Detecting Changes in Slope With an L_0 Penalty

Bastien LE CHENADEC bastien.le-chenadec@eleves.enpc.fr
Sofiane EZZEHI sofiane.ezzehi@eleves.enpc.fr

January 9, 2024

1 Introduction and contributions

Changepoints detection is an important problem in time series analysis. It consists in finding the points in time where the statistical properties of the signal change. The problem can be modeled as a parametric approximation of the original signal to which a cost function is associated [1]. Finding the approximation with minimal cost is generally a dynamic programming problem. In [2], the authors propose a piecewise continuous model for the signal, penalized by an L_0 penalty on the number of changepoints.

Our contributions are the following :

- Bastien L.C. implemented the algorithm in Python and optimized it to make it run in a reasonable time.
- Sofiane E. debugged the algorithm, notably errors in the original paper, and ran the experiments.

We did not reuse the original code (in R and C++). We chose to test the algorithm on Tunisian stock market data, and we experimented with different values of the penalty parameter β and the segment cost h.

2 Method

Let $y = y_1, \dots, y_n$[1] $\in \mathbb{R}^n$ be successive data points in time. We aim to find m changepoints $\tau_1, \dots, \tau_m$ such that the data is divided in $m+1$ segments. We let $\tau_0 = 0$ and $\tau_{m+1} = n$; the segment j is then $y_{\tau_{j-1}+1}, \dots, y_{\tau_j}$. The number of changepoints m is assumed unknown.

2.1 Model

The parametric model considered is a continuous piecewise linear model. We denote ϕ_{τ_j} the value taken by the model at time τ_j. Under these assumptions, the model writes :

$$\forall i \in [\![0, m]\!], \forall t \in [\![\tau_i + 1, \tau_{i+1}]\!], \quad Y_t = \phi_{\tau_i} + \frac{\phi_{\tau_{i+1}} - \phi_{\tau_i}}{\tau_{i+1} - \tau_i}(t - \tau_i) + Z_t \tag{1}$$

where $(Z_t)_{1 \le t \le n}$ is assumed to be a gaussian white noise with variance σ^2.

[1] A word on notations : if we have a sequence $x = x_1, \dots, x_n$, we denote $x[s:t]$ the subsequence $x_s, \dots, x_t$.

2.2 Penalized cost approach

In order to find the parameters $m, \tau_1, \ldots, \tau_m, \phi_{\tau_0}, \ldots, \phi_{\tau_{m+1}}$, we adopt a penalized cost approach. The cost function is a squarred error loss, penalized by a term that depends on the segments length, and an L_0 penalty on the number of changepoints. The cost function writes :

$$\sum_{i=0}^{m} \left[\frac{1}{\sigma^2} \sum_{t=\tau_i+1}^{\tau_{i+1}} \left(y_t - \phi_{\tau_i} - \frac{\phi_{\tau_{i+1}} - \phi_{\tau_i}}{\tau_{i+1} - \tau_i} (t - \tau_i) \right)^2 + h(\tau_{i+1} - \tau_i) \right] + \beta m \tag{2}$$

To simplify the notations we introduce the cost function for fitting a segment that takes the value ϕ at time s and ψ at time t :

$$\mathcal{C}(y[s+1:t], \phi, \psi) = \frac{1}{\sigma^2} \sum_{j=s+1}^{t} \left(y_j - \phi - \frac{\psi - \phi}{t - s} (j - s) \right)^2 \tag{3}$$

The optimisation problem then writes :

$$\min_{\substack{m \\ \tau_1, \ldots, \tau_m \\ \phi_0, \ldots, \phi_{m+1}}} \sum_{i=0}^{m} \left[\mathcal{C}(y[\tau_i + 1 : \tau_{i+1}], \phi_{\tau_i}, \phi_{\tau_{i+1}}) + h(\tau_{i+1} - \tau_i) \right] + \beta m \tag{4}$$

2.3 Dynamic programming

Problem (4) can be solved by dynamic programming. Compared to other changepoints detection methods, the difficulty lies in the continuity constraint between segments. Let $f^t(\phi)$ be the minimum cost for segmenting $y_1, \ldots, y_t$ conditionally on the model taking the value ϕ at time t :

$$\begin{aligned} f^t(\phi) = \min_{\substack{k \\ \tau_1, \ldots, \tau_k \\ \phi_0, \ldots, \phi_k}} & \sum_{i=0}^{k-1} \left[\mathcal{C}(y[\tau_i + 1 : \tau_{i+1}], \phi_{\tau_i}, \phi_{\tau_{i+1}}) + h(\tau_{i+1} - \tau_i) \right] \\ & + \mathcal{C}(y[\tau_k + 1 : t], \phi_{\tau_k}, \phi) + h(t - \tau_k) + \beta k \end{aligned} \tag{5}$$

$f^t(\phi)$ can be expressed recursively :

$$f^t(\phi) = \min_{\phi', s} f^s(\phi') + \mathcal{C}(y[s+1:t], \phi', \phi) + h(t-s) + \beta \tag{6}$$

To handle the cost of a specific segmentation let $f^t_{\boldsymbol{\tau}}(\phi)$ be the minimum cost for segmenting $y_1, \ldots, y_t$ conditionally on the model taking the value ϕ at time t and the changepoints being $\boldsymbol{\tau} = (\tau_0, \tau_1, \ldots, \tau_k)$. $f^t_{\boldsymbol{\tau}}$ is a quadratic polynomial in ϕ and its coefficients can be expressed recursively.

2.4 Pruning

Given the exponential complexity of the dynamic programming algorithm ($\mathcal{O}(2^n)$ possible segmentations), the authors introduce two pruning techniques to reduce the complexity.

2.4.1 Functional pruning

Let $\mathcal{T}_t$ be the set of all possible changepoint vectors for segmenting $y_1, \ldots, y_t$, and $\mathcal{T}^*_t = \left\{ \boldsymbol{\tau} \in \mathcal{T}_t \,\middle|\, \exists \phi \in \mathbb{R}, f^t(\phi) = f^t_{\boldsymbol{\tau}}(\phi) \right\}$ the set of changepoint vectors at time t that are optimal for some ϕ. The following result reduces the number of segmentations to consider at time $t+1$.

Theorem 1. *Let $\boldsymbol{\tau} \in \mathcal{T}_s$ such that $\boldsymbol{\tau} \notin \mathcal{T}^*_s$. Then $\boldsymbol{\tau} \notin \mathcal{T}^*_t$ for all $t > s$.*

2.4.2 Inequality based pruning

Let $K = 2\beta + h(1) + h(n)$. Suppose that h is non-negative and non-decreasing. The following result reduces the number of segmentations to consider at time $t+1$.

Theorem 2. *Let $\boldsymbol{\tau} \in \mathcal{T}_s$ such that :*

$$\min_{\phi} f^s_{\boldsymbol{\tau}}(\phi) > K + \min_{\phi'} f^s(\phi')$$

Then $\boldsymbol{\tau} \notin \mathcal{T}^*_t$ for all $t > s$.

This theorem allows to further prune the set $\hat{\mathcal{T}}_t$.

2.5 Algorithm

The algorithm consists in computing the coefficients of the polynomials $f^t_{\boldsymbol{\tau}}(\phi)$ iteratively, starting from $f^0(\phi)$. At each step, the set of possible segmentations is pruned using the two pruning techniques. The algorithm is detailed in appendix 6.1.

The main difficulty of this algorithm is that we need to store polynomial coefficients instead of cost values. We also have to be carful to store only the coefficients that will be needed in recursive calls ($f^{\tau_k}_{\tau_1,\dots,\tau_{k-1}}$) to avoid memory issues. Overall avoiding re-allocating memory at each step was a challenge to implement the algorithm efficiently.

The authors provided relatively detailed pseudo-code for the algorithm, but we still found it difficult to implement. One difficulty was that the coefficients updates given in the paper were not correct, which took a lot of time to figure out.

3 Data

3.1 Data description : Tunisian stock market

Financial data, and more specifically stock prices across time, are a good candidate to test the method. Indeed, stock prices, taken at a relatively low frequency (daily) are known to exhibit clear trends, while being noisy enough to make the detection of changepoints interesting. And since the method is designed to detect changepoints in a piecewise linear model, it is well suited for this kind of data.

We test the method on the stock prices of all the listed companies on the Bourse de Tunis, which is Tunisia stock exchange. The data was taken from Kagglee, and is available at `https://www.kaggle.com/datasets/amariaziz/tunisian-stock-market`. It contains the Open, High, Low and Close prices of 88 companies, as well as the volume of transactions. We only use the Close prices, which are the prices at the end of the trading day.

3.2 Data preprocessing

Since the data was partially collected by web scraping, it naturally requires some cleaning and preprocessing. We detail here the steps we took to clean the data. We are left with 85 companies.

Stocks with too few data points We see on figure **??** that the number of data points per company is very variable. We remove the stocks that have less than 200 data points, since they are not interesting for our analysis.

Missing values and duplicates Missing values are days where we have corrupted data for a given stock. We see in table **??** that the data is relatively clean of missing values, with only 3 companies having no more than a few percent of missing values. We see on figure **??** that each stock has between 1% and 3% of duplicates. We remove the missing values as well as the duplicates.

Missing days Missing days are days where we have no data at all for a given stock. We see on figure **??** that the number of missing days is clearly the biggest issue with the data, with most stocks having more than 40% of missing days. We deal with this issue by interpolating the missing days, using a linear interpolation between the last and next available data points. We note that this is not a perfect solution, since it assumes that the stock price evolves linearly between two days, which is never the case for stock prices. However, it is a simple solution that allows us to keep the data for all the stocks.

Data shrinkage by sub-sampling Since our method is relatively slow for time series longer than a few thousand points, and since our available computational power is limited, we sub-sample the data to reduce its size. We sub-sample the data by only keeping at most 1000 points per stock, by sampling uniformly on the time axis. We see on figure **??** that this sub-sampling very slightly affects the data.

4 Results

We present here a few experiments that we conducted on the 85 Tunisian stocks we preprocessed. First, we zoom on a few stocks to see how the method behaves and we qualitatively compare it to a best-fitting piecewise constant mean model. Second, we investigate the influence of the penalty parameter β on the number of changepoints detected. Third, we investigate the influence of the segment cost h on the number of changepoints detected. Finally, we briefly discuss the computational complexity of the method.

4.1 Qualitative comparison with a piecewise constant mean model

We compare the method to a piecewise constant mean model, which is the simplest model that can be used to detect changepoints. We note that the nature of the data (stock prices) puts the piecewise constant mean model at a great disadvantage, since stock prices are known to exhibit piecewise linear trends. We only perform this comparison qualitatively.

We see on figure 1 that the CPOP method is able to detect the changepoints much more accurately than the PELT method. More precisely, the PELT method exhibits a clear tendency to detect multiple changepoints between two consecutive CPOP changepoints. This is due to the fact that the PELT method is, by design, only able to detect changes in the mean of the signal; Therefore, along a steep trend, as the signal rapidly and monotonically varies, it will detect multiple changepoints, while the CPOP method will only detect one changepoint at the beginning of the trend and one at the end.

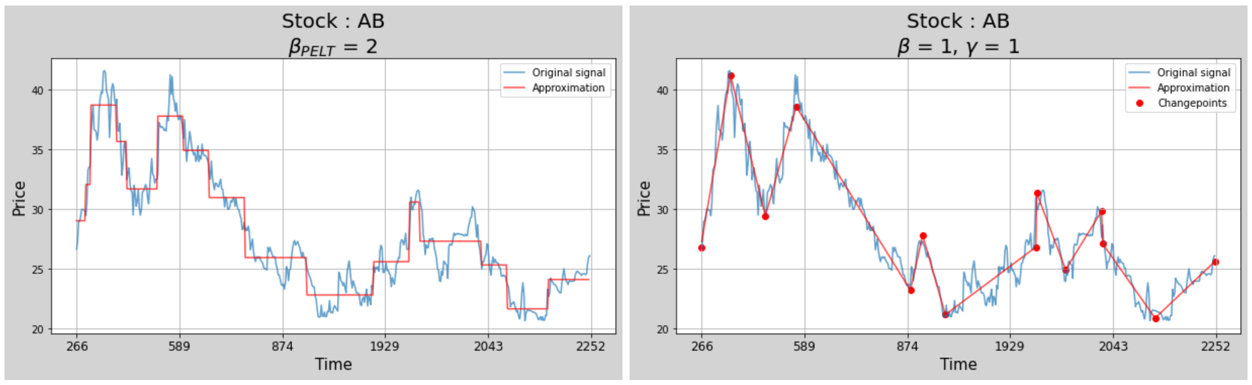

Figure 1: Comparison of the PELT method (left) and the CPOP method (right) on the stock of the company "Amen Bank" (AB).

4.2 Influence of the penalty parameter β

Next, we investigate the influence of the penalty parameter β on the number of changepoints detected. We see on figure **??** that the number of changepoints detected is a decreasing function of β. This is consistent with the fact β penalizes the number of changepoints.

4.3 Influence of the segment cost h

4.4 Computational complexity

5 Bibliography

References

[1] Charles Truong, Laurent Oudre, and Nicolas Vayatis. Selective review of offline change point detection methods. *arXiv.org*, Mar 2020. URL https://arxiv.org/abs/1801.00718.

[2] Robert Maidstone, Paul Fearnhead, and Adam Letchford. Detecting changes in slope with an l_0 penalty. *arXiv.org*, Feb 2017. URL https://arxiv.org/abs/1701.01672.

6 Appendix

6.1 Algorithms

This algorithm keeps the polynomials that are optimal on some part of the real curve, by starting at $-\infty$ and iteratively finding the next optimal polynomial.

Algorithm 1 Functional pruning algorithm

```
1: Input: Candidate segmentations $\hat{\mathcal{T}}_t$, costs $f^t_\tau(\phi)$ for $\tau \in \hat{\mathcal{T}}_t$.
2: Initialize: $\phi_{\text{curr}} = -\infty \quad \tau_{curr} = \arg\min_{\tau \in \hat{\mathcal{T}}_t} \left[ f^t_\tau(\phi_{curr}) \right] \quad \mathcal{T}_{temp} = \hat{\mathcal{T}}_t \setminus \{\tau_{curr}\} \quad \mathcal{T}^*_t = \{\tau_{curr}\}$
3: while $\mathcal{T}_{temp} \neq \emptyset$ do
4:     for $\tau \in \mathcal{T}_{temp}$ do
5:         $x_\tau = \min \left\{ \phi : f^t_\tau(\phi) - f^t_{\tau_{curr}}(\phi) = 0 \;\; \& \;\; \phi > \phi_{curr} \right\}$
6:         if $x_\tau = \emptyset$ then
7:             $\mathcal{T}_{temp} = \mathcal{T}_{temp} \setminus \{\tau\}$
8:     $\mathcal{T}_{temp} = \mathcal{T}_{temp} \setminus \{\tau_{curr}\}$
9:     $\tau_{curr} = \arg\min_{\tau \in \mathcal{T}_{temp}} (x_\tau)$
10:    $\phi_{curr} = x_{\tau_{curr}}$
11:    $\mathcal{T}^*_t = \mathcal{T}^*_t \cup \{\tau_{curr}\}$
12: Output: $\mathcal{T}^*_t$
```

This algorithm is the main algorithm, it computes the coefficients of the polynomials and prunes the set of segmentations. The coefficients updates are detailed the appendix.

Algorithm 2 CPOP

```
1: Input: Data $y = y_1, \ldots, y_n$, penalty $\beta$ and segment cost $h$.
2: Initialize: $\hat{\mathcal{T}}_1 = \{\{0\}\}$
3: for $t = 1, \ldots, n$ do                                    ▷ Cost computation
4:     for $\tau \in \hat{\mathcal{T}}_t$ do
5:         if $\tau = \{0\}$ then                                 ▷ No changepoint
6:             $f^t_\tau(\phi) = \min_{\phi'} \mathcal{C}(y[1:t], \phi', \phi) + h(t)$
7:         else                                                ▷ Compute recursively
8:             $f^t_\tau(\phi) = \min_{\phi'} \left( f^{\tau_k}_{\tau_1,\ldots,\tau_{k-1}}(\phi') + \mathcal{C}(y[\tau_k + 1 : t], \phi', \phi) + h(t - \tau_k) + \beta \right)$
9:     $\mathcal{T}^*_t = \text{functional\_pruning}(\hat{\mathcal{T}}_t, f^t)$
10:    $\hat{\mathcal{T}}_{t+1} = \left\{ \tau \in \hat{\mathcal{T}}_t | \min_\phi f^t_\tau(\phi) \leq \min_{\phi',\tau'} f^t_{\tau'}(\phi') + K \right\} \cup \{(\tau, t) | \tau \in \mathcal{T}^*_t\}$
11: Output: $\arg\min_{\tau \in \hat{\mathcal{T}}_n} [\min_\phi f^n_\tau(\phi)]$
```