Apprentissage de dictionnaire invariant aux déformations temporelles pour le clustering de séries temporelles

Colin Coërchon colin180302@gmail.com
Chloé Habasque habasquechloe29@gmail.com

January 9, 2025

1 Introduction et contributions

L'analyse de séries temporelles joue un rôle central dans de nombreuses applications, allant de la détection d'anomalies à la reconnaissance de motifs, en passant par le traitement de signaux complexes. Au cours de notre formation, nous avons étudié à plusieurs reprises des concepts tels que le **codage parcimonieux** et l'**apprentissage de dictionnaires**, qui sont des outils puissants pour représenter des données de manière compacte et interprétable. Ces approches permettent, entre autres, de reconstruire un signal à partir d'un petit nombre d'atomes appris à partir des données. Cependant, l'article scientifique *"Time Warp Invariant Dictionary Learning for Time Series Clustering"* [1] propose un nouveau point de vue sur ce problème en introduisant une notion clé : l'**invariance par décalage temporel**.

Le problème abordé dans cet article est celui du **clustering de séries temporelles** dans un contexte où les données peuvent être de longueurs différentes et inclurent des décalages temporels. Ces variations compliquent la comparaison directe entre séries temporelles et nécessitent des méthodes capables de s'adapter dynamiquement à ces disparités. Pour résoudre ce problème, l'article propose une approche basée sur :

- **Un codage parcimonieux invariant au décalage temporel** : un algorithme, appelé TWI-OMP (*Time Warp Invariant Orthogonal Matching Pursuit*), permet de représenter chaque série temporelle comme une combinaison linéaire parcimonieuse d'atomes alignés dynamiquement via une opération de warping temporel.
- **Un apprentissage de dictionnaires discriminants** : chaque cluster est associé à un sous-dictionnaire contenant les atomes les plus représentatifs, appris itérativement à partir des données tout en minimisant l'erreur de reconstruction.

L'article [1] présente ainsi une méthode qui permet de regrouper des séries temporelles similaires tout en étant robuste aux variations temporelles. Cette méthode est décrite en détail à travers l'algorithme final **TWI-DLCLUST**, qui permet donc de réaliser le clustering souhaité.

Contributions : Bien que nous ayons chacun contribué à toutes les étapes de ce projet, Chloé Habasque s'est particulièrement investie dans l'analyse des données ainsi que dans l'implémentation de certaines fonctionnalités, notamment la mise à jour des dictionnaires, tandis que Colin Coërchon s'est concentré sur le développement et l'optimisation des algorithmes *TWI-OMP* et *TWI-DLCLUST*. Le code a été entièrement conçu et implémenté par nos soins.

2 Méthode

2.1 Formalisation mathématique du problème

Le problème de clustering de séries temporelles peut être formulé comme la minimisation d'un critère d'inertie, où l'objectif est de partitionner un ensemble $X = \{x_i\}_{i=1}^{N}$ de séries temporelles d'entrée $x_i \in \mathbb{R}^{q_i}$ en K clusters $\{C_l\}_{l=1}^{K}$. Chaque cluster C_l est associé à un sous-dictionnaire $D_l = \{d_j^l\}_{j=1}^{K_l}$, composé de K_l atomes représentatifs. Le problème est défini comme suit :

$$\min_{C,D} \sum_{l=1}^{K} \sum_{x_i \in C_l} E(x_i, D_l)$$

où $E(x_i, D_l)$ représente **l'erreur de reconstruction** sous décalage temporel, donnée par :

$$E(x_i, D_l) = \min_{\alpha_i^l} \|x_i - \mathcal{F}_i(D_l)\alpha_i^l\|_2^2 \quad \text{sous la contrainte } \|\alpha_i^l\|_0 \leq \tau$$

La principale innovation par rapport au codage parcimonieux classique réside dans la transformation $\mathcal{F}_i(D_l)$, qui aligne dynamiquement les atomes d_j^l du dictionnaire D_l sur x_i pour résorber les décalages temporels. Cette transformation est définie comme :

$$\mathcal{F}_i(D_l) = [f_i(d_1^l), \ldots, f_i(d_{K_l}^l)] \in \mathbb{R}^{q_i \times K_l},$$

où $f_i(d_j^l) = \Delta_{ij}^l d_j^l$, et Δ_{ij} est une matrice de projection obtenue par warping temporel.

Le warping temporel permet de définir un alignement (Appendix B) π entre les indices temporels de d_j^l et x_i, c'est-à-dire une correspondance qui ajuste les séquences temporelles pour minimiser les différences entre elles malgré les décalages.

2.2 Étape 1 : Codage parcimonieux invariant au décalage temporel (TWI-OMP)

Le codage parcimonieux consiste à représenter chaque série temporelle x_i comme une combinaison linéaire de quelques atomes du dictionnaire D_l associé à son cluster. Pour ce faire, nous utilisons l'algorithme **TWI-OMP** (*Time Warp Invariant Orthogonal Matching Pursuit*), une extension de l'algorithme classique OMP adaptée pour traiter les séries temporelles alignées dynamiquement.

Présentation de TWI-OMP L'algorithme TWI-OMP sélectionne itérativement les atomes $f_i(d_j^l)$ alignés pour représenter x_i de manière parcimonieuse. Les principales étapes de l'algorithme sont les suivantes :

1. **Alignement des atomes :** Calculer la transformation $\mathcal{F}_i(D_l)$ qui aligne les atomes du dictionnaire D_l sur la série temporelle x_i en maximisant la similarité, mesurée par le **COSTW** (*Cost of Time Warping*) (Appendix C), une métrique basée sur le cosinus de similarité après alignement.
2. **Sélection des atomes :** Identifier les atomes les plus significatifs qui réduisent le plus l'erreur de reconstruction $E(x_i, D_l)$.
3. **Estimation des coefficients :** Calculer les coefficients parcimonieux α_i^l correspondant aux atomes sélectionnés.

2.3 Étape 2 : Apprentissage de dictionnaires invariants au décalage temporel

L'apprentissage des dictionnaires vise à optimiser les atomes d_j^l de chaque dictionnaire D_l associé aux clusters C_l, de manière à minimiser l'erreur de reconstruction des séries temporelles appartenant à chaque cluster. Dans cette étape, nous supposons que les classes C_l, les coefficients α_i^l, et les projections Δ_i^l obtenus lors de l'étape de codage parcimonieux (Étape 1) sont fixés. L'objectif est donc de mettre à jour les atomes d_j^l en utilisant une méthode de **descente de gradient** pour réduire le critère d'inertie $\mathcal{J}_l$.

Pour chaque cluster C_l, le critère d'inertie $\mathcal{J}_l$ est défini comme :

$$\mathcal{J}_l = \sum_{x_i \in C_l} E(x_i, D_l) = \sum_{x_i \in C_l} \left\| x_i - \mathcal{F}_i(D_l)\alpha_i^l \right\|_2^2$$

où $\mathcal{F}_i(D_l)$ aligne dynamiquement les atomes de D_l sur x_i, et α_i^l sont les coefficients de reconstruction.

Mise à jour des atomes par descente de gradient :

La mise à jour d'un atome d_j^l par descente de gradient se calcule de cette manière :

$$d_j^{l(m+1)} = d_j^{l(m)} - \eta \frac{\partial \mathcal{J}_l}{\partial d_j^{l(m)}} \qquad \text{avec} \qquad \frac{\partial \mathcal{J}_l}{\partial d_j^l} = -2 \sum_{x_i \in C_l} \alpha_{ij}^l \Delta_i^l \left(x_i - \mathcal{F}_i(D_l)\alpha_i^l \right)$$

où α_{ij}^l est le coefficient de parcimonie associé à d_j^l pour x_i.

Les étapes sont décrites dans l'Algorithme 2 avec plus de précisions.

2.4 Étape 3 : Clustering de séries temporelles (TWI-DLCLUST)

Après l'étape de codage parcimonieux et l'apprentissage des dictionnaires, l'algorithme **TWI-DLCLUST** procède au clustering des séries temporelles en alternant entre l'attribution des séries aux clusters et la mise à jour des dictionnaires. Cette étape est cruciale car elle permet de regrouper les séries similaires tout en affinant les représentations par atomes spécifiques à chaque cluster.

Après l'initialisation des différents clusters et dictionnaires, l'algorithme TWI-DLCLUST itère entre deux étapes principales :

1. **Attribution des séries temporelles aux clusters** :
   (a) Pour chaque série temporelle x_i, effectuer un codage parcimonieux en utilisant **TWI-OMP** avec chaque dictionnaire D_l.
   (b) Assigner la série temporelle x_i au cluster C_l qui minimise $E(x_i, D_l)$.
2. **Mise à jour des dictionnaires** : Pour chaque cluster C_l, mettre à jour les atomes d_j^l du dictionnaire D_l en utilisant la descente de gradient, comme décrit dans l'Étape 2.

Critère d'arrêt : L'algorithme se termine lorsque les clusters ne changent plus entre deux itérations consécutives, indiquant ainsi une convergence.

Les étapes complètes sont décrites dans l'Algorithme 3.

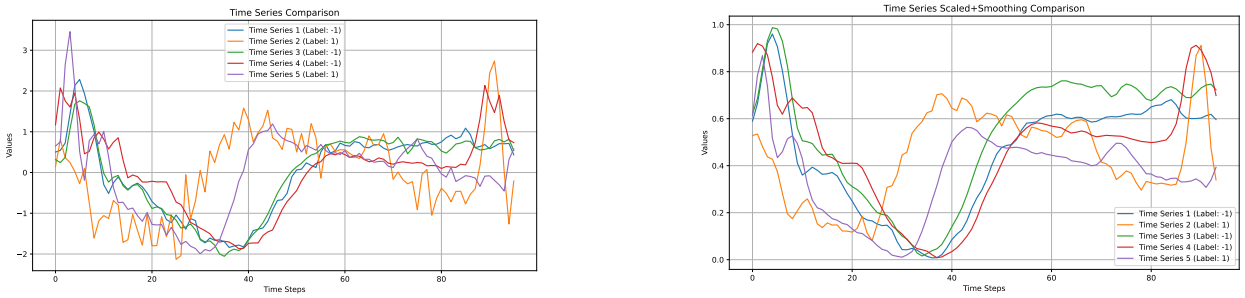
3 Données utilisées

Le papier étudié utilise au total 12 datasets comprenant différents types de données telles que des images converties en vecteurs ou des séries temporelles. Le dataset principal *deezer* n'étant pas disponible, nous avons choisi d'utiliser deux autres datasets pour analyser cette méthode : *ECG200* et *Trace*.

3.1 ECG200

Le dataset *ECG200* contient des séries temporelles utilisées pour la classification binaire de signaux électrocardiographiques (ECG). Chaque série représente l'activité électrique enregistrée pendant un battement de cœur sur 96 points temporels. Le dataset contient un total de 100 séries, réparties en deux classes : **battement de cœur normal** ou **infarctus du myocarde** (crise cardiaque). Ce jeu de données est particulièrement utilisé dans le domaine médical pour développer des modèles de diagnostic automatique capables de détecter des anomalies cardiaques à partir de signaux ECG.

Pour utiliser ce dataset, nous avons appliqué quelques transformations, notamment une normalisation pour ramener les données à une même échelle et faciliter la comparaison entre les séries, ainsi qu'un lissage afin de réduire le bruit et mettre en évidence les tendances et motifs sous-jacents (Figure 1).



(a) Avant

(b) Après

Figure 1: Séries temporelles du dataset ECG200 avant et après pré-traitement

3.2 Trace

Ce jeu de données à 4 classes est un sous-ensemble du Transient Classification Benchmark (projet TRACE), il s'agit d'un jeu de données synthétique conçu pour simuler des défaillances d'instrumentation dans une centrale nucléaire.
Ces 100 séries temporelles de longueur 275 sont réparties en 4 classes de 1 à 4, auxquelles nous avons effectué les mêmes changement : normalisation et lissage.

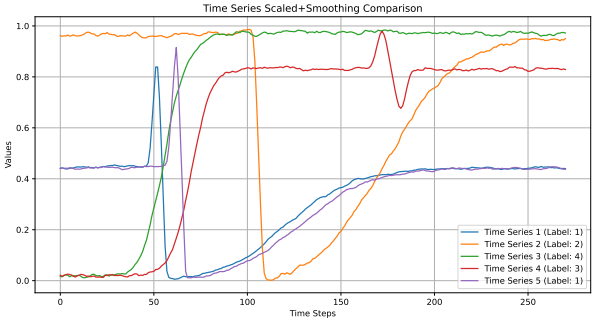


Figure 2: Différentes séries temporelles du dataset Trace

4 Résultats

4.1 Résultats de la fonction TWI-OMP

Les premiers résultats ont été obtenus en testant l'algorithme TWI-OMP, décrit dans l'1. Pour cela, une série temporelle synthétique a été générée, contenant trois motifs explicites, et un dictionnaire composé de trois atomes correspondant à ces motifs a été utilisé. Comme illustré dans la Figure 5, l'algorithme identifie correctement les motifs similaires et les associe aux positions adéquates. En combinant les atomes sélectionnés, la série temporelle reconstruite s'aligne étroitement sur la série temporelle d'origine.

4.2 Résultats de la mise à jour des dictionnaires

Nous avons ensuite testé notre algorithme de mise à jour des sous-dictionnaires, présenté dans l'Appendix D. Cet algorithme repose sur une descente de gradient permettant d'ajuster les atomes pour chaque classe. La mise en œuvre de cette méthode s'est avérée particulièrement complexe, et nous pensons que ces difficultés expliquent en partie les résultats moyens obtenus. En effet, nous avons atteint un ARI maximum de 0.55, alors que l'article original rapporte des performances allant jusqu'à 0.8 ou 0.9.

Les résultats illustrés dans la Figure 6 montrent l'entraînement du dictionnaire pour la classe 4 du dataset `Trace`, après 50 itérations de l'algorithme.

4.3 Résultats du Clustering

Nous avions ainsi 2 Datasets, et nous les avons largement utilisés pour nos nombreux essais. L'évaluation des modèles se fait avec le Adjusted Rand Index (ARI), présenté dans l'article [1]. Voici les résultats que nous avons obtenu :

Table 1: Comparaison des résultats pour différents jeux de données avec TWI-DCLUST

Jeu de données	K (Clusters)	Taille des atomes	Kl (Atomes par dic.)	τ	ARI
ECG	2	10	5	1	0.0294
Trace	4	25	5	3	0.5486

Dataset `ECG200` : Nos résultats sur le dataset `ECG200` ne sont pas très concluant. Cela peut s'expliquer par plusieurs facteurs, mais le plus probable semble être que ce jeu de données n'est simplement pas adapté à ce type de clustering. En effet, la Figure 3 illustre cette homogénéité dans ces séries temporelles. Les seules petites différences entre les deux classes sera fortement ignoré par le caractère invariant aux décalages temporels de l'algorithme TWI-DLCLUST.

Nous obtenons donc un résultat avec un ARI de **0.0294**, ce qui correspond quasiment à un clustering aléatoire (*ARI proche de 0*).

Dataset `Trace` : Contrairement au Dataset `ECG200`, celui-ci présente des séries plus variées, formant des groupes distincts, qui devrait faciliter l'identification des atomes propres à chaque cluster. Cela est illustré par la Figure 4.

L'ARI pour le dataset Trace est de **0.5486** en utilisant un dictionnaire de 5 atomes de taille 25 et avec un tau de 1. Ce résultat n'est pas parfait (*encore loin du résultat parfait avec un ARI de 1*), mais il est déjà bien plus convainquant que celui obtenu avec le dataset `ECG200`.

References

[1] Saeed Varasteh Yazdi et al. "Time warp invariant dictionary learning for time series clustering: application to music data stream analysis". In: *Joint european conference on machine learning and knowledge discovery in databases*. Springer. 2018, pp. 356–372.

Appendix

A Algorithmes

Algorithme TWI-OMP

Algorithm 1 TWI-OMP : Time Warp Invariant Orthogonal Matching Pursuit

Require: Série temporelle x, dictionnaire $D = \{d_j\}_{j=1}^{K}$, facteur de parcimonie τ.
Ensure: Coefficients parcimonieux α, projections Δ.

$r \leftarrow x$ ▷ Initialiser le résidu
$\Omega \leftarrow \emptyset$ ▷ Initialiser l'ensemble des indices des atomes sélectionnés
while $|\Omega| < \tau$ **do**
    Pour chaque atome d_j **dans** D :
    1. Calculer le warping temporel Δ_{ij}^{l} via un algorithme de programmation dynamique similaire à la DTW.
    2. Calculer le **COSTW** : $\text{COSTW}(r, d_j) = \frac{r^T(\Delta_{ij}^{l} d_j)}{\|r\|_2 \|\Delta_{ij}^{l} d_j\|_2}$.
    Sélectionner l'atome d_{j^*} qui maximise $\text{COSTW}(r, d_j)$.
    Mettre à jour $\Omega \leftarrow \Omega \cup \{j^*\}$.
    Calculer $S_\Omega = [\Delta_{ij}^{l} d_j^{l}]_{j \in \Omega}$ ▷ Aligner les atomes sélectionnés
    Estimer les coefficients : $\alpha_\Omega = (S_\Omega^T S_\Omega)^{-1} S_\Omega^T x$ ▷ Méthode des moindres carrées
    Mettre à jour le résidu : $r \leftarrow x - S_\Omega \alpha_\Omega$.
end while
return α, Δ

Algorithme d'apprentissage des dictionnaires L'algorithme suivant décrit les étapes de mise à jour des dictionnaires invariants au décalage temporel :

Algorithme Time Warp Invariant Dictionary Learning for Clustering

Algorithm 2 Apprentissage de dictionnaires invariants au décalage temporel

Require: Dictionnaires actuels $D = \{D_l\}_{l=1}^K$, classes $C = \{C_l\}_{l=1}^K$, coefficients α_i^l, projections Δ_i^l, taux d'apprentissage η.
Ensure: Dictionnaires mis à jour D.

for chaque cluster $l = 1, \ldots, K$ **do**
  for chaque atome d_j^l dans D_l **do**
    Calculer le gradient :

$$\frac{\partial \mathcal{J}_l}{\partial d_j^l} = -2 \sum_{x_i \in C_l} \alpha_{ij}^l \Delta_i^l (x_i - \mathcal{F}_i(D_l)\alpha_i^l)$$

    Mettre à jour l'atome :

$$d_j^l \leftarrow d_j^l - \eta \frac{\partial \mathcal{J}_l}{\partial d_j^l}$$

    Normaliser l'atome :

$$d_j^l \leftarrow \frac{d_j^l}{\|d_j^l\|_2}$$

  end for
end for
return D

Algorithm 3 TWI-DLCLUST : Time Warp Invariant Dictionary Learning for Clustering

Require: Séries temporelles $X = \{x_i\}_{i=1}^N$, nombre de clusters K, facteur de parcimonie τ.
Ensure: Clusters $\{C_l\}_{l=1}^K$, dictionnaires $\{D_l\}_{l=1}^K$.

Initialiser $\{C_l\}_{l=1}^K$ via un clustering spectral ou une propagation d'affinité
Initialiser les sous-dictionnaires $\{D_l\}_{l=1}^K$ aléatoirement
repeat
  for all $x_i \in X$ **do**
    Effectuer un codage parcimonieux **TWI-OMP** avec chaque D_l ▷ Étape d'attribution des séries aux clusters
    Assigner x_i au cluster C_l minimisant $E(x_i, D_l)$ ▷ Attribution basée sur l'erreur de reconstruction minimale
  end for
  for all D_l **do**
    Mettre à jour D_l par descente de gradient ▷ Étape de mise à jour des dictionnaires
  end for
until Convergence (pas de changement dans les clusters) ▷ Critère d'arrêt
return $\{C_l\}_{l=1}^K, \{D_l\}_{l=1}^K$

B Alignement

Un alignement π entre deux séries temporelles x et y est une bijection partielle qui associe des points de x à des points de y, permettant ainsi de comparer des motifs similaires même s'ils sont décalés ou étirés dans le temps. Formellement, π est une fonction qui mappe les indices de x vers ceux de y, en respectant l'ordre temporel :

$$\pi : \{1, \ldots, q_i\} \to \{1, \ldots, q_j\}, \quad \text{tel que } t_1 < t_2 \implies \pi(t_1) \leq \pi(t_2).$$

Cet alignement permet de comparer les séries temporelles de manière flexible, en s'adaptant aux variations locales dans le temps.

C COSTW

Définition du COSTW Le **COSTW** (*Cost of Time Warping*) est une mesure de similarité entre une série temporelle x_i et un atome d_j^l après alignement temporel via la matrice de projection Δ_{ij}^l. Avant sa définition formelle, il peut être exprimé comme :

$$\text{COSTW}(x_i, d_j^l) = s(\pi^*)$$

où $s(\pi^*)$ représente la similarité cosinus maximale obtenue avec l'alignement optimal π^*.

Formellement, le **COSTW** est défini à partir de la similarité cosinus, une mesure couramment utilisée pour évaluer l'alignement directionnel entre deux vecteurs. La similarité cosinus entre deux vecteurs a et b est donnée par :

$$\cos(\theta) = \frac{a^T b}{\|a\|_2 \|b\|_2}.$$

Ainsi, le **COSTW** peut être défini comme :

$$\text{COSTW}(x_i, d_j^l) = \cos(\theta(x_i, \Delta_{ij}^l d_j^l)) = \frac{x_i^T (\Delta_{ij}^l d_j^l)}{\|x_i\|_2 \|\Delta_{ij}^l d_j^l\|_2}.$$

Maximiser le **COSTW** revient donc à maximiser la similarité cosinus entre x_i et l'atome aligné $\Delta_{ij}^l d_j^l$.

Le **COSTW** partage avec la DTW l'objectif de gérer les variations temporelles, mais se distingue par son intégration directe dans le processus de codage parcimonieux. Contrairement à la DTW qui cherche un alignement global entre deux séries, le **COSTW** est utilisé localement pour aligner chaque atome du dictionnaire avec la série temporelle en cours de traitement. De plus, le **COSTW** est basé sur la similarité cosinus, ce qui le rend particulièrement adapté pour sélectionner des atomes ayant une orientation similaire après alignement, facilitant ainsi la reconstruction parcimonieuse.

D Apprentissage de dictionnaires

Pour chaque cluster C_l, le critère d'inertie $\mathcal{J}_l$ est défini comme la somme des erreurs de reconstruction des séries temporelles appartenant à ce cluster :

$$\mathcal{J}_l = \sum_{x_i \in C_l} E(x_i, D_l) = \sum_{x_i \in C_l} \|x_i - \mathcal{F}_i(D_l)\alpha_i^l\|_2^2$$

où $\mathcal{F}_i(D_l)$ est la transformation qui aligne dynamiquement les atomes du dictionnaire D_l sur la série temporelle x_i, et α_i^l sont les coefficients de reconstruction obtenus lors de l'étape précédente.

Mise à jour des atomes par descente de gradient Pour minimiser $\mathcal{J}_l$, nous appliquons une descente de gradient sur chaque atome d_j^l du dictionnaire D_l. La mise à jour de l'atome d_j^l à l'itération $m+1$ est formulée comme suit :

$$d_j^{l(m+1)} = d_j^{l(m)} - \eta \frac{\partial \mathcal{J}_l}{\partial d_j^{l(m)}}$$

où :

- η est le taux d'apprentissage.
- $\frac{\partial \mathcal{J}_l}{\partial d_j^{l(m)}}$ est le gradient de $\mathcal{J}_l$ par rapport à l'atome d_j^l à l'itération m.

Après chaque mise à jour, les atomes sont normalisés pour conserver une norme unité, ce qui est essentiel pour maintenir la stabilité de l'apprentissage et assurer une comparaison cohérente entre les atomes :

$$\|d_j^{l(m+1)}\|_2 = 1$$

Calcul du gradient Le gradient du critère d'inertie $\mathcal{J}_l$ par rapport à l'atome d_j^l est calculé en différenciant l'erreur de reconstruction :

$$\frac{\partial \mathcal{J}_l}{\partial d_j^l} = -2 \sum_{x_i \in C_l} \alpha_{ij}^l \Delta_i^l \left(x_i - \mathcal{F}_i(D_l)\alpha_i^l \right)$$

où α_{ij}^l est le coefficient associé à l'atome d_j^l pour la série temporelle x_i.

E Comparaison des classes des séries temporelles

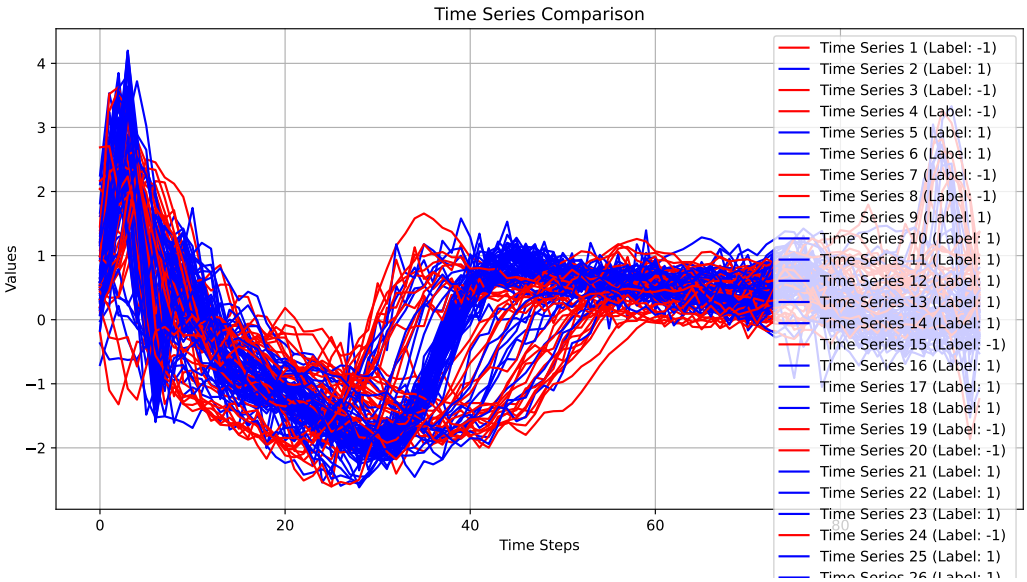


Figure 3: Comparaison des séries temporelles de ECG en fonction des labels

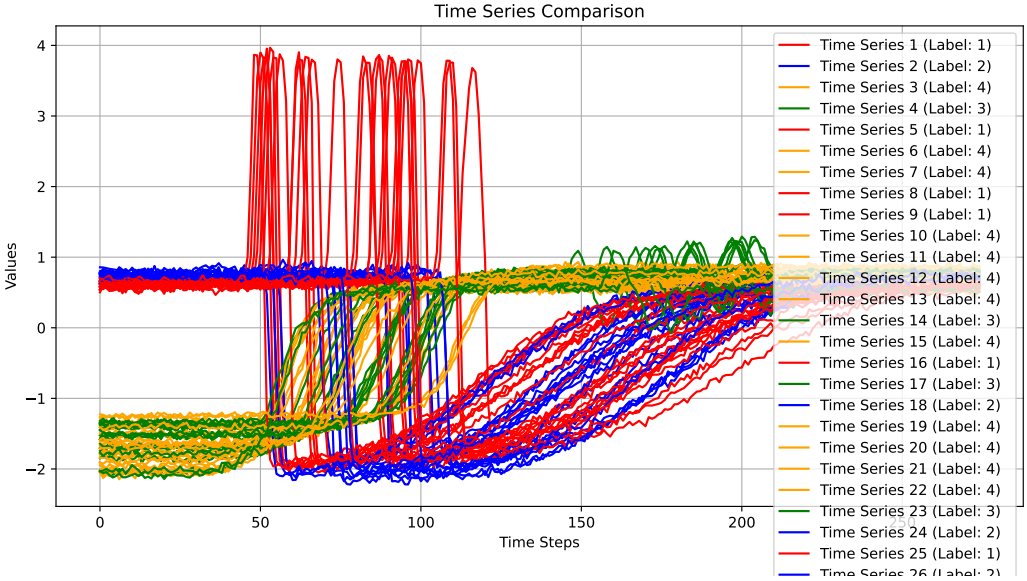


Figure 4: Comparaison des séries temporelles de Trace en fonction des labels

F Images

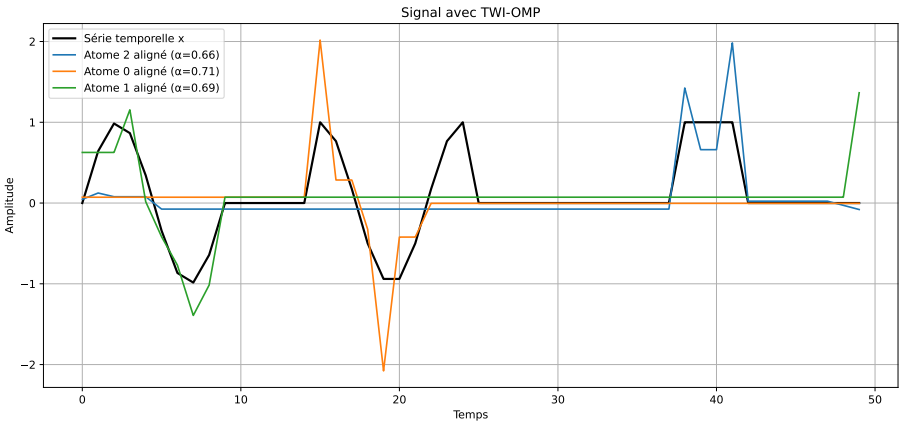


Figure 5: Atomes renvoyés par la fonction TWI-OMP

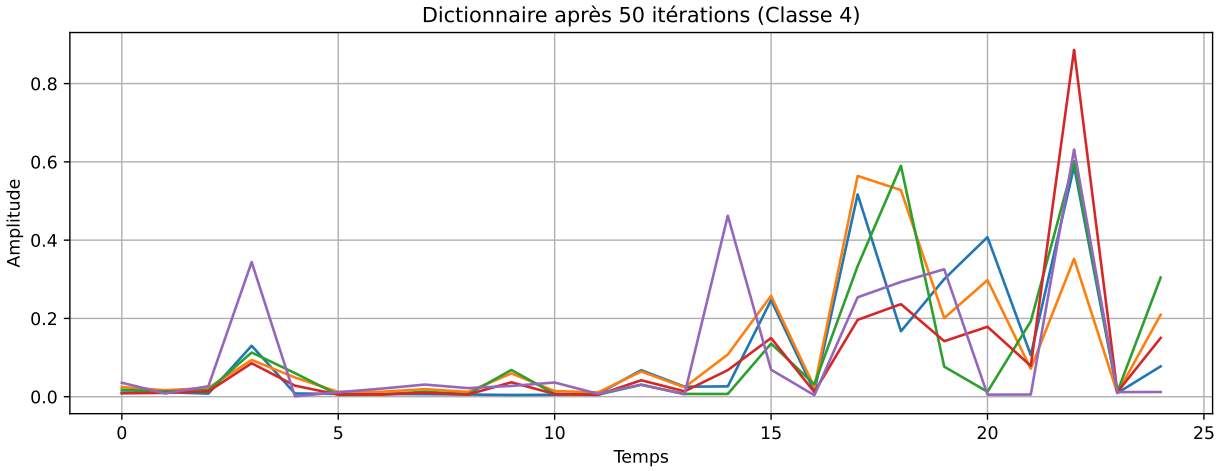


Figure 6: Mise à jour de dictionnaire

F.1 ECG200

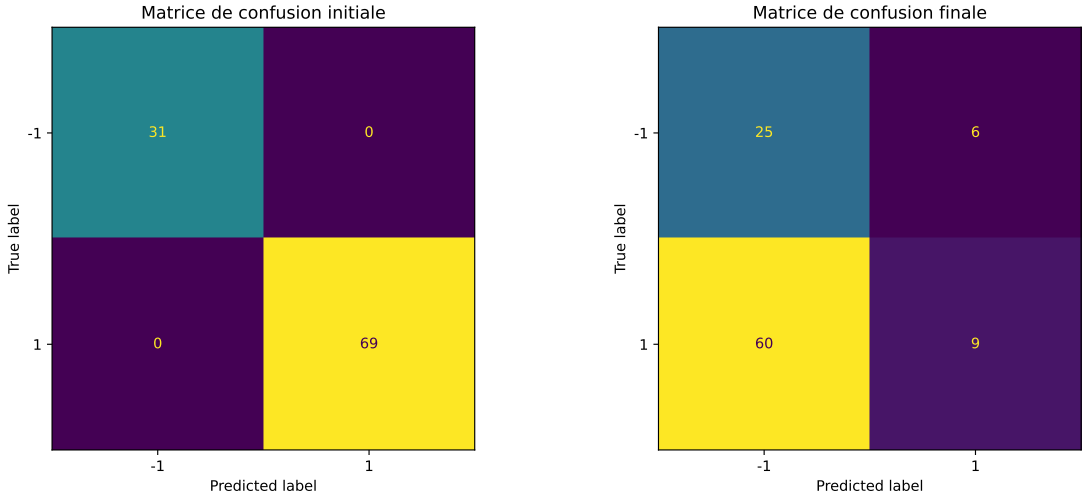


Figure 7: Matrice de confusion du clustering de ECG200

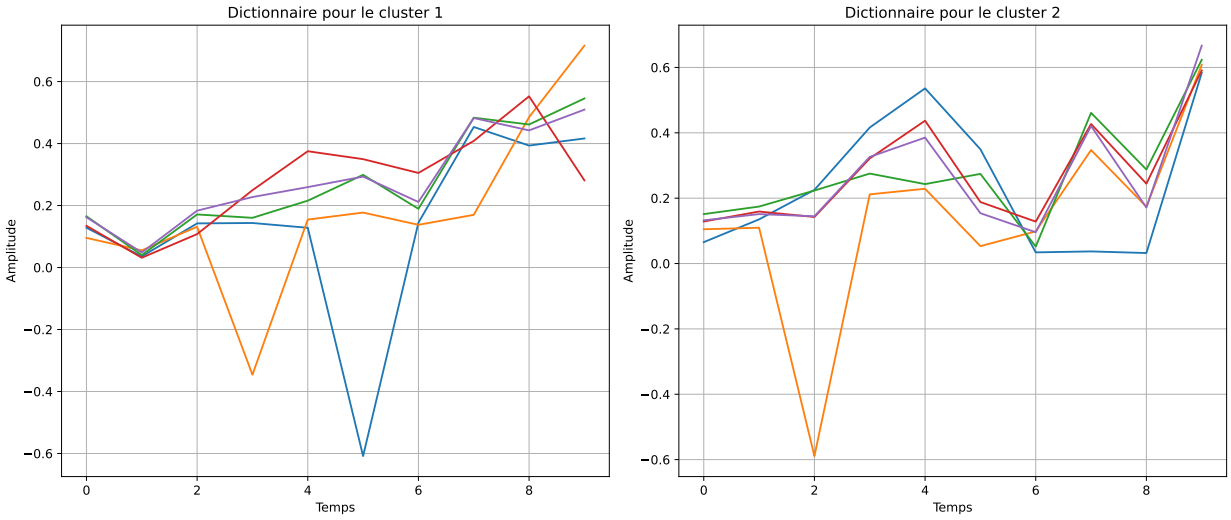


Figure 8: Dernière mise à jour des dictionnaires pour ECG200

F.2 Trace

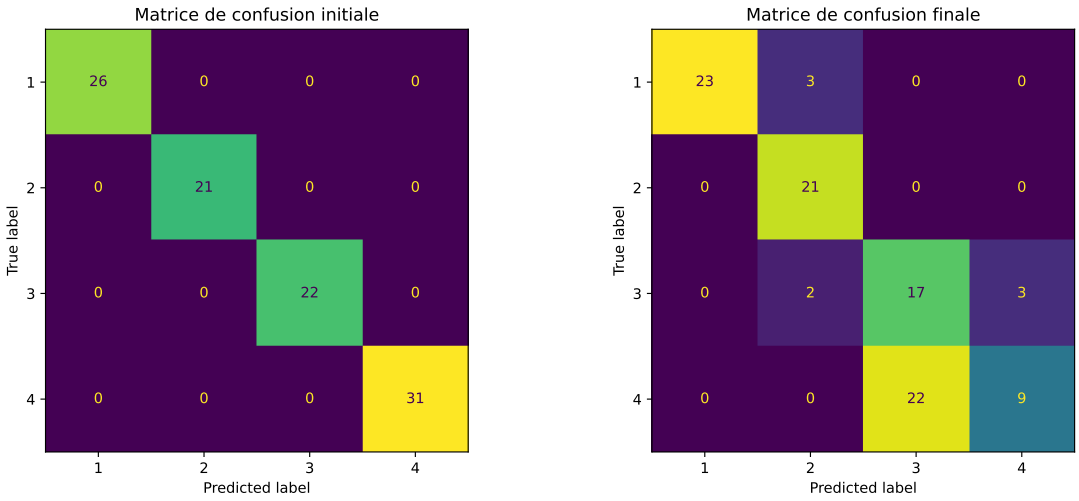


Figure 9: Matrice de confusion du clustering de Trace

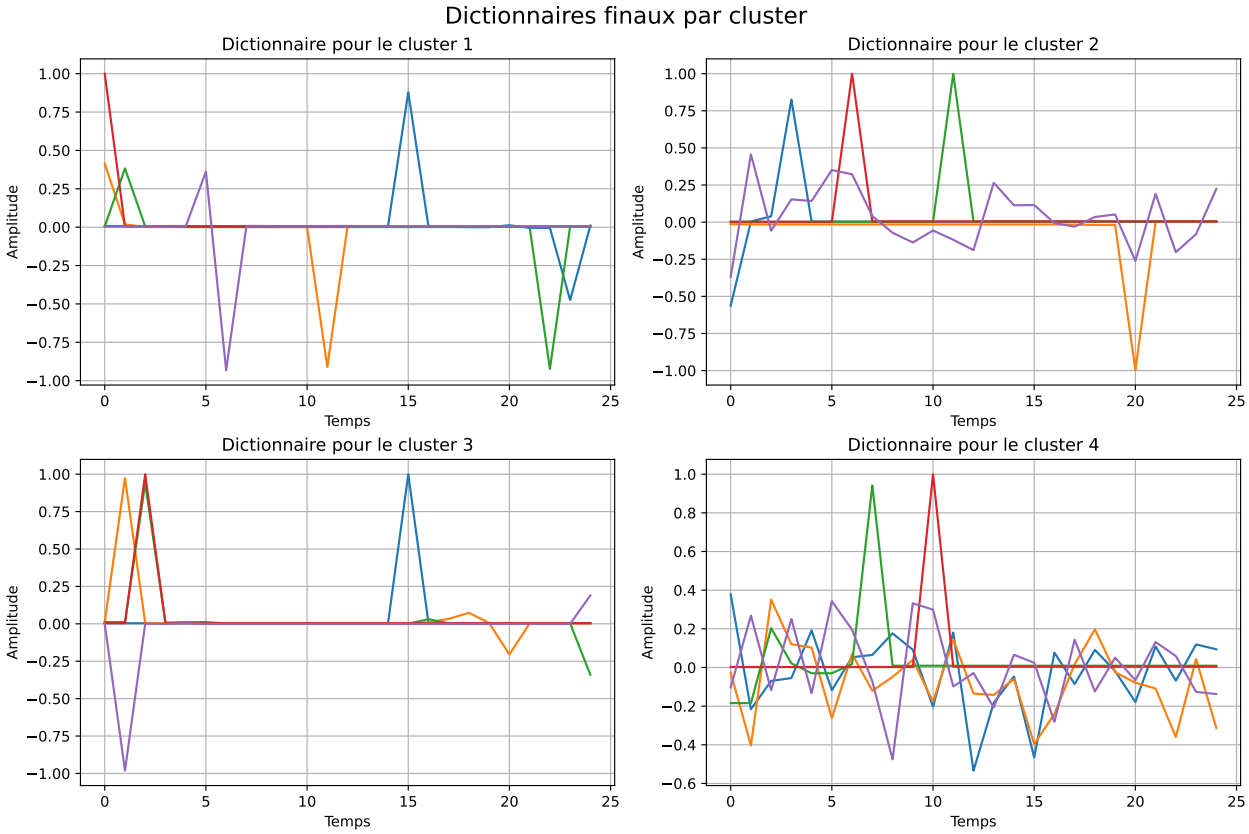


Figure 10: Dernière mise à jour des dictionnaires pour Trace