Optimizing Changepoint Detection through Deep Learning-based Penalty Tuning

Author: Tung Nguyen

1. Problem

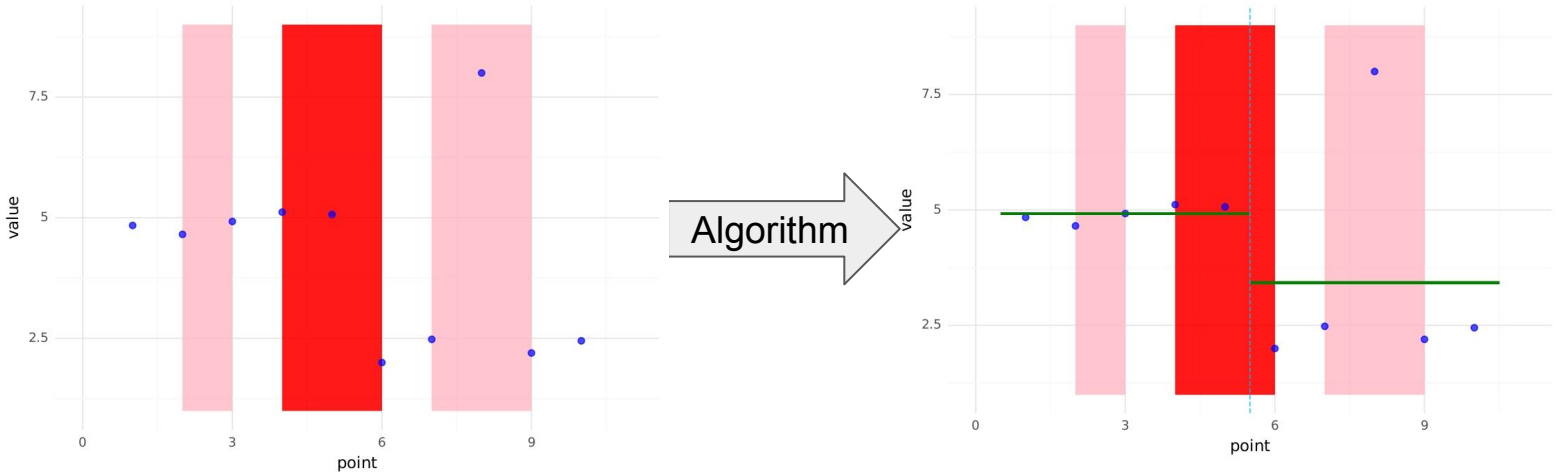
2. Review

3. Method

4. Results

We have a sequence with labels (negative or positive regions)

We apply the algorithm (OPART, LOPART) and return a set of changepoints (or breakpoints)

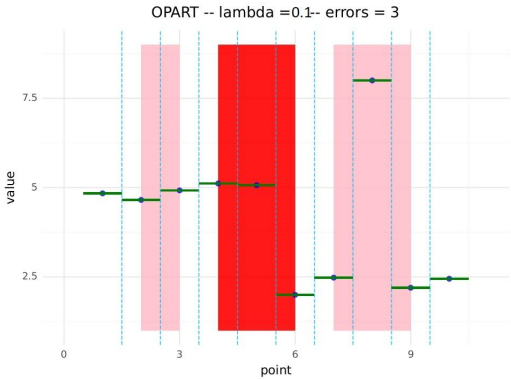


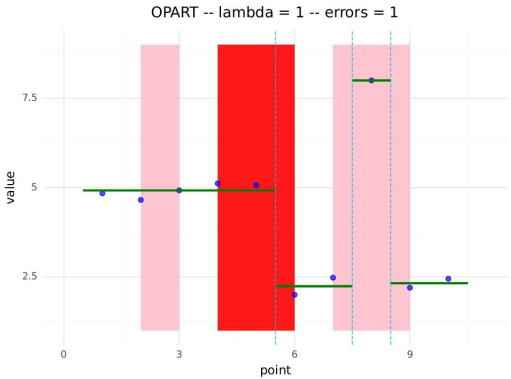
There is a fixed penalty parameter of the algorithms (OPART, LOPART), which is λ

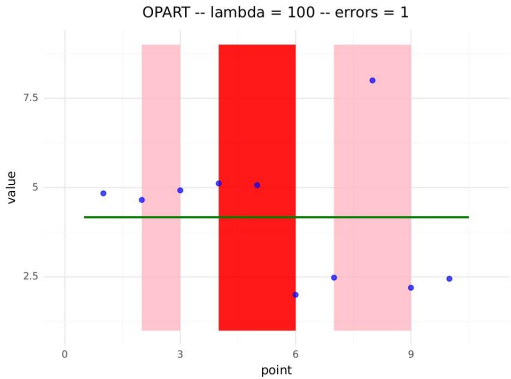
```
OPART(sequence, λ)          = set_of_changepoints

LOPART(sequence, labels, λ) = set_of_changepoints
```

With varying values of λ, different results are produced. See examples below:





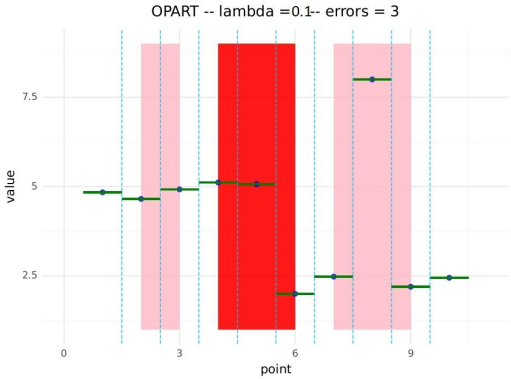


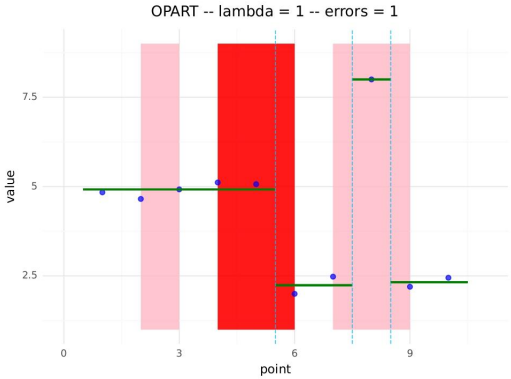
1. Problem

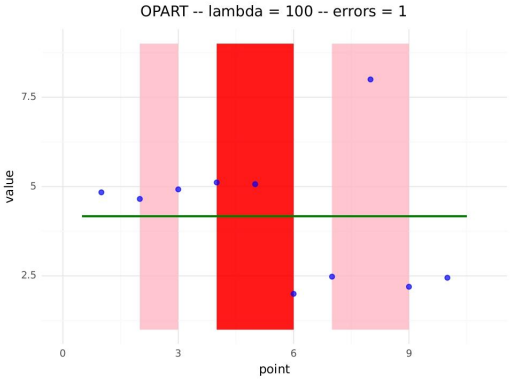
2. Review

3. Method

4. Results





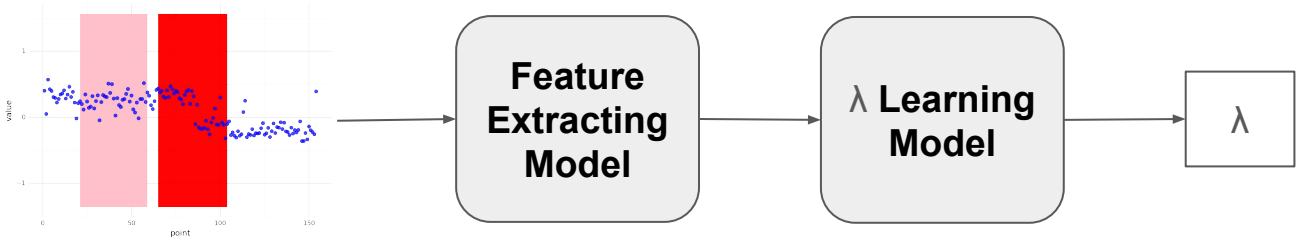


So from the example, we see that λ=0.1 is not good, λ=1 is maybe better.

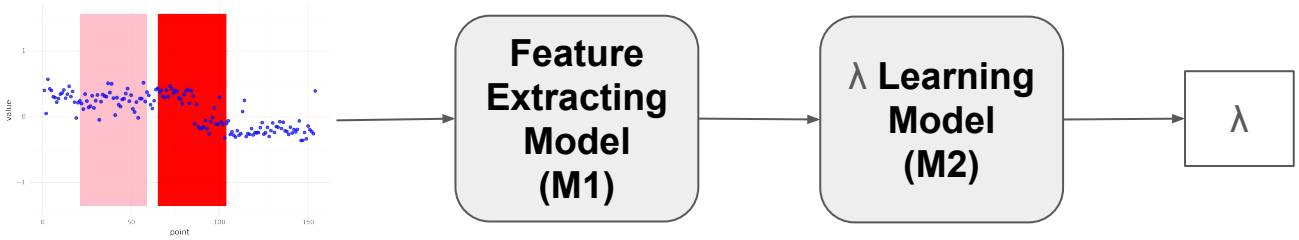
My research goal is to identify the optimal λ that minimize the number of predicted label errors

From the sequences, how can we learn λ

Below is the general model

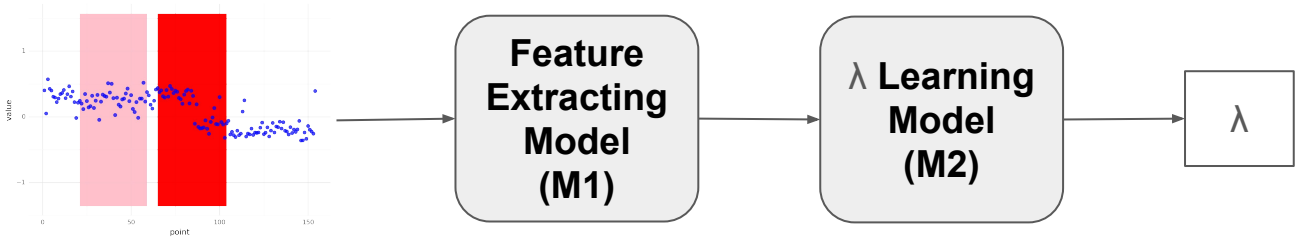


Sequences



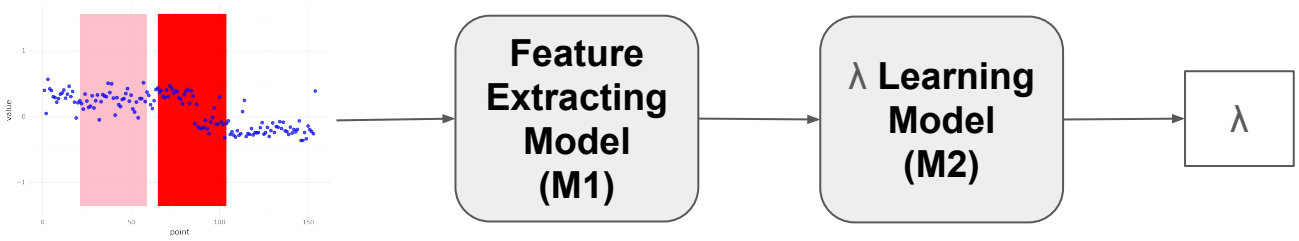
1. Bayesian Information Criterion (BIC)

- M1: `x = log(N)` where `N` is the sequence length.
- M2: `λ = x`



2. Linear

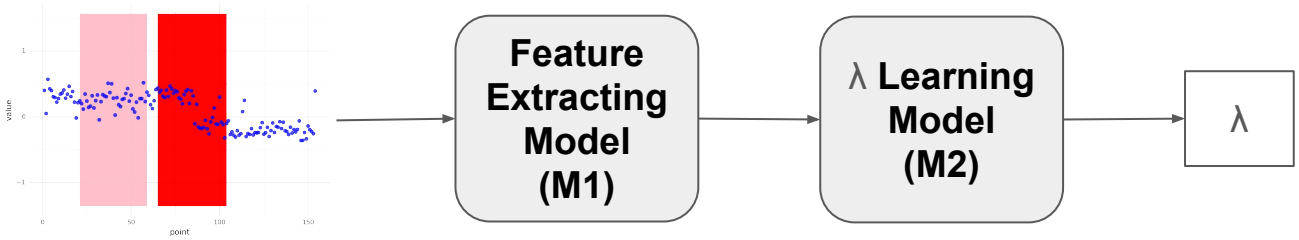
- M1: `x = log(log(N))` where `N` is the sequence length.
- M2: `λ = exp(xw + β)` where `w` and `β` are the model's parameters.



Summary

	BIC	Linear
M1	$x = \log(N)$	$x = \log(\log(N))$
M2	$\lambda = x$	$\lambda = \exp(xw + \beta)$

Innovation: Using different type of models for M1 and M2

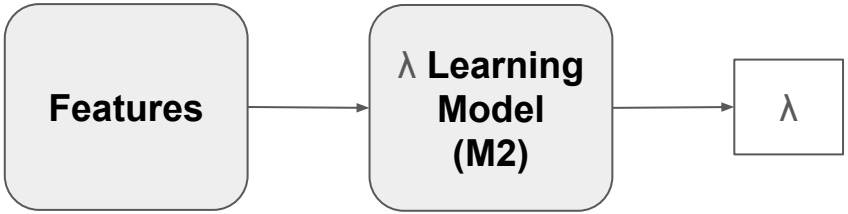


I have two approaches for selecting models M1 and M2

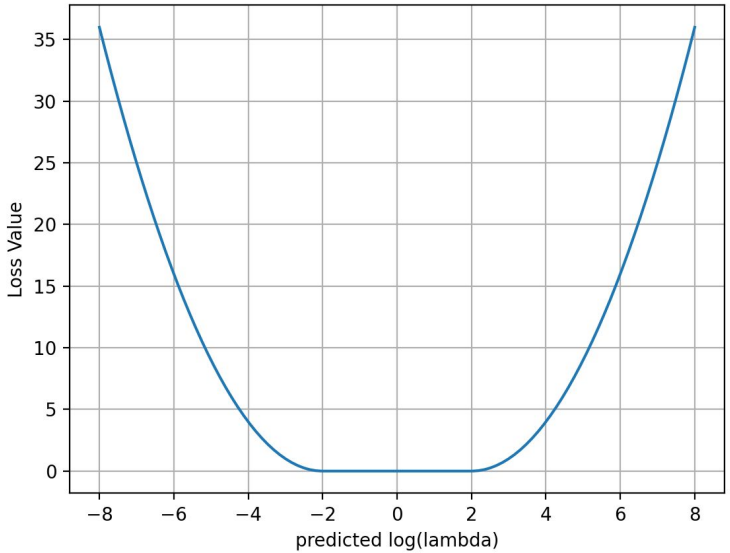
- Approach 1:
  - M1: consider a set of features that I can manually extract from a sequence (length, mean, variance, …), then using visualization to select the useful ones.
  - M2: multilayer perceptron (MLP)
- Approach 2:
  - M1: Recurrent Neural Network (RNN)
  - M2: linear model

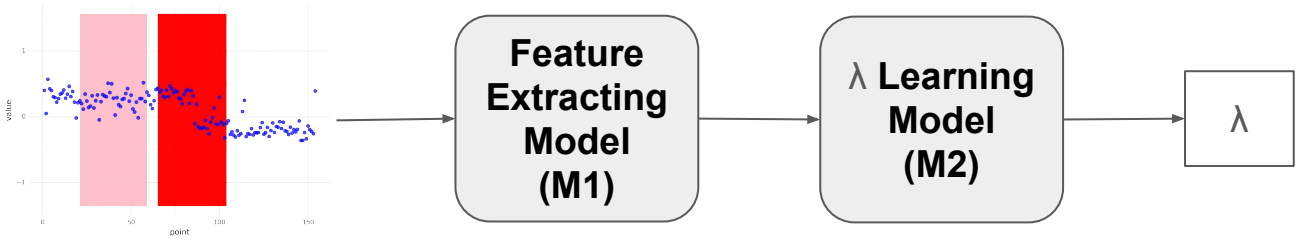
To learn a model M2, we need data X and target y:

- X is the set of features (output of M1)
- y is the `(m1, m2)` : for each sequence, I run OPART with `λ = 10^m` where `m ∈ {-5, -4.5, …, 5}`. Then choose the interval `(m1, m2)` that minimizes the number of train label errors. (For example, for sequence 1, `(m1, m2)=(-3, 3)`)



Loss Function: Mean Square Hinge Error (for example, `(m1, m2)=(-3, 3)`) with margin = 1





Approach 1:

- M1: I consider 7 features
  - Standard deviation
  - Mean
  - Range Value
  - Absolute skewness
  - Kurtosis
  - Length
  - Sum of difference between two consecutive points

  Then I visualize each of them with respect to the target interval `(m1, m2)` to pick the useful ones.

- M2: multilayer perceptron (MLP)

1. Problem

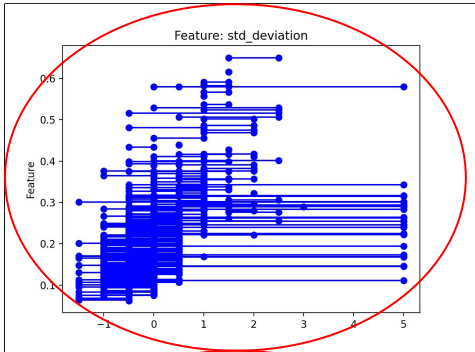
2. Review

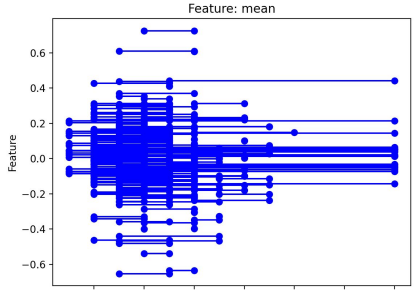
3. Method

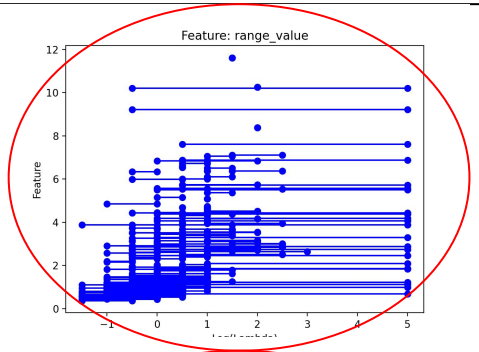
a. Approach 1

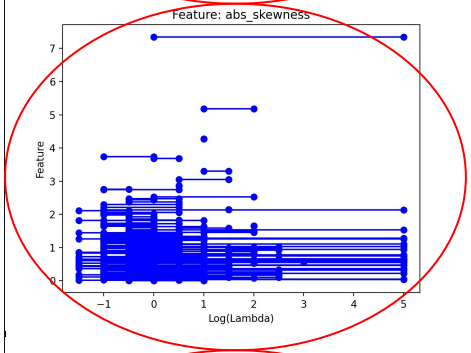
b. Approach 2

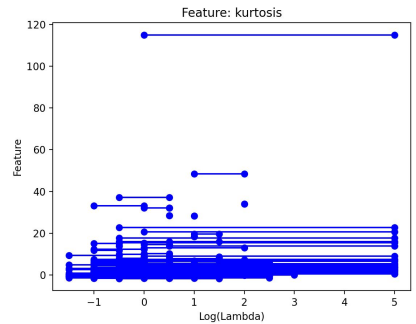
4. Results

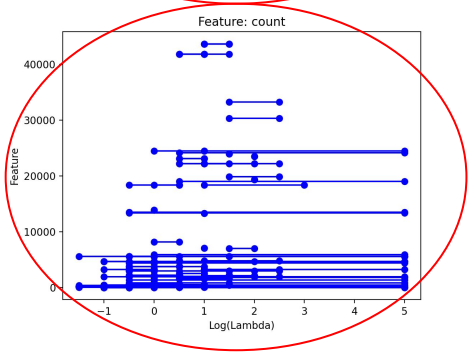


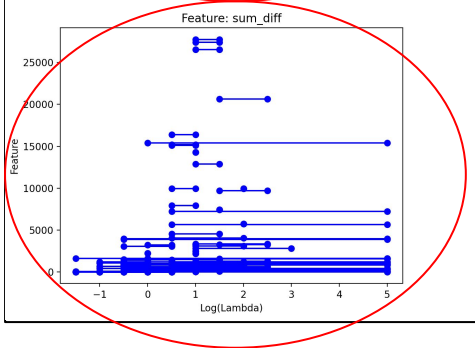


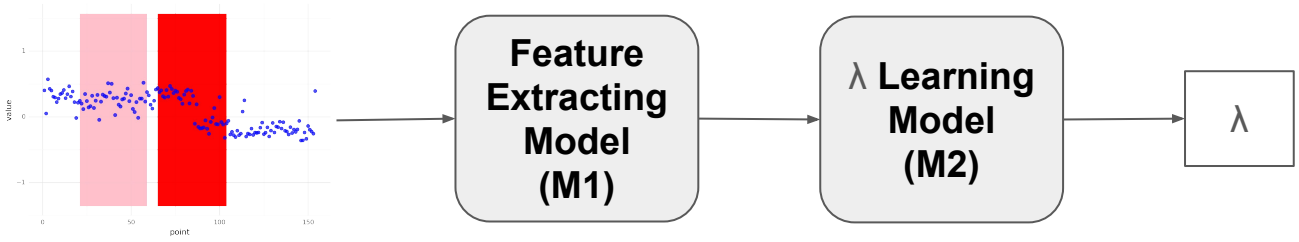








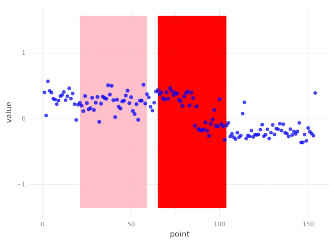




Approach 1 comparison with the previous works

	BIC	Linear	Approach 1
M1	x = log(N)	x = log(log(N))	x1 = std(seq) x2 = range(seq) x3 = |skew(seq)| x4 = len(seq) = N x5 = sum_diff(seq)
M2	λ = x	λ = exp(xw + β)	log(λ) = MLP(x1, x2, x3, x4, x5)

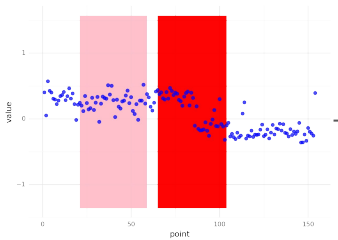
Approach 1



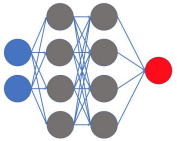
Feature Extracting Model (M1)

λ Learning Model (M2)

λ



x1
x2
x3
x4
x5



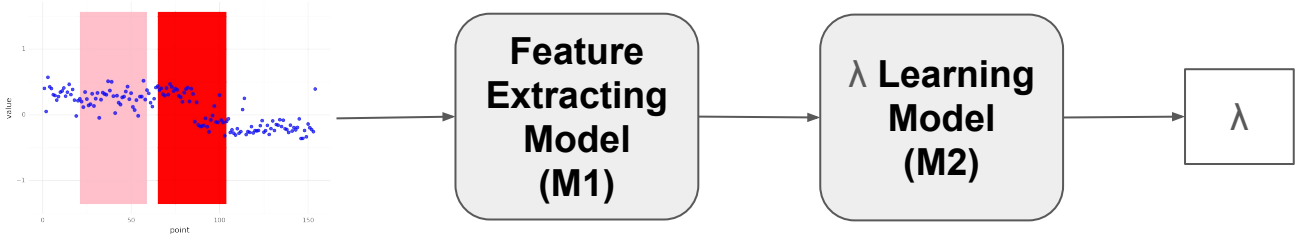
λ

Approach 2

- M1: Recurrent Neural Network (RNN)
- M2: Linear model

The reason I chose RNN to be the M1 (extracting features from sequences) because this type of model can:

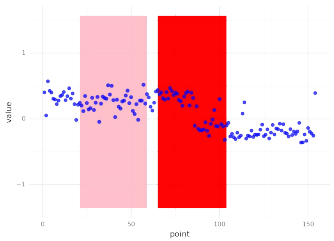
- Take into inputs with different length
- Produce the fixed length output
- Can learn features from sequences (like mean, standard variation, length, …) so I hope this model can produce some useful features that I did not consider on the approach 1.



Approach 2 comparison with the previous works

	BIC	Linear	Approach 2
M1	$x = \log(N)$	$x = \log(\log(N))$	$\mathbf{x} = RNN(seq)$
M2	$\lambda = x$	$\lambda = \exp(xw + \beta)$	$\lambda = \exp(\mathbf{xw} + \beta)$

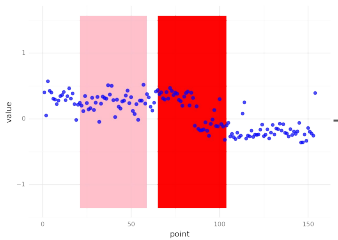
Approach 2

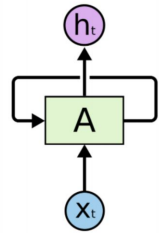


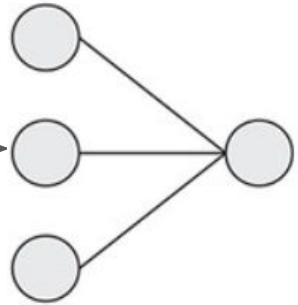
Feature Extracting Model (M1)

λ Learning Model (M2)

λ







λ

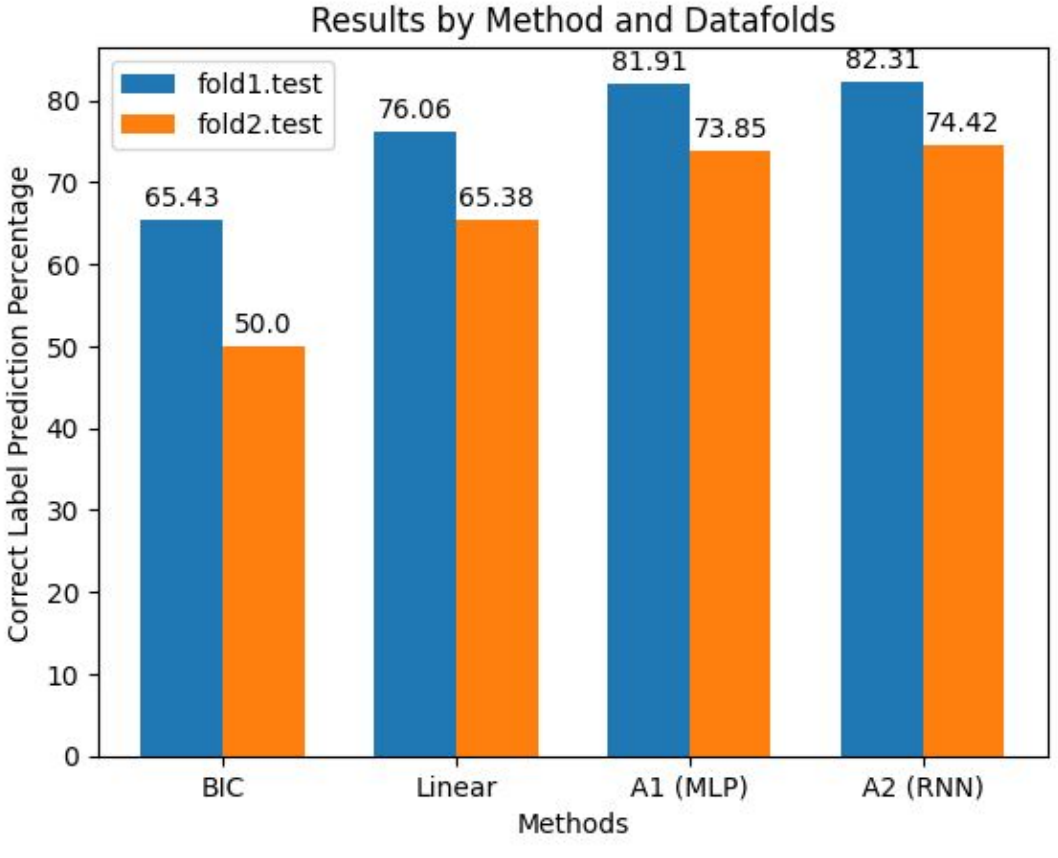
1. Problem

2. Review

3. Method

4. Results

Using each method (4 of them) to predict λ, and I apply `OPART(seq_i, λ_i)` for each sequence, get the set of changepoints. Then I can calculate the correct predicted label percentage.



THANK YOU FOR LISTENING