Changepoint detection

Inference and applications

Etienne Caprioli
etienne.caprioli23@imperial.ac.uk
Imperial College London

June 11, 2024

Outline of this presentation

Inference

- Problem setting
  - → Statistical test
  - → Online/Offline inference
- What are we interested in ?
  - → Single changepoint
  - → Multiple changepoints
- Bayesian approach

Applications

- Application in market risk monitoring
- Other areas of applications
- Useful libraries, packages and repositories

Inference

Problem setting

Piecewise homogeneous sequences

Let $(X_i)_i$ be a sequence of random variables. We model this signal with the equation

$$X_t = \sum_{j=0}^{q} X_t^{(j)} \, 1_{\tau_j \leq t \leq \tau_{j+1}}$$

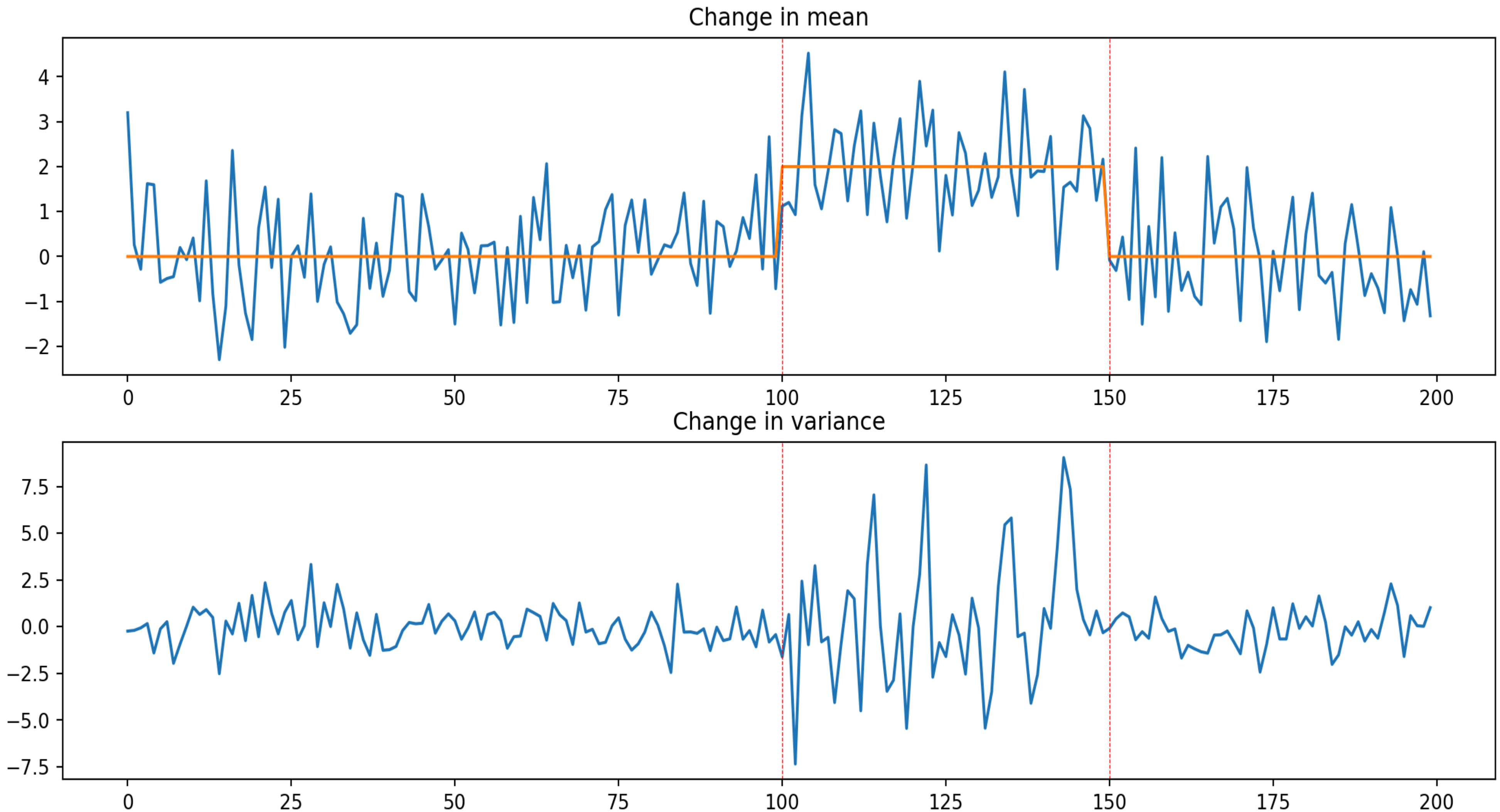
Where each $(X_t^{(j)})_j$ can be modeled individually. We are interested in inferring the *changepoints* $(\tau_j)_j$ and the j *joint distributions* of $(X_t^{(j)})_j$.

Note

The collections $(X_t^{(j)})_j$ can be modeled in very different ways: IID sequences, (non-) stationary time series, IID + trend/seasonality…

Problem setting

Example of datasets



Problem setting

Statistical test

We will adopt here the most used framework of change in mean, and assume that the variance σ^2 is known (or estimated).

Let q be the assumed number of changepoints in the dataset, we are interested in testing:

$$H_0 : q = 0$$
$$H_1 : q = m$$

To perform this test, we need a *test statistic* to be compared to a certain *threshold*. We will focus here on the likelihood ratio (LR) statistic evaluated at some time point τ:

$$LR_\tau = \frac{1}{\sigma^2}\left[\sum_{i=1}^{n}(X_i - \bar{X}_{1:n})^2 - \sum_{i=1}^{n}(X_i - \bar{X}_{1:\tau})^2 - \sum_{i=\tau+1}^{n}(X_i - \bar{X}_{\tau+1:n})^2\right]$$

Problem setting

Statistical test

CUSUM statistic

If we define C_τ such that

$$C_\tau = \sqrt{\frac{\tau(n-\tau)}{n}}\left|\bar{X}_{1:\tau} - \bar{X}_{\tau+1:n}\right|$$

We note that $LR_\tau = \frac{C_\tau^2}{\sigma^2}$

Problem setting

Online/Offline inference

Online inference

The dataset keeps growing through time. We have a continous flow of of datapoints coming. The goal is still to estimate the localisation of the changepoint, but usually the quantity of data *after* the changepoint is limited. The data is analysed by flow.

> **Example**
>
> Seismographs continuously monitors the seismic activity in a given zone, stock market quotes are continuously updated.

Offline inference

The size of the dataset is fixed, the aim is still to detect the changepoints, but this time we have access to possibly much more data after the changepoint.

> **Example**

What are we interested in ?

Single changepoint

In some settings we know that there is at most one changepoint in the dataset. In this setting, the inference of the changepoint is well documented and an essentially complete theory exist.

We use the LR statistic and set a threshold $c > 0$. If the threshold is exceeded, than we conclude that there is a changepoint in the data and estimate this changepoint τ with :

$$\hat{\tau} = \underset{\tau \in \{1,...,n-1\}}{\text{argmax}} \; LR_\tau = \underset{\tau \in \{1,...,n-1\}}{\text{argmax}} \; \frac{C_\tau^2}{\sigma^2}$$

How to choose the threshold c?

Some theoretical results that ensure that the false positive rate tends to 0 as $n \to \infty$ suggest that the threshold should be set to at least $c = 2 \log n$. However, this threshold can appear be to be quite conservative in some applications, and some papers suggest to use $c = 2 \log \log n$ instead.

What are we interested in ?

Multiple changepoints

- Unfortunately this framework becomes less efficient when there are several changepoints.
  - $\rightarrow$ The presence of other changepoints affects the value of the CUSUM statistics, pushing it sometimes below the threshold even when there is an actual changepoint
  - $\rightarrow$ This framework doesn't help to estimate the number of changepoints

The penalised cost function

To deal with the possibility of multiple changepoints, we define the notion of cost function $f(\tau_1, \dots, \tau_q | X_1, \dots, X_n)$, which is a function that we aim to minimize in order to find the changepoints. One example of this function is simply the negative log-likelihood if we have a model for IID samples (e.g. Gaussian IID samples):

$$f(\tau_1, \dots, \tau_q | X_1, \dots, X_n) = \frac{n}{2} \log \sigma^2 + \frac{\sum_{j=1}^{q+1} \sum_{i=\tau_{j-1}+1}^{\tau_j} (X_i - \bar{X}_{\tau_{j-1}+1:\tau_j})^2}{2\sigma^2}$$

And aim to minimise the penalised cost:

$$\min_{\tau_1, \dots, \tau_q} f(\tau_1, \dots, \tau_q | X_1, \dots, X_n) + \beta(q)$$

What are we interested in ?

Multiple changepoints

Example of penalisations

- BIC : $\beta(q) = q\log(n)$
- AIC : $\beta(q) = 2q$

Each type of penalisation has its own pros/cons and theoretical results. For instance, under some regularity condition, the BIC penalisation ensures that the infered number of changepoints $\hat{q}$ converges to the true number of changepoints q^*.

What are we interested in ?

Multiple changepoints

Another idea estimates multiple changepoints is to use a recursive approach. Since we have an efficient method to detect one changepoint, why not use this method iteratively on a succession of intervals ?

Binary segmentation algorithm

BS algorithm

- Find $\hat{\tau}_1 = \underset{\tau \in \{1,...,n-1\}}{\operatorname{argmax}} \; C_\tau$
  - $\rightarrow$ If $C_{\hat{\tau}_1} > c_n$ then keep $\hat{\tau}_1$ in memory and run the first step again on the intervals $[1, \hat{\tau}_1]$ and $[\hat{\tau}_1 + 1, n]$.
  - $\rightarrow$ If $C_{\hat{\tau}_1} \leq c_n$ then stop the algorithm and return the list of the kept changepoints.

Or

- Find $\hat{\tau}_1 = \underset{\tau \in \{1,...,n-1\}}{\operatorname{argmax}} \; LR_\tau$
- Run again the first step again on the intervals $[1, \hat{\tau}_1]$ and $[\hat{\tau}_1 + 1, n]$, until the length of the interval is less then 2.
- Select the best changepoints collection using the penalised cost method.

What are we interested in ?

Binary segementation algorithm

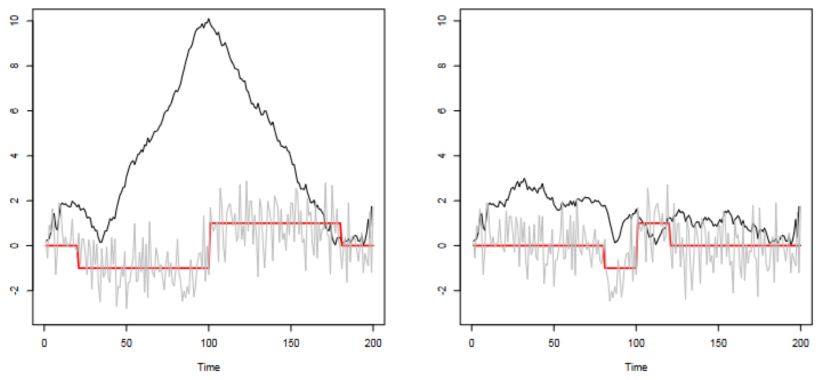
Notes

A good algorithm

- Easy to implement (recusrive algorithm)
- Computationally efficient (complexity in $\mathcal{O}(n \log n)$)

But

- Relies on the ability of the CUSUM statistic to detect one changepoint among possibly several changepoints...
- Need to select the right threshold c_n in the first version of the algorithm...



What are we interested in ?

Multiple changepoints

Wild binary segmentation (WBS) algorithm

To try to prevent the problem of ending up working in an interval that contains several changepoints and disturb the changepoint estimation, the Wild Binary Segmentation algorithm extend the BS algorithm to M random sub-intervals $[s_m, e_m] \subset [1, n]$ and only keeping the highest CUSUM statistic among the batch of CUSUM statistic calculated. This way, we hope that at least one of the random sub-interval only contains one true changepoint.

BS algorithm

- Draw M sub-intervals of $[1, n]$ and let $\mathcal{M}$ be the set of the indices m such that $[s_m, e_m] \subset [1, n]$.
- Compute $m_0, \hat{\tau}_1 = \underset{m \in \mathcal{M},\, \tau \in [s_m, e_m]}{\operatorname{argmax}} C_\tau(X_{s_m:e_m})$

  $\rightarrow$ If $LR_{\hat{\tau}_1} > c_n$ then keep $\hat{\tau}_1$ in memory and run the first step again on the intervals $[1, \hat{\tau}_1]$ and $[\hat{\tau}_1 + 1, n]$.

  $\rightarrow$ If $LR_{\hat{\tau}_1} \leq c_n$ then stop the algorithm and return the list of the kept changepoints.

Or

- Draw M sub-intervals of $[1, n]$ and let $\mathcal{M}$ be the set of the indices m such that $[s_m, e_m] \subset [1, n]$.
- Compute $m_0, \hat{\tau}_1 = \underset{m \in \mathcal{M},\, \tau \in [s_m, e_m]}{\operatorname{argmax}} C_\tau(X_{s_m:e_m})$
- Run again the first step again on the intervals $[1, \hat{\tau}_1]$ and $[\hat{\tau}_1 + 1, n]$, until the length of the interval is less then 2.
- Select the best changepoints collection using the penalised cost method.

What are we interested in ?

Multiple changepoints

Wild binary segmentation (WBS) algorithm

Notes

A good algorithm

- Solve the initial problem of perturbation of the CUSUM statistics when there are several changepoints
- Computationally sill quite efficient

But

- The computational cost increases with M.
- Need to select the right threshold c_n in the first version of the algorithm...

What are we interested in ?

Multiple changepoints

Optimal partitionning

Let us come back to

$$\min_{\tau_1,...,\tau_q} f(\tau_1, \ldots, \tau_q | X_1, \ldots, X_n) + \beta(q)$$

This minimisation problem greatly simplifies if we assume that the criteria f can be broken down into a sum of costs over several segments:

$$f(\tau_1, \ldots, \tau_q | X_1, \ldots, X_n) = \sum_{j=1}^{q+1} c_{\tau_{j-1},\tau_j}(X_{1:n})$$

Let us assume that $\beta(q) = \lambda q$ where λ is a constant. We can therefore write the minimisation problem above as

$$Q_{n,\lambda}(q;\tau_1,\ldots,\tau_q) = \sum_{j=1}^{q+1} c_{\tau_{j-1},\tau_j}(X_{1:n}) + q\lambda$$

What are we interested in ?

Multiple changepoints

Optimal partitionning

Now denote

$$Q_{t,\lambda} = \min_{q;\ \tau_1 < \dots < \tau_q < t} \sum_{j=1}^{q} c_{\tau_{j-1},\tau_j}(X_{1:n}) + c_{\tau_q,t}(X_{1:n}) + q\lambda$$

This can be interpreted as the minimum segmentation cost of the time series between times 0 and t. It can be be rewritten:

$$Q_{t,\lambda} = \min \left\{ c_{0,t}(X_{1:n}), \min_{\tau=1,\dots,t-1} Q_{\tau,\lambda} + c_{\tau,t}(X_{1:n}) + \lambda \right\}$$

Which simplifies further if we set $Q_{0,\lambda} = -\lambda$ (this value is arbitrary as it represents the minimum segmenting cost of the series between 0 and 0):

What are we interested in ?

Multiple changepoints

Optimal partitionning

This last expression has a recursive structure that one can exploit to solve this minimisation problem and estimate the changepoints $(\hat{\tau}_j)_j$ and their number $\hat{q}$:

$$\hat{\tau}_1 = \underset{\tau=0,...,n-1}{\mathbf{argmin}}\; Q_{\tau,\lambda} + c_{\tau,n}(X_{1:n}) + \lambda$$

And then

$$\hat{\tau}_{j+1} = \underset{\tau=0,...,\hat{\tau}_j-1}{\mathbf{argmin}}\; Q_{\tau,\lambda} + c_{\tau,\hat{\tau}_j}(X_{1:n}) + \lambda$$

We recursively computes the $(\hat{\tau}_j)_j$ until we fine $\hat{\tau} = 0$.

What are we interested in ?

Multiple changepoints

Pruned exact linear time (PELT) algorithm

The problem of the optimal partitioning is its computational cost: $\mathcal{O}(n^2)$ (which can become prohibitive in some applications). The PELT algorithm has been introduced to reduce the computational cost of optimal partitioning by adding a *pruning* rule that prunes parts of the search space that are deemed to have a higher segmentation cost.

Indeed if at some point τ we have:

$$Q_{\tau,\lambda} + c_{\tau+1:t}(X_{1:n}) + a > Q_t, \quad \textbf{for some } a > 0$$

Then τ will never be an acceptable changepoint, and can thus be eliminated from the search space.

Complexity of the PELT algorithm

Bayesian approach

Bayesian online changepoint detection

This method, based on Adams and MacKay (2007), adopts a Bayesian approach to compute the posterior probability distribution of the “run length” r_t (i.e. the time after the last change point):

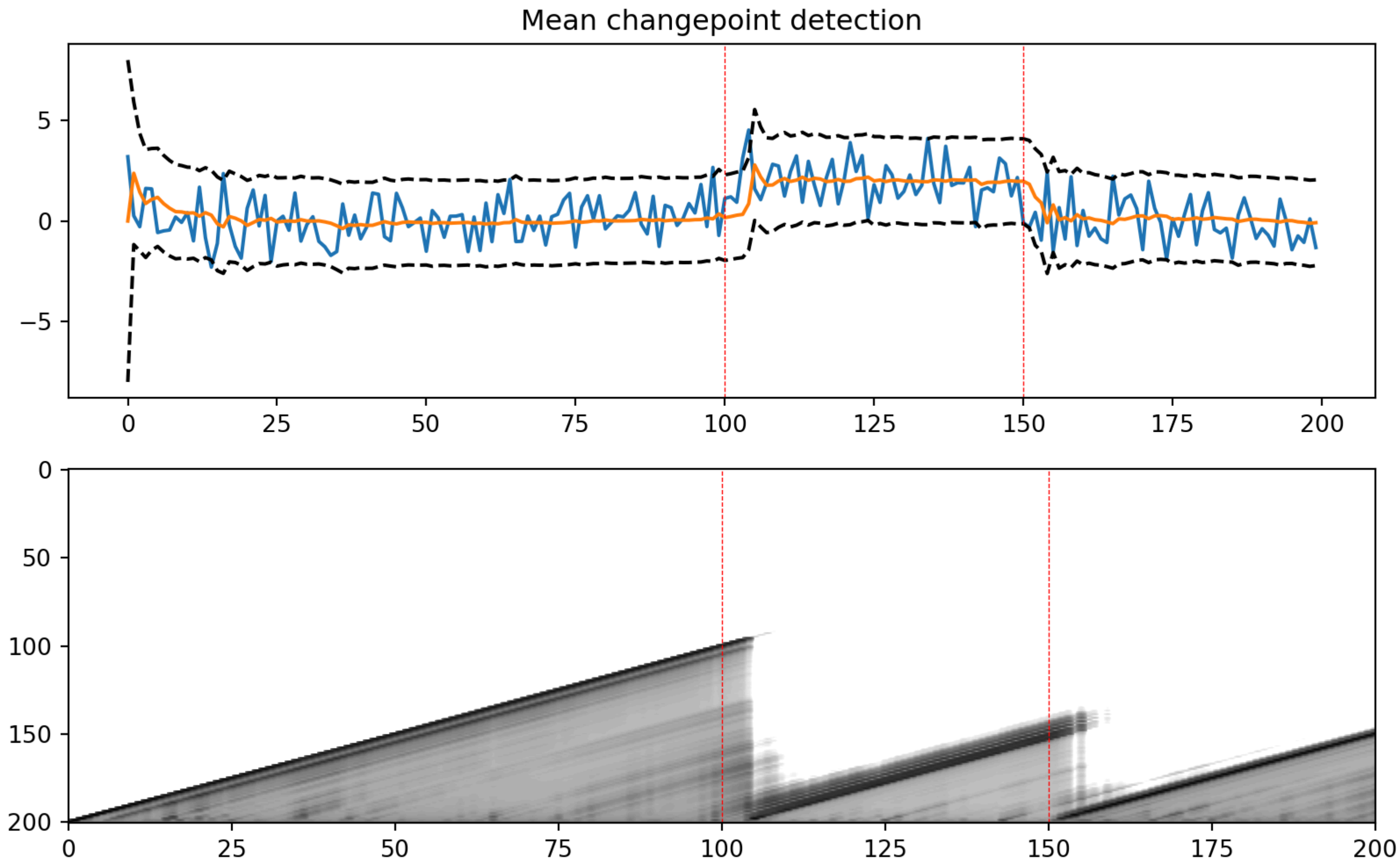
$$r_t = \begin{cases} 0, & \textbf{if } t \textbf{ is a changepoint} \\ r_{t-1} + 1, & \textbf{otherwise} \end{cases}$$

The idea is to assign a prior to the “hazard rate” (i.e. the frequency at which the changepoints occur) and use the exponential family posterior predictive closed formula to compute the posterior probability distribution of r_t at every t:

$$p(r_t | X_{1:t}) = \frac{p(r_t, X_{1:t})}{\sum_{r_{t'}} p(r_{t'}, X_{1:t})}$$

Bayesian approach

Bayesian online changepoint detection

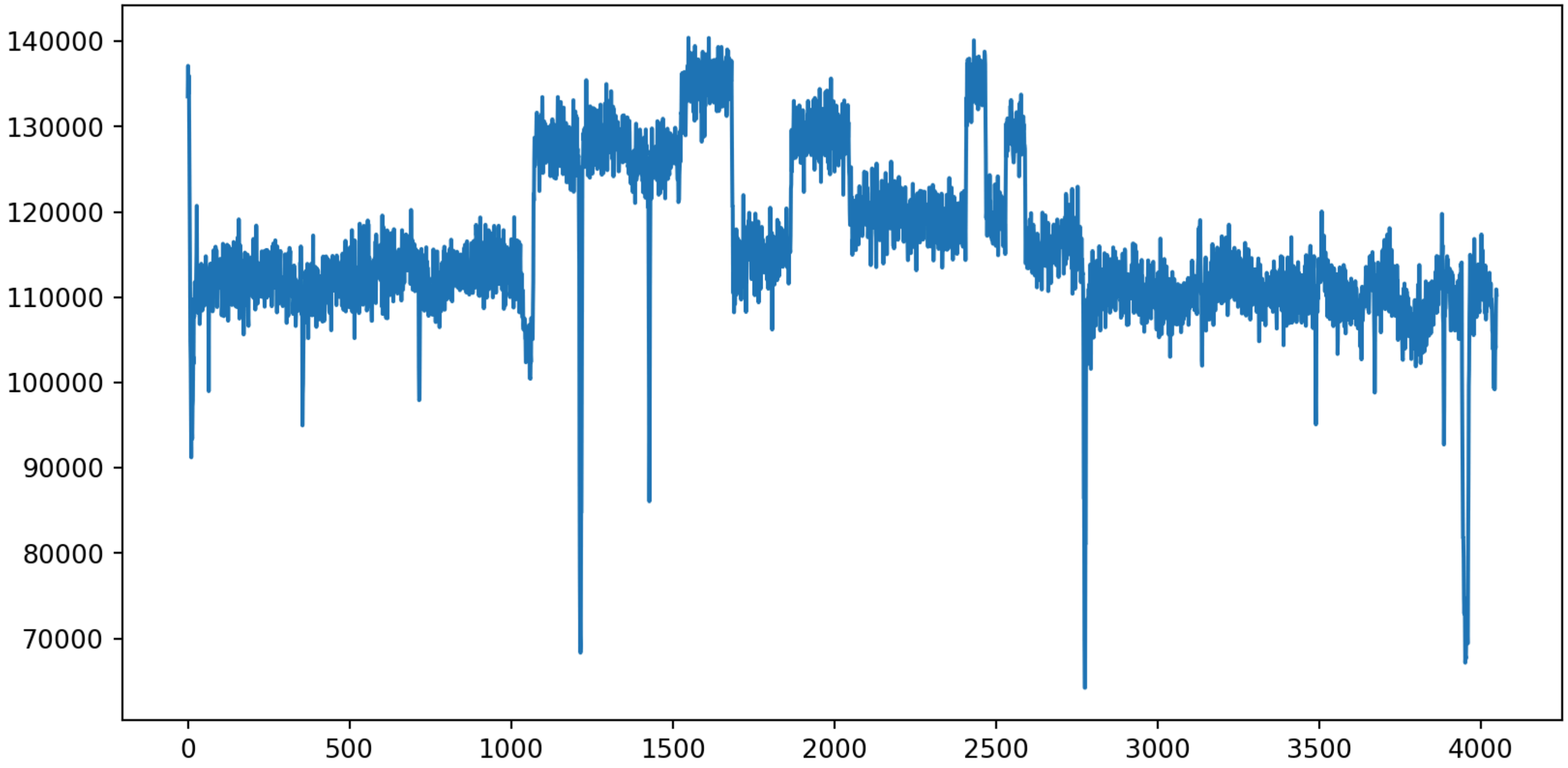


Example of mean changepoint detection with the BOCD algorithm. The bottom plot represent the posterior probability distribution of r_t.

Applications

Canonical datasets

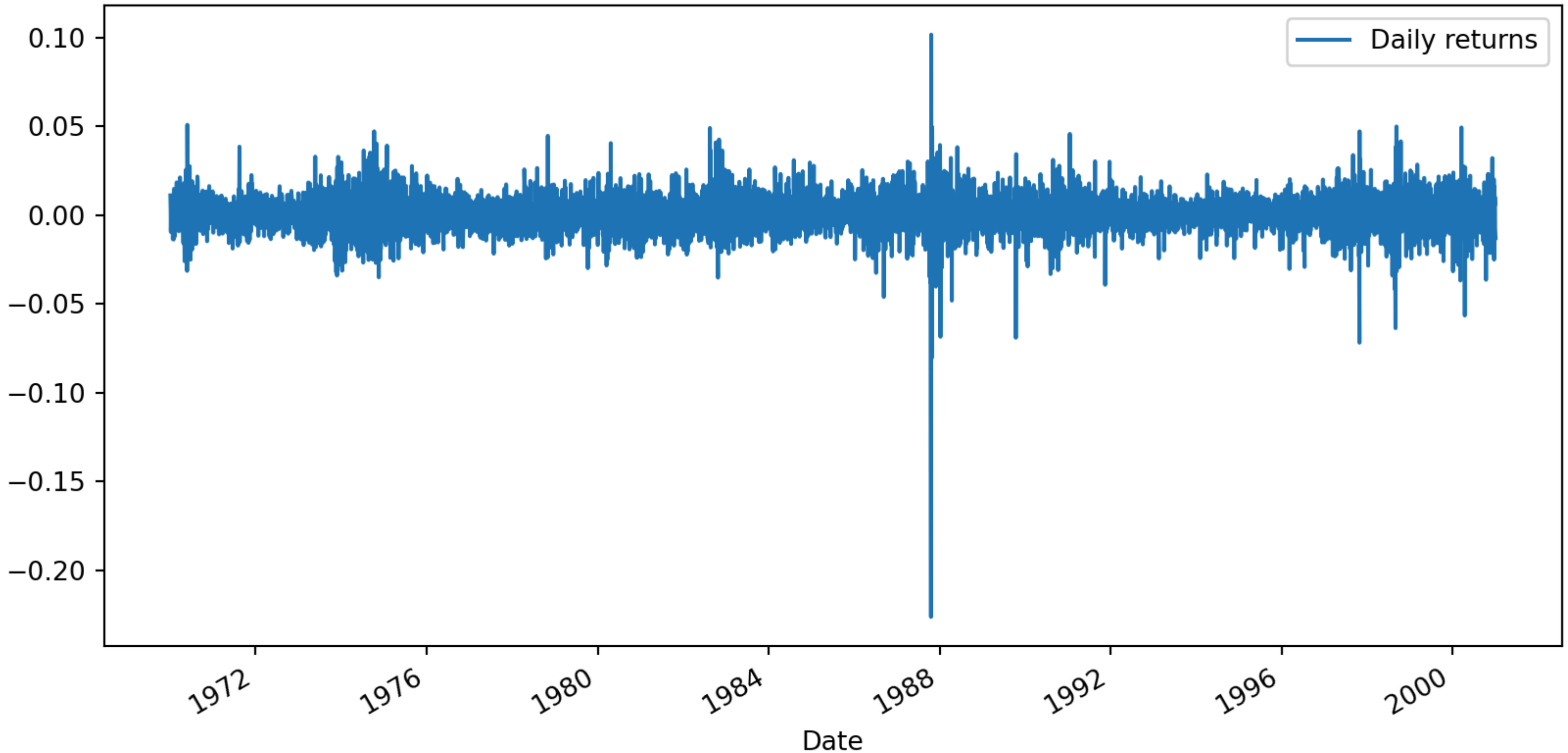
Well log



Material response vs depth during an oil well drilling. This type of data is used to analyse the soil and determine what is the best depth and orientation for the forage.

Canonical datasets

Dow Jones Industrial Average

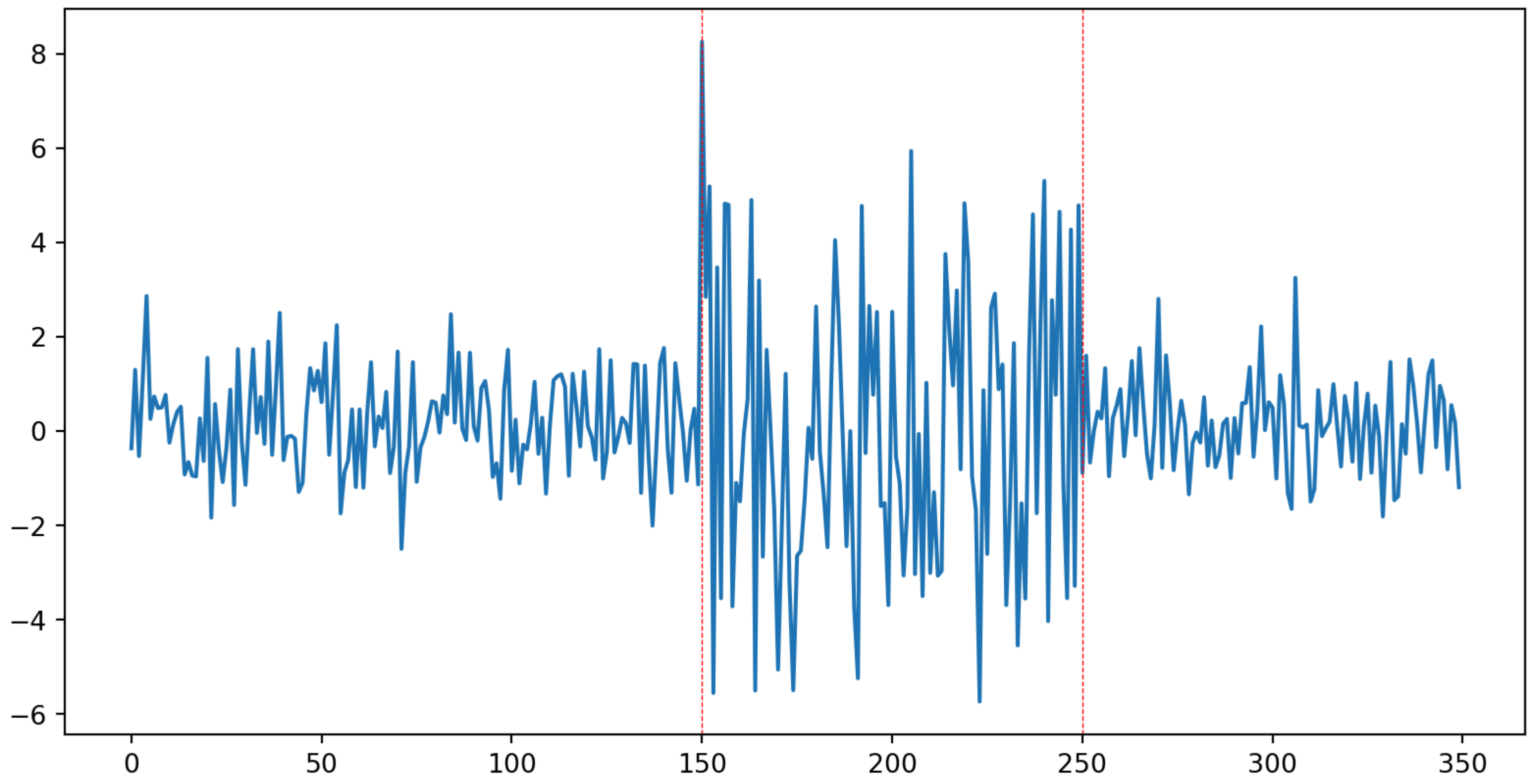


Daily returns of the Dow Jones Industrial Average index between 1970 and 2000. This can be one building block for market risk monitoring.

Application in risk monitoring in Finance

Dataset

To monitor financial markets and potentially take action to protect financial stability, trading venues monitor the volatility of the different stock index, to do so, they need some sort of *online* changepoint detection.



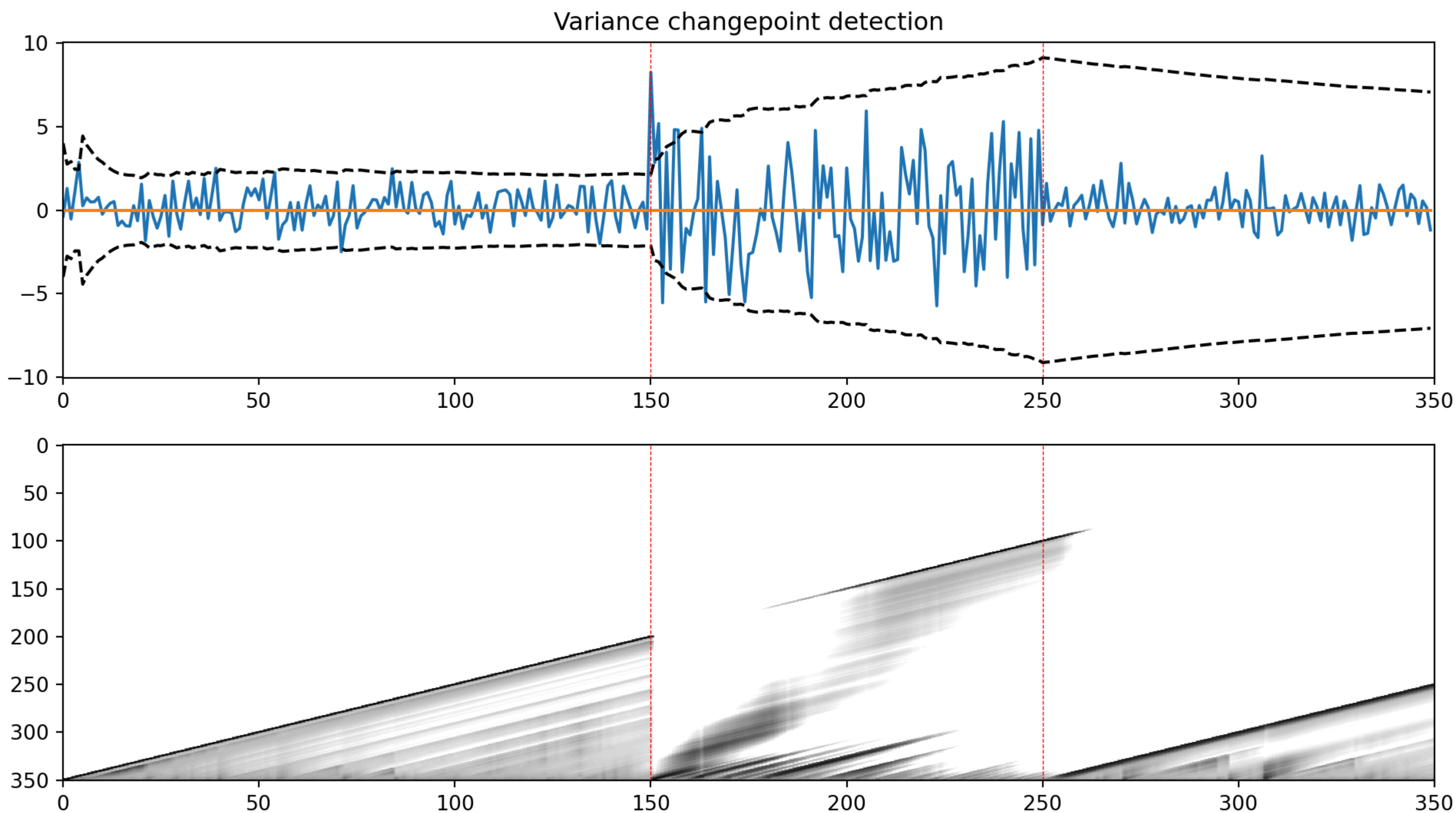
Application in risk monitoring in Finance

Changepoint algorithm

```python
def BOCD_var(signal, alpha0, beta0, hazard_prob, mu = 0):

    # Initialisation
    T = len(signal)
    log_R = -np.inf * np.ones((T+1, T+1)) # Run length posterior log-probability matrix
    log_R[0, 0] = 0 # At time 0, the posterior proability is initialised to 1 at R = 0

    posterior_precision = np.nan * np.empty(T) # Mean of the posterior distribution of the precision


    log_message = np.array([0]) # message initialised at 1
    log_H = np.log(hazard_prob) # Constant prior on changepoint probability.
    log_1_minus_H = np.log(1-hazard_prob)

    # Prior's parameters for the previous data point
    prior_shape = np.array([alpha0])
    prior_rate = np.array([beta0])


    # Online posterior distribution of the run length update:
    for t in range(1, T+1):
```

Application in risk monitoring in Finance

Results



Result of the BOCD algorithm on the variance changepoint dataset.

Application in risk monitoring in Finance

Results

Remarks

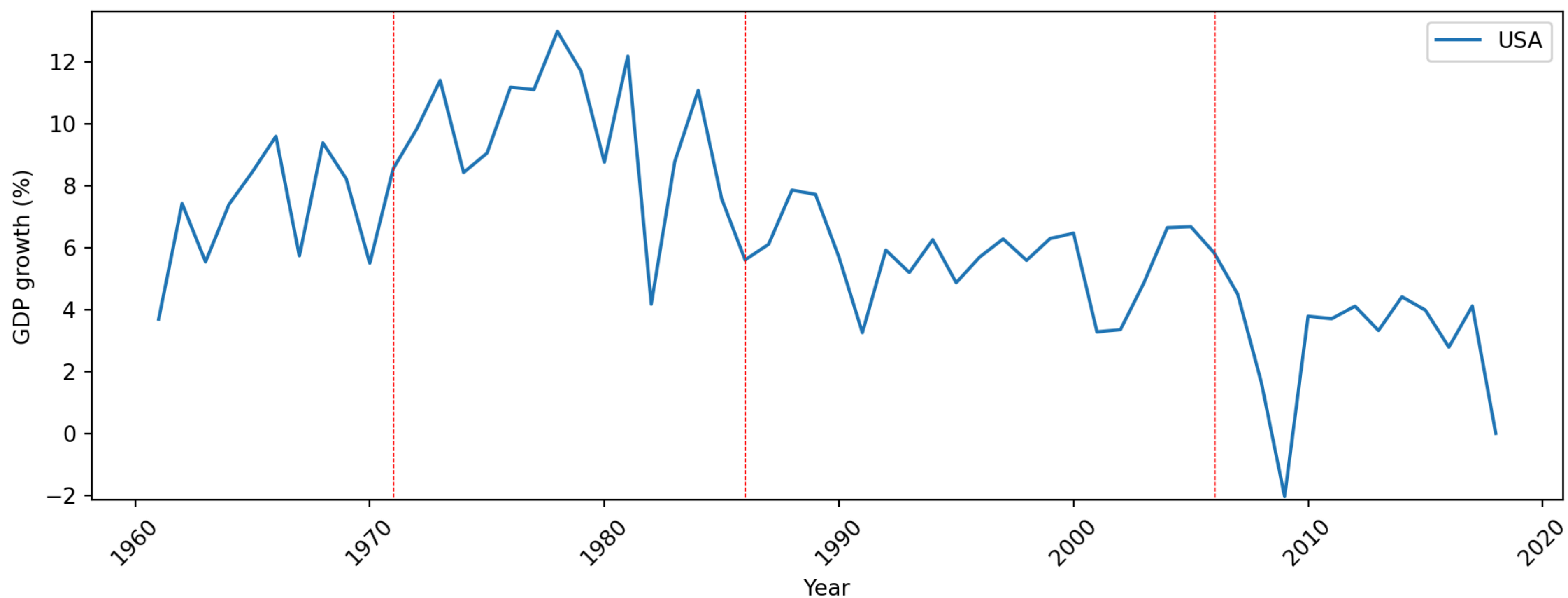
Note that the inferred variance seem to do great at the beginning, but then lacks reactivity. This is beacause of the online nature of the BOCD algorithm: at every time-step it keeps memory of what happened in the past, and the longer the past is, the stronger the memory gets.

To prevent that, the *restarted* BOCD algorithm was developed by Alami, Maillard, and Feraud (2020) which restarts all over again every time a changepoint is detected.

Other possible examples of applications

Retrospective analysis of GDP growth

To analyse the effects of economic policies or the economic cycles, one can apply an *offline* changepoint analysis with a PELT algorithm to infer the number and the location of the changepoints, which can indicate a change in the economic cycle.



Other possible examples of applications

Retrospective analysis of GDP growth

```
import ruptures as rpt

data = pd.read_csv("../Datasets/gdp.csv")

country_gdp = data.loc[data["Country Code"] == "USA"].melt(id_vars="Country Name", value_vars = [str(i) fo
country_gdp["Growth"] = country_gdp["GDP"].pct_change()

growth_data = np.array(country_gdp['Growth'])[1:]*100

algo = rpt.Pelt(model = "l1", min_size = 2).fit(growth_data)
changepoints = algo.predict(pen = np.log(len(growth_data)))


plt.figure(figsize = (12, 5))
plt.plot(np.array(country_gdp["Year"], dtype = float)[1:], growth_data, label = "USA")
plt.vlines([np.array(country_gdp["Year"], dtype = float)[1:][i] for i in changepoints[:-1]], linestyle = '
plt.xticks(rotation = 45);
plt.ylim((1.05*min(growth_data), 1.05*max(growth_data)));
plt.xlabel("Year")
plt.legend()
plt.ylabel("GDP growth (%)");
```

Useful libraries, package and repository

In R

- `mcp`
- `segmented`

In Python

- `ruptures`
- `changepoynt`

Useful GitHub repositories

https://github.com/gwgundersen/bocd https://github.com/alan-turing-institute/TCPD/tree/master

References

Code and presentation

GitHub repository

Textbooks and articles

Adams, Ryan Prescott, and David J. C. MacKay. 2007. "Bayesian Online Changepoint Detection." arXiv. https://doi.org/10.48550/arXiv.0710.3742.

Alami, Reda, Odalric Maillard, and Raphael Feraud. 2020. "Restarted Bayesian Online Change-Point Detector Achieves Optimal Detection Delay." In *Proceedings of the 37th International Conference on Machine Learning*, edited by Hal Daume III and Aarti Singh, 119:211–21. Proceedings of Machine Learning Research. PMLR. https://proceedings.mlr.press/v119/alami20a.html.

"Bayesian Online Changepoint Detection." n.d. Accessed June 6, 2024. https://gregorygundersen.com/blog/2019/08/13/bocd/#adams2007bayesian.

Fearnhead, Paul, and Piotr Fryzlewicz. 2022. "Detecting A Single Change-Point." arXiv. http://arxiv.org/abs/2210.07066.

———. 2024. "The Multiple Change-in-Gaussian-Mean Problem." arXiv. http://arxiv.org/abs/2405.06796.

Feasel, Kevin. 2022. *Finding Ghosts in Your Data: Anomaly Detection Techniques with Examples in Python*. Berkeley, CA: Apress. https://doi.org/10.1007/978-1-4842-8870-2.