Wild Binary Segmentation

A Monte Carlo-like approach to localizing changepoints in data

Magnus Berg Sletfjerding

March 7, 2019

The Problem

Let's say we have a noisy time series data.

- Financial data
- Student satisfaction over time
- Traffic Data
- Biomolecular activity

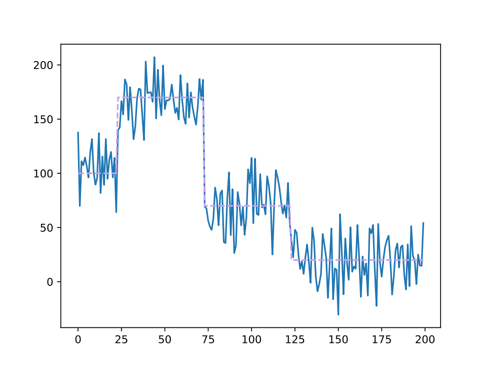
We would prefer to separate this data into separate chunks that are easier to work with. But how?!

The Problem

Let's say we have a noisy time series data.

- Financial data
- Student satisfaction over time
- Traffic Data
- Biomolecular activity

We would prefer to separate this data into separate chunks that are easier to work with. But how?!

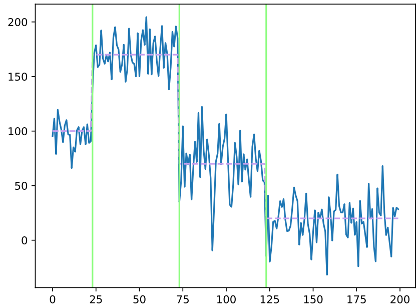


The Problem

Let's say we have a noisy time series data.

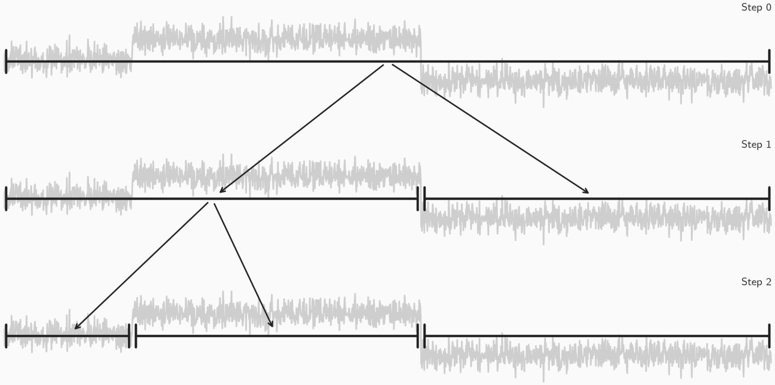
- Financial data
- Student satisfaction over time
- Traffic Data
- Biomolecular activity

We would prefer to separate this data into separate chunks that are easier to work with. But how?!



Binary Segmentation

This is the classical way of finding changepoints in data.



It's quick, easy to understand, and easy to conceptualize.

Binary Segmentation - math

Assume that our data can be like so:

$$X_t = f_t + \varepsilon_t, t = 1, \ldots, T$$

where f_t is a one-dimensional signal which has an unknown number of changepoints N, with unknown locations $\eta_1, \ldots, \eta_N$, and ε_t is a normal random variable centered at 0.

Binary Segmentation maximizes this statistic:

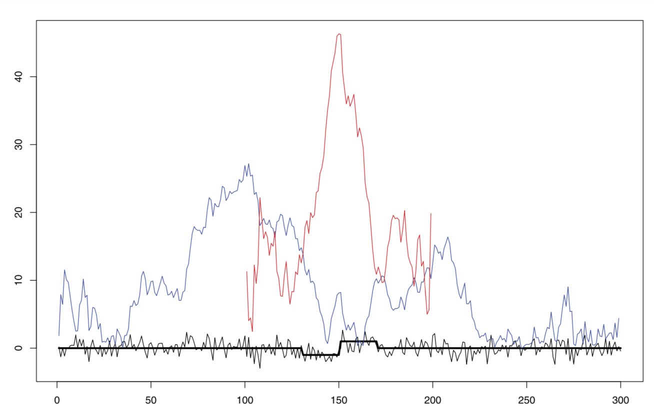
$$\tilde{X}^b_{s,e} = \sqrt{\frac{e-b}{n(b-s+1)}} \sum_{t=s}^{b} X_t - \sqrt{\frac{b-s+1}{n(e-b)}} \sum_{t=b+1}^{e} X_t$$

where $s \leq b < e$ and $n = e - s + 1$.

What are we left with? The **most likely changepoint,** b_0

So Binary Segmentation is perfect?

Not quite. Sometimes, the test statistic used can end up being maximized at places where there is no changepoint!!



Hopefully there will be a better method in the next slide!

Wild Binary Segmentation

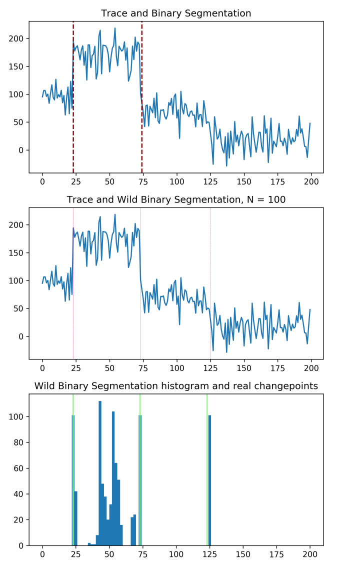
Before doing any Binary Segmentation, make a set F_T^M of M randomly sampled **intervals** $[s_m, e_m], m = 1, \ldots, M$ where $[s_m, e_m]$ have been drawn from $1, \ldots, M$.

Once that's done, do binary segmentation on ALL the intervals, but only choose the b_0^m that completely maximizes $\tilde{X}_{s,e}^b$

Higher chance that the fit will find the "right" values.

An example

- I generated some data to work on, and implemented the two algorithms
- This looks more or less like what I work with on the daily
- Normal Binary Segmentation does not find the third(last) changepoint
- Wild Binary Segmentation (1000 iterations) found the true changepoints for the most part



Conclusion

- $WBS > BS$
- Implementing algorithms yourself is hard
- BS is implemented in the `ruptures` package
- No WBS in Python 3.7