Detección de puntos de cambio en diversas trayectorias

CHAIR Soulaiman

May 2025

Índice

1. Introducción

La detección de puntos de cambio consiste en identificar momentos en el tiempo donde las propiedades estadísticas de una serie de datos presentan variaciones. Esta técnica es clave en distintos ámbitos, como las finanzas, el monitoreo ambiental y el control de calidad, ya que entender los cambios en las tendencias de los datos permite mejorar la toma de decisiones y el análisis predictivo. Detectar estos cambios ayuda a los analistas a reaccionar de forma anticipada ante patrones o anomalías que puedan surgir en los datos.

Lo importante de detectar puntos de cambio es que ayuda a entender mejor los datos y tomar decisiones acertadas. Por ejemplo, en el clima, una variación repentina en la temperatura o la presión puede anticipar tormentas o cambios bruscos en el tiempo. En la salud, notar alteraciones en los signos vitales de un paciente puede servir para prevenir problemas graves. En general, identificar estos cambios a tiempo permite reaccionar rápido y adaptarse mejor.

Existen varios métodos para detectar puntos de cambio, como las pruebas estadísticas, que comparan los datos antes y después del cambio, o los algoritmos de aprendizaje automático, que encuentran patrones en datos más complejos. También se utilizan enfoques bayesianos, que ajustan la detección a medida que se recibe nueva información. Cada uno de estos métodos tiene sus ventajas dependiendo del tipo de datos y el objetivo del análisis.

Aunque la detección de puntos de cambio tiene muchas ventajas, también presenta varios desafíos. Uno de los problemas más comunes es elegir el método adecuado, porque cada técnica puede dar resultados distintos dependiendo del tipo de datos que tengas. Además, el ruido y los valores atípicos pueden complicar el proceso, provocando falsos positivos o incluso dejando cambios sin detectar. Por eso, es importante que los profesionales tomen en cuenta las características de sus datos al aplicar estos métodos.

Hay varias herramientas y programas que puedes usar para detectar puntos de cambio en los datos. Lenguajes como Python y R tienen bibliotecas especiales para esto. Por ejemplo, en Python puedes usar la biblioteca ruptures, que te permite aplicar diferentes algoritmos para detectar cambios, visualizar los resultados y ajustar los parámetros según lo necesites. En R, existe el paquete changepoint, que también facilita este tipo de análisis.

2. Estado del arte

El problema de los puntos de cambio ha sido y sigue siendo un tema importante en el estudio de trayectorias, ya que permite identificar momentos clave en los que un proceso cambia. Los primeros trabajos sobre este tema fueron de Page en 1954 y 1955, quien presentó esquemas de inspección continua para detectar cambios en los datos. En 1955, también propuso una prueba estadística para identificar cambios en parámetros sin conocer el punto exacto. Más tarde, Chernoff y Zacks (1964) utilizaron un enfoque bayesiano para estimar la media de una distribución normal que cambia con el tiempo. Más recientemente, Gichuhi, Franke y Weizsacker (2008) utilizaron redes neuronales para detectar puntos de cambio en datos binarios, lo que abrió nuevas posibilidades con el aprendizaje automático.
Muchos investigadores también han trabajado en los problemas relacionados con los múltiples puntos de cambio, entre ellos. Bai y Perron (1998) propusieron métodos para estimar y probar modelos con múltiples cambios estructurales. Pan y Chen (2006) usaron un criterio de información modificado para detectar varios puntos de cambio en los datos. Por su parte, Yao (1984) planteó un enfoque bayesiano para estimar funciones escalón ruidosas, mientras que Barry y Hartigan (1992) introdujeron modelos de partición para manejar los puntos de cambio. Lee (1998) trabajó en estimar el número de puntos de cambio utilizando un enfoque bayesiano. Más recientemente, Lavielle (1999) se centró en detectar múltiples puntos de cambio en secuencias de variables dependientes, y Lai, Liu y Xing (2005) analizaron modelos autorregresivos con volatilidad constante para tratar los cambios en los parámetros de los datos.

Uno de los más conocidos es la segmentación binaria, que es bastante popular en este tipo de estudios. Es un método aproximado con una complejidad computacional de $O(n \log n)$, donde n es el número de puntos de datos. Por otro lado, el algoritmo de vecindad de segmentos (Scott y Knott, 1974) es más preciso, ya que busca de manera exacta en todo el espacio de segmentación, pero su costo computacional es mucho mayor, $O(Qn^2)$, donde Q es el número máximo de puntos de cambio y n es el número de datos. Si el número de puntos de cambio crece linealmente con el aumento de los datos, el costo puede llegar a ser $O(n^3)$, lo que lo hace menos eficiente a medida que se acumulan más datos.

El método de partición óptima (Auger y Lawrence, 1989) es otra opción, que mejora la eficiencia de la vecindad de segmentos, pero sigue sin igualar la rapidez de la segmentación binaria. A pesar de esto, es un enfoque exacto, con una complejidad computacional de $O(n^2)$, y se aplica en un tipo más reducido de problemas.
El método PELT (Pruned Exact Linear Time) es una técnica bastante eficiente para detectar múltiples puntos de cambio en series temporales. En el estudio de Killick, Eckley y Jonathan (2011), este método se aplicó a series de datos oceánicos y mostró ser muy rápido en comparación con otras técnicas, ya que puede manejar grandes volúmenes de datos en un tiempo relativamente corto.

La clave de su eficiencia está en que PELT realiza una poda en el proceso de búsqueda, lo que significa que no tiene que revisar todas las posibles divisiones de los datos, sino solo las más relevantes, haciendo que funcione en un tiempo O(n). Por otro lado, en el trabajo de Madon y Hingrat (2014), PELT se utilizó en combinación con un árbol de clasificación para analizar el comportamiento de los animales. Este enfoque ayudó a detectar cambios significativos en sus movimientos de manera rápida y precisa, lo que es especialmente útil cuando se tiene que manejar una gran cantidad de datos sobre los patrones de movimiento de los animales.

3. Evaluación

Los métodos para detectar puntos de cambio se pueden evaluar de dos formas: una es demostrando ciertas propiedades matemáticas de los algoritmos y la otra es hacerlo de forma empírica, calculando distintas métricas.

En lo que sigue, al conjunto de los puntos de cambio verdaderos lo denoto como $\mathcal{T}^* = \{t_1^*, \ldots, t_K^*\}$, y al conjunto de los puntos de cambio estimados lo denoto como $\widehat{\mathcal{T}} = \{\hat{t}_1, \ldots, \hat{t}_{\hat{K}}\}$.

3.1. F1-Score

La métrica F1-Score emerge como indicador robusto para evaluar el rendimiento en esta tarea. Su cálculo se basa en dos componentes esenciales:

Precisión (Prec): Mide la fiabilidad de las detecciones

$$\text{Prec} = \frac{\text{Detecciones correctas}}{\text{Total de detecciones}} = \frac{|\text{Tp}|}{\hat{K}}$$

Exhaustividad (Rec): Evalúa la capacidad de descubrimiento

$$\text{Rec} = \frac{\text{Detecciones correctas}}{\text{Total de puntos reales}} = \frac{|\text{Tp}|}{K^*}$$

Considero que una detección es válida cuando existe coincidencia dentro de un margen M muestral:

$$\text{Tp} = \left\{ t^* \in \mathcal{T}^* \mid \exists\, \hat{t} \in \widehat{\mathcal{T}} \,:\, |\hat{t} - t^*| < M \right\}$$

F1-score se define como la media armónica entre la precisión y el recall:

$$\text{F1} = 2 \cdot \frac{\text{Prec} \cdot \text{Rec}}{\text{Prec} + \text{Rec}} \in [0, 1]$$

El mejor valor posible para esta métrica es 1, indicando una detección perfecta, mientras que su peor valor es 0.

3.2. Hausdorff

Desde un punto de vista formal, esta métrica corresponde a la mayor distancia temporal entre un punto de cambio y su correspondiente estimación:

$$\text{Hausdorff}(\mathcal{T}^*, \widehat{\mathcal{T}}) = \text{máx} \left\{ \underbrace{\underset{\hat{t}\in\widehat{\mathcal{T}}}{\text{máx}} \underset{t^*\in\mathcal{T}^*}{\text{mín}} |\hat{t} - t^*|}_{\text{Error máximo de detección}} \quad , \quad \underbrace{\underset{t^*\in\mathcal{T}^*}{\text{máx}} \underset{\hat{t}\in\widehat{\mathcal{T}}}{\text{mín}} |\hat{t} - t^*|}_{\text{Error máximo de omisión}} \right\}$$

donde:

- El primer término evalúa la máxima distancia de cualquier punto detectado al punto real más cercano
- El segundo término mide la máxima distancia de cualquier punto real al punto detectado más cercano

Este valor representa el peor error cometido por el algoritmo que genera el conjunto de puntos estimados $\widehat{\mathcal{T}}$, y se expresa en número de muestras. Cuando su valor es cero, significa que ambos conjuntos de puntos de cambio coinciden exactamente. Por el contrario, cuanto mayor sea su valor, mayor será la distancia existente entre algún punto de cambio verdadero en $\mathcal{T}^*$ y el punto estimado más cercano en $\widehat{\mathcal{T}}$, o viceversa.

3.3. Índice de Rand

La métrica fundamental Índice de Rand cuantifica la precisión en la detección de puntos de cambio. Esta medida estadística compara la similitud entre la segmentación obtenida $\widehat{\mathcal{T}}$ y la segmentación de referencia $\mathcal{T}^*$, proporcionando una evaluación global del rendimiento del algoritmo.

El Índice de Rand calcula la proporción de pares de muestras que son:

- **Concordantes**:
  - Pertenecen al mismo segmento en ambas segmentaciones
  - Pertenecen a segmentos diferentes en ambas segmentaciones
- **Discordantes**:
  - Asignados al mismo segmento en una segmentación y a diferentes en la otra

Para formalizar esta idea, se definen las siguientes relaciones para un conjunto de puntos de cambio $\mathcal{T}$:

$$\text{SameSeg}(\mathcal{T}) := \{(s,t) \mid 1 \leq s < t \leq T \text{ tales que } s \text{ y } t \text{ se encuentran en el mismo segmento según } \mathcal{T}\}$$
$$\text{DiffSeg}(\mathcal{T}) := \{(s,t) \mid 1 \leq s < t \leq T \text{ tales que } s \text{ y } t \text{ pertenecen a segmentos distintos según } \mathcal{T}\}$$

A partir de estas definiciones, el Índice de Rand se expresa como:

$$\mathrm{RI}(\mathcal{T}^*, \widehat{\mathcal{T}}) := \frac{|\mathrm{SameSeg}(\widehat{\mathcal{T}}) \cap \mathrm{SameSeg}(\mathcal{T}^*)| + |\mathrm{DiffSeg}(\widehat{\mathcal{T}}) \cap \mathrm{DiffSeg}(\mathcal{T}^*)|}{T(T-1)/2}$$

Este valor se encuentra normalizado en el intervalo entre 0 (cuando no existe ningún acuerdo entre las segmentaciones) y 1 (cuando las segmentaciones son idénticas).

4. funciones de costo

Esta sección presenta el primer elemento definitorio de los métodos de detección de cambios, que son las funciones de costo. En la mayoría de los casos, estas funciones se derivan a partir de un modelo de señal. A continuación, se agrupa los modelos y sus funciones de costo asociadas en dos categorías: paramétricas y no paramétricas.

4.1. Función de Costo L1

Esta función de costo detecta cambios en la mediana de una señal. En general, es un estimador robusto para detectar desplazamientos en el punto central (ya sea media, mediana o moda) de una distribución . Formalmente, dado un segmento de señal $\{y_t\}_{t\in I}$ donde I representa el intervalo de análisis, el costo se calcula como:

$$c(y_I) = \sum_{t\in I} \|y_t - \tilde{y}\|_1$$

donde $\tilde{y}$ corresponde a la mediana muestral del segmento.

4.2. Función de Costo L2

La función CostL2 cuantifica la variabilidad alrededor de la media muestral mediante la norma euclídea al cuadrado. Para un segmento de señal $\{y_t\}_{t\in I}$ donde I denota el intervalo de estudio, el costo se define como:

$$c(y_I) = \sum_{t\in I} \|y_t - \bar{y}\|_2^2$$

donde $\tilde{y}$ corresponde a la mediana muestral del segmento.

4.3. Función de Costo Normal

Esta función de costo permite detectar cambios tanto en la media como en la matriz de covarianza de una secuencia de variables aleatorias gaussianas multivariadas. Formalmente, para un segmento de señal $\{y_t\}_{t\in I}$ con $y_t \in \mathbb{R}^d$, la función de costo se define como:

$$c(y_I) = |I| \cdot \log\det(\widehat{\Sigma}_I + \epsilon I_d)$$

donde:

- $\widehat{\Sigma}_I = \frac{1}{|I|-1} \sum_{t \in I} (y_t - \bar{y}_I)(y_t - \bar{y}_I)^\top$ es la matriz de covarianza muestral del segmento.
- $\bar{y}_I$ es la media empírica del segmento.
- $\epsilon > 0$ es un término de regularización (típicamente $\epsilon = 10^{-6}$) que se añade para evitar problemas numéricos en matrices mal condicionadas.
- I_d es la matriz identidad de dimensión $d \times d$.

4.4. Cambio de media con kernel (CostRbf)

La función `CostRbf` opera mediante el mapeo de los datos a un espacio de características $\mathcal{H}$ mediante la función $\Phi(\cdot)$, donde se analizan las propiedades estadísticas. Para un segmento $\{y_t\}_{t \in I}$ con $y_t \in \mathbb{R}^d$, el costo se calcula como:

$$c(y_I) = \sum_{t \in I} \|\Phi(y_t) - \bar{\mu}\|^2_{\mathcal{H}}$$

donde $\bar{\mu} = \frac{1}{|I|} \sum_{t \in I} \Phi(y_t)$ representa la media en el espacio de características. El kernel radial (RBF) implementado tiene la forma:

$$k(x, y) = \exp\left(-\gamma \|x - y\|^2\right), \quad \gamma > 0$$

donde $\gamma = 1/\text{mediana}(\{\|y_i - y_j\|^2\}_{i,j})$.

4.5. Cambio de media con kernel (CostCosine)

La función de costo evalúa la variabilidad en el espacio de características $\mathcal{H}$ generado por el kernel coseno:

$$c(y_{a..b}) = \sum_{t=a}^{b-1} \|\Phi(y_t) - \bar{\mu}_{a..b}\|^2_{\mathcal{H}}$$

donde:

- $\Phi : \mathbb{R}^d \to \mathcal{H}$ es el mapeo al espacio de características
- $\bar{\mu}_{a..b} = \frac{1}{b-a} \sum_{t=a}^{b-1} \Phi(y_t)$ es la media empírica en $\mathcal{H}$
- El kernel coseno se define como:

$$k(x, y) = \frac{\langle x, y \rangle}{\|x\|_2 \|y\|_2} \in [-1, 1]$$

donde $\langle \cdot \mid \cdot \rangle$ y $\| \cdot \|$ corresponden al producto escalar y la norma euclidiana respectivamente. Dicho de otro modo, equivale al producto punto normalizado en norma L2 de los vectores.

4.6. Cambio en modelo lineal (CostLinear)

Consideremos una serie temporal $\{y_t\}_{t=1}^n$ con posibles puntos de cambio en $t_1, t_2, \ldots, t_k$. El modelo de regresión por segmentos se define como:

$$y_t = x_t' \delta_j + \varepsilon_t, \quad t_j \leq t < t_{j+1}$$

donde:

- $y_t \in \mathbb{R}$: Variable respuesta
- $x_t \in \mathbb{R}^p$: Vector de covariables
- $\delta_j \in \mathbb{R}^p$: Coeficientes de regresión para el j-ésimo segmento
- ε_t: Término de error con $\mathbb{E}[\varepsilon_t] = 0$

Las estimaciones por mínimos cuadrados de las fechas de ruptura se obtienen minimizando la suma de los residuos al cuadrado. Formalmente, la función de costo asociada a un intervalo I se define como:

$$c(y_I) = \min_{\delta \in \mathbb{R}^p} \sum_{t \in I} \|y_t - \delta' x_t\|_2^2$$

4.7. Cambio lineal continuo (CostCLinear)

Dado un conjunto de knots $\{t_k\}_{k=1}^K$, el spline lineal continuo $f : \mathbb{R} \to \mathbb{R}^d$ se define mediante:

- **Comportamiento afín por intervalos**:

$$f(t) = \alpha_k (t - t_k) + \beta_k, \quad \alpha_k, \beta_k \in \mathbb{R}^d, \quad t \in [t_k, t_{k+1})$$

- **Condición de continuidad**:

$$\lim_{t \to t_k^-} f(t) = \lim_{t \to t_k^+} f(t), \quad \forall k$$

La función de costo `CostCLinear` mide el error al aproximar la señal con una spline lineal. Formalmente, se define para $0 < a < b \leq T$ como:

$$c(y_{a,b}) := \sum_{t=a}^{b-1} \left\| y_t - y_{a-1} - \frac{t - a + 1}{b - a} (y_{b-1} - y_{a-1}) \right\|^2$$

y se toma $c(y_{0,b}) = c(y_{1,b})$.

4.8. Función de costo basada en rangos (CostRank)

La clave de este método reside en la transformación de los datos originales $\{y_t\}_{t=1}^T$ a sus rangos $\{r_t\}_{t=1}^T$, donde:

$$r_t = \text{rango}(y_t \text{ en } \{y_1, \ldots, y_T\})$$

Para un segmento $y_{a..b}$, la función de costo se define como:

$$c_{\text{rank}}(a, b) = -(b-a)\bar{r}'_{a..b}\widehat{\Sigma}_r^{-1}\bar{r}_{a..b}$$

donde:

- $\bar{r}_{a..b} = \frac{1}{b-a}\sum_{t=a+1}^{b} r_t$ es la media de rangos en el segmento
- $\widehat{\Sigma}_r$ es la matriz de covarianza estimada de los rangos completos

4.9. Detección de cambios con una métrica de tipo Mahalanobis (CostMl)

Dada una matriz semidefinida positiva $M \in \mathbb{R}^{d\times d}$, definimos la pseudométrica:

$$\|x - y\|_M^2 = (x-y)^T M(x-y)$$

Para un segmento de señal $\{y_t\}_{t\in I}$, la función de costo se calcula como:

$$c(y_I) = \sum_{t\in I} \|y_t - \bar{\mu}\|_M^2$$

donde $\bar{\mu} = \frac{1}{|I|}\sum_{t\in I} y_t$ es la media empírica del segmento.

4.10. Cambio de modelo autorregresivo (CostAR)

Considerando una serie temporal $\{y_t\}_{t=1}^n$ con posibles puntos de cambio en $\{t_j\}_{j=1}^k$, el modelo AR(p) por segmentos se define como:

$$y_t = \sum_{i=1}^{p} \delta_{j,i} y_{t-i} + \epsilon_t, \quad t_j \le t < t_{j+1}$$

donde:

- p: Orden del modelo (seleccionado mediante AIC en mi implementación)
- $\delta_j \in \mathbb{R}^p$: Coeficientes AR para el j-ésimo segmento
- ϵ_t: Innovaciones con $\mathbb{E}[\epsilon_t] = 0$, $\text{Var}(\epsilon_t) = \sigma^2$

La función de costo implementada minimiza la suma de residuos al cuadrado:

$$c(y_I) = \min_{\delta\in\mathbb{R}^p} \sum_{t\in I} (y_t - \delta' z_t)^2$$

con $z_t = [y_{t-1}, \ldots, y_{t-p}]'$.

5. Métodos de búsqueda de puntos de cambio

Esta sección presenta el segundo elemento definitorio de los métodos de detección de cambios, concretamente el **método de búsqueda**. Los algoritmos de búsqueda determinan cómo se exploran las posibles configuraciones de puntos de cambio en la serie temporal, afectando tanto a la precisión como a la eficiencia computacional del método.

5.1. Segmentación Binaria (BinSeg)

El algoritmo de **Segmentación Binaria (BinSeg)** es un método iterativo para detectar puntos de cambio en series temporales que opera mediante una estrategia greedy. En cada iteración, el algoritmo identifica el punto de cambio óptimo que minimiza la suma de costos de los segmentos adyacentes:

$$\hat{t}^{(k)} = \underset{a<t<b}{\operatorname{argmin}} \left[c(y_{a..t}) + c(y_{t..b}) \right] \tag{1}$$

donde $c(\cdot)$ representa típicamente el error cuadrático medio. A pesar de su eficiencia computacional ($\mathcal{O}(n \log n)$), el enfoque greedy implica que cada punto de cambio se estima condicionado a los cambios anteriores, lo que puede afectar la optimalidad global.

Algorithm 1 Algoritmo BinSeg

Require: Señal $\{y_t\}_{t=1}^{T}$, función de costo $c(\cdot)$, criterio de parada
Ensure: Conjunto L de puntos de cambio estimados

Inicializar $L \leftarrow \emptyset$ ▷ Lista de puntos de cambio
repeat
  $k \leftarrow |L|$ ▷ Número actual de puntos de cambio
  $t_0 \leftarrow 0$, $t_{k+1} \leftarrow T$ ▷ Límites del segmento completo
  if $k > 0$ **then**
    Ordenar $L = \{t_1, \ldots, t_k\}$ ascendentemente
  end if
  Inicializar arreglo G de longitud $k+1$ ▷ Ganancias por segmento
  for $i \leftarrow 0$ **to** k **do**
    Calcular ganancia:
    $G[i] \leftarrow c(y_{t_i..t_{i+1}}) - \min\limits_{t_i<t<t_{i+1}} \left[c(y_{t_i..t}) + c(y_{t..t_{i+1}}) \right]$
  end for
  $\hat{i} \leftarrow \operatorname{argmax}_i G[i]$ ▷ Segmento con máxima ganancia
  $\hat{t} \leftarrow \underset{t_{\hat{i}}<t<t_{\hat{i}+1}}{\operatorname{argmin}} \left[c(y_{t_{\hat{i}}..t}) + c(y_{t..t_{\hat{i}+1}}) \right]$
  $L \leftarrow L \cup \{\hat{t}\}$ ▷ Añadir nuevo punto de cambio
until criterio de parada sea satisfecho
return L

El algoritmo termina cuando se alcanza un número máximo de cambios o cuando la máxima ganancia $G[i]$ está por debajo de un umbral predefinido. Esta aproximación balancea eficiencia computacional con capacidad de detección, siendo particularmente útil cuando el número de segmentos es desconocido a priori.

5.2. Segmentación PELT (Pruned Exact Linear Time)

El método PELT es un algoritmo de detección exacta de puntos de cambio que combina optimalidad global con eficiencia computacional mediante técnicas de poda dinámica. A diferencia de métodos aproximados como BinSeg, PELT garantiza encontrar la partición óptima de la serie temporal minimizando:

$$\sum_{i=1}^{m+1} c(y_{\tau_{i-1}:\tau_i}) + \beta m \tag{2}$$

donde $c(\cdot)$ es la función de costo, β el parámetro de penalización y m el número de cambios. La clave del algoritmo reside en su capacidad para descartar particiones subóptimas manteniendo únicamente las soluciones relevantes.

Algorithm 2 Algoritmo PELT

Require: Señal $\{y_t\}_{t=1}^{T}$, función de costo $c(\cdot)$, penalización β
Ensure: Conjunto $L[T]$ de puntos de cambio estimados

1: Inicializar $Z[0] \leftarrow -\beta$ ▷ Costos acumulados
2: Inicializar $L[0] \leftarrow \emptyset$ ▷ Lista de cambios
3: Inicializar $\chi \leftarrow \{0\}$ ▷ Conjunto activo de índices
4: **for** $t \leftarrow 1$ **hasta** T **do**
5: Encontrar el punto óptimo previo:
6: $\hat{t} \leftarrow \arg\min_{s \in \chi} [Z[s] + c(y_{s:t}) + \beta]$
7: Actualizar costo acumulado:
8: $Z[t] \leftarrow Z[\hat{t}] + c(y_{\hat{t}:t}) + \beta$
9: Registrar cambios:
10: $L[t] \leftarrow L[\hat{t}] \cup \{\hat{t}\}$
11: Poda dinámica:
12: $\chi \leftarrow \{s \in \chi : Z[s] + c(y_{s:t}) \leq Z[t]\} \cup \{t\}$
13: **end for**
14: **return** $L[T]$

El algoritmo mantiene un conjunto activo χ de índices candidatos, actualizando en cada iteración tanto los costos acumulados $Z[t]$ como la lista de cambios $L[t]$. La regla de poda (línea 9) asegura que solo se conserven las particiones que potencialmente pueden llevar a la solución óptima global, reduciendo así la complejidad computacional sin sacrificar precisión.

5.3. Segmentación Bottom-Up

El método Bottom-Up es un enfoque no greedy para la detección de puntos de cambio que opera mediante fusión progresiva de segmentos. A diferencia de métodos como BinSeg que dividen la señal, este algoritmo sigue una estrategia de unificación: comienza con una partición inicial fina (definida por un parámetro de grilla δ) y fusiona iterativamente los pares de segmentos más similares hasta satisfacer un criterio de parada. Matemáticamente, el proceso optimiza:

$$\min_{L} \sum_{i=0}^{k} c(y_{t_i:t_{i+1}}) \tag{3}$$

donde $L = \{t_1, \ldots, t_k\}$ son los puntos de cambio y $c(\cdot)$ es la función de costo. La clave del método reside en su métrica de fusión:

$$G[i] = c(y_{t_{i-1}:t_{i+1}}) - \left[c(y_{t_{i-1}:t_i}) + c(y_{t_i:t_{i+1}})\right] \tag{4}$$

que cuantifica la ganancia al fusionar dos segmentos adyacentes.

Algorithm 3 Algoritmo Bottom-Up

Require: Señal $\{y_t\}_{t=1}^{T}$, función de costo $c(\cdot)$, tamaño de grilla $\delta > 2$, criterio de parada
Ensure: Conjunto L de puntos de cambio estimados
Inicializar $L \leftarrow \{\delta, 2\delta, \ldots, (\lfloor T/\delta \rfloor - 1)\delta\}$
repeat
  $k \leftarrow |L|$
  Ordenar $L = \{t_1, \ldots, t_k\}$ ascendentemente
  Inicializar array G de longitud $k - 1$
  for $i \leftarrow 1$ **to** $k - 1$ **do**
    Calcular ganancia de fusión:
    $G[i-1] \leftarrow c(y_{t_{i-1}:t_{i+1}}) - \left[c(y_{t_{i-1}:t_i}) + c(y_{t_i:t_{i+1}})\right]$
  end for
  $\hat{i} \leftarrow \operatorname{argmin}_i G[i]$ ▷ Seleccionar fusión óptima
  $L \leftarrow L \setminus \{t_{\hat{i}+1}\}$ ▷ Eliminar punto de cambio
until criterio de parada sea satisfecho
return L

5.4. Detección Window-Based

El método *Window-Based* es un algoritmo eficiente para detectar puntos de cambio mediante el análisis de discrepancia entre **segmentos adyacentes de tamaño** w. Utiliza dos fragmentos de la señal $\{y_t\}_{t=1}^{T}$ que se deslizan a lo largo de ella, comparando sus propiedades estadísticas mediante una medida de discrepancia derivada de la función de costo $c(\cdot)$.

$$d(y_{u..w}, y_{v..w}) = c(y_{u..w}) - [c(y_{v..w}) + c(y_{u..v})] \tag{5}$$

donde $u < v < w$ son índices temporales. La curva de discrepancia se define para cada punto t como:

$$Z[t] = c(y_{t-w..t+w}) - [c(y_{t-w..t}) + c(y_{t..t+w})] \tag{6}$$

Los picos en esta curva indican potenciales puntos de cambio, detectados mediante un procedimiento de búsqueda de máximos locales (PKSearch).

Algorithm 4 Algoritmo Window-Based

Require: Señal $\{y_t\}_{t=1}^{T}$, función de costo $c(\cdot)$, ancho de media ventana w, procedimiento PKSearch
Ensure: Conjunto L de puntos de cambio estimados

```
Inicializar Z ← [0, ..., 0]                          ▷ Array de longitud T
for t ← w to T − w do
    p ← (t − w)..t                                   ▷ Ventana izquierda
    q ← t..(t + w)                                   ▷ Ventana derecha
    r ← (t − w)..(t + w)                             ▷ Ventana combinada
    Z[t] ← c(y_r) − [c(y_p) + c(y_q)]                ▷ Discrepancia
end for
L ← PKSearch(Z)                                      ▷ Detección de picos
return L
```

6. Organización de los Notebooks

Esta sección se organiza en dos carpetas principales dentro de `notebooks/`:
La primera es varianza constante que contiene los notebooks que trabajan con una serie temporal de varianza constante:

- `Generacion_varianza_constante.ipynb` → Generación de la serie.
- `Deteccion_PELT.ipynb` → Aplicación del algoritmo **PELT**.
- `Deteccion_ChangeFinder.ipynb` → Aplicación del algoritmo **ChangeFinder**.
- `Deteccion_BinSeg.ipynb` → Aplicación del método **Binary Segmentation**.
- `Deteccion_WindowBased.ipynb` → Aplicación del método **Window-Based**.
- `Deteccion_BottomUp.ipynb` → Aplicación del método **Bottom-Up**.

La segunda es varianza variable que contiene los notebooks que trabajan con una serie temporal de varianza variable, donde los puntos de cambio son más difíciles de identificar:

- `Generacion_varianza_variable.ipynb` → Generación de la serie.
- `Deteccion_PELT.ipynb` → Evaluación del algoritmo **PELT**.
- `Deteccion_ChangeFinder.ipynb` → Evaluación del algoritmo **ChangeFinder**.
- `Deteccion_BinSeg.ipynb` → Aplicación del método **Binary Segmentation**.
- `Deteccion_WindowBased.ipynb` → Aplicación del método **Window-Based**.
- `Deteccion_BottomUp.ipynb` → Aplicación del método **Bottom-Up**.