Segmentation supervisée de signaux Méthodes à noyaux et annotations partielles

Charles TRUONG[1,2], Laurent OUDRE[3], Nicolas VAYATIS[1,2]

[1]CMLA (ENS Paris Saclay, CNRS) 61 Avenue du Président Wilson, 94230 Cachan, France

[2]COGNAC-G (Université Paris Descartes, CNRS, SSA) 45 Rue des Saints-Pères, 75006 Paris, France

[3]L2TI (Université Paris 13) 99 avenue Jean Baptiste Clément 93430 Villetaneuse, France

truong@cmla.ens-cachan.fr, laurent.oudre@univ-paris13.fr, vayatis@cmla.ens-cachan.fr

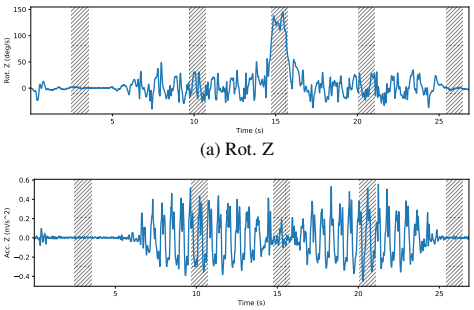
Résumé – Cet article décrit une procédure automatique de calibrage d'algorithmes de détection de rupture. Notre approche utilise la capacité d'un expert à fournir des annotations partielles, c'est-à-dire une estimation très approximative de la position des ruptures sur quelques signaux. Notre contribution consiste en une stratégie supervisée d'apprentissage de métrique basée sur noyau reproduisant. Une fois combinée à un algorithme de détection, la métrique peut reproduire la stratégie de segmentation de l'expert sur de nouveaux signaux.

Abstract – This article describes an automatic procedure to calibrate change point detection algorithms. Our approach expands on the ability of an expert to provide very rough segmentation estimates, called partial annotations, for a few signal examples. Our contribution consists in a supervised strategy to learn a kernel Mahalanobis metric, which, once combined with a detection algorithm, can replicate the expert's segmentation strategy on new signals.

1 Introduction

La détection de rupture (ou segmentation de signal) est une tâche centrale en traitement du signal. Elle consiste à déterminer les limites temporelles des régimes successifs d'un signal univarié ou multivarié. Les applications sont nombreuses et vont de l'analyse ADN [4] à la surveillance d'équipements industriels [2]. En pratique, l'expert (économiste, biologiste, etc.) doit choisir, parmi la riche littérature, une méthode de segmentation appropriée. Un paramètre particulièrement important est le type de rupture à détecter, qui est déterminé par la représentation du signal ou, de manière équivalente, par la métrique choisie. Ce processus complexe pourrait être automatisé en utilisant la capacité de l'expert à segmenter manuellement quelques signaux, ou au moins à fournir des annotations partielles, c'est-à-dire le début et la fin d'une partie de chaque régime. Par exemple, sur la figure 1, l'expert a annoté des portions d'une seconde d'un signal collecté en surveillant, avec un capteur inertiel, un sujet effectuant une séquence d'activités simples (se tenir debout, marcher, faire demi-tour) [1]. L'objectif de ce travail est de concevoir un mécanisme pour calculer automatiquement, à partir d'exemples de segmentations (signaux et annotations partielles), une métrique appropriée. Un algorithme de détection de rupture qui utiliserait cette métrique serait en mesure de reproduire la stratégie de segmentation de l'expert.

Segmentation et méthodes à noyau. Pour un signal $y := \{y_t\}_1^T$ avec T échantillons et à valeurs dans $\mathbb{R}^d$, la détection d'un nombre K de ruptures consiste à trouver les indices $\hat{t}_k$ $(k = 1, \ldots, K)$ tels que



(a) Rot. Z

(b) Acc. Z

Figure 1 – Exemple de signal partiellement annoté, tiré du jeu de signaux *Marche* (voir la section 3.1). Les annotations (parties grises hachurées) sont les portions de signal qui sont considérées comme homogènes, c'est-à-dire ne contenant aucune rupture.

$$\hat{t}_1, \ldots, \hat{t}_K := \underset{t_1 < \cdots < t_K}{\arg\min} \sum_{k=0}^{K} \sum_{t=t_k+1}^{t_{k+1}} \left\| y_t - \bar{\mu}_{t_k..t_{k+1}} \right\|^2 \quad (1)$$

où $\bar{\mu}_{a..b}$ est la moyenne empirique du sous-signal $\{y_t\}_{t=a+1}^b$, $t_0 := 0$ et $t_{K+1} := T$ sont donnés par convention, et $\|\cdot\|$ est une norme sur $\mathbb{R}^d$ définie par l'utilisateur (la norme euclidienne par exemple). De nombreux algorithmes de la littérature minimisent cette somme de résidus, sous des contraintes diverses. Les méthodes optimales trouvent les ruptures qui optimisent exactement le critère (1). La procédure la plus largement utilisée est basée sur la programmation dynamique [3, 4]. Des méthodes plus rapides mais approximatives ont aussi été développées; les procédures par fenêtre glissante et la segmen-

tation binaire [2] en sont les exemples les plus connus.
Cet article porte sur le calibrage de la norme $\|\cdot\|$. À notre connaissance, un seul travail [8] traite de segmentation supervisée : les auteurs utilisent, dans le critère (1), une (pseudo-)norme linéaire de type Mahalanobis $\|\cdot\|_M$ donnée par $\|u\|_M := u'Mu$ ($\forall u \in \mathbb{R}^d$) où la matrice de métrique M est positive semi-définie (psd). La matrice de métrique optimale est apprise en minimisant une perte convexe entre les partitions d'un signal. Initialement, cet algorithme nécessite des annotations complètes (c'est-à-dire les positions des ruptures). Une stratégie non convexe est proposée pour gérer les annotations partielles. Cependant, elle dépend de la capacité de l'utilisateur à initialiser correctement la métrique et n'est sensible qu'aux changements de moyenne. Les méthodes à noyau ont, elles, vu le jour car elles sont capables de détecter des changements dans les moments d'ordre supérieur des distributions de probabilité [4]. Formellement, soit $k(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$ un noyau et $\mathcal{H}$ l'espace de Hilbert à noyau reproduisant associé (désigné par l'acronyme anglais rkhs). La fonction de reproduction $\phi : \mathbb{R}^d \to \mathcal{H}$ est définie implicitement par $\phi(y_t) = k(y_t,\cdot) \in \mathcal{H}$, avec $\langle \phi(y_s) | \phi(y_t) \rangle_{\mathcal{H}} = k(y_s,y_t)$ et $\|\phi(y_t)\|^2_{\mathcal{H}} = k(y_t,y_t)$. La segmentation par méthode à noyau revient à minimiser un critère de la forme (1) où le signal y est remplacé par sa transformée en grande dimension $\{\phi(y_t)\}_t$ et la norme par $\|\cdot\|_{\mathcal{H}}$. Ces techniques ont été largement étudiées d'un point de vue algorithmique [5] et théorique [4]. Néanmoins, elles sont non-supervisées par nature.
Nous proposons d'étendre le critère (1) à la classe générale de normes de type Mahalanobis basées sur un noyau $\|\cdot\|_{\mathcal{H},M}$ définies par $\|\phi(u)\|_{\mathcal{H},M} := \phi(u)'M\phi(u)$ ($\forall u \in \mathbb{R}^d$) où M est une matrice psd. La somme des résidus (1) peut être réécrite comme suit pour tout ensemble de K indices de rupture $\mathcal{T} = \{t_1, \dots, t_K\}$:

$$V(\mathcal{T}) := \sum_{k=0}^{K} \sum_{t=t_k+1}^{t_{k+1}} \left\| \phi(y_t) - \bar{\mu}_{t_k..t_{k+1}} \right\|^2_{\mathcal{H},M}. \qquad (2)$$

La norme $\|\cdot\|_{\mathcal{H},M}$ contrôle le type de rupture pouvant être détectée et doit être calibrée algorithmiquement à l'aide des annotations partielles disponibles. Notre approche s'appuie sur les méthodes d'apprentissage de métrique, les travaux de [7] en particulier, qui ont été appliqués avec succès dans les domaines de la classification et clustering [7], mais n'ont pas encore été mis en pratique à la segmentation de signaux.

Contributions. Cet article présente une procédure pour apprendre une norme de type Mahalanobis basée sur un noyau en utilisant une base d'apprentissage (les signaux et leurs annotations partielles). La méthode décrite offre une nouvelle perspective sur la segmentation supervisée. Par rapport aux travaux précédents, notre approche est non paramétrique et s'accommode d'un noyau arbitraire. Une fois apprise, la métrique peut être combinée avec tout algorithme de détection basé sur la minimisation (exacte ou approximative) du critère (2). Des expériences sur des données réelles montrent que l'apprentissage améliore la précision de plusieurs algorithmes de segmentation.

2 Méthode

Notre approche consiste en une étape d'apprentissage, au cours de laquelle une matrice de métrique optimale $\widehat{M}$ est estimée, et en une étape de prédiction, au cours de laquelle la segmentation est effectuée sur de nouveaux signaux, en utilisant le critère V. Cette section décrit les étapes successives pour apprendre une métrique et l'appliquer sur de nouveaux signaux.

Des annotations aux contraintes. L'apprentissage de métrique s'exprime comme une optimisation sous contraintes : des contraintes de similarité et de dissimilarité, que nous construisons à partir d'annotations partielles. Une contrainte de *similarité* est une paire d'échantillons qui doivent être proches selon la distance apprise. Une contrainte de *dissimilarité* est une paire d'échantillons qui doivent être éloignés selon la distance apprise. Précisément, soit $y^{\text{train}} := [y_1^{\text{train}}, y_2^{\text{train}}, \dots]$ la concaténation de tous les échantillons d'apprentissage, i.e. les échantillons qui appartiennent à une partie annotée d'un signal d'apprentissage. Les deux vecteurs y_s^{train} et y_t^{train} sont considérés comme "similaires" s'ils appartiennent au même régime et "dissemblables" s'ils appartiennent à deux régimes consécutifs du même signal. Seules des paires d'échantillons provenant du même régime ou de deux régimes consécutifs créent une contrainte de similarité/dissimilarité. Les échantillons n'appartenant pas à une partie homogène du signal (selon les annotations) ne créent aucune contrainte. Ce principe est illustré à la figure 2.

Apprentissage d'une métrique à noyau. Une fois les contraintes générées, la matrice de métrique optimale $\widehat{M}$ peut être estimée. Pour cela, nous définissons $\widehat{M}$ comme étant la solution du problème d'optimisation sous contrainte suivant:

$$\begin{aligned} &\min_{M \succeq 0} \quad D_{LD}(M,I) \quad \text{s.c.} \\ &\left\| \phi(y_s^{\text{train}}) - \phi(y_t^{\text{train}}) \right\|^2_{\mathcal{H},M} \leq u, \, y_s^{\text{train}} \text{ and } y_t^{\text{train}} \text{ similaires} \\ &\left\| \phi(y_s^{\text{train}}) - \phi(y_t^{\text{train}}) \right\|^2_{\mathcal{H},M} \geq v, \, y_s^{\text{train}} \text{ and } y_t^{\text{train}} \text{ dissimilaires} \end{aligned} \qquad (3)$$

où $D_{LD}(M,M_0) := \operatorname{tr}(MM_0^{-1}) - \log\det(MM_0^{-1})$ est la divergence LogDet qui agit comme une distance sur l'ensemble des matrices psd et $u > 0$ (resp. $v > 0$) est une borne supérieure (resp. inférieure) sur les distances intra-régime (resp. inter-régime). La divergence entre M et la matrice identité s'apparente à une régularisation.La résolution du problème (3) a été étudiée d'un point de vue théorique et algorithmique [7]. Nous présentons ici les aspects clés de l'algorithme d'optimisation. Tout d'abord, l'optimisation (3) est effectuée dans l'espace des matrices psd à action sur $\mathcal{H}$, qui peut être de dimension infinie et définie implicitement par le noyau $k(\cdot,\cdot)$. Une formulation équivalente, mais en dimension finie et plus efficace, est proposée, dans laquelle la matrice du noyau est apprise au lieu de la matrice métrique [7]. Dans ce contexte, l'algorithme d'apprentissage de métrique à noyau *ne calcule pas la matrice métrique optimale* $\widehat{M}$ mais plutôt

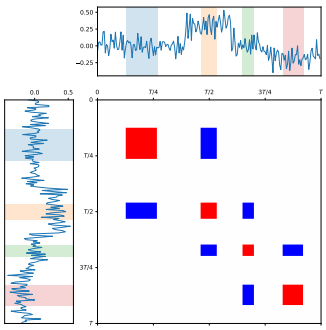


Figure 2 – Illustration de la méthode permettant de transformer des annotations en contraintes. Les annotations sont mises en surbrillance dans les zones colorées du signal. Les contraintes de similarité/dissimilarité sont stockées dans une matrice A telle que $A_{st} = 1$ (en rouge) si y_s et y_t sont similaires, $A_{st} = -1$ (en bleu) si y_s et y_t sont dissimilaire, 0 sinon (en blanc).

la matrice de Gram $\widehat{G}$ telle que $Gst := \phi(y_s^{\text{train}})'\widehat{M}\phi(y_t^{\text{train}})$ pour tous les échantillons y_s^{train} et y_t^{train}. Ensuite, la formulation équivalente est résolue à l'aide de la méthode itérative de Bregman [7], avec la règle de mise à jour suivante:

$$\widehat{G} \longleftarrow \widehat{G} + \beta\widehat{G}(e_s - e_t)(e_s - e_t)'\widehat{G} \tag{4}$$

où e_t est le t-ème vecteur de la base canonique et où $\beta \in \mathbb{R}$ dépend de $\phi(y_s^{\text{train}})$, $\phi(y_t^{\text{train}})$ et de s'ils sont similaires ou non. Chaque mise à jour a une complexité de l'ordre de $\mathcal{O}(T_{\text{train}}^2)$, où T_{train} est le nombre d'échantillons d'apprentissage.
Après l'apprentissage de la métrique, il ne reste plus qu'à combiner la métrique de Mahalanobis $\|\cdot\|_{\widehat{M},\mathcal{H}}$ avec un algorithme de détection de rupture. À cette fin, il faut pouvoir calculer les produits scalaires $\phi(z_s)'\widehat{M}\phi(z_t)$ pour tous les échantillons z_s et z_t pour un nouveau signal z. Une difficulté réside dans le fait que la matrice de métrique $\widehat{M}$ n'est pas explicitement disponible. Grâce à un théorème du représentant [7, Theorem 1], la connaissance de $\widehat{G}$ s'avère suffisante. En détail, l'expression suivante peut être utilisée pour tous les échantillons z_s et z_t:

$$\phi(z_s)'\widehat{M}\phi(z_t) = k(z_s,z_t) + k_s' \, G^{-1}(\widehat{G} - G)G^{-1} \, k_t \tag{5}$$

où $k_\bullet := [k(z_\bullet,y_1^{\text{train}}), k(z_\bullet,y_2^{\text{train}}), \dots]'$ et G est la matrice de Gram des échantillons d'apprentissage dans l'espace non transformé, i.e. $G_{st} := k(y_s^{\text{train}},y_t^{\text{train}})$.

Intuition sur la métrique apprise. Utiliser la norme $\|\cdot\|_{\mathcal{H},M}$ revient à effectuer les opérations suivantes : les échantillons sont d'abord envoyés vers un espace de grande dimension (via ϕ) ensuite, ils sont transformés linéairement, puis des sauts de moyenne sont détectés. En effet, la décomposition de la matrice symétrique $M = U'U$ donne $\|\phi(y_t)\|_{\mathcal{H},M} = \|U\phi(y_t)\|_{\mathcal{H}}$. Par conséquent, mesurer les distances avec la norme $\|\cdot\|_{\mathcal{H},M}$ équivaut à appliquer une transformation $U\phi(\cdot)$ aux données. La somme des résidus V (2) mesure l'erreur d'approximation du signal transformé $\{U\phi(y_t)\}_t$ par une fonction constante par morceaux. La première transformation ϕ est non supervisée (i.e. non spécifique à une tâche) et extrait un grand nombre de descripteurs (éventuellement infinis) tandis que la seconde transformation U est linéaire et spécifique à une tâche. Il faut noter que si le noyau k définit implicitement un rkhs de dimension infinie, la transformation, déterminée par M (également de dimension infinie), n'est pas paramétrique.

3 Résultats

Nous combinons la procédure d'apprentissage de métrique pour améliorer les performances de quatre méthodes de détection non supervisées sur un jeu de données réelles, appelé *Marche*.

3.1 Cadre expérimental

Données. Les données *Marche* sont composées de 262 enregistrements annotés (fréquence d'échantillonnage: 100 Hz) provenant d'un capteur inertiel placé dans le bas du dos de sujets effectuant une séquence d'activités simples [1]. Les régimes successifs sont les suivants: ≪Debout≫, ≪Marche≫, ≪Demi-tour≫, ≪Marche≫, ≪Arrêt≫. La tâche consiste à détecter les instants où l'activité du sujet change. Pour cette étude, deux dimensions sont utilisées: la vitesse angulaire autour de l'axe vertical (≪Rot. Z≫) et l'accélération verticale (≪Acc. Z≫). Un exemple est présenté sur la figure 1. Les deux dimensions sont mises à l'échelle pour avoir une moyenne nulle et une variance égale à 1. Dans la suite, la représentation temps-fréquence des signaux de *Marche* est définie comme la transformée de Fourier à court terme (STFT), calculée avec 300 échantillons par segment et un chevauchement de 299 échantillons. Les annotations partielles sont composées de 50 échantillons (soit 0,5 s) prélevés au milieu de chaque régime.

Algorithmes de détection. Les quatre algorithmes de détection non-supervisés sont `OptLin` [4] (programmation dynamique et norme euclidienne), `OptGau` [5] (programmation et la norme rkhs induite par le noyau gaussien), `WinLin` [2] (fenêtre glissante et norme euclidienne) et `WinGau` [6] (fenêtre glissante et norme rkhs induite par le noyau gaussien)[1]. Les méthodes à fenêtre glissante utilisent une fenêtre longue de 100 échantillons. Les algorithmes supervisés associés ($M = \widehat{M}$) sont notés de manière transparente ♡`OptLin`, ♡`WinLin`, ♡`OptGau` et ♡`WinGau`. Les algorithmes utilisant le noyau linéaire (se terminant en `Lin`) prennent en entrée la représentation temps-fréquence des signaux. Les algorithmes utilisant le noyau gaussien (se terminant en `Gau`) prennent en entrée les signaux originaux (mis à l'échelle). Les méthodes supervisées (celles avec un ♡) sont évaluées par validation croisée (avec 10 sous-ensembles).

Métriques d'évaluation. Pour évaluer la précision de la segmentation, deux métriques sont introduites: HAUSDORFF et F1 SCORE. La métrique HAUSDORFF mesure la pire erreur de prédiction entre un ensemble d'indices de rupture $\{t_1,t_2,\dots\}$ et leurs estimations $\{\hat{t}_1,\hat{t}_2,\dots\}$. HAUSDORFF$(\{t_k\}_k,\{\hat{t}_k\}_k)$

[1] Tous les algorithmes sont implémentés dans la librairie Python `ruptures`, disponible à github.com/deepcharles/ruptures.

Table 1 – Résultats de la segmentation (moyenne et écart type)

	HAUSDORFF	F1 SCORE
WinLin	2,92 (±3,21)	0,81 (±0,17)
♡WinLin	2,04 (±2,54)	0,82 (±0,18)
OptLin	1,80 (±2,35)	0,84 (±0,17)
♡OptLin	1,10 (±0,72)	0,85 (±0,13)
WinGau	5,79 (±2,86)	0,64 (±0,17)
♡WinGau	3,77 (±3,29)	0,78 (±0,19)
OptGau	1,44 (±2,12)	0,90 (±0,15)
♡OptGau	0,99 (±1,59)	0,94 (±0,13)

(a) Résultats globaux

	Debout/Marche	Marche/Demi-tour	Demi-tour/Marche	Marche/Arrêt
WinLin	2,00 (±2,81)	0,94 (±1,58)	1,28 (±2,11)	1,42 (±2,39)
♡WinLin	1,00 (±2,02)	0,85 (±1,09)	0,66 (±0,86)	1,33 (±2,03)
OptLin	0,60 (±0,92)	0,38 (±0,62)	0,38 (±0,43)	1,69 (±2,24)
♡OptLin	0,42 (±0,42)	0,66 (±0,42)	0,61 (±0,35)	0,93 (±0,73)
WinGau	5,21 (±3,05)	1,20 (±1,64)	2,27 (±2,75)	1,50 (±2,34)
♡WinGau	2,98 (±3,22)	1,25 (±1,91)	2,37 (±2,96)	1,20 (±2,23)
OptGau	0,51 (±0,38)	0,41 (±1,05)	0,42 (±0,78)	1,23 (±2,11)
♡OptGau	0,42 (±0,41)	0,42 (±1,17)	0,42 (±0,90)	0,74 (±1,50)

(b) Erreur absolue (en seconde) par type de rupture

Table 2 – Influence de la largeur d'annotation (pour ♡OptGau)

	0,1 s	0,5 s	1,5 s
HAUSDORFF	1,22 (±1,90)	0,99 (±1,59)	1,15 (±1,80)
F1 SCORE	0,92 (±0,14)	0,94 (±0,13)	0,92 (±0,14)

est exprimé en seconde et est égal à

$$\max\{\max_k \min_l |t_k - \hat{t}_l|\,, \max_k \min_l |\hat{t}_k - t_l|\}. \qquad (6)$$

Le F1 SCORE est la moyenne géométrique de la precision $\text{PR} := \#\text{TP}/\#\{\hat{t}_l\}_l$ du rappel $\text{RA} := \#\text{TP}/\#\{t_k\}_k$ où l'ensemble des vrais positifs $\text{TP} := \{t_k \,|\, \exists\, \hat{t}_l \text{ t.q. } |\hat{t}_l - t_k| < M\}$ contient les ruptures détectées, à une marge $M = 1$ s près.

3.2 Résultats et discussion

Plusieurs observations sont faites à partir des résultats rapportés dans le tableau 1. Le plus important est que la supervision améliore considérablement les performances de la segmentation : pour les deux métriques, les tests de Wilcoxon signés entre les scores d'une méthode de détection et son équivalent supervisé produisent des p-valeurs bien inférieures à 1%. La méthode la plus précise, et la seule avec un HAUSDORFF en dessous de 1 seconde, est ♡OptGau. À titre de comparaison, les signaux provenant des données *Marche* durent entre 17 et 40 secondes. Cela prouve que notre approche est capable d'apprendre une métrique appropriée à partir des signaux bruts et de remplacer une étape de pré-traitement telle qu'une STFT. Il est intéressant de noter que pour cet algorithme, la supervision améliore principalement la détection de la dernière rupture (Marche/Arrêt) d'environ 0,5 seconde (de 1,23 à 0,74, voir le tableau 1-b), même si elle est la moins bien détectée en l'absence de supervision. Cela est dû au fait que les contraintes (de similarité ou dissimilarité) qui sont le moins respectées par la norme d'origine déterminent le plus la métrique apprise [7]. En conséquence, la rupture la moins précisément détectée par l'algorithme non supervisé est susceptible de bénéficier de la plus grande amélioration de la précision de la détection. Cela peut entraîner une diminution de la précision lors de l'estimation des autres ruptures. Par exemple, OptLin détecte les changements entre "Marche" et "Demi-tour", mieux que ♡OptLin. Néanmoins, les deux autres ruptures sont mieux estimées par ♡Optlin et, selon HAUSDORFF et F1 SCORE, la détection est encore considérablement améliorée par notre approche.

Ces résultats ont été obtenus en utilisant des annotations de 0,5 seconde au milieu de chaque régime. Le tableau 2 indique pour différentes largeurs d'annotation partielle la précision de ♡OptGau. Lorsque seulement $0{,}1$ s de chaque régime est annoté, la segmentation est toujours plus précise que pour OptGau, mais pas aussi bonne que lorsque $0{,}5$ s est annoté, car moins d'informations sont fournies. Inversement, il peut aussi y avoir trop d'annotation: pour $1{,}5$ s, les performances diminuent par rapport à $0{,}5$ s. En effet, le régime le plus court ("Arrêt") est souvent inférieur à $1{,}5$ s, ce qui signifie que les annotations incluent des échantillons ambigus proches des ruptures et susceptibles de générer des contraintes fortes mais non voulues. En résumé, davantage d'annotations améliorent la précision de la segmentation, mais les échantillons situés autour des changements de régime doivent être supprimés.

4 Conclusion

Nous avons étendu une procédure d'apprentissage de métrique à base de noyau à la détection de ruptures. Notre approche apprend une transformation de signal non linéaire, non paramétrique et spécifique à une tâche, en utilisant des annotations partielles fournies par un expert. Cette méthodologie offre ainsi une nouvelle perspective pour la segmentation supervisée de séries chronologiques, Des expériences sur des données réelles ont montré que la supervision améliore considérablement les performances de détection pour plusieurs algorithmes de segmentation.

References

[1] R. Barrois-Müller, T. Gregory, L. Oudre, T. Moreau, C. Truong, A. Aram Pulini, A. Vienne, C. Labourdette, N. Vayatis, S. Buffat, A. Yelnik, C. de Waele, S. Laporte, P.-P. Vidal, and D. Ricard. An automated recording method in clinical consultation to rate the limp in lower limb osteoarthritis. *PLoS One*, 11(10):e0164975, 2016.

[2] M. Basseville and I. Nikiforov. *Detection of abrupt changes: theory and application*, volume 104. Prentice Hall Englewood Cliffs, 1993.

[3] R. Bellman. On a routing problem. *Quaterly of Applied Mathematics*, 16(1):87–90, 1955.

[4] D. Garreau and S. Arlot. Consistent change-point detection with kernels. *Electronic Journal of Statistics*, 12(2):4440–4486, 2018.

[5] Z. Harchaoui and O. Cappé. Retrospective mutiple change-point estimation with kernels. In *Proceedings of the IEEE/SP Workshop on Statistical Signal Processing*, pages 768–772, Madison, Wisconsin, USA, 2007.

[6] Z. Harchaoui, F. Vallet, A. Lung-Yut-Fong, and O. Cappé. A regularized kernel-based approach to unsupervised audio segmentation. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1665–1668, Taipei, Taiwan, 2009.

[7] P. Jain, B. Kulis, J. V. Davis, and I. S. Dhillon. Metric and kernel learning using a linear transformation. *Journal of Machine Learning Research (JMLR)*, 13:519–547, 2012.

[8] R. Lajugie, F. Bach, and S. Arlot. Large-margin metric learning for constrained partitioning problems. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 297–395, Beijing, China, 2014.