(s_t^i)_{i\in\{1,N\}}$, $\text{WeightedSum}(s)_t = \sum_{i\in\{1,N\}} \lambda^i s_t^i$ where $\lambda^i = \frac{\max_t s_t^i - \min_t s_t^i}{\mu_t s_t^i - \min_t s_t^i}$ with μ is the empirical average. This aggregation function allow the cost functions not to vote equally, in deed, the cost functions with a peaking score will have a high λ^i. Thus, those cost functions will be more taken into account in the voting process.

A critic about this method can already been done even before seeing the results. Indeed, we have seen that for the aggregation functions Min and Max, all the cost functions have equal power during the voting process. This means that if a cost function is not performing well, this would harm the overall result.

In the paper, the number of changepoints is assumed to be known in advance.

3 Data

3.1 Tennessee Eastman Process (TEP) [4]

This dataset, recorded in 1993, is made thanks to sensors on real industrial engines. In the problem studied in this report, 21 signals of 50 variables can be used. Indeed, the method is unsupervised so the training set is of no use. For the testing signals, the sensors have been recorded every 3 min-

utes during 48 hours. A fault has been introduced 8 hours after the beginning of the recordings. This means that this dataset is composed of time series with only one changepoint.

Analysis: The fact that this dataset contains only one changepoint brings the dynamic programming and the binary segmention search methods to be equal. A more problematic property of this dataset is the fact that all the single changepoints are happening at the beginning of the time series. This is a problem because in the method, the predicted changepoint is the result of an argmin or an argmax function. In Python, the usual convention is to output the argument of the first minimum or maximum. This introduced a bias in the case where the extremum is reached several times. In our case, it happens, for example, when the scaling function MinMax is chosen with the aggregation function Min.

3.2 SKoltech Anomaly Benchmark (SKAB) [2]

In order to evaluate the presented method on time series containing more than one changepoint, the authors chose to use SKAB dataset. SKAB has been crafted by the same authors. It is composed of 34 signals of 8 variables. The signals are also the result of an industrial process monitored by censors. Each signal contains between 2 and 4 changepoints.

Metric: The usual metric used for this dataset is called the Numenta Anomaly Benchmark (NAB) [3]. This metric is a little complex to explain in details. The main idea behind it is that it is better to predict a changepoint a bit ealier than the true timestep, than predicting it latter. It provides the user with three scores: standard, low False Positive (FP), and low False Negative (FN). The standard score takes into the four possible outcomes and the latter two give more importance to FP and FN respectively.

4 Code enhancement

By going deeper into the details of the method and the code, I encountered four mistakes that I reported as issues in the original GitHub repository. Here is the list of the spotted issues:

- Mistake: Cost New https://github.com/YKatser/CPDE/issues/8
- Mistake: Cost functions - Linear VS RBF https://github.com/YKatser/CPDE/issues/9
- Mistake: ar(1) vs ar(5) in Win https://github.com/YKatser/CPDE/issues/10
- Implementation details: (Win Dynp) vs Binseg https://github.com/YKatser/CPDE/issues/11

Besides fixing these issues in a forked repository, a particular attention was taken to make the code readable, reliable and easy to use. Indeed, Black formatting was used. The proposed implementation relies more on the source code of **ruptures**. For example, each of the ensemble method is a subclass of the non-ensemble search method in **ruptures**. Finally, the experiments can be run in a single click on Google Colab.

5 Results

All the results of the original paper have been ran again with the modifications described in section 4. Tables 1 for TEP and 2 for SKAB are presenting the outcomes of the experiments. The lines "ensemble bounds" have been added. There are the average of the best score among the single cost methods for each signal. This is why there are called "ensemble bound". It is clear that they are far above the best achieved results by the ensemble models. This shows that the idea of crafting an ensemble model that is better than all the single cost for any signal is far from being simple. Indeed, this results can be understood with inequalities. Let Y be the score of an ensemble method. Let R_i be the scores of the single costs methods. A first inequality is the following:

$$\mathbb{E}[\max_i R_i] \geq \max_i(\mathbb{E}[]R_i]) \tag{1}$$

Here, the left term corresponds to the ensemble bound and the right term is the bold number in the table of results for the single costs. By looking at the tables, we can see that this inequality is strict which is a good sign because it means that there is not only one cost that is better than the others. SKAB dataset is the one having more than one changepoint. Table 2 shows the following inequality for the search method Binseg and Dynp:

$$\max_i(\mathbb{E}[R_i]) \geq \mathbb{E}[Y] \tag{2}$$

By putting equation 1 and 2 together, we get $\mathbb{E}[\max_i R_i] \geq \mathbb{E}[Y]$ which means that the idea of crafting an ensemble model that is better than all the single cost for any signal is not happening in practise.

In the tables 1 and 2, it is noticable that all the scores have changed compared to the original paper except for the single cost experiments since no isssue was found for that part. There is no clear aggregation and scaling function that is working better than an other. This means that this choice might depend and the dataset. An interesting idea would be to allow the algorithm to have some signals with annotations to train on. The theory of multi-armed bandit with best arm identification might be of great use here. This would allow the ensemble model to avoid selecting some cost function that are actually harming the overall result. A further study could lean on finding the best convex combination of scores. The field of sequential learning provides theoretical bounds.

References

[1] Iurii Katser, Viacheslav Kozitsin, Victor Lobachev, and Ivan Maksimov. Unsupervised offline changepoint detection ensembles. *Applied Sciences*, 11(9):4280, 2021.

[2] Iurii D. Katser and Vyacheslav O. Kozitsin. Skoltech anomaly benchmark (skab). https://www.kaggle.com/dsv/1693952, 2020.

[3] Alexander Lavin and Subutai Ahmad. Evaluating real-time anomaly detection algorithms - the numenta anomaly benchmark. *CoRR*, abs/1510.03336, 2015.

[4] Cory A. Rieth, Ben D. Amsel, Randy Tran, and Maia B. Cook. Additional Tennessee Eastman Process Simulation Data for Anomaly Detection Evaluation, 2017.

[5] Charles Truong, Laurent Oudre, and Nicolas Vayatis. Selective review of offline change point detection methods. *Signal Processing*, 167:107299, 2020.

model		Win	Binseg	Dynp
ar(1)		21.27	30.15	30.15
mahalabonis		27.29	**36.88**	**36.88**
l1		20.63	32.53	32.53
l2		22.09	30.30	30.30
rbf		**28.96**	22.85	22.85
Ensemble bound		40.13	42.28	42.28
Aggregation	Scaling			
Min	Raw	26.35	32.60	22.85
	MinMax	21.83	32.60	32.60
	Znorm	28.31	30.06	27.70
	MinAbs	28.96	4.50	22.85
	Rank	16.34	**41.79**	**41.79**
Sum	Raw	27.28	30.20	30.20
	MinMax	27.28	32.14	32.50
	Znorm	27.28	32.14	32.50
	MinAbs	27.28	32.14	30.20
	Rank	32.04	20.55	29.99
WeightedSum	Raw	27.28	30.20	30.20
	MinMax	27.28	34.11	32.50
	Znorm	22.76	1.62	32.50
	MinAbs	27.28	34.11	30.20
	Rank	32.04	13.47	22.56
Max	Raw	28.96	4.50	37.29
	MinMax	32.04	20.58	26.17
	Znorm	**36.47**	6.72	28.03
	MinAbs	32.04	18.30	31.97
	Rank	21.78	10.85	20.68

Table 1: Standard NAB scores on TEP dataset

model		Win	Binseg*	Dynp*
ar(1)		15.54	21.39	19.40
mahalabonis		15.55	**24.10**	**22.37**
l1		**18.40**	17.87	18.64
l2		14.78	17.46	18.96
rbf		16.21	20.86	19.33
Ensemble bound		29.37	30.35	28.34
Aggregation	Scaling			
Min	Raw	15.89	19.33	19.33
	MinMax	17.90	19.33	16.43
	Znorm	17.41	14.28	19.71
	MinAbs	**19.61**	7.05	19.33
	Rank	15.98	-0.72	19.23
Sum	Raw	14.85	20.71	19.86
	MinMax	15.98	15.86	19.40
	Znorm	17.04	12.07	20.94
	MinAbs	16.71	15.90	19.90
	Rank	15.10	15.73	21.62
WeightedSum	Raw	16.48	19.82	19.01
	MinMax	16.80	12.28	19.40
	Znorm	16.32	11.44	20.94
	MinAbs	17.56	11.81	19.90
	Rank	15.91	15.73	21.62
Max	Raw	16.21	**20.86**	**21.64**
	MinMax	12.87	15.61	20.64
	Znorm	14.78	18.20	19.40
	MinAbs	12.87	15.70	19.99
	Rank	16.30	16.99	21.14

*the admissible changepoints were considered every 5 timesteps instead of every 1 timestep to speed up training.

Table 2: Standard NAB scores on SKAB dataset.