Anomaly Detection

大綱

1. 動機
2. 簡介
3. Ruptures 理論介紹
4. 實測

動機

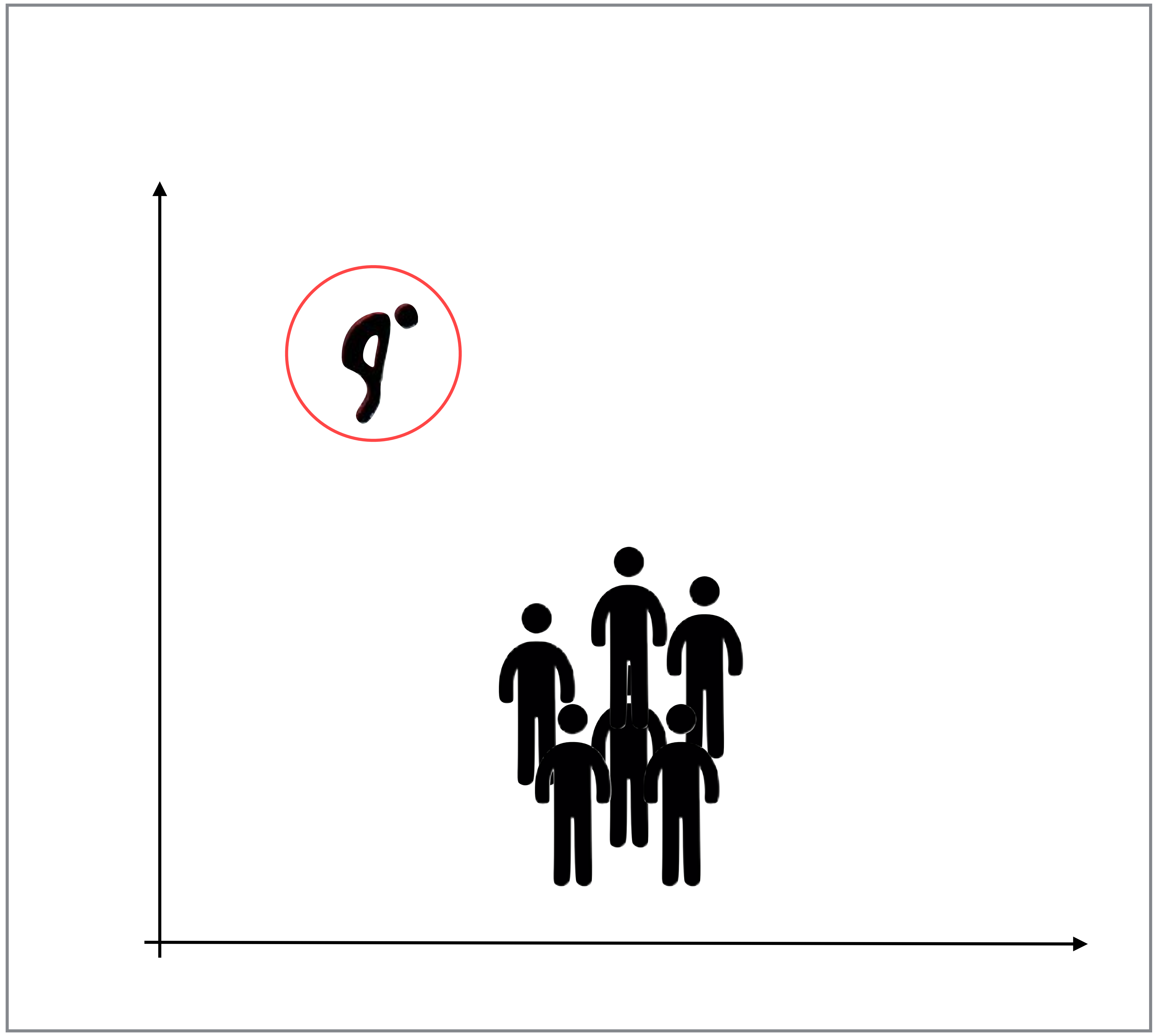
若做的案子沒有Label或Label稀少怎麼辦？從非監督著手協助業務單位快速找到可疑資料，並從過程搜集Label

1. 稽核案件、洗防名單都屬於事後監控的例子
2. 調查過程通常都有大量未標註標籤、正負類極度不平衡
3. 使用Anomaly Detection的架構協助業務單位迅速歸納出資料集中的異常值
4. 從非監督走向監督的一條路程

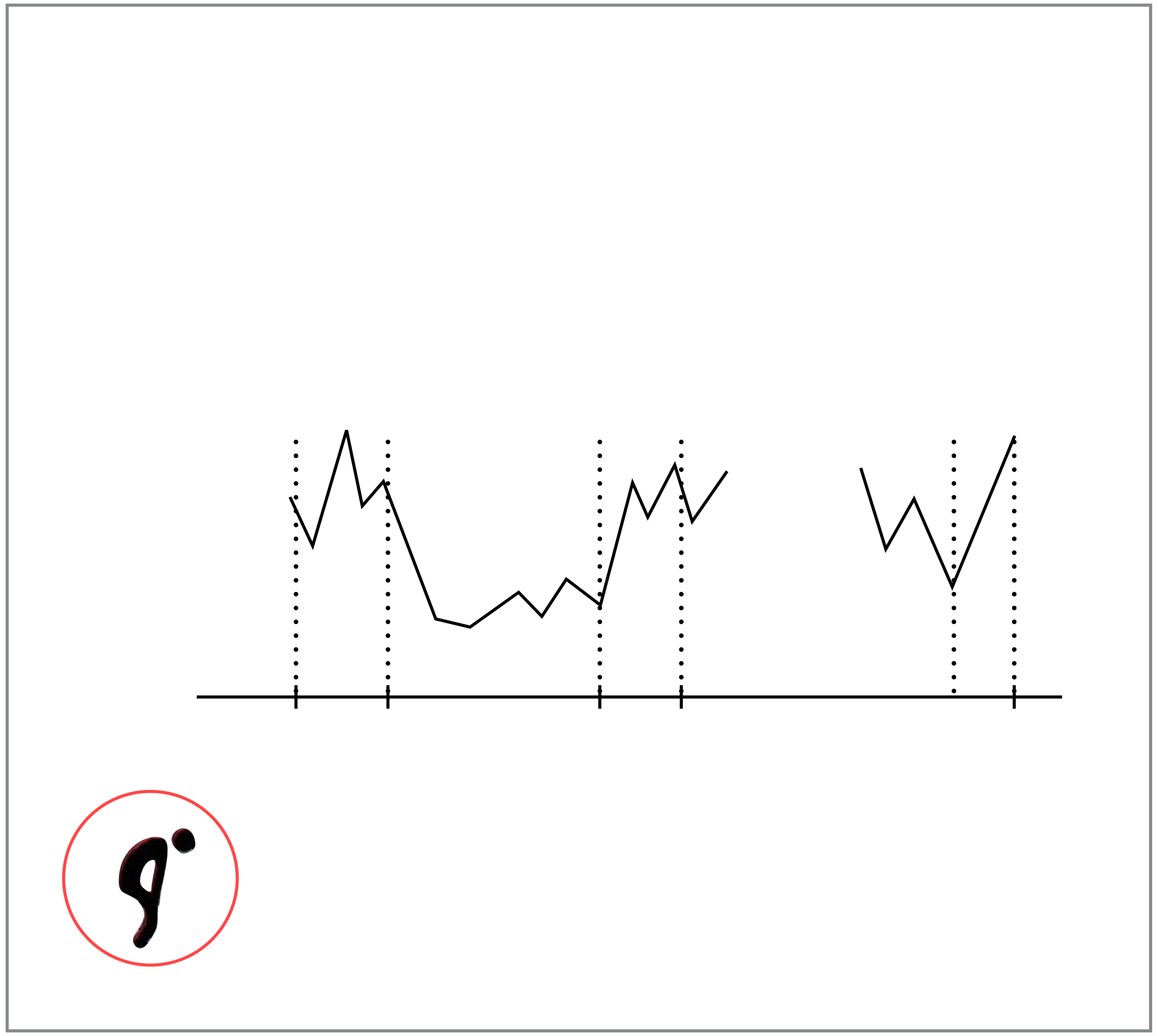
簡介

希望可以在沒有Label的狀況下，快速提供異常分佈資料、異常發生時間，提供更多調查線索

異常分佈資料



異常時間



Ruptures

Change Point Detection是一個針對訊號或者時間序列，偵測背後生成模型改變點的任務

1. change point detection是偵測結構改變點的任務
2. Ruptures 是python中實作change point detection的套件
3. 最重要的假設：piecewise stationary

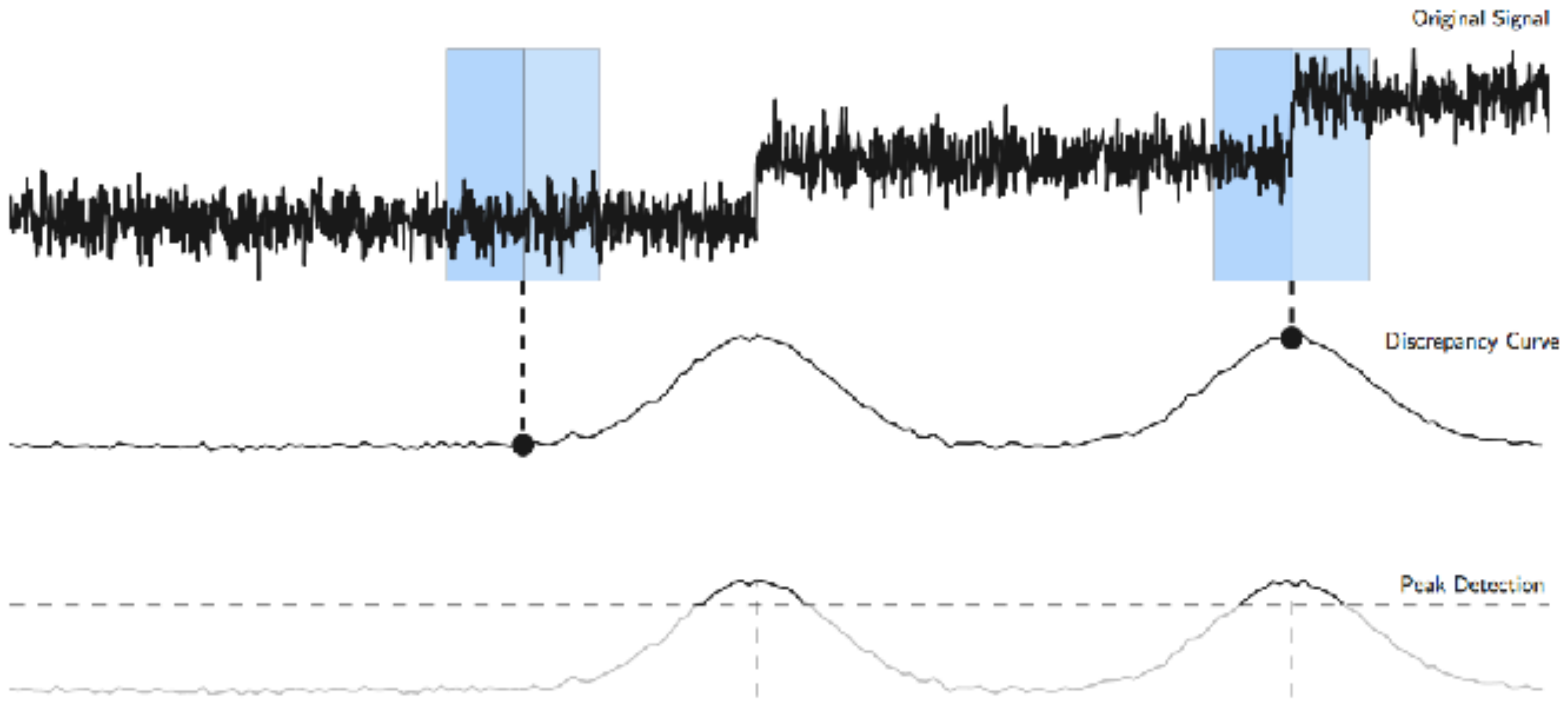


Figure 8: Schematic view of Win

Ruptures 理論

定義問題 -> divide and conquer

1. Q1：如何找到最佳的分割？
2. Q2：給定任一分割如何計算損失？
3. Q3：如何透過演算法算完每種可能性？

- **Problem 1 : known number of changes K.** The change point detection problem with a fixed number K of change points consists in solving the following discrete optimization problem

$$\min_{|\mathcal{T}|=K} V(\mathcal{T}). \tag{P1}$$

- **Problem 2 : unknown number of changes.** The change point detection problem with an unknown number of change points consists in solving the following discrete optimization problem

$$\min_{\mathcal{T}} V(\mathcal{T}) + \text{pen}(\mathcal{T}) \tag{P2}$$

where $\text{pen}(\mathcal{T})$ is an appropriate measure of the complexity of a segmentation $\mathcal{T}$.

Q1

分成兩種不同問題，第一種是給定change point個數下找尋最佳解，另一種是change point unknown

1. 如何找到最佳的Partition？
2. 最佳代表有評價方式 -> 評價函數
3. P1：給定change point個數K，找尋最小化評價函數的partition
4. P2：change point個數未知，找尋最小化評價函數的partition

- **Problem 1 : known number of changes K.** The change point detection problem with a fixed number K of change points consists in solving the following discrete optimization problem

$$\min_{|\mathcal{T}|=K} V(\mathcal{T}). \tag{P1}$$

- **Problem 2 : unknown number of changes.** The change point detection problem with an unknown number of change points consists in solving the following discrete optimization problem

$$\min_{\mathcal{T}} V(\mathcal{T}) + \mathrm{pen}(\mathcal{T}) \tag{P2}$$

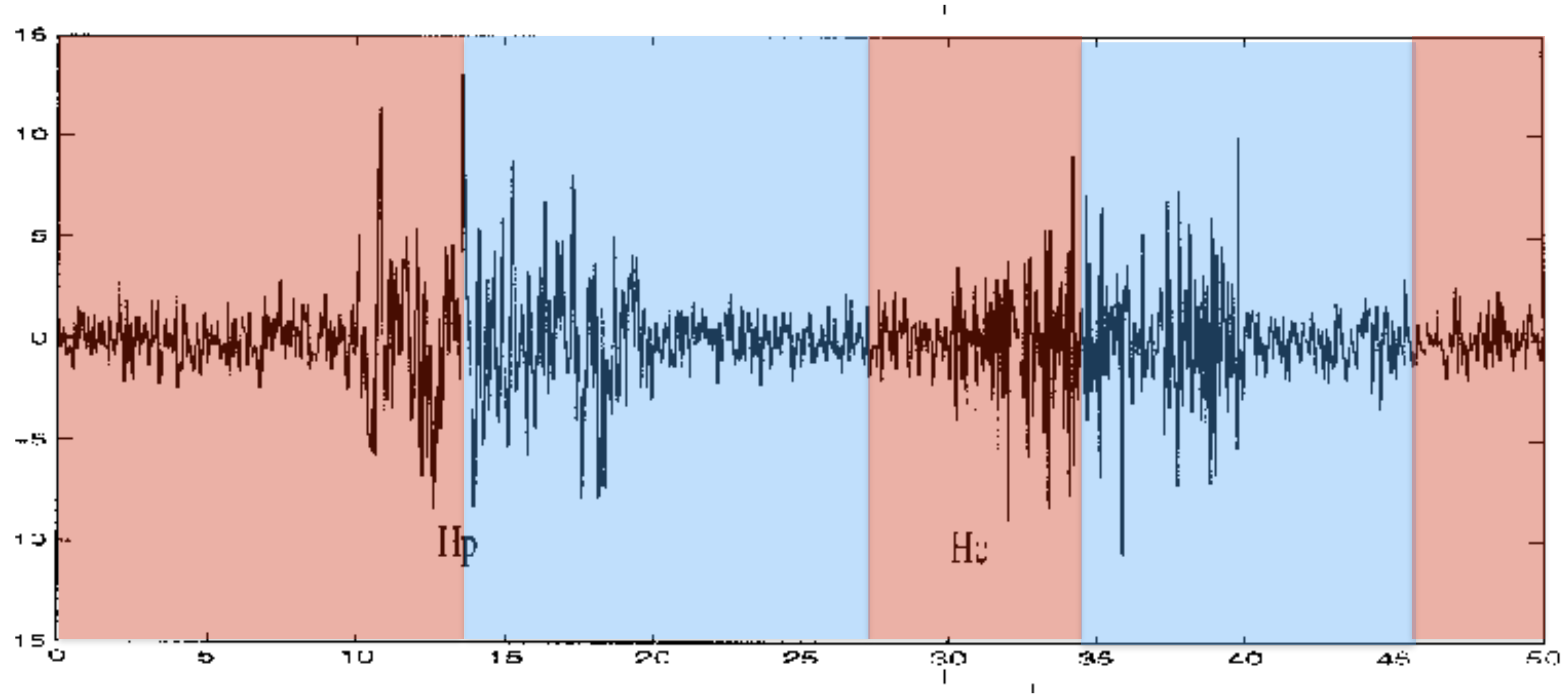
where $\mathrm{pen}(\mathcal{T})$ is an appropriate measure of the complexity of a segmentation $\mathcal{T}$.

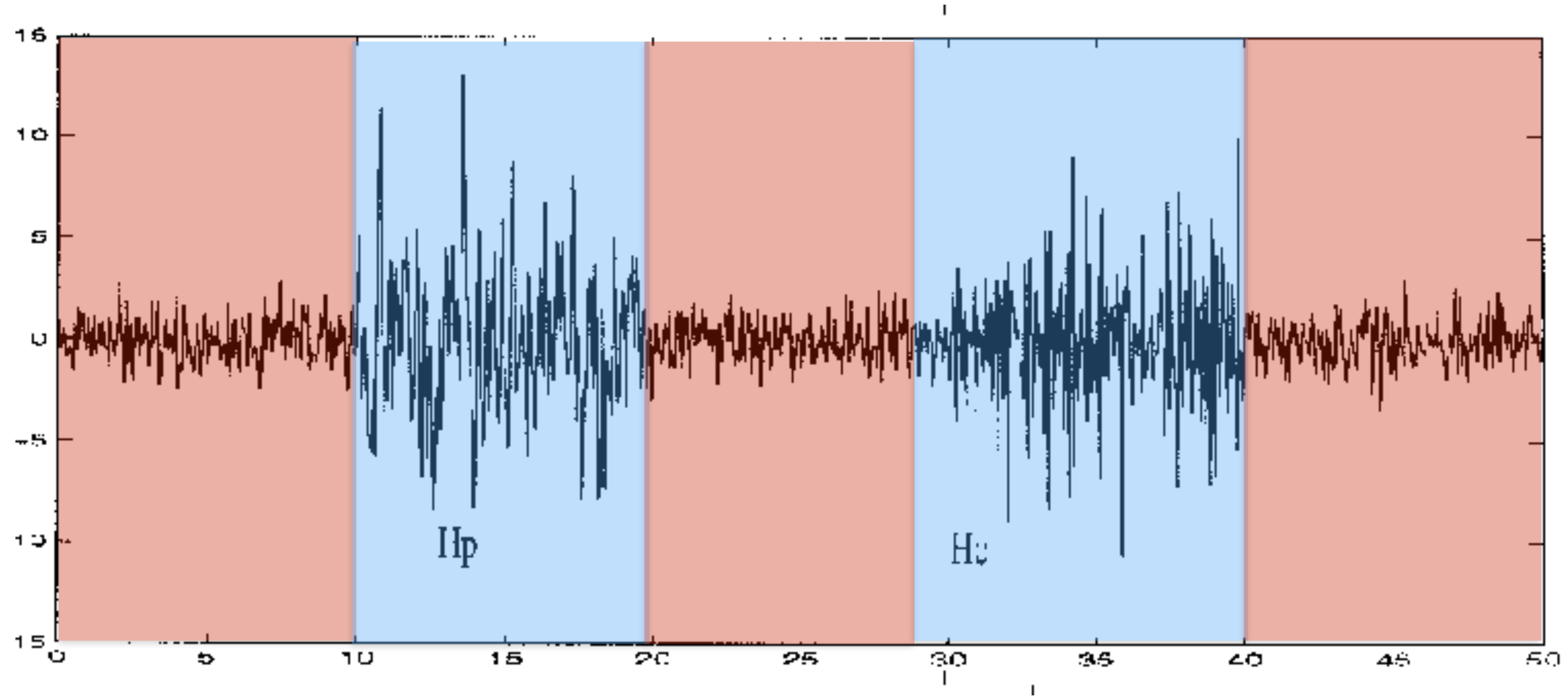
Q2

給定任一分割如何計算損失？

$$V(\mathcal{T}, y) := \sum_{k=0}^{K} c(y_{t_k..t_{k+1}})$$

1. 兩張圖有一樣的改變點數
2. 哪一張分的比較好？
3. 因為同一個區段內的序列行為較相近
4. 同個區段內的生成機制較類似





Q2

給定任一分割如何計算損失？

1. 同區段內行為較類似
2. 同區段內為相同的機率分佈
3. 因此可將MLE當成cost function
4. 選擇可能性最大的組合

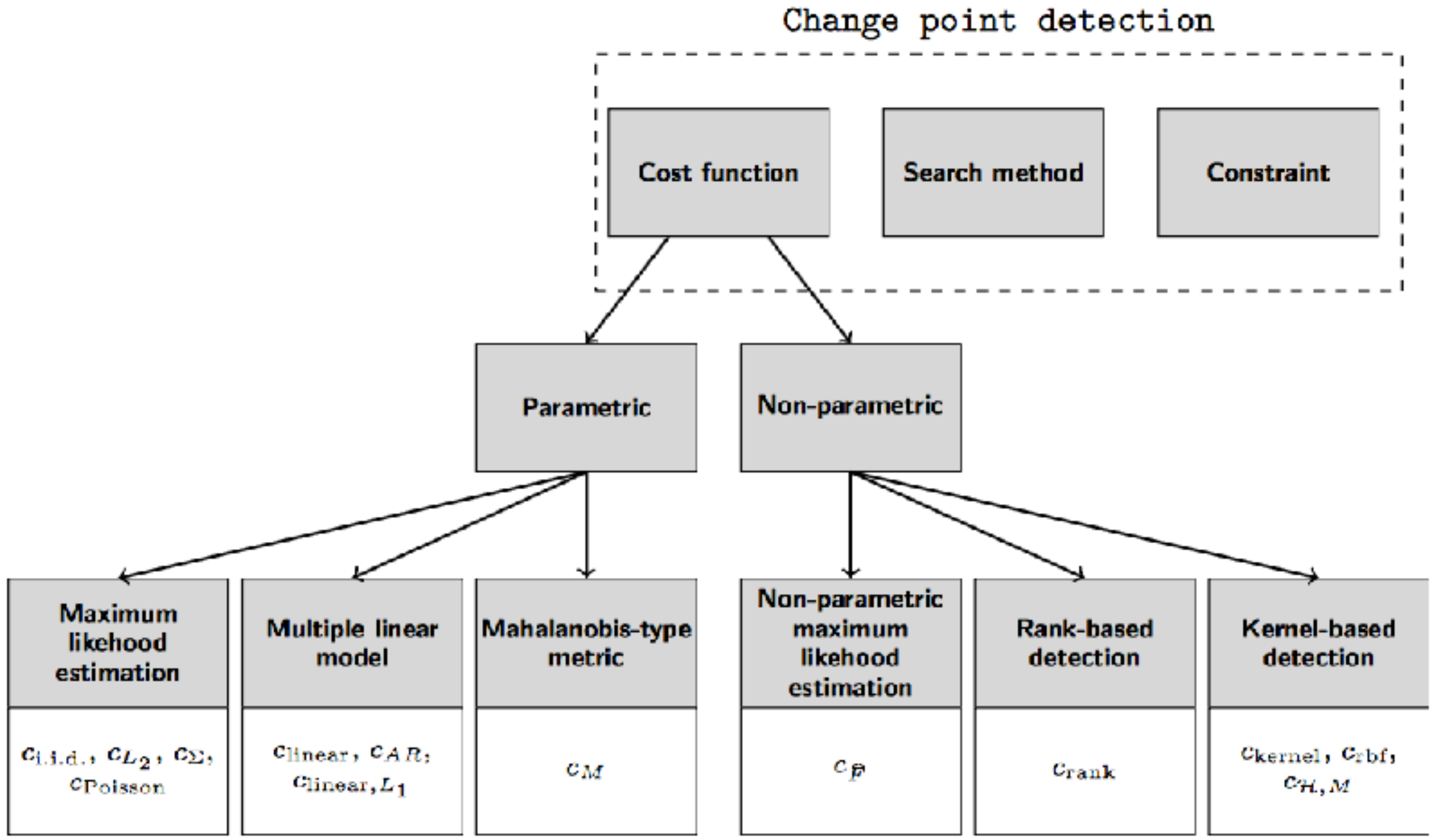


Figure 6: Typology of the cost functions described in Section 4.

Q2

給定任一分割如何計算損失？

1. 同區段內行為較類似
2. 同區段內為相同的機率分佈
3. 因此可將MLE當成cost function
4. 選擇可能性最大的組合

$$c_{i.i.d.}(y_{a..b}) := -\sup_{\theta} \sum_{t=a+1}^{b} \log f(y_t|\theta).$$

$$c_{L_2}(y_{a..b}) := \sum_{t=a+1}^{b} \|y_t - \bar{y}_{a..b}\|_2^2$$

$$c_{\Sigma}(y_{a..b}) := (b-a) \log \det \widehat{\Sigma}_{a..b} + \sum_{t=a+1}^{b} (y_t - \bar{y}_{a..b})' \widehat{\Sigma}_{a..b}^{-1} (y_t - \bar{y}_{a..b})$$

Q2

給定任一分割如何計算損失？

1. $C_{i.i.d}$：一個時間區段內，序列每個值iid同一個density function f

$$c_{i.i.d.}(y_{a..b}) := -\sup_{\theta} \sum_{t=a+1}^{b} \log f(y_t|\theta).$$

2. C_{L2}：對平均值取2-norm（maximize Normal distribution MLE） -> mean shift

$$c_{L_2}(y_{a..b}) := \sum_{t=a+1}^{b} \|y_t - \bar{y}_{a..b}\|_2^2$$

3. C_{sig}：考慮二階動差，考慮mean shift, scale shift

$$c_{\Sigma}(y_{a..b}) := (b-a)\log\det\widehat{\Sigma}_{a..b} + \sum_{t=a+1}^{b}(y_t - \bar{y}_{a..b})'\widehat{\Sigma}_{a..b}^{-1}(y_t - \bar{y}_{a..b})$$

Q3

透過演算法設計，將所有change point可能性計算完、提供Approximate方式計算，找出最優解

1. Optimal方式將所有可能性都做計算，並計算全局最優解，計算成本高
2. Approximate方式採取window, greedy方式找尋局部最優解

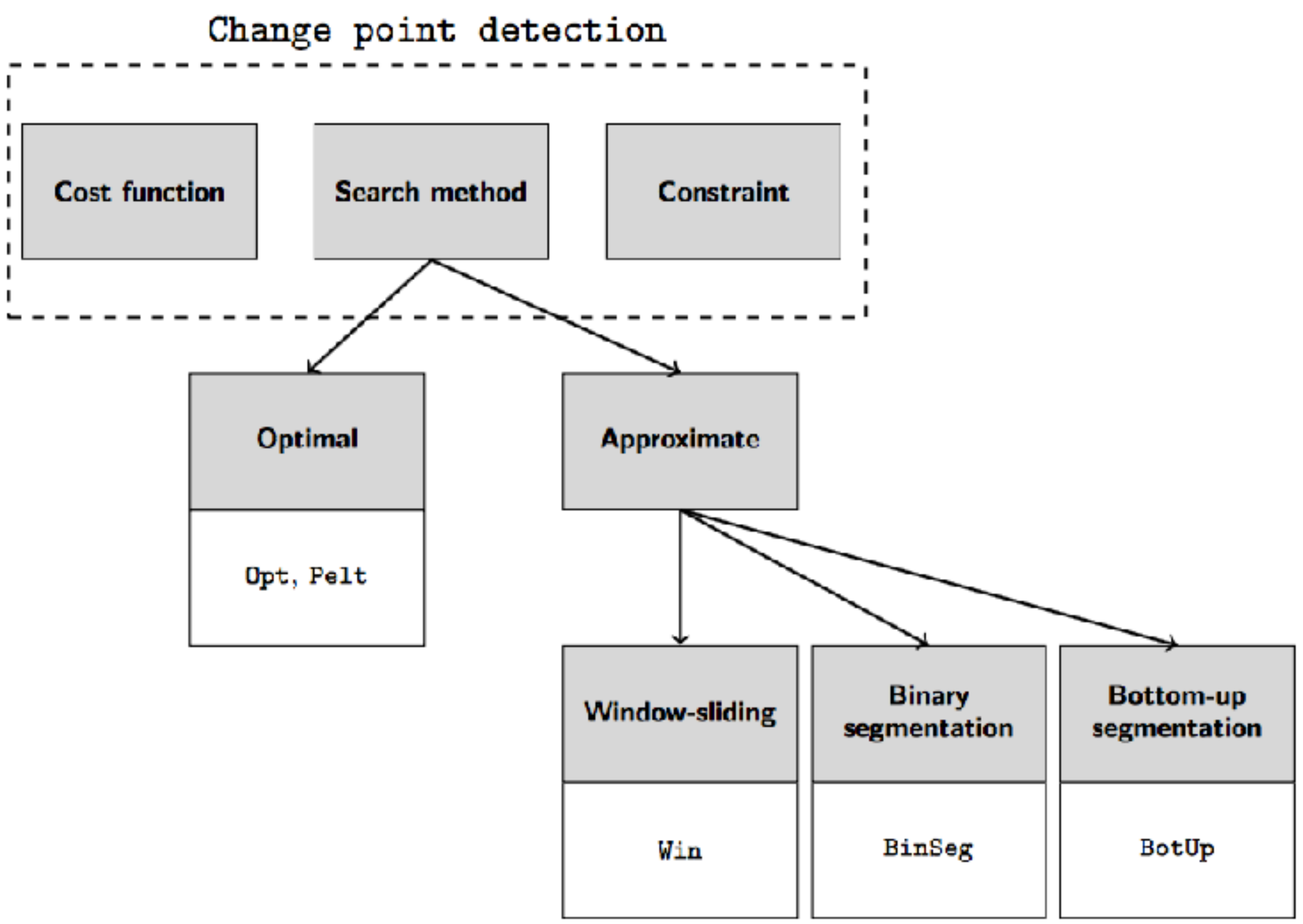


Figure 7: Typology of the search methods described in Section 5.

Optimal

以Optimal方式找尋P1,P2的解

Opt

K known

$$\min_{|\mathcal{T}|=K} V(\mathcal{T}, y=y_{0..T}) = \min_{0=t_0<t_1<\cdots<t_K<t_{K+1}=T} \sum_{k=0}^{K} c(y_{t_k..t_{k+1}})$$
$$= \min_{t\le T-K} \left[c(y_{0..t}) + \min_{t=t_0<t_1<\cdots<t_{K-1}<t_K=T} \sum_{k=0}^{K-1} c(y_{t_k..t_{k+1}}) \right]$$
$$= \min_{t\le T-K} \left[c(y_{0..t}) + \min_{|\mathcal{T}|=K-1} V(\mathcal{T}, y_{t..T}) \right] \quad (21)$$

評價函數可以寫成遞迴形式，因此可以使用Dynamic Programming求解

Pelt(Pruned Exact Linear Time)

K unknown

$$\text{if } \left[\min_{\mathcal{T}} V(\mathcal{T}, y_{0..t}) + \beta|\mathcal{T}|\right] + c(y_{t..s}) \ge \left[\min_{\mathcal{T}} V(\mathcal{T}, y_{0..s}) + \beta|\mathcal{T}|\right] \text{ holds,}$$
$$\text{then } t \text{ cannot be the last change point prior to } T. \quad (23)$$

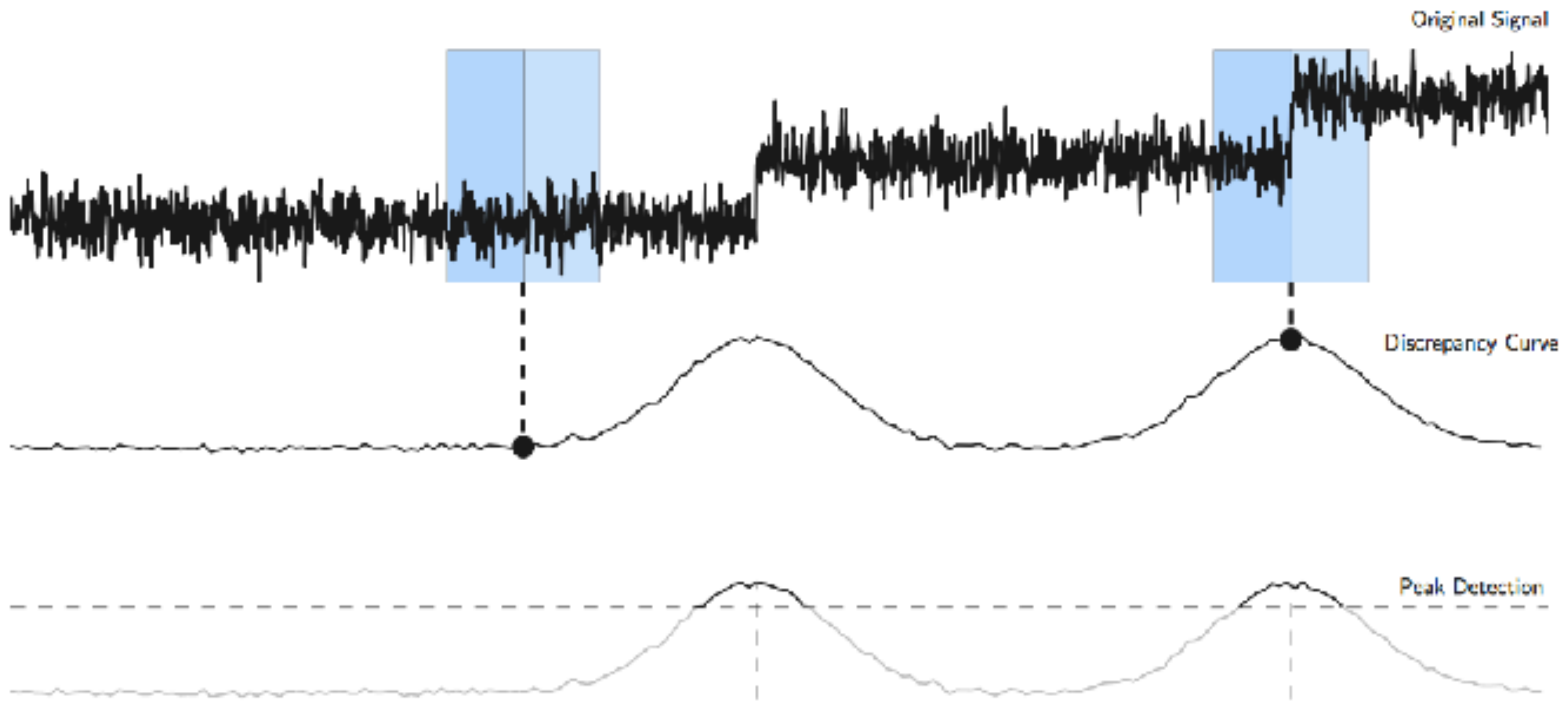
若上式成立，則將上述組合排除，因此可以大幅降低運算數量。

Approximate

以Approximate方式找解，找到的
可能非最佳解但速度較快

Window sliding

K known

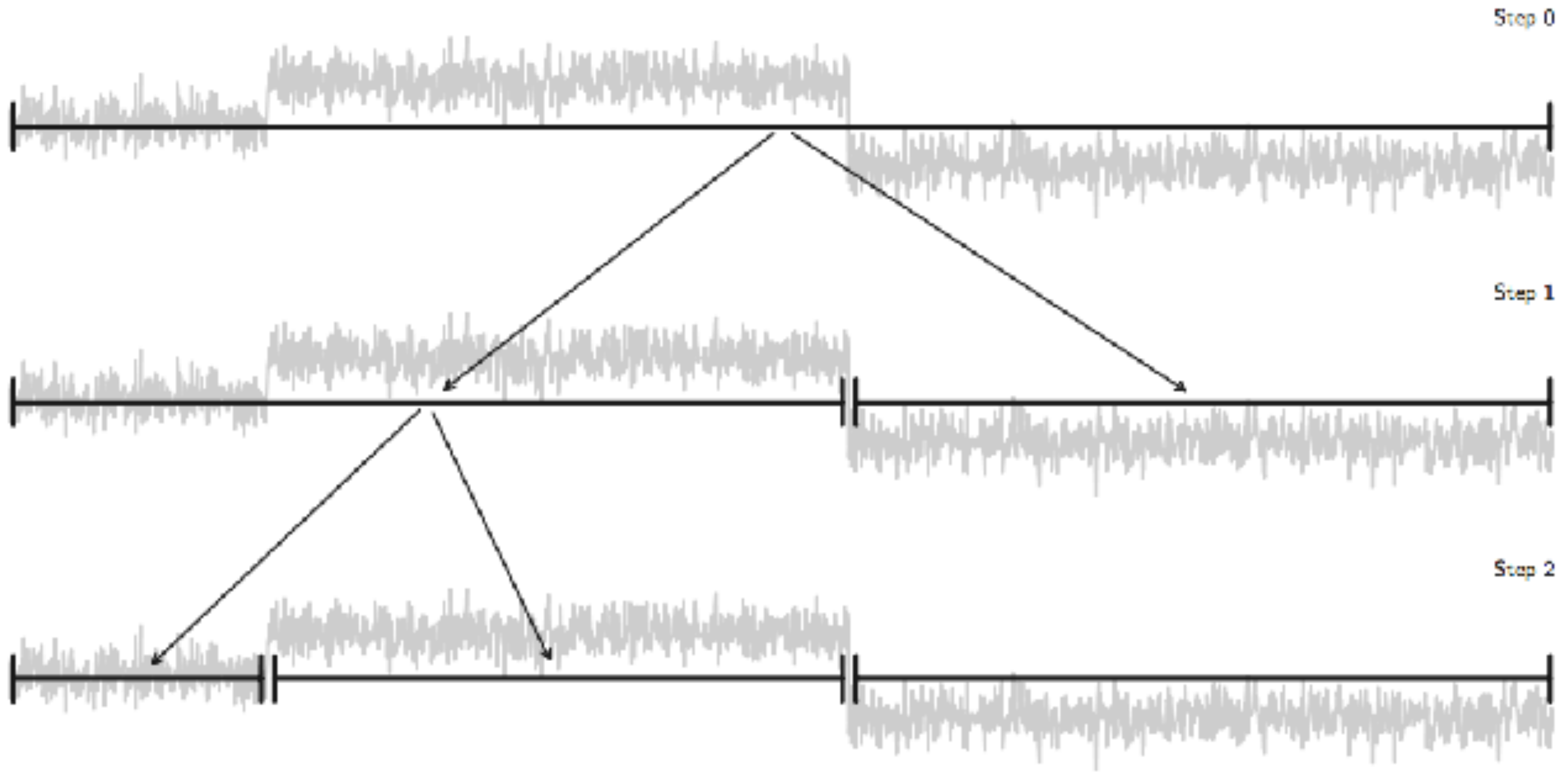


$$d(y_{a..t}, y_{t..b}) = c(y_{a..b}) - c(y_{a..t}) - c(y_{t..b})$$

使用Sliding Window，計算每點t的
discrepency分數，最後再取peak

Binary segmentation

K known



$$\hat{t}^{(1)} := \mathrm{argmin}_{1 \le t < T-1} \underbrace{c(y_{0..t}) + c(y_{t..T})}_{V(\mathcal{T}=\{t\})}.$$

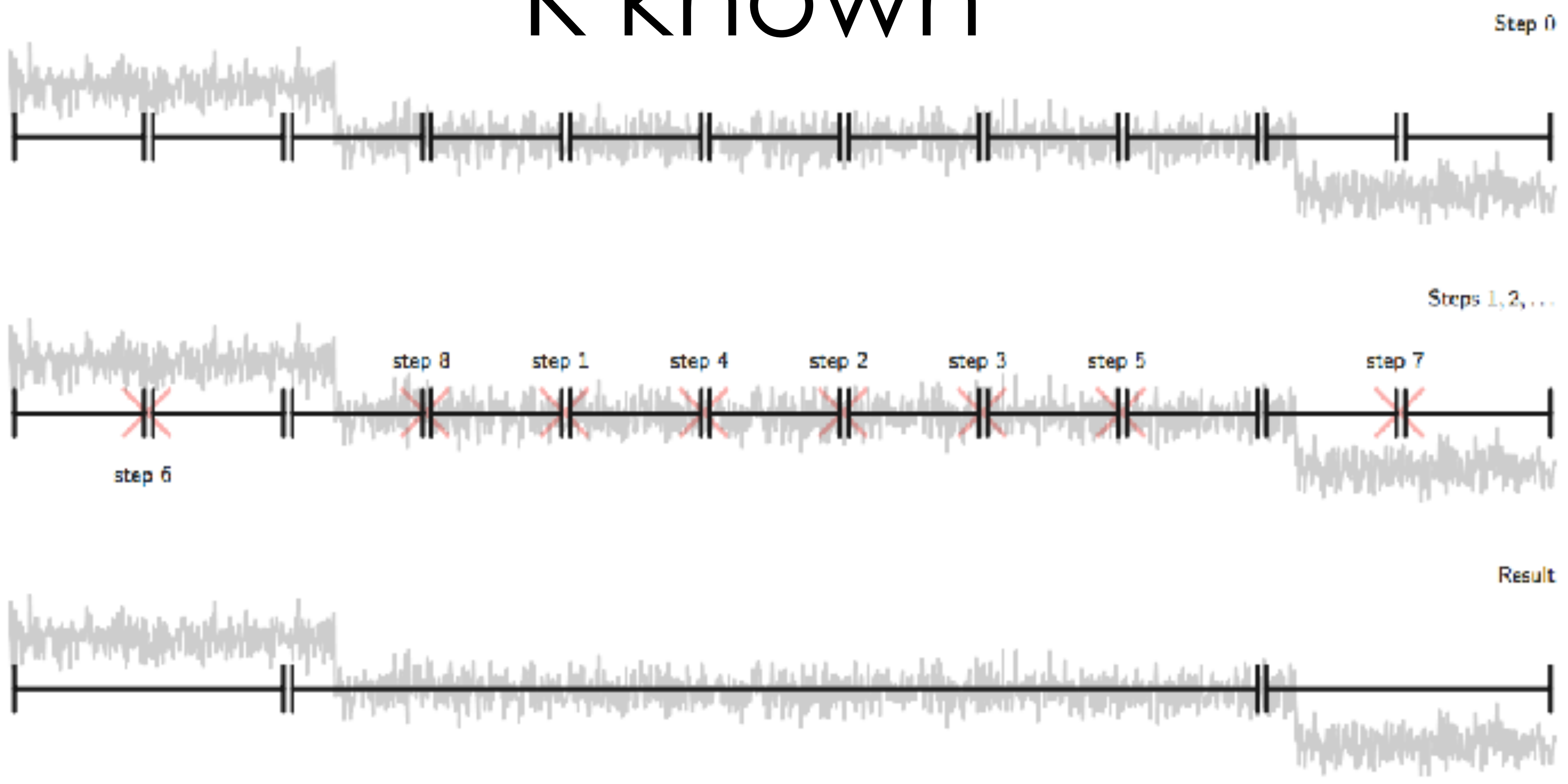
每次都找尋切分後cost總和最小的
點

Approximate

以Approximate方式找解，找到的
可能非最佳解但速度較快

Bottom-up segmentation

K known



$$d(y_{a..t}, y_{t..b}) = c(y_{a..b}) - c(y_{a..t}) - c(y_{t..b})$$

一開始將序列分成若干份，將相鄰的合併，計算discrepency，每計算一輪將discrepency最小的pair合併直到剩下K個point

測試結果

Piecewise Stationary的假設很強，若違背不容易看出來效果

1. 目前Pelt(K unknown)cost僅支援l1,l2,rbf，若要使用其他的cost需自行開發
2. 若想使用須將序列轉為piecewise stationary再做會有較佳效果